Representation Learning for Uncertainty-Aware Clinical Decision Support

Dissertation von Zhiliang Wu

München 2022

Representation Learning for Uncertainty-Aware Clinical Decision Support

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

eingereicht von
Zhiliang Wu

aus
Shanghai, China

December 8, 2021

Erstgutachter:	Prof. Dr. Volker Tresp
Zweitgutachter:	Prof. Dr. Thorsten Joachims
Drittgutachter:	Prof. Dr. Florian Büttner
Tag der mündlichen Prüfung:	22.03.2022

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Wu, Zhiliang

Name, Vorname

München, 08.12.2021

Ort, Datum

Wu, Zhiliang

Unterschrift Doktorand/in

Formular 3.2

Contents

Abstract

Over the last decade, there has been an increasing trend towards digitalization in healthcare, where a growing amount of patient data is collected and stored electronically. These recorded data are known as electronic health records. They are the basis for state-of-the-art research on clinical decision support so that better patient care can be delivered with the help of advanced analytical techniques like machine learning. Among various technical fields in machine learning, representation learning is about learning good representations from raw data to extract useful information for downstream prediction tasks. Deep learning, a crucial class of methods in representation learning, has achieved great success in many fields such as computer vision and natural language processing. These technical breakthroughs would presumably further advance the research and development of data analytics in healthcare. This thesis addresses clinically relevant research questions by developing algorithms based on state-of-the-art representation learning techniques. When a patient visits the hospital, a physician will suggest a treatment in a deterministic manner. Meanwhile, uncertainty comes into play when the past statistics of treatment decisions from various physicians are analyzed, as they would possibly suggest different treatments, depending on their training and experiences. The uncertainty in clinical decision-making processes is the focus of this thesis. The models developed for supporting these processes will therefore have a probabilistic nature. More specifically, the predictions are predictive distributions in regression tasks and probability distributions over, e.g., different treatment decisions, in classification tasks. The first part of the thesis is concerned with prescriptive analytics to provide treatment recommendations. Apart from patient information and treatment decisions, the outcome after the respective treatment is included in learning treatment suggestions. The problem setting is known as learning individualized treatment rules and is formulated as a contextual bandit problem. A general framework for learning individualized treatment rules using data from observational studies is presented based on state-of-the-art representation learning techniques. From various offline eval-

uation methods, it is shown that the treatment policy in our proposed framework can demonstrate better performance than both physicians and competitive baselines. Subsequently, the uncertainty-aware regression models in diagnostic and predictive analytics are studied. Uncertainty-aware deep kernel learning models are proposed, which allow the estimation of the predictive uncertainty by a pipeline of neural networks and a sparse Gaussian process. By considering the input data structure, respective models are developed for diagnostic medical image data and sequential electronic health records. Various pre-training methods from representation learning are adapted to investigate their impacts on the proposed models. Through extensive experiments, it is shown that the proposed models delivered better performance than common architectures in most cases. More importantly, uncertainty-awareness of the proposed models is illustrated by systematically expressing higher confidence in more accurate predictions and less confidence in less accurate ones. The last part of the thesis is about missing data imputation in descriptive analytics, which provides essential evidence for subsequent decision-making processes. Rather than traditional mean and median imputation, a more advanced solution based on generative adversarial networks is proposed. The presented method takes the categorical nature of patient features into consideration, which enables the stabilization of the adversarial training. It is shown that the proposed method can better improve the predictive accuracy compared to traditional imputation baselines.

Zusammenfassung

In den vergangenen zehn Jahren entwickelte sich eine steigende Tendenz zur Digitalisierung im Gesundheitswesen, wobei eine zunehmende Anzahl der Patientendaten elektronisch gesammelt und gespeichert wird. Diese aufgezeichneten Daten sind auch bekannt als elektronische Gesundheitsakten. Sie bilden die Grundlage für die moderne Forschung über klinische Entscheidungsunterstützung, sodass eine bessere Patientenversorgung mithilfe von fortschrittlichen Analysetechniken wie maschinellem Lernen gewährleistet werden kann. Einer der vielen technischen Bereiche des maschinellen Lernens ist Repräsentation-Lernen, das gute Repräsentationen aus Rohdaten lernt, um nützliche Informationen für diverse Aufgaben zu extrahieren. Deep Learning, eine wichtige Untergruppe des Repräsentation-Lernens, hat große Erfolge in verschiedenen Bereichen erzielt, wie z.B. in Computervision und Verarbeitung natürlicher Sprache. Dieser technische Durchbruch kann wahrscheinlich auch die Forschung und Entwicklung der Datenanalyse im Gesundheitswesen weiter fördern. Die vorliegende Arbeit konzentriert sich auf die Behandlung klinisch relevanter Forschungsfragen mittels moderner Techniken des Repräsentation-Lernens. Wenn die Patientin oder der Patient das Krankenhaus besucht, wird die Ärztin oder der Arzt eine Behandlung deterministisch empfehlen. Dabei kommt die Ungewissheit ins Spiel, wenn die vergangenen Statistiken der Behandlungsentscheidungen von verschiedenen Ärztinnen und Ärzten analysiert werden, weil sie aufgrund unterschiedlicher Ausbildung und Erfahrung eventuell andere Behandlungen empfehlen würden. Diese Ungewissheitsperspektive in klinischen Entscheidungsprozessen ist der Schwerpunkt dieser Arbeit. Die für klinische Entscheidungsunterstützung entwickelten Modelle haben deswegen einen probabilistischen Charakter. Genauer gesagt sind die Vorhersagen aus den Modellen in der klinischen Entscheidungsunterstützung prädiktive Verteilungen bei Regressionsaufgaben und Wahrscheinlichkeitsverteilungen über z.B. verschiedene Behandlungsentscheidungen bei Klassifizierungsaufgaben. Der erste Teil dieser Arbeit befasst sich mit präskriptiver Analytik, um geeignete Behandlungen zu empfehlen. Neben Patienteninformationen und

Behandlungsentscheidung wird ebenfalls das Ergebnis der jeweiligen Behandlung beim Lernen der Behandlungsempfehlung eingeschlossen. Die Fragestellung ist auch bekannt als das Lernen von individualisierten Behandlungsstrategien und wird als ein Contextual Bandit Problem formuliert. Ein allgemeiner Rahmen für das Lernen individualisierter Behandlungsstrategien mit Daten aus Beobachtungsstudien wird basierend auf modernen Techniken des Repräsentation-Lernens präsentiert. Von verschiedenen Offline-Evaluierungsmethoden wird gezeigt, dass die Behandlungsstrategie aus unserem vorgeschlagenen Rahmen eine bessere Leistung als Ärztinnen, Ärzte und wettbewerbsfähige Baselines demonstriert. Anschließend werden ungewissheitsbewusste Regressionsmodelle von diagnostischer und prädiktiver Analytik erforscht. Ungewissheitsbewusste Deep Kernel Learning Modelle werden vorgeschlagen, die die Einschätzung der Ungewissheit durch eine Pipeline von neuronalen Netzwerke und einem sparse Gaussian Process ermöglichen. Unter Berücksichtigung der Eingabedaten-Struktur werden die jeweiligen Modelle für diagnostische medizinische Bilddaten und longitudinale elektronische Gesundheitsakten entwickelt. Unterschiedliche Pre-trainingsmethoden aus Repräsentation-Lernen werden angepasst, um die Auswirkungen auf die vorgeschlagenen Modelle zu untersuchen. Nach zahlreichen Experimente wird gezeigt, dass die vorgeschlagenen Modelle in den meisten Fällen eine bessere Leistung als übliche Architekturen liefern. Noch wichtiger wird das Ungewissheitsbewusstsein der vorgeschlagenen Modelle durch systematisch höheres Vertrauen in präzisere Vorhersagen und niedrigeres Vertrauen in weniger präzisere Vorhersagen verdeutlicht. Der letzte Teil dieser Arbeit ist über das Problem der fehlenden Daten in der deskriptiven Analytik, die wichtige Hinweise für die nachfolgenden Entscheidungsprozesse bietet. Anstatt traditioneller Methoden wie Mittelwert- und Medianwert-Imputation wird eine fortschrittlichere Lösung basierend auf Generative Adversarial Networks vorgeschlagen. Die vorlegte Methode berücksichtigt den kategorischen Charakter der Patientenmerkmale, womit die Stabilisierung des Adversarial Trainings ermöglicht wird. Es wurde gezeigt, dass die vorgeschlagene Methode die Vorhersagegenauigkeit gegenüber traditionellen Imputation-Baselines verbessern konnte.

Acknowledgement

This dissertation summarizes my past three years' Ph.D. study at Siemens Technology and LMU Munich. It would not be possible without the support of many people.

First, I would like to express my deepest gratitude to my supervisor and mentor, Prof. Dr. Volker Tresp. Volker has provided me with the best possible platform to conduct my research, where I can always get support from various colleagues and feel at home when talking to them. After introducing me to the topic of machine learning in clinical decision support, Volker has given me great trust and freedom to pursue various ideas, including causality analysis and Gaussian Processes. Each time I need any help or guidance, he always shows strong interest, gives insightful comments, and encourages me to work on innovative ideas. I appreciate his efforts in helping me prepare the publications, where I can closely learn how a world-class researcher thinks and works. In addition, I am very honored that Prof. Dr. Thorsten Joachims and Prof. Dr. Florian Büttner have agreed to be the external examiners of the thesis.

Four years ago, I was admitted to the Siemens Mentoring Program organized by Dr. Fabian Rhein. From that, I got the chance to know Prof. Dr. Thomas Runkler at TU Munich and Dr. Sigurd Spiekermann at Siemens Technology, who later took me as a working student and introduced me to Volker for a possible Ph.D. position. I am deeply grateful to them for their continuous support so that I can make good decisions by talking to the right people at the right time.

I feel privileged to work with many excellent colleagues, including Dr. Yinchong Yang, Yushan Liu, Jindong Gu, and Dr. Yunpu Ma. Throughout my Ph.D. study, Yinchong has been like a second mentor to me. He offered me the chance to discuss any complicated topics with him, which kept me on the right track in the research. His great work about representation learning in healthcare forms the backbones of my research work. I will never forget the midnight where we discussed generalized linear models. Yushan's outstanding mathematical knowledge has facilitated many topics during my Ph.D. study, where I can al-

ways ask trivial questions but get insightful comments. Her perseverance on mathematical precision has broadly impacted my attitude towards writings and publications. Jindong's strong expertise in computer vision has encouraged me to work on vision-based topics in the medical domain. His enthusiastic attitude towards research has always been inspiring to me. Yunpu's profound knowledge in causality analysis has prevented me from getting lost in this big research field at the beginning of my Ph.D. study. Apart from them, I have also received a lot of help and advice from other colleagues, including but not limited to Dr. Rui Zhao, Dr. Michael Moor, Sophia Althammer, Prof. Dr. Peter A. Fasching, Dr. Denis Krompass, Dr. Ulli Waltinger and Dr. Michael May.

Last but not least, I would like to thank my parents for being supportive throughout my study overseas. Special thanks to Huan for her love, tolerance, and patience during my stressful times. I am fortunate to have such strong support from my family throughout this unique journey.

List of Publications and Declaration of Authorship

- **Zhiliang Wu**, Yinchong Yang, Yunpu Ma, Yushan Liu, Rui Zhao, Michael Moor, and Volker Tresp. Learning Individualized Treatment Rules with Estimated Translated Inverse Propensity Score. In *2020 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–11, 2020. doi: 10.1109/ICHI48887.2020.9374397
  Best Paper Award at ICHI 2020 Analytics Track

  I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I regularly discussed this work with the co-authors, Yinchong Yang, Yunpu Ma, Yushan Liu, Rui Zhao, Michael Moor, and Volker Tresp. All co-authors contributed to improving the manuscript.

  This published work serves as Chapter 2.

- **Zhiliang Wu**, Yinchong Yang, Jindong Gu, and Volker Tresp. Quantifying Predictive Uncertainty in Medical Image Analysis with Deep Kernel Learning. In *2021 IEEE 9th International Conference on Healthcare Informatics (ICHI)*, pages 63–72, 2021b. doi: 10.1109/ICHI52183.2021.00022

  I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I regularly discussed this work with the co-authors, Yinchong Yang, Jindong Gu, and Volker Tresp. All co-authors contributed to improving the manuscript.

  This published work serves as Chapter 3.

- **Zhiliang Wu**, Yinchong Yang, Peter A Fashing, and Volker Tresp. Uncertainty-Aware Time-to-Event Prediction using Deep Kernel Accelerated Failure Time Models. In *Proceedings of the 6th Machine Learning for Healthcare Conference*, volume 149 of *Proceedings of Machine Learning Research*, pages 54–79. PMLR, 06–07 Aug 2021a

  I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I regularly discussed this work with the co-authors, Yinchong Yang, Peter A. Fasching, and Volker Tresp. All co-authors contributed to improving the manuscript.

  This published work serves as Chapter 4.

- Yinchong Yang, **Zhiliang Wu**, Volker Tresp, and Peter A. Fasching. Categorical EHR Imputation with Generative Adversarial Nets. In *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–10. IEEE, 2019. doi: 10.1109/ICHI.2019.8904717

  Yinchong Yang conceived of the original research contributions, where I helped him on performing implementations and evaluations. Yinchong Yang wrote the initial draft of the manuscript. Yinchong Yang and I worked on the subsequent corrections. Yinchong Yang and I regularly discussed this work with the other co-authors, Volker Tresp and Peter A. Fasching. All co-authors contributed to improving the manuscript.

  This published work serves as Chapter 5.

Other publications

- Jindong Gu, **Zhiliang Wu**, and Volker Tresp. Introspective Learning by Distilling Knowledge from Online Self-explanation. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020. doi: 10.1007/978-3-030-69538-5_3

- Malte Feucht, **Zhiliang Wu**, Sophia Althammer, and Volker Tresp. Description-based Label Attention Classifier for Explainable ICD-9 Classification. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 62–66. Association for Computational Linguistics, November 2021. doi: 10.18653/v1/2021.wnut-1.8

Chapter 1

Introduction

The purpose of this chapter is to offer a technical background of all contributions covered in the following chapters. Section 1.1 includes an overview of deep learning as a powerful representation learning technique. The topic is motivated from the perspective of general machine learning and artificial intelligence in subsection 1.1.1. With notations introduced in subsection 1.1.2, different neural network architectures are summarized in subsection 1.1.3. The respective implications in the medical applications are presented in subsection 1.1.4. Section 1.2 provides an introduction from the perspective of analytics in healthcare. In its subsequent subsections, the technical background of each chapter is explained, including motivations, problem settings, related works, and highlighted contributions.

1.1 Deep Learning as Representation Learning

1.1.1 Motivation

Machine learning (ML) is a technical field addressing problems of constructing computer programs to improve with experience automatically (Mitchell, 1997). Meanwhile, the goal of artificial intelligence (AI) is to build intelligent entities being capable of perceiving, understanding, predicting, and even manipulating the outside world (Russell and Norvig, 2009). Over the past decades, ML has found most interest in achieving these goals and therefore serves as a foundation for the modern AI among different technical fields (Russell and Norvig, 2009; Murphy, 2012). Unlike statistical modeling, the ML community lays more emphasis on predictive accuracy instead of data models. This is also regarded as the

algorithmic modeling culture in the statistics community (Breiman, 2001).

In conventional machine learning, the performance of a learning algorithm largely depends on the quality of hand-crafted features from, e.g., domain experts, which is also regarded as sample representations. The pre-processing and transformation of data from raw inputs to sample representations is known as feature engineering. In addition, many algorithms assume a low-dimensional set up of sample representations to avoid the curse of dimensionality (Bellman, 1966). If the dimensionality of the feature space is too high, the standard recipe involves algorithms of either dimensionality reduction or feature selection. However, for most real-world tasks, it is not easy to know how features could be extracted. Especially with unstructured data, it remains difficult even for those who have substantial prior knowledge. Examples include raw pixel valued images or a sequence of words, corresponding to the problem settings in computer vision (CV) and natural language processing (NLP), respectively. It is thus tempting to ask whether it is possible to automate learning representations from raw inputs.

To learn good representations, several priors are proven to be helpful, including smoothness and hierarchical organization of explanatory factors (Bengio et al., 2013). In particular, with an increasing amount of data being collected and the advance of modern computing hardware modules like Graphics Processing Units (GPUs), deep learning (DL) shows "unreasonable" effectiveness in many real-world use cases (Sejnowski, 2020). As an important class of methods in representation learning, DL belongs to ML but enables the learning of very complex functions in an end-to-end fashion from raw inputs (LeCun et al., 2015). The breakthroughs with DL arguably started from the encouraging results on speech recognition by Mohamed et al. (2009) and state-of-the-art performance in object recognition on ImageNet from Krizhevsky et al. (2012), while the origin of Deep Neural Networks (DNNs) dates back to the perceptron in the mid-twentieth century from Rosenblatt (1961). The inductive bias (model assumptions) of DNNs enables the self-organized representation learning from high-dimensional input data, which nicely addresses the above-discussed limitations of traditional machine learning.

1.1.2 Notation

In this chapter, scalars are mostly denoted by lowercase letters x; vectors are denoted by bold lowercase letters $\boldsymbol{x}$ with elements x_i; matrices are denoted by bold uppercase letters $\boldsymbol{X}$ with entries x_{ij} in the i-th row and j-th column; sets are denoted by calligraphic letters such as $\mathcal{X}$ with the cardinality $|\mathcal{X}|$. Vectors are assumed to be column vectors, i.e., $\boldsymbol{x} \in \mathbb{R}^p$.

Concatenating n vectors horizontally results in a matrix as $\boldsymbol{X} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{p \times n}$.

1.1.3 Neural Networks-Based Building Blocks

Various NN architectures have been proposed to address the challenges of data having a structured or unstructured nature. The inductive bias from Multilayer Perceptrons, Convolutional Neural Networks, and Recurrent Neural Networks turns out to be most widely used to learn latent representations from raw inputs. These networks are involved as building blocks in the published work in the following chapters. In this subsection, we introduce the definitions and summarize characteristics of these NNs.

Multilayer Perceptrons

Multilayer Perceptrons (MLP), also known as Deep Feedforward Networks (DFN) and Fully Connected Networks (FCN), are essential building blocks for modern DNNs. From the very first idea of linear perceptrons by Rosenblatt (1961) to the more recent variants like gMLP by Liu et al. (2021) and MLP-Mixer by Tolstikhin et al. (2021), MLPs have always been attracting much attention in the neural network research community. As a universal approximator, a single-layer MLP with an infinite number of hidden units has been proven to be able to well approximate any reasonable function (Cybenko, 1989; Hornik, 1991; Pinkus, 1999). In practice, we use MLPs with a finite number of hidden units as a powerful approximation model.

As shown in Figure 1.1, an MLP has at least three layers of nodes, an input layer of nodes, several hidden layers of nodes and an output layer of nodes. In supervised learning, the input and output layers of nodes correspond to raw inputs and labels, respectively, and the naming of hidden is based on the fact that these hidden layers of nodes are not observed in the datasets. Within each hidden layer of nodes, activation functions are applied to introduce a non-linearity between the layers. Note that the non-linear activation function is critical as otherwise, the composition of multiple linear layers can be merged into a single layer.

Formally, the mapping between each layer of nodes corresponds to a function $f^{(l)}(\cdot)$, where l refers to the index of the respective output layer of nodes. With the default activation being the rectified linear unit (ReLU) (Nair and Hinton, 2010), the l-th mapping function can be expressed as

$$f^{(l)}(\boldsymbol{x}; \boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}) = \max\{\boldsymbol{0}, \boldsymbol{W}^{(l)}\boldsymbol{x} + \boldsymbol{b}^{(l)}\},$$

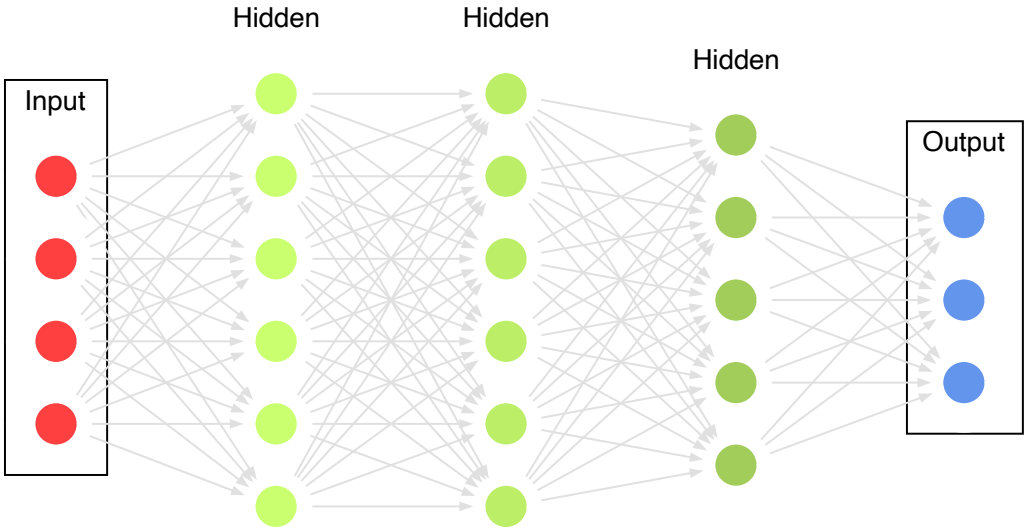


Figure 1.1: An MLP with three hidden layers, which consumes a four-dimensional vector as input and produces a three-dimensional target variable.

where $\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}$ are the weighting matrix and the bias term, respectively.

The function of a complete MLP can be expressed compactly as a composition of functions. For example, the entire function of an MLP with three hidden layers in Fig. 1.1 is

$$f(\boldsymbol{x}; \mathcal{W}, \mathcal{B}) := f^{(4)}\left(f^{(3)}\left(f^{(2)}\left(f^{(1)}(\boldsymbol{x})\right)\right)\right),$$

where $\mathcal{W}, \mathcal{B}$ denotes the sets of weighting matrices $\{\boldsymbol{W}^{(i)}\}_{i=1}^{4}$ and bias terms $\{\boldsymbol{b}^{(i)}\}_{i=1}^{4}$, respectively. And the activation function in $f^{(4)}$ is usually not ReLU but depends on the type of target variables, such as the softmax function for multi-class classification tasks.

The learning of the parameters is formulated as an optimization problem, which refers to using backpropagation (BP) algorithms to minimize a task-related loss function, e.g., Mean Squared Error (MSE) in regression problems and cross-entropy in classification problems. More concretely, the optimization algorithms are gradient descent-based methods, ranging from the classical stochastic gradient descent (SGD) (Robbins and Monro, 1951; Bottou et al., 1998) to the popular Adam optimizer (Kingma and Ba, 2015).

It has been shown that many functions can be represented much more efficiently with a deep network structure (multiple layers) than a shallow one (one hidden layer) (Delalleau and Bengio, 2011), which motivates the building of neural networks with an increasing depth. However, with a growing depth of the network, the training of an MLP becomes

more difficult. A careful initialization of the trainable parameters has shown to be beneficial to train ReLU-based deep networks (Glorot and Bengio, 2010; He et al., 2015). In addition, batch normalization has been proposed to accelerate the training by reducing internal covariate shift (Ioffe and Szegedy, 2015). Meanwhile, a deeper network structure introduces inevitably more trainable parameters, which could result in problems like overfitting. When the number of trainable parameters vastly exceeds the number of data samples, it is called over-parameterized. To retain the generalization ability of the over-parameterized models, regularization is an important and effective solution, which refers to adding a penalty term in the loss function to constrain the value of these parameters (Gelman and Vehtari, 2021). Especially for neural networks, Dropout is proposed as a simple form of regularization, which randomly drops units in the neural network during the training (Srivastava et al., 2014).

Convolutional Neural Networks

The convolution operation can be viewed as a weighted average operation between two functions. In NNs' terminology, the first function refers to the input, the second is a kernel or a filter with trainable parameters, and the respective output is a feature map (Goodfellow et al., 2016). If there are one or more convolutional operations within a neural network, it is named after Convolutional Neural Networks (CNNs) (LeCun et al., 1989, 1995). CNNs turn out to be especially useful for processing data having a grid-like topology, e.g., one-dimensional time series data lying on the grid of a time axis, two- or three-dimensional image data lying on the grid of pixels.

Formally, we denote the convolution operation with an asterisk, $*$. The feature map $\boldsymbol{s}$ from a one-dimensional (discrete) convolution operation is defined as

$$s_i = (\boldsymbol{x} * \boldsymbol{k})_i = \sum_{l=0}^{L-1} x_{i+l} k_l,$$

where $\boldsymbol{x} \in \mathbb{R}^p$ is the input vector, $\boldsymbol{k} \in \mathbb{R}^L$ is the kernel vector, $(\cdot)_i$ is the i-th element in a vector, and $L \in \mathbb{Z}_+$ is the size of the kernel.

Note that the input $\boldsymbol{x}$ has the index $i + m$ instead of $i - m$ to avoid the flipping of the kernel, which makes it indeed a cross-correlation operation (Goodfellow et al., 2016). However, in DL, the difference between these operations is negligible due to the trainable nature of the kernel parameters. Similarly, the feature map $\boldsymbol{S}$ from a two-dimensional

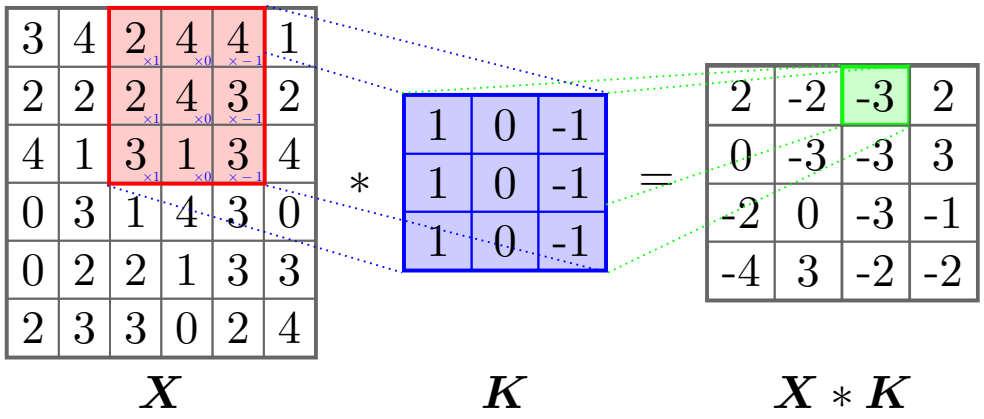


Figure 1.2: A 3x3 convolutional kernel applied on a randomly generated 6x6 matrix with no padding and stride being one.

(discrete) convolution operation is defined as

$$s_{ij} = (\boldsymbol{X} * \boldsymbol{K})_{ij} = \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} x_{i+h,j+w} k_{hw},$$

where $\boldsymbol{X} \in \mathbb{R}^{p \times q}$ is the input matrix, $\boldsymbol{K} \in \mathbb{R}^{H \times W}$ is the kernel matrix, $(\cdot)_{ij}$ is the ij-th entry in a matrix, $H, W \in \mathbb{Z}_+$ are the height and width of the kernel. An example of the 2D convolution operation is shown in Figure 1.2, where Dumoulin and Visin (2016) provide more technical details, such as different paddings or strides.

Compared with matrix multiplications in MLPs, the convolution operations enjoy the property of parameter sharing (the same kernel applies to different locations) and sparse connections (output values only depend on a few inputs). Consequently, there are usually fewer trainable parameters in CNNs than in MLPs for the same task. Similar to MLPs, non-linear activation functions are applied to the feature maps as convolution operations are, in essence, also linear. On the other hand, another particular operator in CNNs is the pooling function, which replaces the neighboring outputs with their summary statistics, such as averaging or maximum values, corresponding to average pooling and max pooling, respectively. The pooling functions facilitate learning representations being invariant to small translations of the inputs, which is vital for capturing the presence of certain features rather than their locations. In addition, we can use the pooling functions to learn latent representations with a fixed size, no matter how large the size of inputs is.

With these basic CNN-specific functions, the architecture of CNNs has evolved for a long time. In short, there is a trend of an increasing number of layers with more trainable parameters. For example, LeNet (LeCun et al., 1998) has 60 thousand trainable parameters

while VGG (Simonyan and Zisserman, 2014) owns 138 million. Meanwhile, the innovation from new building blocks further pushes forward the limit of CNNs, such as the skip connections in ResNet (He et al., 2016) and cross-layer information flow in DenseNet (Huang et al., 2017). Khan et al. (2020) offer a complete overview of the recent development of deep CNNs.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are the specialized NNs for processing sequential data, such as text data in NLP. Similar to CNNs, the modeling advantage of RNNs over MLPs is also attributed to parameter sharing. On the other hand, RNNs model the sequential data in a very different way to CNNs. Concretely speaking, given a sequence of input vectors, each element from the 1D convolutional operation outputs is based on some neighboring features in the input multiplied by the kernel. By comparison, each output element from an RNN is a function of the input vectors at their respective previous time steps. The later time step an output element refers to, the larger amount of input vectors it depends on. The gist of parameter sharing comes to RNNs by repeatably applying the function with the same parameters to compute latent representations and outputs.

Formally, the latent representations $\boldsymbol{h}_t$ and output predictions $\hat{\boldsymbol{y}}_t$ of a simple RNN in the Elmen architecutre (Elman, 1990) are learned by

$$\begin{aligned}\boldsymbol{h}_t &= g_1(\boldsymbol{U}\boldsymbol{x}_t + \boldsymbol{V}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\boldsymbol{h}}) \\ \hat{\boldsymbol{y}}_t &= g_2(\boldsymbol{W}\boldsymbol{h}_t + \boldsymbol{b}_{\boldsymbol{y}}),\end{aligned}$$

where $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$ are the input-to-hidden, hidden-to-hidden, hidden-to-output weighting matrices, $\boldsymbol{b}_{\boldsymbol{h}}, \boldsymbol{b}_{\boldsymbol{y}}$ are the bias terms, $g_1(\cdot)$ is an activation function, e.g., a hyperbolic tangent function, tanh, and $g_2(\cdot)$ is a target dependent activation function like a softmax function.

As shown in Figure 1.3, the latent representations $\boldsymbol{h}_t$ and the output prediction $\hat{\boldsymbol{y}}_t$ are functions of all its previous input vectors from $\boldsymbol{x}_0$ to $\boldsymbol{x}_t$. The trainable parameters in the functions, such as the weighting matrices $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}$, are shared across all time steps, which facilitates the generalization to sequential inputs with variable lengths. If there is only one single time step in RNN, the model degenerates to an MLP with one hidden layer. Moreover, the length of the input vectors does not have to match the length of the outputs, which results in various task-specific RNN types, such as many-to-many, many-to-one, or one-to-many[1].

[1]More RNN types can be found at `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`.

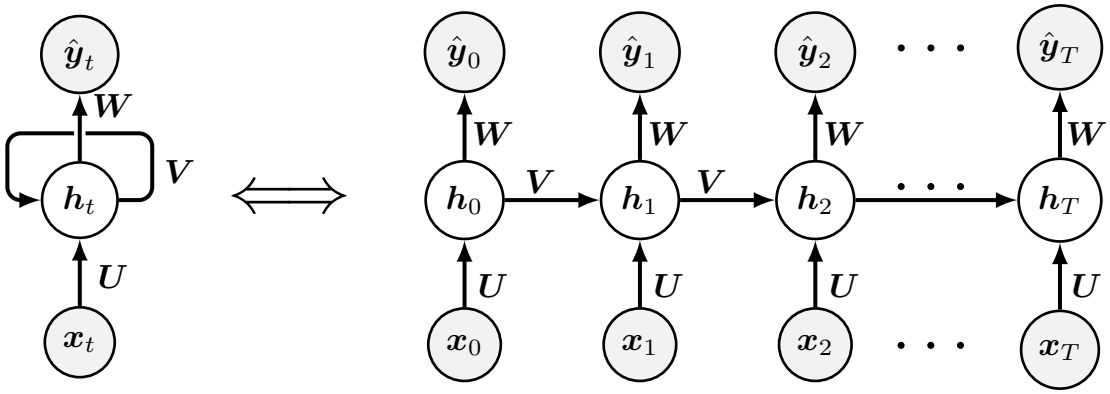


Figure 1.3: Recurrent Neural Networks in a rolled (left) and an unrolled view (right). Given sequential inputs $\{\boldsymbol{x}_t\}_{t=0}^{T}$, the RNN output predictions $\{\hat{\boldsymbol{y}}_t\}_{t=0}^{T}$ through latent representations $\{\boldsymbol{h}_t\}_{t=0}^{T}$.

However, the long-term dependency of output elements in later time steps could also hinder the information flow, i.e., the input vectors in earlier time steps have a relatively smaller influence than the more recent ones. When RNNs are trained with gradient-based algorithms such as back-propagation through time (Werbos, 1990), the long-term dependency leads to practical training issues known as vanishing gradients (Bengio et al., 1994; Hochreiter, 1998), which means the gradient from the error caused in later predictions are difficult to be propagated to trainable parameters in earlier time steps. To facilitate the training with long-term dependencies, more advanced RNNs are proposed with gating mechanisms, where long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Chung et al., 2014) are the two most successful models. Formally, if we follow definitions in Chung et al. (2014) and Graves et al. (2013), the modeling of the latent representations $\boldsymbol{h}_t$ in LSTM is

$$
\begin{aligned}
\boldsymbol{i}_t &= \sigma\left(\boldsymbol{W}_{\boldsymbol{xi}}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{hi}}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{\boldsymbol{ci}}\boldsymbol{c}_{t-1} + \boldsymbol{b}_{\boldsymbol{i}}\right) \\
\boldsymbol{f}_t &= \sigma\left(\boldsymbol{W}_{\boldsymbol{xf}}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{hf}}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{\boldsymbol{cf}}\boldsymbol{c}_{t-1} + \boldsymbol{b}_{\boldsymbol{f}}\right) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh\left(\boldsymbol{W}_{\boldsymbol{xc}}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{hc}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\boldsymbol{c}}\right) \\
\boldsymbol{o}_t &= \sigma\left(\boldsymbol{W}_{\boldsymbol{xo}}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{ho}}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{\boldsymbol{co}}\boldsymbol{c}_t + \boldsymbol{b}_{\boldsymbol{o}}\right) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh\left(\boldsymbol{c}_t\right)
\end{aligned}
$$

while in GRU, it is

$$
\begin{aligned}
\boldsymbol{z}_t &= \sigma\left(\boldsymbol{W}_{\boldsymbol{xz}}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{hz}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\boldsymbol{z}}\right) \\
\boldsymbol{r}_t &= \sigma\left(\boldsymbol{W}_{\boldsymbol{xr}}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{hr}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{\boldsymbol{r}}\right) \\
\hat{\boldsymbol{h}}_t &= \tanh\left(\boldsymbol{W}_{\boldsymbol{xh}}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{rh}}\left(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}\right) + \boldsymbol{b}_{\boldsymbol{h}}\right) \\
\boldsymbol{h}_t &= \left(1 - \boldsymbol{z}_t\right) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \hat{\boldsymbol{h}}_t
\end{aligned}
$$

where $\sigma(\cdot)$ denotes the sigmoid function, $\odot$ denotes the element-wise multiplication, $\boldsymbol{i}_t, \boldsymbol{f}_t, \boldsymbol{o}_t$ are the input, forget and output gates in LSTM, and $\boldsymbol{r}_t, \boldsymbol{z}_t$ are the update or reset gates in GRU.

1.1.4 Implications in Medical Applications

Since the introduction of the Electronic Health Records (EHRs), a similar trend of collecting and creating big datasets has been observed in the healthcare domain (Halpern et al., 2016; Tresp et al., 2016; Topol, 2019). The series of Medical Information Mart for Intensive Care (MIMIC) databases can be seen as a representative example of this movement (Moody and Mark, 1996; Saeed et al., 2011; Johnson et al., 2016, 2019, 2021). At the same time, physicians are becoming overwhelmed by the increasing amount of patient data, as it becomes wider in terms of features and longer in terms of timestamps. To this end, data-driven methods such as ML are expected to play an important role in providing better patient care. Unlike some other online advertisement domains, where several wrong predictions of the model are possibly tolerant, the treatment decisions in healthcare go through a complex process and require careful administration. Thus, data-driven clinical decision support (CDS) is preferred over fully automated solutions, where physicians' decisions can benefit from the second opinions offered by the data-driven models (Zan et al., 2010). The EHRs can be collected and saved in many forms, ranging from structured data like tabular data to unstructured data like X-ray images, clinical notes, and genomic sequences (Yu et al., 2018; Rajkomar et al., 2018; Esteva et al., 2019). The NN-based building blocks in Section 1.1.3 can be used to transform the raw input data from different forms of EHRs to latent representations, which would then be consumed by the downstream prediction tasks. In the following, we will provide an overview on the applications of these building blocks in the medical domain.

Multilayer Perceptrons

In medical statistics, linear regression and logistic regression are two popular models thanks to their white-box nature, where the values of the fitted coefficients can be interpreted as the importance of respective features. MLPs have been used as their nonlinear generalizations to further increase the predictive performance, where a sigmoid or a softmax activation function is added to the output layer of nodes for binary classification tasks and multi-class classification tasks, respectively. For example, Dreiseitl and Ohno-Machado (2002) present a methodology review of various classification algorithms with an emphasis on the comparison between logistic regression and MLPs. In the review, it is found that MLPs show better results than logistic regression in 18% sampled studies, although there is no significant difference between them in 42% cases and 39% cases are not considered due to inadequate statistical testing. In addition, Kuzmanovski and Aleksovska (2003) use both linear regression and MLPs to predict the unit cell parameters, where MLPs deliver much better results due to the non-linearity between the input features and target variables. In this work, MLPs serve as our default modeling choice for the learning representations with structured data in vectorized forms (i.e., $\boldsymbol{x}_i$ denotes the feature vector of the i-th data sample). In Chapter 2 and 4, the raw input data includes both static features and dynamic features, which differs in whether the feature has a time-stamp information. Following Esteban et al. (2016), We use MLPs to learn latent representations from static features for downstream tasks. For dynamic features, we use Recurrent Neural Networks to capture the time-dependency between features in different time steps, which will be discussed in more details in the following paragraphs.

Convolutional Neural Networks

In the medical domain, the applications of CNNs have gained the most attention and have advanced many state-of-the-art performances. Thanks to the similarity between the tasks using standard benchmark datasets like ImageNet (Deng et al., 2009) and the ones using medical image data, several CNN-based models have achieved human-level performance. Esteva et al. (2017) have trained a single CNN end-to-end using a dataset of $129,450$ clinical images, and the model shows a level of competence comparable to dermatologists, which facilitates the low-cost access to important diagnostic care. In radiology, Nam et al. (2019) showed that the automatic detection algorithm based on CNNs outperforms physicians in radiograph classification. If the algorithm is deployed as CDS to offer second opinions, the

physician's performance is expected to improve. With these success stories, many AI-based medical imaging services have received approval from the Food and Drug Administration (FDA) in the US and CE Marking in Europe (Benjamens et al., 2020). At the same time, the high resolution nature of the medical images and the safety-critical nature of the tasks have also in return highlighted many important aspects for future research, e.g., transparency, robustness, and fairness (Asan et al., 2020). In Chapter 3, we will include these state-of-the-art CNNs as backbones in our proposed models, such as ResNet (He et al., 2016) and DenseNet (Huang et al., 2017). Furthermore, we will enhance the model regarding uncertainty-awareness to address problems in transparency and robustness.

Recurrent Neural Networks

RNNs have demonstrated ground-breaking performance in many sequential modeling tasks (Lipton et al., 2015), including speech recognition, music generation, machine translation. For advanced CDS, the sequential EHR data is a critical type of information, consisting of recorded clinical events during each hospital visit. More specifically, the events at each visited time step can be formulated as a feature vector $\boldsymbol{x}_t$ to denote the patient status. This formulation has a high similarity with the tasks in machine translation, where clinical events correspond to words and hospital visits correspond to sentences. Since the number of clinical events varies from patient to patient, the length of the resulting sequence is also not fixed. To learn a fixed-size latent representation, the modeling advantage of RNNs is therefore of great interest. For example, suppose we want to model the respective treatment decision after a sequence of clinical events. In that case, we can use a many-to-one RNN structure and take the last hidden state $\boldsymbol{h}_{T_i}$ as the abstract covariates of the patients, where T_i denotes the sequence length of the i-th patient. More generally, there have been many successful applications using RNNs to model the sequential EHR data. Esteban et al. (2016) proposed to use an RNN with a many-to-many structure to predict the endpoints that occurred during kidney transplantation. Meanwhile, Choi et al. (2016a) proposed RNN-based Doctor AI to predict the diagnosis and medication based on encounter records, where the attention-based mechanism was developed in Choi et al. (2016b) to further improve the model interpretability. Afterward, Yang et al. (2017) proposed to use a many-to-one RNN to predict the therapy suggestions for patients who suffered from breast cancer. Based on these works, we will propose in Chapter 2 an RNN-based framework for learning problems in contextual bandit problems, which provides the treatment suggestions by integrating the outcome information. Furthermore, in Chapter 4, we will combine the

RNN-based models with sparse Gaussian Processes to equip the uncertainty-awareness in the time-to-event predictions.

1.2 Representation Learning for Analytics in Healthcare

The analytics landscape of advanced CDS (El Morr and Ali-Hassan, 2019) consists of four key aspects as shown in Table 1.1. All types of analytics can be instantiated and formulated as challenging machine learning problems. In the following, we will discuss the motivations, problem settings, related works, and contributions in respective chapters.

	Contributions in this thesis
Prescriptive Analytics	Learning Individualized Treatment Rules in Chapter 2.
Diagnostic Analytics	Diagnostic Medical Image Analysis in Chapter 3.
Predictive Analytics	Time-to-event Prediction in Chapter 4.
Descriptive Analytics	Missing data imputation in Chapter 5.

Table 1.1: Types of analytics and our respective contributions.

1.2.1 Propensity Score-Based Models for Prescriptive Analytics

In prescriptive analytics, we are interested in what we should do under some new situations. In probabilistic terms, it refers to $p(a|\boldsymbol{x})$, the conditional probability of the treatment decisions a given the covariates of a patient $\boldsymbol{x}$. In a clinical setting, we have two different mindsets to tackle the problems. On the one hand, we can assume that physicians always follow the clinical guidelines and give the best possible treatment decisions to the patients. In such cases, we can have CDS to mimic the physicians' decisions as well as possible, which is translated into a supervised learning problem. Under such a setting, we are essentially building a propensity score model, where the propensity score refers to the conditional probability of assigning a specific treatment given a vector of observed covariates (Rosenbaum and Rubin, 1983). Previous works such as Esteban et al. (2016); Yang et al. (2017) have shown that RNN-based solutions can deliver much stronger performance than traditional methods when building propensity score models. Recently, McIntosh et al. (2021) show that ML-based solutions are even more acceptable than physicians' original

prescriptions in retrospective simulations, which encourages the deployment of suggestions from AI-based CDS to offer second options. On the other hand, when no clear guidelines are available, or some decisions must be made under time pressure, treatment decisions from doctors can be inconsistent and prone to mistakes. Examples include decision-making situations in the intensive care unit (ICU) environment as in Komorowski et al. (2018), which corresponds to one of the use cases we study in this thesis. Instead of approximating the observed treatments, we are interested in studying the causal effects of different treatment decisions, i.e., which treatment may lead to a better outcome for a patient or what treatment strategy would result in an improved average outcome.

In causal inference, we are interested in differentiating the outcomes between different treatments, while the fundamental problem is that we only observe one treatment and one outcome for each patient. Formally, the problem can be formulated as studying the average treatment effects (ATE) using the potential outcomes framework (Rubin-Neyman Causal Model) (Rubin, 1974, 2005). Within the potential outcomes framework, the identifiabiilty of causal effects requires some untestable assumptions including Stable Unit Treatment Value Assumption (SUTVA), consistency, ignorability (no unmeasured confounders), and positivity (common support). Recent NN-based works include Balancing Neural Network (BNN) from Johansson et al. (2016) and Generative Adversarial Nets for inference of Individualized Treatment Effects (GANITE) from Yoon et al. (2018b), where Bica et al. (2021) offer a comprehensive overview.

Equivalently, the problem can be formulated as contextual bandits (CBs) (Langford and Zhang, 2007), while many other names exist, such as reinforcement learning with immediate reward (Abe et al., 2003), bandit problems with side observations (Wang et al., 2005), bandit problems with covariates (Rigollet and Zeevi, 2010). CBs concern decision-making processes in an environment where feedback is received only for a chosen action under a given context (Dudík et al., 2011). In our clinical setting, the context refers to the patient covariates, the action refers to the treatment decision, and the feedback refers to the outcome observed from the given treatment. Instead of studying the ATE in causal inference, the goal in CBs is to learn a new policy that leads to better expected outcomes. The policy here is the conditional probability of the treatment decisions given the covariates of a patient, which shares the same formulation as the one in supervised learning. However, it is not trained in a supervised fashion but includes the outcome information during the optimization. In precision medicine, these policies are also known as individualized treatment rules (ITRs) (Zhou et al., 2017).

CBs have demonstrated strong performance in many online applications, ranging from personalized news recommendation (Li et al., 2010), advertisement placement (Bottou et al., 2013) to Just-in-time adaptive interventions in mobile health (Nahum-Shani et al., 2018). Meanwhile, in most observational studies, we have datasets passively collected from clinical routines. We cannot perform any new policy directly on patients due to safety and ethical reasons. Therefore, we follow the setting of batch learning from logged bandit feedback (BLBF) (Swaminathan and Joachims, 2015) to tackle the problem in an offline fashion. Existing approaches to solve BLBF problems fall under two categories. The first approach is called Direct Method (DM), which transforms the problem into a supervised learning problem and consists of two steps. In the first step, a supervised learning problem is formulated to fit an outcome model, mapping the treatment decisions and patient covariates to outcome values. The treatment policy is derived from the outcome model in the second step by taking the treatment corresponding to the best-estimated outcome. However, since supervised learning is optimized to predict outcomes rather than differentiate the treatment decisions, the feature of treatment decisions may be neglected for the sake of higher accuracy. Therefore, this approach is found not to generalize well (Beygelzimer and Langford, 2009; Zhao et al., 2012). The situations could possibly be improved if we take advantage of the potential outcome framework, as in the causal inference setting. The other approach is known as Inverse Propensity Score (IPS), which casts the problem as a policy optimization problem using the propensity score with the interaction information like treatments and their respective outcomes. In this thesis, we develop our method using an IPS-based approach.

Following the IPS method, Swaminathan and Joachims (2015) formulate the BLBF as a risk minimization problem, which translates the ITRs learning into finding a new policy $\pi_{\boldsymbol{w}}$ that minimizes the risk

$$\begin{aligned} r(\pi_{\boldsymbol{w}}) &= \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\mathbb{E}_{a\sim\pi_{\boldsymbol{w}}(a|\boldsymbol{x})}\Big[\delta(\boldsymbol{x},a)\Big] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\mathbb{E}_{a\sim p(a|\boldsymbol{x})}\Big[\delta(\boldsymbol{x},a)\frac{\pi_{\boldsymbol{w}}(a|\boldsymbol{x})}{p(a|\boldsymbol{x})}\Big], \end{aligned} \tag{1.1}$$

where $\boldsymbol{w}$ denotes the parameters of the new policy. The loss $\delta(\boldsymbol{x}, a)$ is an indicator function, which is 1 for negative outcome and 0 for positive outcome. The derivation in the second line involves the importance sampling to remove the distribution mismatch between the historical policy (physicians' policy) from the propensity score $p(a|\boldsymbol{x})$ and the new policy $\pi_{\boldsymbol{w}}(a|\boldsymbol{x})$. From Equation 1.1, we can see that a policy having higher probability for treatments with positive outcomes and a lower probability for treatment with negative

outcomes will result in lower expected risk.

The Inverse Propensity Score (IPS) estimator applies Monte Carlo sampling to estimate the expected risk in Equation 1.1 as

$$\hat{r}_{\text{IPS}}(\pi_{\boldsymbol{w}}) = \frac{1}{n}\sum_{i=1}^{n} \delta_i \frac{\pi_{\boldsymbol{w}}(a_i|\boldsymbol{x}_i)}{p(a_i|\boldsymbol{x}_i)},$$

where $\delta_i, a_i, \boldsymbol{x}_i$ denotes the observed loss, the assigned treatment, and covariates of the i-th patient sample, respectively. However, the IPS estimator is not an ideal training objective for mainly two reasons. Firstly, the estimator suffers from large variances, especially when there is a large discrepancy between the new policy and the physicians' policy (Dudík et al., 2011). More importantly, the optimization is prone to the problem of propensity overfitting, which means the optimized policy tends to be dominated by the physicians' policy instead of focusing on treatments with low risks. A toy example from a multi-class classification task is illustrated in Figure 1.4. A good new policy should behave like case (a), which suggests treatments with lower losses and results in an overall risk of 0. Meanwhile, due to the propensity overfitting problem in the IPS estimator, the new policy can achieve the same minimum risk by simply avoiding the treatment assigned by physicians as shown in case (b). Moreover, if we translate the loss by, e.g., -1, the minimizer of the IPS estimator will be completely different, which tends to over-present the physician's policy.

The problem of propensity overfitting originates from the lack of equivariance of the IPS estimator, i.e., the minimizer of the IPS estimator is dependent on the translation of the loss. It can be reflected by the value of the treatment matching factor (TMF) defined as

$$s(\pi_{\boldsymbol{w}}) \coloneqq \frac{1}{n}\sum_{i=1}^{n} \frac{\pi_{\boldsymbol{w}}(a_i|\boldsymbol{x}_i)}{p(a_i|\boldsymbol{x}_i)}, \tag{1.2}$$

which in expectation equals to one. The unconstrained TMF in the IPS estimator leads to its lack of equivariance. To resolve the problems, Swaminathan and Joachims (2015) proposed the self-normalized IPS (SNIPS) estimator by introducing the TMF as a multiplicative control variate

$$\hat{r}_{\text{SNIPS}}(\pi_{\boldsymbol{w}}) = \frac{\frac{1}{n}\sum_{i=1}^{n} \delta_i \frac{\pi_{\boldsymbol{w}}(a_i|\boldsymbol{x}_i)}{p(a_i|\boldsymbol{x}_i)}}{s(\pi_{\boldsymbol{w}})}. \tag{1.3}$$

It was proven to be asymptotically unbiased and has the property of equivariance so that the respective optimization can focus on searching for treatment decisions with low risks.

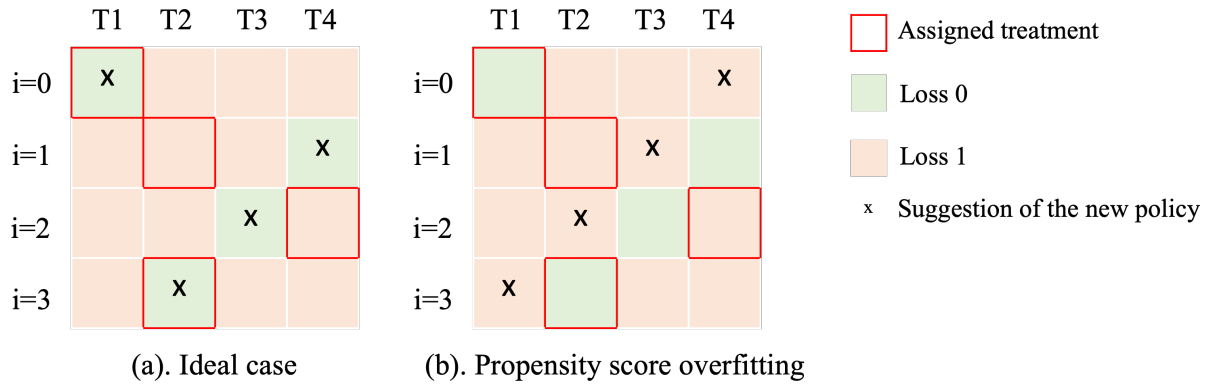


Figure 1.4: A toy example to illustrate the propensity overfitting problem. Here, we assume a deterministic assignment or suggestion of treatment decisions (probability being one). Rows correspond to different patient samples, and columns correspond to treatments decisions. For example, the first sample $i = 0$ has the lowest risk (loss 0) at the treatment T1, which is also assigned by physicians (red square). Both case (a) and (b) can achieve the theoretical minimum risk of 0, but case (b) suffers from the propensity overfitting problem.

Contributions

Standard estimators like IPS, SNIPS have been introduced with a focus on the issues found in IPS. Meanwhile, several problems remain to be solved. First, the SNIPS estimator in its current form in Equation 1.3 is not compatible with SGD training, which hinders its integration with NN-based models. In addition, all these estimators assume true propensity scores are part of the datasets, which are unavailable in most observational studies. Lastly, due to safety considerations, we cannot apply any newly learned policy to patients before going through rigorous evaluations. In Chapter 2, we will present a general framework to tackle these challenges for learning optimal Individualized Treatment Rules (ITRs). The framework consists of two parts, a predictive model and an ITR model. For the predictive model, we will take advantage of the state-of-the-art predictive modeling of treatment decisions (Yang et al., 2017), which is used for propensity score estimation. The ITR model will be based on the latest BLBF formulation, BanditNet (Joachims et al., 2018), so that the policy can be optimized using SGD. The policy will be learned by encouraging treatments with positive outcomes and discouraging treatments with negative ones. To validate the effectiveness of the proposed framework, we will conduct experiments both in simulation studies and real-world use cases. In simulation studies, we have ground-truth

values to conduct evaluation using metrics like accuracy. For the real-world datasets, we will do evaluation with various offline methods. The encouraging experimental results will justify the effectiveness of the proposed framework.

1.2.2 Uncertainty-Aware Models for Diagnostic and Predictive Analytics

Although deep learning models introduced in Section 1.1.3 define many state-of-the-art performances for diagnostic and predictive analytics in advanced CDS, most proposed models focus on improving the point estimate performance such as accuracy. Nevertheless, in safety-critical areas like healthcare, communicating the uncertainty of the predictions would not hurt but rather facilitate a trustworthy relationship between AI-based models and physicians, where a high uncertainty value indicates low confidence in the respective prediction. From a data-centric ML perspective, the uncertainty of the predictions can help the collection of new data samples for building better systems more efficiently. The problem setting is similar to regression tasks, whose goal is to map the input features to the continuous target variables. However, in our setting, we focus not only on the point estimation of the target variables, but also want to provide respective uncertainty estimation for each prediction.

In general, uncertainties from predictions can arise from two different reasons, where they are correspondingly defined as aleatoric uncertainty and epistemic uncertainty. The aleatoric uncertainty refers to the **data uncertainty**, which can not be reduced even when we collect more data. Examples include intrinsic random noises in sensor measurements, such as Gaussian noises in linear regression models. Meanwhile, epistemic uncertainty is the **model uncertainty**, which can be reduced by observing more data samples. For example, it can be modeled by assuming a probability distribution on the weights in the linear regression models, which is also known as Bayesian linear regression. A more comprehensive introduction can be found in Yarin (2016) and Kendall and Gal (2017). The modeling of uncertainty has been studied as a fundamental problem in the field of probabilistic machine learning (Ghahramani, 2015), where Gaussian Processes (GP) are one of the flexible non-parametric models considering both model uncertainty and data uncertainty. As a non-parametric model, GP has an infinite amount of parameters, which we have to handle probabilistically and integrate out properly. In the following, we will introduce GP, sparse GP and see how they can be integrated with deep learning for diagnostic and predictive

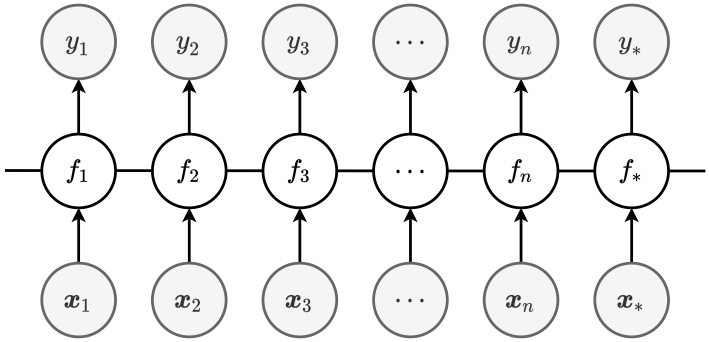


Figure 1.5: Graphical model of a Gaussian Process for regression. Nodes are variables where shaded ones are observed (a training dataset $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ with a test pair $(\boldsymbol{x}_*, y_*)$), and non-shaded ones are latent function values ($\{f_i\}_{i=1}^n$ for training and f_* for testing).

analytics.

As shown in Figure 1.5, a GP is a collection of random variables, any finite number of which have a joint zero-mean Gaussian distribution (Rasmussen and Williams, 2006). Formally, the (latent) function values $\boldsymbol{f}$ behave according to

$$p(\boldsymbol{f}|\boldsymbol{X}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{ff}}),$$

where we denote all function values as a vector $\boldsymbol{f} := [f(\boldsymbol{x_1}), \ldots, f(\boldsymbol{x}_n)] \in \mathbb{R}^n$, all input samples $\boldsymbol{x}_i \in \mathbb{R}^p$ as a design matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ (there is a transpose operation after the concatenation), a pre-defined covariance function $k(\cdot, \cdot)$ specifying the covariance between pairs of input samples, and the covariance matrix

$$\boldsymbol{K}_{\boldsymbol{ff}} := k(\boldsymbol{X}, \boldsymbol{X}) = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

In a regression problem, we have the likelihood model

$$p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{X}) = \mathcal{N}\left(\boldsymbol{y}|\boldsymbol{f}, \sigma_{\text{obs}}^2 \boldsymbol{I}\right),$$

where we denote all target variables in a vector as $\boldsymbol{y} := [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$ and σ_{obs}^2 as the variance of the observational noise in the training data.

With a GP prior over function values and a Gaussian likelihood, the marginal likelihood $p(\boldsymbol{y}|\boldsymbol{X})$ is analytically tractable by integrating the function values $\boldsymbol{f}$ out

$$p(\boldsymbol{y}|\boldsymbol{X}) = \int p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{X}) p(\boldsymbol{f}|\boldsymbol{X}) d\boldsymbol{f} = \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{ff}} + \sigma_{\text{obs}}^2 \boldsymbol{I}).$$

Though GP is a non-parametric model, the hyperparameters in the kernel function could largely influence the model performance, such as the scaling and length-scale parameter in a radial basis function (RBF) kernel. The hyperparameters $\boldsymbol{\theta}$ can be learned by maximizing the log marginal likelihood as

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log \mathcal{N}\left(\boldsymbol{y} \mid \mathbf{0}, \boldsymbol{K}_{\boldsymbol{ff}} + \sigma^2_{\text{obs}}\boldsymbol{I}\right).$$

With the optimized hyperparameters $\boldsymbol{\theta}^*$, GP provides a predictive distribution for a test sample $\boldsymbol{x}_*$ as

$$p(f_*|\boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta}^*) = \mathcal{N}\left(f_*|\boldsymbol{k}_{\boldsymbol{f}*}^\top \left(\boldsymbol{K}_{\boldsymbol{ff}} + \sigma^2_{\text{obs}}\boldsymbol{I}\right)^{-1}\boldsymbol{y}, k_{**} - \boldsymbol{k}_{\boldsymbol{f}*}^\top \left(\boldsymbol{K}_{\boldsymbol{ff}} + \sigma^2_{\text{obs}}\boldsymbol{I}\right)^{-1}\boldsymbol{k}_{\boldsymbol{f}*}\right),$$

where $k_{**} = k(\boldsymbol{x}_*, \boldsymbol{x}_*) \in \mathbb{R}, \boldsymbol{k}_{\boldsymbol{f}*} = k(\boldsymbol{X}, \boldsymbol{x}_*) \in \mathbb{R}^n$. In probabilistic terms, the predictive distribution above can be derived by conditioning a joint GP prior on the noisy observations, where we have the joint GP prior as

$$p(\boldsymbol{f}, f_*|\boldsymbol{X}, \boldsymbol{x}_*) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f} \\ f_* \end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{ff}} & \boldsymbol{k}_{*\boldsymbol{f}} \\ \boldsymbol{k}_{\boldsymbol{f}*} & k_{**} \end{bmatrix}\right). \tag{1.4}$$

However, the $\mathcal{O}(n^2)$ storage complexity and $\mathcal{O}(n^3)$ computational complexity from the inverse operation of the covariance matrix $\boldsymbol{K}_{\boldsymbol{ff}}$ hinder the application of GPs to large-scale datasets. Many approximation methods have been proposed to solve the scalability issue in exact GPs. Generally speaking, the approximation can be categorized into the prior approximation and the posterior approximation with inducing points. Following Quinonero-Candela and Rasmussen (2005), we define inducing points as inducing inputs and inducing variables, $\{\boldsymbol{z}_i, u_i\}_{i=1}^m =: \boldsymbol{Z}, \boldsymbol{u}$, corresponding to the inputs and (latent) function values in the original dataset, $\{\boldsymbol{x}_i, f_i\}_{i=1}^n =: \boldsymbol{X}, \boldsymbol{f}$, where the number of inducing points, m, is pre-defined and $m \ll n$.

Under the prior approximation, Williams and Seeger (2000) propose the Nyström approximation, one influential solution based on using a subset of the original dataset to constitute the covariance matrix. The covariance matrix is assumed to have a low rank and approximated by the inducing points as

$$\boldsymbol{K}_{\boldsymbol{ff}} \approx \boldsymbol{Q}_{\boldsymbol{ff}} = \boldsymbol{K}_{\boldsymbol{fu}}\boldsymbol{K}_{\boldsymbol{uu}}^{-1}\boldsymbol{K}_{\boldsymbol{uf}},$$

where $\boldsymbol{K}_{\boldsymbol{fu}} = k(\boldsymbol{X}, \boldsymbol{Z}) \in \mathbb{R}^{n\times m}, \boldsymbol{K}_{\boldsymbol{uu}} = k(\boldsymbol{Z}, \boldsymbol{Z}) \in \mathbb{R}^{m\times m}$. More generally, we denote $\boldsymbol{Q}_{\boldsymbol{ab}} := \boldsymbol{K}_{\boldsymbol{au}}\boldsymbol{K}_{\boldsymbol{uu}}^{-1}\boldsymbol{K}_{\boldsymbol{ub}}$ as the approximated covariance matrix using the inducing points. In

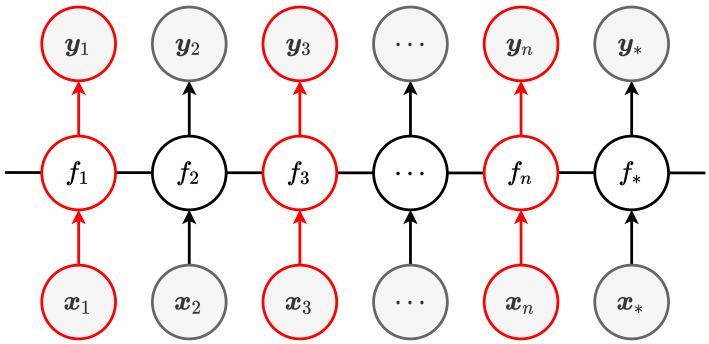


Figure 1.6: Graphical model of the Nyström approximation. Nodes with red circles are (randomly) selected samples to approximate the original covariance matrix.

comparison to Equation 1.4, the joint prior is approximated by

$$q_{\text{Nyström}}\left(\boldsymbol{f}, f_*\right) = \mathcal{N}\left(\left[\begin{array}{c} \boldsymbol{f} \\ f_* \end{array}\right] \middle| \mathbf{0}, \left[\begin{array}{cc} \boldsymbol{Q}_{\boldsymbol{f}\boldsymbol{f}} & \boldsymbol{k}_{*\boldsymbol{f}} \\ \boldsymbol{k}_{\boldsymbol{f}*} & k_{**} \end{array}\right]\right), \tag{1.5}$$

where from now on we drop the explicit conditioning on (inducing) inputs to simplify the notation.

As shown in Figure 1.6, the inducing points in the Nyström approximation refer to a subset of the dataset, which can be selected, e.g., randomly or through clustering algorithms like K-means. It equals an exact GP when the whole dataset is treated as the inducing points. The Woodbury formula further speeds up the inversion operation related to the covariance matrix, and the final computational cost is reduced to $\mathcal{O}(nm^2)$. Nevertheless, the Nyström approximation can not be viewed as a well-formed probabilistic model due to the inconsistency of $\boldsymbol{Q}_{\boldsymbol{f}\boldsymbol{f}}$ and $\boldsymbol{k}_{\boldsymbol{f}*}$ in Equation 1.5. The problem is later fixed by Deterministic Training Conditional (DTC) approximation from Seeger et al. (2003).

Following the idea of approximating the generative model with the exact inference, Snelson and Ghahramani (2005) propose Sparse Pseudo-input Gaussian Processes (SPGP), which is later known as the Fully Independent Training Conditional (FITC) Approximation. Unlike the Nyström approximation, the inducing points are trainable parameters learned by gradient-based optimization to better represent the dataset. Furthermore, as shown in Figure 1.7, the function values $\boldsymbol{f}$ are assumed to be conditionally independent given the inducing variables $\boldsymbol{u}$, which reflects the naming of FITC.

The joint prior of FITC takes the form

$$q_{\text{FITC}}\left(\boldsymbol{f}, f_*\right) = \mathcal{N}\left(\left[\begin{array}{c} \boldsymbol{f} \\ f_* \end{array}\right] \middle| \mathbf{0}, \left[\begin{array}{cc} \boldsymbol{Q}_{\boldsymbol{f}\boldsymbol{f}} + \operatorname{diag}(\boldsymbol{K}_{\boldsymbol{f}\boldsymbol{f}} - \boldsymbol{Q}_{\boldsymbol{f}\boldsymbol{f}}) & \boldsymbol{q}_{*\boldsymbol{f}} \\ \boldsymbol{q}_{\boldsymbol{f}*} & k_{**} \end{array}\right]\right), \tag{1.6}$$

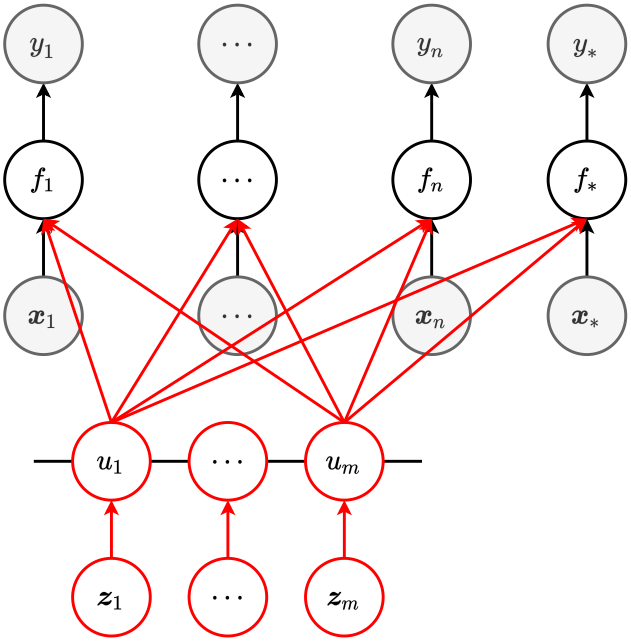


Figure 1.7: Graphical model of the Fully Independent Training Conditional Approximation. Nodes with red circles are trainable inducing points.

where $\boldsymbol{q}_{\boldsymbol{f}*}$ denotes $\boldsymbol{K}_{\boldsymbol{fu}}\boldsymbol{K}_{\boldsymbol{uu}}^{-1}\boldsymbol{k}_{\boldsymbol{u}*}$ by following the definition of $\boldsymbol{Q}_{\boldsymbol{ab}}$. From the expression, we can see FITC replaces the prior covariance with the respective approximation at all locations except the values on the diagonal. This derivation of training objective and predictive distribution follows a similar procedure in exact GPs by replacing the covariance matrix with the cheaper approximation in Equation 1.6. With some other matrix inversion tricks, the computational complexity remains to be $\mathcal{O}(nm^2)$.

However, with a large amount of trainable inducing points in the generative model, the training of FITC is prone to overfitting problems. To better approximate exact GPs, Titsias (2009) propose the Variational Free Energy (VFE) to approximate the GP posterior using inducing points. Due to the consistency of GPs, the true data and the inducing points are jointly Gaussian

$$p(\boldsymbol{f}, \boldsymbol{u}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{u} \end{bmatrix} \middle| \boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{ff}} & \boldsymbol{K}_{\boldsymbol{fu}} \\ \boldsymbol{K}_{\boldsymbol{uf}} & \boldsymbol{K}_{\boldsymbol{uu}} \end{bmatrix}\right).$$

The posterior in an exact GP, $p(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{y}) = p(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{y})p(\boldsymbol{u}|\boldsymbol{y})$, is approximated by the variational distribution, $q(\boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u})$. The approximation encourages $\boldsymbol{u}$ to be a sufficient statistic, since $\boldsymbol{u}$ would represent the data well if we can have $p(\boldsymbol{f}|\boldsymbol{u}) \approx p(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{y})$. The learning of the inducing points involves minimizing the Kullback–Leibler (KL) di-

vergence, $\mathrm{KL}[q(\boldsymbol{f},\boldsymbol{u})\|p(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{y})]$, or equivalently[2] maximizing the Evidence Lower Bound (ELBO) of the log marginal likelihood

$$\begin{aligned}
\log p(\boldsymbol{y}|\boldsymbol{\theta},\boldsymbol{Z},q(\boldsymbol{u})) &= \log \iint p(\boldsymbol{y},\boldsymbol{f},\boldsymbol{u})d\boldsymbol{f}d\boldsymbol{u} \\
&= \log \iint \frac{q(\boldsymbol{f},\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})} p(\boldsymbol{y},\boldsymbol{f},\boldsymbol{u})d\boldsymbol{f}d\boldsymbol{u} \\
&\overset{\text{(i)}}{\geq} \iint q(\boldsymbol{f},\boldsymbol{u}) \log \frac{p(\boldsymbol{y},\boldsymbol{f},\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})} d\boldsymbol{f}d\boldsymbol{u} \\
&\overset{\text{(ii)}}{=} \iint p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u}) \log \frac{p(\boldsymbol{y} \mid \boldsymbol{f})\cancel{p(\boldsymbol{f}|\boldsymbol{u})}p(\boldsymbol{u})}{\cancel{p(\boldsymbol{f}|\boldsymbol{u})}q(\boldsymbol{u})} d\boldsymbol{f}d\boldsymbol{u} \\
&= \int q(\boldsymbol{u}) \left[\int p(\boldsymbol{f} \mid \boldsymbol{u}) \log p(\boldsymbol{y} \mid \boldsymbol{f}) d\boldsymbol{f} + \log \frac{p(\boldsymbol{u})}{q(\boldsymbol{u})} \right] d\boldsymbol{u} := \mathcal{J}(\boldsymbol{\theta},\boldsymbol{Z},q(\boldsymbol{u}))
\end{aligned}$$

where (i) follows the Jensen's inequality and (ii) involves the substitution of the variational distribution. To tighten the bound, we can maximize over the variational distribution $q(\boldsymbol{u})$, where the optimal choice $q^*(\boldsymbol{u})$ can be analytically solved and has a Gaussian form. Inserting $q^*(\boldsymbol{u})$ back in the ELBO results in a final bound $\mathcal{J}_{q^*(\boldsymbol{u})}(\boldsymbol{\theta},\boldsymbol{Z})$, whose parameters are to be learned with gradient based methods like in FITC. The predictive distribution for a new sample can be computed from the variational distribution as

$$\begin{aligned}
p\left(f_* \mid \boldsymbol{y}\right) = q\left(f_*\right) &= \iint p\left(f_* \mid \boldsymbol{f},\boldsymbol{u}\right) q(\boldsymbol{f},\boldsymbol{u}) d\boldsymbol{f}d\boldsymbol{u} \\
&= \iint p\left(f_* \mid \boldsymbol{f},\boldsymbol{u}\right) p(\boldsymbol{f} \mid \boldsymbol{u}) q(\boldsymbol{u}) d\boldsymbol{f}d\boldsymbol{u} \\
&= \int p\left(f_* \mid \boldsymbol{u}\right) q(\boldsymbol{u}) d\boldsymbol{u},
\end{aligned}$$

where the last line involves the GP consistency as $\int p\left(f_* \mid \boldsymbol{f},\boldsymbol{u}\right) p(\boldsymbol{f} \mid \boldsymbol{u}) d\boldsymbol{f} = p(f_* \mid \boldsymbol{u})$.

Unlike FITC, increasing the number of inducing points would not cause any overfitting problem but only improve the approximation to an exact GP. Meanwhile, like other variational methods, VFE is even prone to underfit the dataset. In practice, VFE offers better mean estimates, while FITC returns better variance predictions (Bauer et al., 2016). From a DL's perspective, the predictive variance and training objective are the most important parts of the respective GP models, corresponding to the forward and backward propagation. Therefore, we summarize them in Table 1.2.

[2] $$\iint q(\boldsymbol{f},\boldsymbol{u}) \log \frac{p(\boldsymbol{y},\boldsymbol{f},\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})} d\boldsymbol{f}d\boldsymbol{u} = \iint q(\boldsymbol{f},\boldsymbol{u}) \log \frac{p(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{y})p(\boldsymbol{y})}{q(\boldsymbol{f},\boldsymbol{u})} d\boldsymbol{f}d\boldsymbol{u} = p(\boldsymbol{y}) - \mathrm{KL}\big[q(\boldsymbol{f},\boldsymbol{u})||p(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{y})\big]$$

Table 1.2: A summary of the exact GP, sparse GPs and SVGPs with their respective training objective, test time predictive distributions and computational complexity. We follow the notations in Quinonero-Candela and Rasmussen (2005) for exact and sparse GPs, and Jankowiak et al. (2020) for SVGPs.

Method	Training Objective	Test Time Predictive Distribution of f_*	Complexity
GP	$\log \mathcal{N}\left(\boldsymbol{y} \mid \mathbf{0}, \boldsymbol{K_{ff}} + \sigma^2_{\text{obs}}\boldsymbol{I}\right)$	$\mathcal{N}\left(\boldsymbol{k}_{\boldsymbol{f}*}^\top \left(\boldsymbol{K_{ff}} + \sigma^2_{\text{obs}}\boldsymbol{I}\right)^{-1}\boldsymbol{y}, \boldsymbol{k}_{**} - \boldsymbol{k}_{\boldsymbol{f}*}^\top \left(\boldsymbol{K_{ff}} + \sigma^2_{\text{obs}}\boldsymbol{I}\right)^{-1}\boldsymbol{k}_{\boldsymbol{f}*}\right)$	$\mathcal{O}(n^3)$
Nyström	$\log \mathcal{N}\left(\boldsymbol{y} \mid \mathbf{0}, \boldsymbol{Q_{ff}} + \sigma^2_{\text{obs}}\boldsymbol{I}\right), \boldsymbol{Q_{ff}} = \boldsymbol{K_{fu}}\boldsymbol{K_{uu}^{-1}}\boldsymbol{K_{uf}}$	$\mathcal{N}\left(\boldsymbol{k}_{\boldsymbol{f}*}^\top \left(\boldsymbol{Q_{ff}} + \sigma^2_{\text{obs}}\boldsymbol{I}\right)^{-1}\boldsymbol{y}, \boldsymbol{k}_{**} - \boldsymbol{k}_{\boldsymbol{f}*}^\top \left(\boldsymbol{Q_{ff}} + \sigma^2_{\text{obs}}\boldsymbol{I}\right)^{-1}\boldsymbol{k}_{\boldsymbol{f}*}\right)$	$\mathcal{O}(nm^2)$
FITC	$\log \mathcal{N}\left(\boldsymbol{y} \mid \mathbf{0}, \boldsymbol{Q_{ff}} + \boldsymbol{\Lambda} + \sigma^2_{\text{obs}}\boldsymbol{I}\right), \boldsymbol{\Lambda} = \text{diag}\left(\boldsymbol{K_{ff}} - \boldsymbol{Q_{ff}}\right)$	$\mathcal{N}\left(\boldsymbol{q}_{*\boldsymbol{f}}(\boldsymbol{Q_{ff}} + \boldsymbol{\Lambda})^{-1}\boldsymbol{y}, \boldsymbol{k}_{**} - \boldsymbol{q}_{*\boldsymbol{f}}(\boldsymbol{Q_{ff}} + \boldsymbol{\Lambda})^{-1}\boldsymbol{q}_{\boldsymbol{f}*}\right)$	$\mathcal{O}(nm^2)$
VFE	$\log \mathcal{N}\left(\boldsymbol{y} \mid \mathbf{0}, \boldsymbol{Q_{ff}} + \sigma^2_{\text{obs}}\boldsymbol{I}\right) - \frac{1}{2\sigma^2_{\text{obs}}}\text{trace}\left(\boldsymbol{K_{ff}} - \boldsymbol{Q_{ff}}\right)$	$\mathcal{N}\left(\boldsymbol{q}_{*\boldsymbol{f}}(\boldsymbol{Q_{ff}} + \sigma^2_{\text{obs}}\boldsymbol{I})^{-1}\boldsymbol{y}, \boldsymbol{k}_{**} - \boldsymbol{q}_{*\boldsymbol{f}}(\boldsymbol{Q_{ff}} + \sigma^2_{\text{obs}}\boldsymbol{I})^{-1}\boldsymbol{q}_{\boldsymbol{f}*}\right)$	$\mathcal{O}(nm^2)$
SVGP	$\sum_{i=1}^{n}\left\{\log \mathcal{N}\left(y_i \mid \mu_f\left(\boldsymbol{x}_i\right), \sigma^2_{\text{obs}}\right) - \frac{\sigma_f\left(\boldsymbol{x}_i\right)^2}{2\sigma^2_{\text{obs}}}\right\} - \text{KL}(q(\boldsymbol{u})\|p(\boldsymbol{u}))^*$	$\mathcal{N}\left(\mu_{\boldsymbol{f}}\left(\boldsymbol{x}_*\right), \sigma^2_{\boldsymbol{f}}\left(\boldsymbol{x}_*\right)\right)^\dagger$	$\mathcal{O}(m^3)$
PPGP	$\sum_{i=1}^{n}\log \mathcal{N}\left(y_i \mid \mu_f\left(\boldsymbol{x}_i\right), \sigma^2_{\text{obs}} + \sigma^2_{\boldsymbol{f}}\left(\boldsymbol{x}_i\right)\right) - \text{KL}(q(\boldsymbol{u})\|p(\boldsymbol{u}))^*$	$\mathcal{N}\left(\mu_{\boldsymbol{f}}\left(\boldsymbol{x}_*\right), \sigma^2_{\boldsymbol{f}}\left(\boldsymbol{x}_*\right)\right)^\dagger$	$\mathcal{O}(m^3)$

$^*\mu_{\boldsymbol{f}}\left(\boldsymbol{x}_i\right) = \boldsymbol{k}_{\boldsymbol{ui}}^T\boldsymbol{K_{uu}^{-1}}\boldsymbol{m}$, $\sigma^2_{\boldsymbol{f}}\left(\boldsymbol{x}_i\right) = k_{ii} - \boldsymbol{k}_{\boldsymbol{ui}}^T\boldsymbol{K_{uu}^{-1}}\boldsymbol{k}_{\boldsymbol{ui}} + \boldsymbol{k}_{\boldsymbol{ui}}^T\boldsymbol{K_{uu}^{-1}}\boldsymbol{S}\boldsymbol{K_{uu}^{-1}}\boldsymbol{k}_{\boldsymbol{ui}}$.
$^\dagger\mu_{\boldsymbol{f}}\left(\boldsymbol{x}_*\right) = \boldsymbol{k}_{\boldsymbol{u}*}^T\boldsymbol{K_{uu}^{-1}}\boldsymbol{m}$, $\sigma^2_{\boldsymbol{f}}\left(\boldsymbol{x}_*\right) = k_{**} - \boldsymbol{k}_{\boldsymbol{u}*}^T\boldsymbol{K_{uu}^{-1}}\boldsymbol{k}_{\boldsymbol{u}*} + \boldsymbol{k}_{\boldsymbol{u}*}^T\boldsymbol{K_{uu}^{-1}}\boldsymbol{S}\boldsymbol{K_{uu}^{-1}}\boldsymbol{k}_{\boldsymbol{u}*}$.

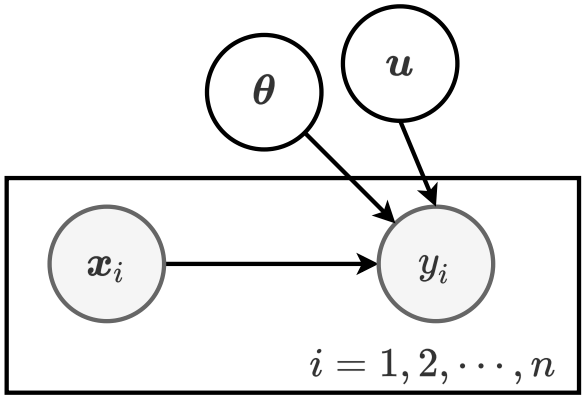


Figure 1.8: Graphical model of the Scalable Variational Gaussian Process.

As shown in Table 1.2, the training and inference of sparse GPs still involve loading the complete training dataset in the memory for the computation of matrices like $\boldsymbol{Q_{ff}}$. It remains to be unsatisfactory for large datasets due to the quadratic storage complexity. For example, large image datasets can easily exceed the memory-size limit in standard computers. Moreover, the training of neural networks is usually done in mini-batches. As a solution, Hensman et al. (2013) propose Scalable Variational Gaussian Process (SVGP), which reduces the space and computational complexity to $\mathcal{O}(m^2)$ and $\mathcal{O}(m^3)$ respectively. As discussed above, the optimal variational distribution in VFE, $q^*(\boldsymbol{u})$, has the form of a Gaussian and the computation of its mean and variance has a dependency on the whole dataset. SVGP drops this dependency by assuming $q(\boldsymbol{u})$ to be a Gaussian distribution, $\mathcal{N}(\boldsymbol{u}|\boldsymbol{m}, \boldsymbol{S})$, with trainable mean and variance. Bad values of $\boldsymbol{m}$ and $\boldsymbol{S}$ would only result in poor variational approximation, but approximation assumptions in SVGP remain unchanged compared to VFE. Consequently, the bound is shown to have a data likelihood term, which can be factorized across samples. As shown in Figure 1.8, SVGP turns to be a fully parametric model of the inducing points $\boldsymbol{u}$ and hyperparameters $\boldsymbol{\theta}$, which enables the training using methods like stochastic gradient descent (SGD).

As a variant of VFE, SVGP inherits nice properties like good mean estimates. Meanwhile, bad properties like over-estimated observational noise σ^2_{obs} in VFE also occur in SVGP. Jankowiak et al. (2020) pointed out that most predictive variance in SVGP is explained by input-independent observational noise and proposed the Parametric Predictive Gaussian Process (PPGP) to improve the predictive variance.

Contributions

To this end, we have introduced exact GP, sparse GPs (FITC, VFE), and SVGPs (SVGP, PPGP). The question of how to combine them with NNs has to be further discussed. In addition, the evaluation of the predictive uncertainty remains to be open. In Chapter 3, we will present a novel deep kernel learning model for regression on medical images, which allows the estimation of uncertainty through a pipeline of state-of-the-art CNNs and SVGPs. The proposed model will be enhanced by introducing various pre-training technologies in representation learning, which we found crucial for the initialization of inducing points in SVGPs. We will show a novel plot, Quantile Performance (QP) plot, to visualize the intuition of uncertainty-awareness: The more confident a model is of its predictions, the more accurate these predictions should be. Through experiments on both univariate and multivariate regression tasks, we will validate the good performance and uncertainty-awareness of our proposed model. In Chapter 4, we will further develop the deep kernel model in the context of time-to-event prediction. Following the idea of a trainable feature extractor with an uncertainty-aware predictive model, we will feed sequential EHRs as input data to the RNN-based feature extractors. The latent representations will be consumed by SVGPs as abstract covariates of the patients. The positive skewness and right-censoring of the time-to-event target variable will be handled by generalizing the linear accelerated failure time models to non-linear SVGP-based models. Experiments conducted on two real-world datasets will show the improved performance of our proposed model compared to common architectures without the loss of scalability. The uncertainty estimates in the proposed model will presumably help establish a trustworthy relationship with physicians since it can express higher confidence in more accurate predictions and vice versa.

1.2.3 GAN-Based Models for Descriptive Analytics

Descriptive analytics can provide evidence for subsequent decision-making processes. Due to the vast amount of workloads in hospitals, missing data is a common yet critical problem in existing EHRs. However, many machine learning models require complete patient vectors as inputs. The ultimate solution would be to enhance the EHR data quality itself through improved data collection processes. Meanwhile, from an algorithmic perspective, the problem can be tackled by imputation methods to estimate missing values based on observed ones. The problem setting is as follows, given a dataset having incomplete patient features, we would like to impute the missing features based on observed ones, so that the

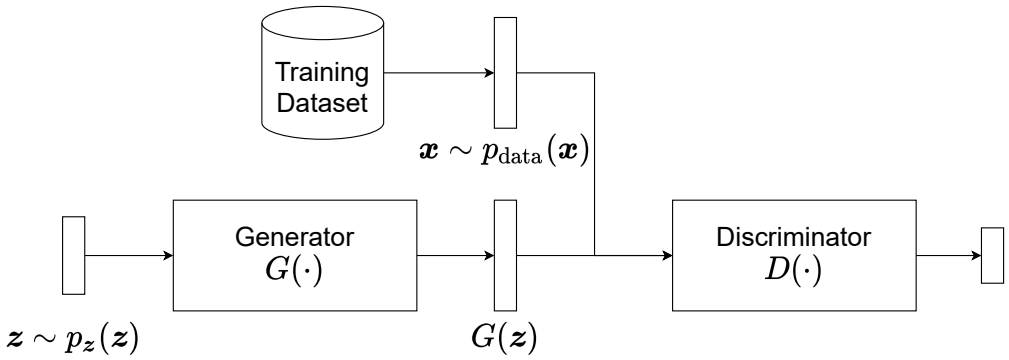


Figure 1.9: Illustration of the architecture of Generative Adversarial Networks.

downstream predictive models can benefit from the completeness of the patient features. In this thesis, we focus on a novel imputation approach based on Generative Adversarial Networks (GANs) (Goodfellow et al., 2014).

GANs are first proposed to capture the data distribution of high-dimensional data like images. In the framework of GANs, there is a generative model (generator) and a discriminative model (discriminator). The discriminator estimates the probability that the input comes from the training data rather than the generator, while the generator tries to fool the discriminator by outputting more realistic samples. In other words, the two models are trained in an adversarial process. The whole learning process can be trained using backpropagation if we involve neural networks for the generator and discriminator.

Figure 1.9 illustrates the architecture of GANs. Formally, we denote the input of the generator as $\boldsymbol{z}$, which is sampled from a pre-defined distribution $p_{\boldsymbol{z}}(\boldsymbol{z})$, e.g., a standard normal distribution. The output of the generator is denoted as $G(\boldsymbol{z})$. Together with samples from the training dataset $\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})$, $G(\boldsymbol{z})$ is fed as input to the discriminator. The output of the discriminator $D(\cdot)$ is the probability that the sample comes from the training data. To correctly classify the samples from real to generated, it is trained by maximizing the log-likelihood of Bernoulli distributions as

$$\mathcal{J}_D(\boldsymbol{\Theta}_D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\Big[\log D(\boldsymbol{x})\Big] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\Big[\log\Big(1 - D(G(\boldsymbol{z}))\Big)\Big], \tag{1.7}$$

where we denote all trainable parameters in the discriminator as $\boldsymbol{\Theta}_D$.

Meanwhile, the generator is trained to fool the discriminator by minimizing the same objective as

$$\mathcal{J}_G(\boldsymbol{\Theta}_G) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\Big[\log\Big(1 - D(G(\boldsymbol{z}))\Big)\Big], \tag{1.8}$$

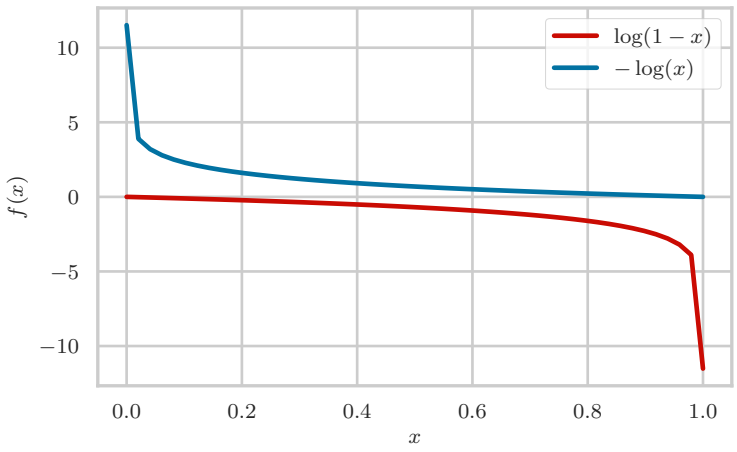


Figure 1.10: Comparison of two different loss functions for the generator. As the value of 0 is undefined for the log function, 10^{-5} is used as an approximation.

where we denote all trainable parameters in the generator as $\boldsymbol{\Theta}_G$. Note that the first term in Equation 1.7 is dropped as it doesn't depend on the parameters from the generator.

Training models by objectives in Equation 1.7 and Equation 1.8 constitutes a zero-sum or minimax game concerning the same value function, which is proven to recover minimizing the Jensen-Shannon divergence between the data and the model distribution. However, the training objective in Equation 1.8 cannot provide sufficiently large gradient for the generator to learn well. Since the discriminator is trained in a supervised learning manner, it can be trained efficiently to reject generated samples confidently, especially at the beginning of the training. As shown in Figure 1.10, with a strong discriminator predicting 0s for the generated samples, the values of $\log(1 - x)$ approaches 0, i.e., the gradient of the generator vanishes. As a fix, the non-saturating loss is proposed for the generator as

$$\mathcal{J}_G(\boldsymbol{\Theta}_G) = -\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\Big[\log D(G(\boldsymbol{z}))\Big], \tag{1.9}$$

which can be interpreted as maximizing the probability that the discriminator makes wrong predictions. As in Figure 1.10, the non-saturating loss offers much larger gradients in regions near $x = 0$, which corresponds to a strong discriminator. After the initial GANs paper, numerous variants of the generator loss function have been proposed. Nevertheless, it was found that the original non-saturating loss could achieve similar results to most of them, given enough hyperparameter optimization and random restarts (Lucic et al., 2018).

Contributions

There are still several obstacles to overcome when we apply GANs to data imputation problems. More specifically, the outputs from the generator are complete feature vectors, which is hard to be used to impute only part of the feature vectors. In addition, one requirement on the generator is that it has to be differentiable so that the gradients from the discriminator can be backpropagated to the generator. Such a requirement makes it hard to generate categorical features due to their discrete nature. In Chapter 5, we will present our solution based on a GANs-based variant for missing data imputation, Generative Adversarial Imputation Nets (GAINs) (Yoon et al., 2018a). We will show that, even if we relax the discrete constraints of categorical variables to enable the backpropagation in GAIN, the training of the generator could still fail. The reason is that a perfect discriminator can be easily achieved by exploiting the apparent difference between the generated data being continuous and the true data being binary. To solve this problem, we will propose a novel way to re-code the categorical features to stabilize the adversarial training based on the GAINs, where we name our proposed models as categorical GAINs. By encoding binary values to continuous values between 0 and 1 with the categorical information being retained, we avoid the "cheating" of the discriminator. In addition, we will propose multiple modifications in GAINs architecture to serve the categorical features and the proposed re-coding method better. The largely improved predictive accuracy will validate the effectiveness of the proposed imputation method.

Chapter 2

Propensity Score-Based Models in Individualized Treatment Rules

Learning Individualized Treatment Rules with Estimated Translated Inverse Propensity Score

Zhiliang Wu[1, 2], Yinchong Yang[1], Yunpu Ma[1, 2], Yushan Liu[1, 2], Rui Zhao[1, 2], Michael Moor[3], Volker Tresp[1, 2]
[1]Siemens AG, [2]LMU Munich, Munich, Germany, {firstname.lastname}@siemens.com
[3]ETH Zurich, Basel, Switzerland, michael.moor@bsse.ethz.ch

***Abstract*—Randomized controlled trials typically analyze the effectiveness of treatments with the goal of making treatment recommendations for patient subgroups. With the advance of electronic health records, a great variety of data has been collected in clinical practice, enabling the evaluation of treatments and treatment policies based on observational data. In this paper, we focus on learning individualized treatment rules (ITRs) to derive a treatment policy that is expected to generate a better outcome for an individual patient. In our framework, we cast ITRs learning as a contextual bandit problem and minimize the expected risk of the treatment policy. We conduct experiments with the proposed framework both in a simulation study and based on a real-world dataset. In the latter case, we apply our proposed method to learn the optimal ITRs for the administration of intravenous (IV) fluids and vasopressors (VP). Based on various offline evaluation methods, we could show that the policy derived in our framework demonstrates better performance compared to both the physicians and other baselines, including a simple treatment prediction approach. As a long-term goal, our derived policy might eventually lead to better clinical guidelines for the administration of IV and VP.**

***Index Terms*—individualized treatment rules, contextual bandit problem, off-policy learning**

I. Introduction

Since the introduction of electronic health records (EHRs), machine learning has increasingly been used to analyze observational clinical data with the goal of individualizing patient care [1]. Compared to traditional rule-based strategies, where all patients with a specific disease in a particular patient group receive similar treatments, the goal of modern personalized medicine is to offer better care to individual patients, taking into account their heterogeneous characteristics. Personalized medicine might be especially important for situations where high-dimensional longitudinal data needs to be analyzed under time pressure, as in an emergency room (ER) or an intensive care unit (ICU). Here, treatment decisions might have to be made without the best medical expert for the case being readily available.

In personalized medicine, individualized treatment rules (ITRs) assign a treatment from a range of possible treatments to an individual patient based on his or her clinical characteristics [2]. Ideally, all patients would have positive outcomes after receiving the treatments suggested by the optimal ITRs. In practice, one is interested in the ITRs' best mean performance. However, the evaluation of ITRs remains challenging, as it is unethical or even dangerous to apply newly learned rules directly to patients. Offline evaluation is the most widely used approach for such tasks. When learning the optimal ITRs, it is implicitly assumed that individualization can lead to better outcomes compared to current guidelines. In clinical practice, physicians might already perform some form of individualization by taking into account patient attributes that are not considered in the guidelines. In *predictive modeling*, one attempts to directly copy the physicians' decision processes by using machine learning [3], which serves as one of our baseline methods.

Recently, many researchers have built powerful machine learning models to predict the physicians' treatment decisions with neural networks [4]–[6]. In particular, recurrent neural networks (and their advanced variants) are the de facto choice when dealing with sequential EHRs. In this paper, we show that recurrent neural networks are also suitable for learning the optimal ITRs within the proposed framework shown in Fig. 1.

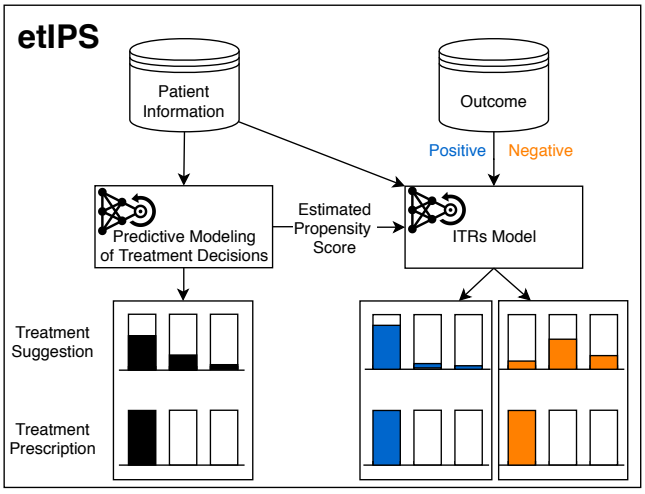


Fig. 1. **etIPS** for learning the optimal ITRs: Both the predictive model (left) and the ITRs model (right) generate treatment suggestions based on available patient information. The predictive model is trained to mimic the physicians' decisions as well as possible. If the predictive model is trained to output probabilistic scores, it essentially estimates propensity scores. The ITRs are trained by encouraging treatments with a positive outcome as well as discouraging treatments with a negative outcome.

From a machine learning perspective, the task of learning optimal ITRs can be formulated as treatment policy optimization based on the observed treatments and their received outcomes for individual patients. Such formulation is closely related to the contextual bandit problem, which concerns

decision making in an environment where feedback is received only for a chosen action under a specific context. The challenge lies in the fact that only the feedback of an assigned action is observed, while the feedback of other actions remains unknown. Most work on the contextual bandit problem in machine learning concerns online services like content recommendations, where the context is a user's profile of interests in different topics, the action is the recommended item, and the feedback is the click action for the recommended item [7]. Online systems also record the model's assigned probability for each recommended item, which plays an essential role in learning a better policy. In the setup of clinical trials, the context can be viewed as the health level and treatment history of patients, the action refers to the treatment decision, and the feedback is the outcome observed after that specific treatment. The probability of assigning a particular treatment to the patient based on his or her covariates is known as the *propensity score* [8]. In randomized clinical trials, the propensity score is usually predefined for the experiments (e.g., 50% for binary randomized clinical trials). However, in observational studies, the propensity score can only be estimated since it is implicit in the observed medical decisions. Many previous studies focus on the learning of ITRs with the predefined propensity scores [2], [9], [10].

Our contribution in this manuscript is threefold:

1) Inspired by previous works in predictive modeling of treatment decisions and contextual bandit problems, we present a general framework, etIPS, for learning ITRs based on sequential EHRs from observational studies by estimating the underlying true propensity score.
2) With experiments on two simulated sequential classification tasks, we empirically verify that the estimated propensity score can replace the true propensity score for learning a better policy in contextual bandit problems.
3) We apply the proposed framework to the MIMIC-III dataset [11] to learn the optimal ITRs for the administration of intravenous (IV) fluids and vasopressors (VP). In various offline evaluations, the ITRs derived from our proposed method show better performance when compared to the physicians' decisions and other baselines.

II. Related work

Predictive modeling with sequential EHRs: Recurrent neural networks (RNNs) have achieved great success on tasks such as machine translation in natural language processing [12]–[16]. In machine translation, sentences are composed of variable number of words, just as EHRs consist of medical events of variable length. Esteban et al. have applied the sequence-to-sequence structure [17] to predict clinical events of patients suffering from kidney failure [5]. In their work, the static EHRs are integrated into the network to achieve better performance. Meanwhile, Choi et al. propose *Doctor AI* to predict diagnosis and medication prescriptions simultaneously [3]. Furthermore, Choi et al. augment the network with attention mechanisms to improve both the accuracy and model interpretability [18]. More recently, Yang et al. have proposed to apply the many-to-one structure to predict the therapy decision for breast cancer [6], which only outputs one prediction for a sequence of events. However, predictive modeling is solely trained to mimic treatment decisions without taking into account the outcome information.

Learning Individualized Treatment Rules (ITRs): The learning of ITRs has attracted much attention in medical research. To get the best average outcome, Qian et al. propose a two-step method [9]. First, an outcome prediction model is fitted with the patient information and treatments. Second, ITRs are derived by selecting the treatment that promises to lead to the best outcome according to the trained model. This approach relies heavily on the correctness of the outcome prediction model. In comparison, Zhao et al. propose the framework of outcome weighted learning (OWL) to construct a model that directly optimizes the outcome without learning an explicit outcome model [10]. In OWL, the learning of ITRs is formulated as a weighted classification problem and is solved by support vector machines. More recently, Zhou et al. have proposed the residual weighted learning (RWL) to improve the robustness of the ITRs learned by OWL [2]. A separate regression model is fitted to estimate the baseline to compute the residual from the outcome. The discussed frameworks mainly focus on linear models and linear classifiers.

Learning the administration of IV and VP: Komorowski et al. propose a reinforcement learning agent to learn the optimal strategies for sepsis management [19]. A k-means algorithm is used to infer the states of the patients, 25 actions are defined by discretizing the dosage of IV and VP, and the mortality is used to define the long-term reward. The optimal policy is derived by solving a Markov decision process with policy iteration. However, mortality is a sparse and noisy long-term reward for both learning and evaluation. In this paper, we have a similar problem setting, but take advantage of an immediate reward to learn and evaluate the optimal ITRs.

Batch learning from bandit feedback (BLBF): Bandit learning is commonly applied in online recommendation systems, where algorithms are evolving by trial and error with real-time feedback from users. In medical applications, it is more common to train algorithms offline, mostly for safety considerations. Batch learning from bandit feedback is one of the offline versions of the contextual bandit problem, where the algorithm is trained with a batch of bandit feedback without online interactions [20]. Under the BLBF setting, the two-step method of deriving an optimal decision by maximizing the best estimated outcome proposed by Qian et al. is called the *Direct Method* (DM), whereas the approaches to optimize weighted outcomes directly proposed by Zhao et al. are known as *Inverse Propensity Score* (IPS) methods [7]. Swaminathan et al. cast BLBF as a counterfactual risk minimization problem. They propose the Policy Optimizer for Exponential Models (POEM) to improve the robustness of IPS methods [21]. Besides, Swaminathan et al. propose to use the self-normalized estimator for counterfactual learning (Norm-POEM) to alleviate the propensity score overfitting problem [22]. Both POEM and Norm-POEM are only applicable to

linear models. More recently, Joachim et al. have proposed to reformulate the self-normalized estimator to train neural networks with bandit feedback [23]. However, all the proposed methods assume that the true propensity score is known.

III. Cohort

In this section, we describe how we define the cohort and process the data to be used in our proposed framework.

A. Cohort Selection

The Medical Information Mart for Intensive Care database (MIMIC-III) is a freely accessible database, which contains data including $53,423$ Intensive Care Unit (ICU) admissions of adult patients between 2001 and 2012 [11]. In this paper, we consider a cohort of patients from MIMIC-III v1.4, who fulfill the Sepsis-3 criteria [24]. We follow the scripts[1] provided by Komorowski et al. [19] to recreate the cohort. In short, the inclusion criteria select those adult patients who are associated with a Sequential Organ Failure Assessment (SOFA) score of 2 or more during the time of interest. The SOFA score ranges from 0 to 24, and a higher value indicates a more severe status of the patient. Further, patients with extreme unusual records or death during the data collection period are excluded from the cohort, as their records would have led to spurious policies. In total, $20,944$ admissions are included in our dataset.

B. Data Description and Processing

Static and sequential information: There are two classes of variables that are relevant for modeling the treatment decisions: 1) static information, e.g., age and gender; 2) sequential information, e.g., time-varying heart rate and respiratory rate. Similar to Komorowski et al. [19], we extract a set of 47 variables, including information about demographics, vital signs, and lab values. Three of those variables are about static information and 44 are about sequential information. More details about the variables can be found in Appendix A. The time of interest is defined as 24 hours before the onset of the sepsis and 48 hours after it. To represent the sequential status of the patient, we aggregate the data by averaging over four-hour windows. As a result, at each time-step, each admission is represented by a multidimensional vector.

Treatments and outcome: We choose to learn the optimal ITRs for the administration of IV and VP, considering the suboptimality of their administration reported in the clinical literature [25]. More specifically, we follow the scripts from Komorowski et al. and define 25 treatment decisions for each four-hour time window, where each decision is an IV-VP pair for discretized dosages. The original dosage is first converted to zero (i.e., zero dosage) and non-zero classes, and the non-zero classes are further divided into quartiles. More statistics of the discretized treatment decisions can be found in Appendix B.

In the bandit problem, each action immediately receives feedback information. Therefore, we compute a clinically guided outcome, denoted by Δ-SOFA (differences between subsequent SOFA scores), as our feedback information to guide the learning of the ITRs. As concluded by Vincent et al. [26], the Δ-SOFA offers an objective evaluation of treatment responses and could be used to reflect patients' responses to therapeutic strategies. Furthermore, if a patient has an unchanged SOFA score in a low range or a decreased SOFA score in subsequent time windows, he/she is associated with a lower mortality rate. Similar applications of the Δ-SOFA have been reported by Raghu et al. [27]. In BLBF, the problem is cast as a risk minimization problem. Thus, we define the loss as 0 (positive outcome) if Δ-SOFA is unchanged in a low SOFA range (0-5) or has decreased. Otherwise, we set the loss as 1 (negative outcome).

Training and test sample generation: To model the treatment decision, we extract samples from the patients' medical history in an expanding window fashion, whenever a treatment is observed. For predictive modeling, the treatment decision is viewed as the target variable for training. All sequential information before the treatment is used as covariates for prediction. The sequential information at the time-step of the treatment is not used for learning, as some variables may not be observed at the time of decision in the ICU. For ITRs learning, the outcome for the treatment decision is required. Therefore, we extract the observed Δ-SOFA in the next time-step and compute the corresponding loss as described in the previous paragraph. As shown in Fig. 2, each sample consists of the sequential information, treatment decision, and the corresponding loss information. In addition, the static information is also extracted but not shown in the figure for the sake of simplicity.

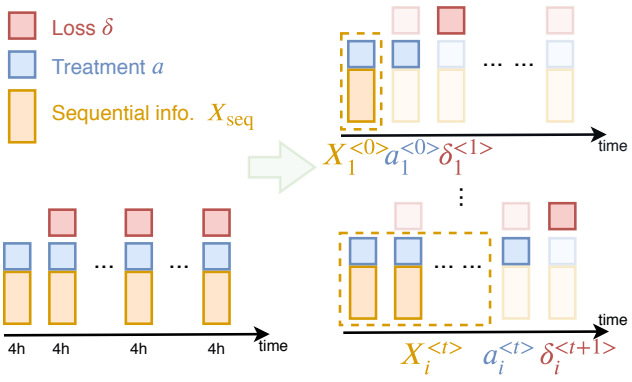


Fig. 2. Illustration of the training and test sample generation from the medical history of each admission: The left-hand side presents the raw data after aggregating for every four-hour time window; the right-hand side shows the generated training and test samples. A sample will be extracted if the following two conditions are fulfilled: 1) A treatment decision is observed at a certain time-step. 2) The feedback information is observed in the following time-step of the treatment. To highlight the relative order between the sequential information, treatment, and the corresponding loss, we add the superscript $< t >$ to indicate the time-step index of the treatment during the admission.

From $20,944$ admissions we could extract in total $224,333$ samples (i.e., 10.7 samples per admission on average). The number of time-steps observed before the treatment varies from 1 to 18 and is on average 7.2. When generating the

[1] https://github.com/matthieukomorowski/AI_Clinician

training samples and test samples, the split is based on the admission level rather than the sample level so that we can achieve a more objective evaluation. With the split admissions, the samples are divided for training and testing accordingly.

IV. Method

Our proposed framework consists of two consecutive parts: a predictive model for the propensity score estimation and an ITRs model trained with an objective function based on the estimated propensity score. After following the preprocessing steps in Fig. 2, we denote our data as $\{(X_{\text{seq}})_i, (\boldsymbol{x}_{\text{sta}})_i, a_i, \delta_i\}_{i=1}^m$, where $X_{\text{seq}} \in \mathbb{R}^{T\times 44}$ represents the (multivariate) random variable for the sequential information with T observed time-steps and $\boldsymbol{x}_{\text{sta}} \in \mathbb{R}^3$ stands for the static information. We denote the treatment decision as $a \in \{1, 2, \ldots, 25\} =: \mathcal{A}$ and the loss of the observed treatment as $\delta \in \{0, 1\}$. Scalars are denoted by lowercase letters such as a, δ; (column) vectors are denoted by bold lowercase letters such as $\boldsymbol{x}_{\text{sta}}$; matrices are denoted by uppercase letters such as X_{seq}; sets are denoted by calligraphic letters such as $\mathcal{A}$.

A. Propensity Score-Based Objective Function for Learning ITRs

Following the formulation in BLBF, the goal of learning the optimal ITRs is to find a policy $\pi_{\boldsymbol{w}}$ that minimizes the risk

$$\begin{aligned} r(\pi_{\boldsymbol{w}}) &= \mathbb{E}_{X\sim\mathbb{P}(X)}\mathbb{E}_{a\sim\pi_{\boldsymbol{w}}(a|X)}\big[\delta(X,a)\big] \\ &= \mathbb{E}_{X\sim\mathbb{P}(X)}\mathbb{E}_{a\sim\mathbb{P}(a|X)}\Big[\delta(X,a)\cdot\frac{\pi_{\boldsymbol{w}}(a|X)}{\mathbb{P}(a|X)}\Big] \end{aligned} \tag{1}$$

where $\boldsymbol{w}$ denotes the parameters of the policy. The loss $\delta(X, a)$ is an indicator function, which is 1 for negative outcome and 0 for positive outcome. The propensity score is reflected in the conditional probability $\mathbb{P}(a|X)$ for different treatments $a \in \mathcal{A}$. For conciseness, we use X to denote the random variable for the complete medical history, including the sequential information X_{seq} and the static information $\boldsymbol{x}_{\text{sta}}$, though it is a slight abuse of notation..

Equation (1) is derived by applying importance sampling to remove the distribution mismatch between the physicians' policy and the new policy $\pi_{\boldsymbol{w}}$. Intuitively, the new policy $\pi_{\boldsymbol{w}}$ will have a lower expected risk $r(\pi_{\boldsymbol{w}})$ when it has a higher probability for treatments with positive outcomes and a lower probability for treatments with negative outcomes.

The Inverse Propensity Score (IPS) estimator

$$\hat{r}_{\text{IPS}}(\pi_{\boldsymbol{w}}) = \frac{1}{m}\sum_{i=1}^{m}\delta_i\frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)} \tag{2}$$

applies Monte Carlo sampling to estimate the expected risk in (1) by taking the observed data points as samples. The IPS estimator will be unbiased if $\mathbb{P}(a_i|X_i)$ describes the physicians' policy. Therefore, it is appealing to use the risk defined by the IPS estimator (IPS risk) as the objective function to learn the optimal ITRs.

However, there are mainly two reasons why it is not possible to optimize the policy using the IPS risk directly. First, it has been shown that the IPS estimator suffers from large variance if there is a large discrepancy between the new policy and the physicians' policy [7], which would be more severe for high-capacity models like neural networks, as it is in our case. Second, directly minimizing an IPS estimator that contains the propensity score is prone to propensity score overfitting [22]. More specifically, the new policy is dominated by the physicians' policy rather than the treatment with low loss. In our setting, the minimal IPS risk in (2) is 0. The new policy will simply put zero probability on all the treatment decisions observed from the physicians. In other words, the new policy achieves minimal IPS risk by recommending any treatment that differs from the physicians' decision. In Sec. V, this phenomenon will also be empirically verified.

Propensity score overfitting originates from *the lack of equivariance* of the IPS estimator (see Appendix D), i.e., the minimizer of the IPS risk is dependent on the translation of the loss. Furthermore, the lack of equivariance is due to the unconstrained treatment matching factor (TMF), defined as

$$s(\pi_{\boldsymbol{w}}) \coloneqq \frac{1}{m}\sum_{i=1}^{m}\frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)} \tag{3}$$

which equals to 1 in expectation (see Appendix C), but will be far from 1 if the propensity score overfitting problem occurs.

As a solution, the self-normalized IPS estimator (SNIPS)

$$\hat{r}_{\text{SNIPS}}(\pi_{\boldsymbol{w}}) = \frac{\frac{1}{m}\sum_{i=1}^{m}\delta_i\frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}}{s(\pi_{\boldsymbol{w}})} \tag{4}$$

is proposed to replace the IPS estimator for the learning of a new policy [22]. It is proven to be asymptotically unbiased [28] and has the property of equivariance (see Appendix E), which enables the new policy to focus on learning the treatment with low loss.

Neural networks are typically trained by mini-batch stochastic gradient descent. Unfortunately, the optimization problem, including the SNIPS estimator, cannot be solved directly by a mini-batch stochastic gradient descent-based method, since all samples are required to compute the denominator. A mini-batch of samples could be used to estimate it, but the result is proven to be biased [23]. Joachim et al. propose the *BanditNet* by reformulating the SNIPS estimator with an additional constraint [23]. In short, optimizing the SNIPS estimator is equivalent to optimizing a λ-translated IPS estimator

$$\hat{r}^{\lambda}_{\text{IPS}}(\pi_{\boldsymbol{w}}) = \frac{1}{m}\sum_{i=1}^{m}(\delta_i-\lambda)\frac{\pi_{\boldsymbol{w}_j}(a_i|X_i)}{\mathbb{P}(a_i|X_i)} \tag{5}$$

where the Lagrange multiplier λ is called the *translation* (more details in Appendix F). The optimal translation λ is found through grid search. As mentioned earlier, a translation of the loss results in a difference among the minimizers of the IPS risk: On the one hand, the new policy tends to avoid the treatments in the physicians' policy if losses are defined as non-negative values; on the other hand, it prefers to over-present the physicians' policy if losses are defined as non-positive values. Taking advantage of the lack of equivariance of the IPS estimator, the reformulation searches the optimal

translation to balance these two tendencies so that the policy can focus on learning the treatment with low loss.

B. Predictive Modeling of Treatment Decisions

In observational studies, the propensity score is not known but can be estimated from the collected data. More specifically, the propensity score can be modeled by any supervised machine learning models that provide probability estimates for the various treatment decisions. We propose to apply state-of-the-art predictive models to produce an estimated propensity score $\hat{\mathbb{P}}(a = a_i|X)$, which is necessary for the optimization problem in (5).

Recurrent neural networks (RNNs) provide an extension of feedforward neural networks to handle sequential inputs. Formally, given an input sequence $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$, an RNN calculates the hidden states $\boldsymbol{h}_t$ at time-step t iteratively by joining the current input at t and the previous hidden state at $t-1$ as

$$\boldsymbol{h}_t = g(W\boldsymbol{x}_t + U\boldsymbol{h}_{t-1}) \tag{6}$$

where $g(\cdot)$ is a non-linear activation function and W and U are parametric weight matrices. Since each hidden state is again dependent on its predecessor, the state at t is theoretically capable of storing all relevant information of the entire history. Downstream models for classification or regression tasks could be implemented to consume the hidden state $\boldsymbol{h}_t$ as their input. However, the classical RNN architecture as in (6) often suffers from the vanishing gradient problem [29], [30] and therefore could fail to capture the long-term dependencies from the previous inputs. More advanced variants of RNNs, such as gated recurrent unit (GRU) [31] or long short-term memory (LSTM) [30], have been proposed to solve the problem with gating mechanisms and have achieved great successes in modeling sequential data with long-term dependencies, such as texts or sensory data [32].

In the case of predictive modeling of treatment decisions, the multidimensional vector $\boldsymbol{x}_t$ at different time-steps constitutes the sequential input data X_{seq}. GRU/LSTM is used to encode X_{seq} into the hidden states $\boldsymbol{h}_t$. Since we are mainly interested in modeling treatment decisions, a many-to-one structure is used [6], i.e., only the representation of the last hidden state $\boldsymbol{h}_T$ is utilized as the input for the treatment prediction, where T is the number of observed time-steps before the treatment. Formally, we have

$$\begin{aligned} \text{GRU/LSTM} : \mathbb{R}^{T\times 44} &\to \mathbb{R}^h \\ X_{\text{seq}} &\mapsto \boldsymbol{h}_T \end{aligned}$$

where h is the dimension of the hidden state and will be tuned as a hyperparameter in the experiments. The static information is concatenated with the hidden state encoded by GRU/LSTM so that the static information is included for the modeling of the treatment decisions [5]. Formally, we have

$$\boldsymbol{z} = (\boldsymbol{h}_T, \boldsymbol{x}_{\text{sta}}).$$

The resulting vector $\boldsymbol{z} \in \mathbb{R}^{h+3}$ represents the patient's complete medical history in a latent vector space and facilitates different subsequent tasks. In our case, a softmax classifier is built on top of it for the treatment prediction, as illustrated in Fig. 3. In our framework, we interpret the probability distribution produced by this model as an estimate of the true propensity score.

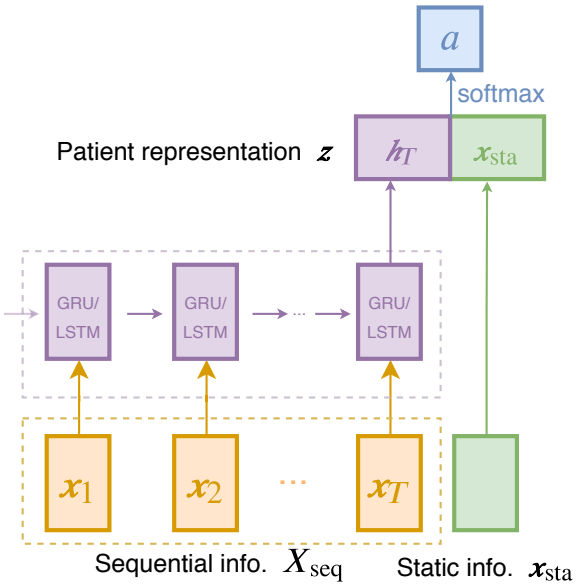


Fig. 3. Illustration of the predictive modeling of treatment decisions with static and sequential information: GRU/LSTM encodes the sequential information into hidden states. The last hidden state is concatenated with the static information, resulting in a vector to represent the patient's complete medical history. On top of it, a softmax classifier is built to predict the treatment decisions of physicians.

C. Estimated Translated Inverse Propensity Score

In this section, we elaborate the entire etIPS framework in Algorithm 1 by inversely joining the modules that have been introduced in Sec. IV-A and IV-B.

In line 1, we train the predictive model as in Sec. IV-B to estimate the physicians' policy. In line 2, we derive the estimated propensity scores on all patient cases from the predictive model in line 1. From line 3 to 6, we train our ITRs model as follows: For the j-th iteration, we select a particular translation $\lambda_j \in (0, 1)$ with grid search. The translation range is defined as $(0, 1)$ because the translation of 0 makes all losses non-negative and the translation of 1 makes all losses non-positive in our setting, which are the two extreme cases for the propensity score overfitting problem. We randomly initialize the trainable parameters $\boldsymbol{w}_j$ in the ITRs model, which has the same network structure as the predictive model in Fig. 3 but is optimized with an objective function based on the estimated propensity score. Depending on the translation λ_j, we minimize the objective functions with respect to the trainable parameters $\boldsymbol{w}_j$ (line 4). For each λ_j, both the minimizer $\boldsymbol{w}_j^*$ and its corresponding treatment matching factor s_j (line 5) are saved. In line 7, the final minimization step outputs the pair $(s^*, \boldsymbol{w}^*)$ that generates the minimum value for the SNIPS risk in (4).

The differences between the minimization goal in line 4 and the IPS risk in (2) are the estimated propensity score $\hat{\mathbb{P}}(a = a_i|X)$ and the translation λ_j. Therefore, we name the algorithm estimated translated Inverse Propensity Score (etIPS). Intuitively, the proposed framework enables the new policy to be trained through encouraging the network to learn from the physicians' treatment decisions with a positive outcome as well as from unsuccessful cases (treatments with a negative outcome).

Algorithm 1: etIPS

Input: A dataset of the form $\{X_i, a_i, \delta_i\}_{i=1}^m$.
Output: The policy of the optimal ITRs $\pi_{\boldsymbol{w}^*}(a|X)$.

1 Learn the physicians' policy $\hat{\mathbb{P}}(a|X)$ with $\{X_i, a_i\}_{i=1}^m$ using the network structure in Fig. 3.
2 Compute the estimated propensity score $\hat{p}_i \coloneqq \hat{\mathbb{P}}(a = a_i|X_i)$ for all i.
3 **for** $\lambda_j \in (0, 1)$ **do**
4 $\quad \boldsymbol{w}_j^* \leftarrow \arg\min_{\boldsymbol{w}_j} \left\{ \frac{1}{m} \sum_{i=1}^m (\delta_i - \lambda_j) \frac{\pi_{\boldsymbol{w}_j}(a_i|X_i)}{\hat{p}_i} \right\}$
5 $\quad s_j \leftarrow \frac{1}{m} \sum_{i=1}^m \frac{\pi_{\boldsymbol{w}_j^*}(a_i|X_i)}{\hat{p}_i}$
6 **end**
7 $s^*, \boldsymbol{w}^* \leftarrow \arg\min_{s_j, \boldsymbol{w}_j^*} \left\{ \frac{1}{s_j} \frac{1}{m} \sum_{i=1}^m \delta_i \frac{\pi_{\boldsymbol{w}_j}(a_i|X_i)}{\hat{p}_i} \right\}$
8 **return** $\pi_{\boldsymbol{w}^*}(a|X)$

V. Experiments

In this section, we provide details of the experiments conducted on three tasks, of which two are tailored to the BLBF setting from the MNIST dataset, which is common for the evaluation of many learning algorithms. As the ground truth labels in the MNIST dataset are available, the performance is evaluated with the metric accuracy. It serves as a simulation study [2], [21], [23]. In contrast, the MIMIC-III dataset only contains the feedback information of assigned treatments, and the offline evaluation is therefore employed, which can estimate the risk of a new policy from data observed from physicians.

A. Implementation details

The neural network-related models are built with the *tensorflow* package [33]. Hyperparameters are tuned with the *hyperopt* package [34]. Five-fold cross-validation is implemented to report the variance of the performance.[2]

B. Simulation studies

In this section, we simulate two controllable modeling tasks that resemble the true data situation. We aim to verify the following hypotheses empirically: a) Without ground truth labels, the propensity score-based objective function is applicable to sequential classification tasks. b) The estimated propensity score could be used to replace the true propensity score in the propensity score-based objective function.

[2]Related scripts see https://github.com/ZhiliangWu/etips.

1) Dataset generation: We define two sequential classification tasks based on the MNIST dataset. The first task, *zeros counting MNIST*, is to predict the number of zeros given a sequence of randomly sampled digit images. During sampling, we limit the maximum number of 0's to be 2 so that the prediction takes the form of a classification task with 3 classes. The second task, *row-by-row MNIST*, is to predict the label of the digit $(0 - 9)$ of the image. Each row of the image is presented sequentially to the neural network, and the classification is performed after reading all rows. Like other supervised learning tasks, the resulting dataset is in the form of $\{X_i, a_i^*\}_{i=1}^m$, where a_i^* is the ground truth label. From that, *the supervised to bandit conversion method* [35] is employed to generate BLBF datasets of the form $\{X_i, a_i, \delta_i, p_i\}_{i=1}^m$. If we view the tasks in a BLBF perspective, the context is a sequence of images X, the action a is the label prediction of the given sequence, the loss δ reflects the correctness of the prediction, and p is the probability of the label prediction. A *logging policy*, which is similar to the physicians' policy in the clinical setting, is required to generate the label prediction for different contexts. Also, we need to set a suboptimal accuracy for it, just like we assume there is still improvement space for physicians. Similar to the conversion procedure [21], we train a neural network to output $\mathbb{P}(a|\cdot)$ based on 5% of the supervised dataset $\{X_i, a_i^*\}_{i=1}^m$ and select the one with an accuracy around 66% as the logging policy. The label prediction a_i is sampled from the output distribution of the logging policy. Meanwhile, the propensity score p_i is also recorded for the sampled action. Finally, the loss δ_i is computed based on the ground truth label a_i^*, i.e., the loss is 0 if the label prediction is the ground truth label and 1 otherwise. More details of these generated datasets can be found in Appendix G.

2) Baselines: For all approaches except the direct method, a many-to-one structure with GRU/LSTM is used to deal with the sequential inputs. The neural network structure in Fig. 3 is not used because there is only sequential image information for the defined tasks. For the direct method, loss prediction is defined as the task for the network, and the action of label prediction is integrated in a way similar to the static information as in Fig. 3.

a. Direct method (DM): This method splits the task into two steps: It first learns the mapping $\mathbb{E}[\delta|X, a]$ to the expected loss given the context and action. The label prediction is then made by selecting the action with the lowest predicted loss $\arg\min_a \mathbb{E}[\delta|X, a]$.
b. Random policy (RP): A dummy policy to perform a label prediction uniformly at random, which serves as a weak baseline.
c. Inverse Propensity Score (IPS): The network is trained to minimize the IPS risk as defined in (2).
d. Translated Inverse Propensity Score (tIPS): The network is trained by minimizing the λ-translated IPS risk as defined in (5).
e. Estimated Inverse Propensity Score (eIPS): The network is trained by minimizing the IPS risk as defined in (2) with the estimated propensity score.

3) Results: Tab. I shows the prediction performance of different approaches. Both tIPS and etIPS achieve more than 90% accuracy, where etIPS yields the best results. RP has an accuracy of around $\frac{1}{\#\text{actions}}$, which is better than DM and IPS/eIPS.

TABLE I
ACCURACY OF DIFFERENT APPROACHES ON SEQUENTIAL CLASSIFICATION TASKS

	propensity score	zeros counting MNIST	row-by-row MNIST
DM	-*	0.343 ± 0.0001	0.098 ± 0.0022
RP	-*	0.363 ± 0.0001	0.103 ± 0.0001
IPS	true	0.301 ± 0.012	0.020 ± 0.0061
tIPS	true	0.899 ± 0.0229	0.931 ± 0.0852
eIPS	estimated	0.319 ± 0.0075	0.016 ± 0.0098
etIPS	estimated	$\mathbf{0.923 \pm 0.0122}$	$\mathbf{0.953 \pm 0.0390}$

*The propensity score is not involved in the algorithm.

4) Discussion: Trained on partial feedback information, an optimal policy should also be able to perform the label prediction with the lowest risk (i.e., the highest accuracy), which works in the same way as an optimal classifier. Furthermore, accuracy serves as a good metric here to evaluate a new policy because the datasets have a balanced distribution for different output classes. The accuracy trained with cross-entropy loss and full label information is around 95% for both tasks. From the performance of tIPS/etIPS, we see that the (estimated) propensity score-based objective function can deliver satisfying performance on the sequential prediction tasks when the ground truth label is not available. In addition, the performance of etIPS is a little better than tIPS. As proven [36], the reason is that the estimated propensity score has the potential to reduce the variance during the learning procedure. Furthermore, the performance of IPS/eIPS is worse than the weak baseline RP. Its poor performance is due to the propensity score overfitting problem, which can be diagnosed by computing the treatment matching factor in (3). For example, in the row-by-row MNIST task, $s(\pi_{\text{IPS}}) = 0.0061$ while $s(\pi_{\text{tIPS}}) = 0.926$. Last but not least, the performance of DM is as poor as RP. In practice, the performance for modeling $\mathbb{E}[\delta|X, a]$ is good with an accuracy of more than 85% (0.843 ± 0.0072 and 0.888 ± 0.0027 respectively). However, for the loss prediction, the network is trained with only one action under a certain context. The knowledge of the losses of different actions under the same context is missing during training. As a result, the trained network would predict similar loss values for different actions under the same context, which accounts for the poor performance on the prediction task.

C. Experiments on the MIMIC-III dataset

1) Evaluation metrics: For the MIMIC-III dataset, the goal is to learn the optimal ITRs for the administration of IV and VP. It is worth mentioning that offline evaluation remains a challenge, and the new policy requires further investigation with domain experts like physicians [37]. A new policy is hereby evaluated with three different evaluation methods:

a. Average Treatment Effects under the new policy (ATENP): This method evaluates the new policy in a deterministic way, i.e., it only considers the treatment suggestion with the highest probability. According to the treatment suggestions of the new policy, the samples in the test set are divided into two groups: those who follow the new policy (group one) and those who do not (group two) [2], [19]. The difference between the average risk in these two groups shows the average treatment effects under the new policy. If a new policy is better than the physicians' policy, the difference should be below zero.
b. Inverse Propensity Score Estimator (IPS): This method estimates the risk of the new policy as in (2). As we discussed earlier, it may suffer from propensity score overfitting problem and thus be strongly biased.
c. Doubly Robust Estimator (DR): The *doubly robust* technique consists of an outcome prediction model and a propensity score model [7] as

$$\hat{r}_{\text{DR}}(\pi_{\boldsymbol{w}}) = \frac{1}{m}\sum_{i=1}^{m}\Bigg[\sum_{a\in\mathcal{A}}\pi_{\boldsymbol{w}}(a|X_i)\hat{\delta}(X_i,a) + \frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\hat{\mathbb{P}}(a_i|X_i)}\big(\delta_i - \hat{\delta}(X_i,a_i)\big)\Bigg]$$

where $\hat{\delta}(X, a)$ is the loss prediction model and $\hat{\mathbb{P}}(a|X)$ is the propensity score model. It protects the mismodeling of either model by combining them to get the best of both.

As the problem is formulated as a risk minimization problem, a lower value of ATENP/IPS/DR is preferred. In Appendix H, the investigation of the correlation between accuracy and the risk estimated by these methods is provided to shed some light on the performance of different evaluation approaches. In short, ATENP shows a consistent correlation with the accuracy and is therefore trustworthy when there is a large sample size in group one. In comparison, the IPS estimator will be strongly biased when the propensity score overfitting problem occurs. In such cases, the DR estimator is more reliable by taking advantage of an outcome prediction model for correction.

2) Baselines: The true propensity score is not available in observational studies, which prevents the application of IPS and tIPS. Instead, we implement DM, RP, and eIPS for evaluation purposes. The network structure for eIPS follows the one in Fig. 3, and DM is defined as a loss prediction task with the treatment as an additional input feature. In addition, the predictive modeling of treatment decisions (cf. Sec. IV-B) and the most frequent policy are also included as baselines. The most frequent policy always suggests the most frequent treatment in the training dataset. In our case, it is the zero dosage of both IV and VP. Besides the evaluation methods, the treatment matching factor (TMF, cf. (3)) is computed based on the estimated propensity score to diagnose the propensity score overfitting problem.

3) Results: Tab. II shows the performance of the policies trained by different approaches. Our proposed approach turns out to have the lowest value in ATENP and DR with a

TABLE II
EVALUATION WITH DIFFERENT RISK ESTIMATORS

	ATENP	IPS	DR	TMF
Predictive Modeling	-0.019 ± 0.0021	0.523 ± 0.0229	0.523 ± 0.0021	1.034 ± 0.0391
Direct Method*	0.032 ± 0.0001	-	-	-
Most Frequent†	-0.023 ± 0.0001	-	-	-
Random Policy	-0.023 ± 0.0001	0.125 ± 0.0001	0.478 ± 0.0026	0.243 ± 0.0001
Estimated Inverse Propensity Score	-0.025 ± 0.1009	$\mathbf{0.009 \pm 0.0019}$	0.504 ± 0.0071	0.018 ± 0.0029
Estimated Translated Inverse Propensity Score	$\mathbf{-0.143 \pm 0.0099}$	0.169 ± 0.0160	$\mathbf{0.471 \pm 0.0060}$	0.438 ± 0.0279

*There is no probability of the treatment suggestion given by $\arg\min_a \mathbb{E}[\delta|X, a]$. The values for IPS/DR/TMF can therefore not be computed.
†A deterministic policy to suggest the most frequent treatment. There is no probability information involved.

high value of TMF (only lower than predictive modeling). In addition, the eIPS have the lowest value in the IPS evaluation with the lowest TMF.

4) Discussion: From a methodological perspective, the baseline approaches DM and eIPS can be viewed as the deep learning variants of the two-step method proposed by Qian et al. [9] and outcome weighted learning (OWL) [10], respectively. Similarly, our proposed method can be understood as a deep learning variant of residual weight learning (RWL) proposed by Zhou et al. [2]. The difference is that instead of learning a baseline by a separate regression model, our method is more efficient by trying different translations λ_j to find the optimal baseline. Furthermore, a predicted baseline in RWL inevitably introduces additional noise in the loss, which can potentially deteriorate the learning.

For ATENP, a value below zero means that the new policy is better than the physicians' policy. In the predictive modeling setting, the policy tries to mimic the physicians' policy as well as possible. The ATENP of it being around zero is therefore expected as it doesn't consider the outcome information. In comparison, the ATENP of etIPS shows a strong negative value of -0.143. It indicates that the observed treatments, which are the same as suggested by the new policy, have a much lower risk than those that are not. Also, the risk in group one of etIPS is estimated by 1929.8 ± 111.33 samples, which is relatively large, compared to 21.8 ± 9.62 for eIPS and 347 ± 0.01 for DM.

The lowest IPS risk for the eIPS is strongly biased, which can be indicated by both the small sample size in group one (21.8 ± 9.62) for ATENP and its lowest treatment matching factor (0.018 ± 0.0029). The DR estimator corrects the bias with an outcome prediction model, resulting in a change from 0.009 to 0.504.

Last but not least, although the TMF value of etIPS is larger than other baselines except the predictive model, it is still a bit away from the expected value of 1 (cf. Appendix C). Two reasons account for it. The first is the suboptimal accuracy of the predictive model, which is 0.571 ± 0.0037 in the test set. As mentioned earlier, the estimated propensity score is used to compute TMF. Therefore, it indicates the alignment between the policies of the predictive model and other models. As the predictive model cannot perfectly reflect the physicians' policy, the TMF value computed based on it does not necessarily have to be strictly around 1 anymore. Nevertheless, the TMF computed from the predictive model is still worth being referenced when the value is extremely low like for eIPS. The second reason is the average risk in the dataset being 0.498. In other words, almost half of the time, the physicians' treatment does not receive a positive outcome. The relatively large amount of negative feedback encourages the algorithm to learn a new policy that is a bit different from the physicians. Besides, there are 25 treatment decisions observed with strong skewness in its distribution (cf. Appendix B). These facts would jointly result in a lower TMF.

VI. CONCLUSION

In this paper, we propose a general framework, etIPS, to learn optimal ITRs. It consists of a predictive model and an ITRs model. The former takes advantage of the state-of-the-art predictive modeling of the treatment decisions while the latter is based on the latest formulation of BLBF problems. By casting the ITRs learning as a problem in BLBF, our proposed approach can discover the optimal policies with sequential EHRs from observational studies. Intuitively speaking, the new policy is learned by encouraging the treatments with a positive outcome and discouraging the treatments with a negative outcome. The reformulation of the SNIPS estimator ensures that such a learning objective is correctly integrated into the objective function of the neural network. The generality of our proposed framework lies in the flexibility to choose an arbitrary propensity score model as well as any ITRs model that would fit the patient features.

With experiments on two simulated BLBF tasks using the MNIST dataset, we have empirically shown that the estimated propensity score can replace the true propensity score when the latter is not known. The result facilitates the usage of data from observational studies without any recorded propensity score. Furthermore, in various offline evaluation methods, our learned policies perform better than the physicians' policy. A true performance evaluation, naturally, would require additional clinical testing.

The proposed framework is compatible with any neural network structures and any data sources, not limiting to recurrent neural networks and sequential EHRs, as we have presented in this paper. With more advanced network structures, the performance of our framework could be further boosted.

As part of future work, we want to study model explainability or interpretability. If the treatment suggestion is provided together with explanations, the physicians would find such clinical decision support systems more transparent and become more encouraged to apply it. For example, the explanation can show which parts of the static or sequential information are especially important for the final treatment suggestion.

ACKNOWLEDGMENT

The authors acknowledge support by the German Federal Ministry for Education and Research (BMBF), funding project "MLWin" (grant 01IS18050).

GEFÖRDERT VOM

REFERENCES

[1] V. Tresp, J. M. Overhage, M. Bundschus, S. Rabizadeh, P. A. Fasching, and S. Yu, "Going digital: a survey on digitalization and large-scale data analytics in healthcare," *Proceedings of the IEEE*, vol. 104, no. 11, pp. 2180–2206, 2016.

[2] X. Zhou, N. Mayer-Hamblett, U. Khan, and M. R. Kosorok, "Residual weighted learning for estimating individualized treatment rules," *Journal of the American Statistical Association*, vol. 112, no. 517, pp. 169–187, 2017.

[3] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, "Doctor ai: Predicting clinical events via recurrent neural networks," in *Machine Learning for Healthcare Conference*, 2016, pp. 301–318.

[4] C. Esteban, D. Schmidt, D. Krompaß, and V. Tresp, "Predicting sequences of clinical events by using a personalized temporal latent embedding model," in *Healthcare Informatics (ICHI), 2015 International Conference on*. IEEE, 2015, pp. 130–139.

[5] C. Esteban, O. Staeck, S. Baier, Y. Yang, and V. Tresp, "Predicting clinical events by combining static and dynamic information using recurrent neural networks," in *2016 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, 2016, pp. 93–101.

[6] Y. Yang, P. A. Fasching, and V. Tresp, "Predictive modeling of therapy decisions in metastatic breast cancer with recurrent neural network encoder and multinomial hierarchical regression decoder," in *2017 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, 2017, pp. 46–55.

[7] M. Dudík, J. Langford, and L. Li, "Doubly robust policy evaluation and learning," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 2011, pp. 1097–1104.

[8] P. R. Rosenbaum and D. B. Rubin, "The central role of the propensity score in observational studies for causal effects," *Biometrika*, vol. 70, no. 1, pp. 41–55, 1983.

[9] M. Qian and S. A. Murphy, "Performance guarantees for individualized treatment rules," *Annals of statistics*, vol. 39, no. 2, p. 1180, 2011.

[10] Y. Zhao, D. Zeng, A. J. Rush, and M. R. Kosorok, "Estimating individualized treatment rules using outcome weighted learning," *Journal of the American Statistical Association*, vol. 107, no. 499, pp. 1106–1118, 2012.

[11] 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*, vol. 3, p. 160035, 2016.

[12] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[13] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[14] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[15] R. Zhao and V. Tresp, "Learning goal-oriented visual dialog via tempered policy gradient," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 868–875.

[16] R. Zhao and V. Tresp, "Efficient dialog policy learning via positive memory retention," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 823–830.

[17] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," *Advances in NIPS*, 2014.

[18] E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart, "Retain: An interpretable predictive model for healthcare using reverse time attention mechanism," in *Advances in Neural Information Processing Systems*, 2016, pp. 3504–3512.

[19] M. Komorowski, L. A. Celi, O. Badawi, A. C. Gordon, and A. A. Faisal, "The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care," *Nature Medicine*, vol. 24, no. 11, p. 1716, 2018.

[20] A. Beygelzimer and J. Langford, "The offset tree for learning with partial labels," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 129–138.

[21] A. Swaminathan and T. Joachims, "Counterfactual risk minimization: Learning from logged bandit feedback," in *International Conference on Machine Learning*, 2015, pp. 814–823.

[22] A. Swaminathan and T. Joachims, "The self-normalized estimator for counterfactual learning," in *advances in neural information processing systems*, 2015, pp. 3231–3239.

[23] T. Joachims, A. Swaminathan, and M. de Rijke, "Deep learning with logged bandit feedback," in *International Conference on Learning Representations*, 2018.

[24] M. Singer, C. S. Deutschman *et al.*, "The third international consensus definitions for sepsis and septic shock (sepsis-3)," *Jama*, vol. 315, no. 8, pp. 801–810, 2016.

[25] L. Byrne and F. Van Haren, "Fluid resuscitation in human sepsis: Time to rewrite history?" *Annals of intensive care*, vol. 7, no. 1, p. 4, 2017.

[26] F. L. Ferreira, D. P. Bota, A. Bross, C. Mélot, and J.-L. Vincent, "Serial evaluation of the sofa score to predict outcome in critically ill patients," *Jama*, vol. 286, no. 14, pp. 1754–1758, 2001.

[27] A. Raghu, M. Komorowski, I. Ahmed, L. Celi, P. Szolovits, and M. Ghassemi, "Deep reinforcement learning for sepsis treatment," *arXiv preprint arXiv:1711.09602*, 2017.

[28] T. Hesterberg, "Weighted average importance sampling and defensive mixture distributions," *Technometrics*, vol. 37, no. 2, pp. 185–194, 1995.

[29] Y. Bengio, P. Frasconi, and P. Simard, "The problem of learning long-term dependencies in recurrent networks," in *IEEE international conference on neural networks*. IEEE, 1993, pp. 1183–1188.

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

[31] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[32] 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*. IEEE, 2013, pp. 6645–6649.

[33] M. Abadi, A. Agarwal *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[34] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: a python library for model selection and hyperparameter optimization," *Computational Science & Discovery*, vol. 8, no. 1, p. 014008, jul 2015.

[35] A. Agarwal, D. Hsu, S. Kale, J. Langford, L. Li, and R. Schapire, "Taming the monster: A fast and simple algorithm for contextual bandits," in *International Conference on Machine Learning*, 2014, pp. 1638–1646.

[36] Y. Xie, B. Liu, Q. Liu, Z. Wang, Y. Zhou, and J. Peng, "Off-policy evaluation and learning from logged bandit feedback: Error reduction via surrogate policy," *arXiv preprint arXiv:1808.00232*, 2018.
[37] O. Gottesman, F. Johansson *et al.*, "Evaluating reinforcement learning algorithms in observational health settings," *arXiv preprint arXiv:1805.12298*, 2018.

APPENDIX

A. Feature description

The included features are chosen to best represent the status of each patient [19]. There could possibly be confounding effects if we haven't included some important features in the model. In the chosen features, most have continuous values except for gender, readmission, and mechanical ventilation being binary.

Static information: age, gender, readmission to intensive care.

Sequential information: weight (kg), Glasgow Coma Scale (GCS), heart rate(HR), Systolic, Mean and Diastolic Blood Pressure(SysBP, MeanBP, DiaBP), Respiratory Rate (RR), SpO2, temperature (celsius), FiO2, Potassium, Sodium, Chloride, Glucose, Blood Urea Nitrogen (BUN), Creatinie, Magnesium, Calcium, SGOT, SGPT, Total Bilirubin, Hemoglobin, count of the white blood cells, count of the platelets, Partial Thromboplastin Time (PTT), Prothorombin Time (PT), International Normalized Ratio (INR), Arterial potential Hydrogen, paO2, paCO2, Arterial Base Excess, Artial lactate, HCO3, mechanical ventilation, shock index, PaO2/FiO2 ratio, maximum dose of vasopressor over 4 hours, intravenous fluids intake over 4 hours, total input, total urine fluid output, urine output over 4 hours, cumulated fluid balance, Sequential Organ Failure Assessment (SOFA) over 4 hours, Systemic Inflammatory Response Syndrome (SIRS) over 4 hours.

B. Treatment decisions

Fig. 4 shows the distribution of different treatment options. The skewness of the distribution is mainly due to the unbalanced distribution of the discretized VP.

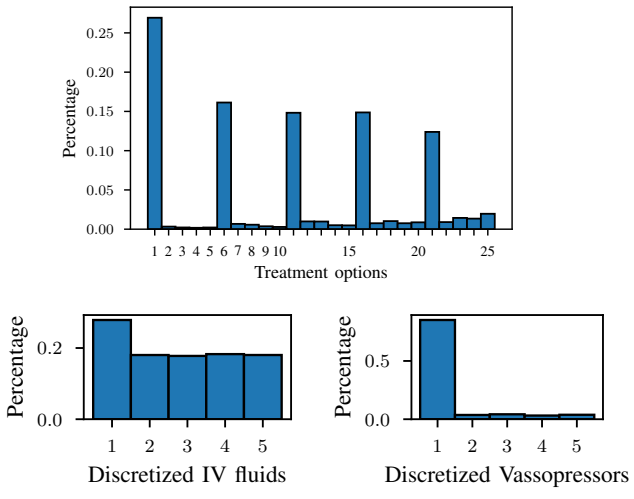


Fig. 4. Distribution of treatment decisions and discretized IV/VP

Due to the possible update of the MIMIC-III database, there are slight differences between the values in Table III compared to the ones reported by Komorowski et al. [19].

TABLE III
RANGE AND MEDIAN OF IV AND VP

Treatment	IV fluids (mL/ 4 hours)		Vasopressors (mcg/kg/min)	
	range	median	range	median
1	0	0	0	0
2	0 - 48	30	0 - 0.08	0.04
3	48 - 150	80	0.08 - 0.2	0.13
4	150 - 500	284	0.2 - 0.45	0.27
5	> 500	874	> 0.45	0.78

C. Treatment matching factor

$$\begin{aligned}
&\mathbb{E}_{X\sim\mathbb{P}(X)}\mathbb{E}_{a\sim\mathbb{P}(a|X)}\left[\frac{\pi_{\boldsymbol{w}}(a|X)}{\mathbb{P}(a|X)}\right] \\
&= \sum_X \mathbb{P}(X)\sum_a \mathbb{P}(a|X)\frac{\pi_{\boldsymbol{w}}(a|X)}{\mathbb{P}(a|X)} \\
&= \sum_X\sum_a \mathbb{P}(X)\pi_{\boldsymbol{w}}(a|X) \\
&= 1
\end{aligned}$$

D. Lack of equvariance of the IPS estimator

$$\min_{\boldsymbol{w}} \frac{1}{m}\sum_{i=1}^m (\delta_i + c)\frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)} \neq c + \min_{\boldsymbol{w}} \frac{1}{m}\sum_{i=1}^m \delta_i \frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}$$

E. Equvariance of the SNIPS estimator

$$\begin{aligned}
&\min_{\boldsymbol{w}} \frac{\frac{1}{m}\sum_{i=1}^m (\delta_i + c)\frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}}{\frac{1}{m}\sum_{i=1}^m \frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}} \\
&= \min_{\boldsymbol{w}} \left(\frac{\frac{1}{m}\sum_{i=1}^m \delta_i \frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}}{\frac{1}{m}\sum_{i=1}^m \frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}} + c\cdot \frac{\cancel{\frac{1}{m}\sum_{i=1}^m \frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}}}{\cancel{\frac{1}{m}\sum_{i=1}^m \frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}}} \right) \\
&= \min_{\boldsymbol{w}} \frac{\frac{1}{m}\sum_{i=1}^m \delta_i \frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}}{\frac{1}{m}\sum_{i=1}^m \frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}}
\end{aligned}$$

F. Reformulation of the SNIPS risk

The optimization objective of the SNIPS risk

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \frac{\frac{1}{m}\sum_{i=1}^m \delta_i \frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}}{\frac{1}{m}\sum_{i=1}^m \frac{\pi_{\boldsymbol{w}}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}}$$

could be reformulated as a two-step optimization problem

$$s^*, \boldsymbol{w}^* = \arg\min_{s_j} \left\{ \arg\min_{\boldsymbol{w}_j} \frac{\frac{1}{m}\sum_{i=1}^m \delta_i \frac{\pi_{\boldsymbol{w}_j}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}}{s_j}, \right.$$
$$\left. \text{s.t. } \frac{1}{m}\sum_{i=1}^m \frac{\pi_{\boldsymbol{w}_j}(a_i|X_i)}{\mathbb{P}(a_i|X_i)} = s_j \right\}$$

where s_j is fixed to different values and $\boldsymbol{w}_j$ represents the corresponding optimization parameters. In other words, the minimizer can be found by 1) fixing s_j to a particular value within a grid search, and 2) solving the corresponding interior constrained optimization problem to find $\boldsymbol{w}_j^*$. The final minimizer is the pair with the lowest SNIPS risk among all (s_j, w_j^*) pairs.

The remaining problem is to solve the interior constrained optimization problem. It is natural to use the Lagrange multiplier to remove the constraint of the fixed s_j. Formally, the problem

$$\boldsymbol{w}_j^* = \arg\min_{\boldsymbol{w}_j} \left\{ \frac{1}{m}\sum_{i=1}^{m} \delta_i \frac{\pi_{\boldsymbol{w}_j}(a_i|X_i)}{\mathbb{P}(a_i|X_i)}, \right. \\ \left. \text{s.t. } \frac{1}{m}\sum_{i=1}^{m} \frac{\pi_{\boldsymbol{w}_j}(a_i|X_i)}{\mathbb{P}(a_i|X_i)} = s_j \right\}$$

is equivalent to

$$\boldsymbol{w}_j^*, \lambda_j^* = \arg\min_{\boldsymbol{w}_j} \max_{\lambda_j} \left\{ \frac{1}{m}\sum_{i=1}^{m} (\delta_i - \lambda_j) \frac{\pi_{\boldsymbol{w}_j}(a_i|X_i)}{\mathbb{P}(a_i|X_i)} + \lambda_j s_j \right\}.$$

Considering the fact that searching for λ_j^* with a fixed s_j is expensive but the inverse is not, reversing the role of λ_j^* and s_j makes the problem more tractable, i.e., fix λ_j first, optimize for $\boldsymbol{w}_j^*$, and compute the corresponding s_j as well as the SNIPS risk $\hat{r}_{\text{SNIPS}}(\pi_{\boldsymbol{w}_j^*})$. Formally, the optimization problem is further reduced to

$$\boldsymbol{w}_j^* = \arg\min_{\boldsymbol{w}_j} \left\{ \frac{1}{m}\sum_{i=1}^{m} (\delta_i - \lambda_j) \frac{\pi_{\boldsymbol{w}_j}(a_i|X_i)}{\mathbb{P}(a_i|X_i)} \right\}.$$

G. Sequential classification tasks from MNIST

Table IV shows some statistics for the tailored sequential classification tasks. The output classes in both tasks have a balanced distribution. Meanwhile, due to the preference of the logging policy, the label predictions show the skewness to some extend in Fig. 5.

TABLE IV
BASIC STATISTICS OF THE TAILORED TASKS WITH MNIST

	zeros counting MNIST	row-by-row MNIST
#samples	$10,000$	$70,000$
input shape (#time-steps, #features)	$(20 \pm 5, 784)$	$(28, 28)$
#output classes	3	10

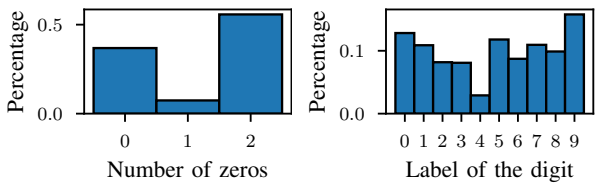


Fig. 5. Distribution of label prediction of the logging policy

H. Different evaluation methods

As the accuracy is computed with the ground truth label, it serves as a good reference to understand the performance of different risk estimators. In Fig. 6 and 7, the blue color denotes the performance of zeros counting MNIST while red the row-by-row MNIST.

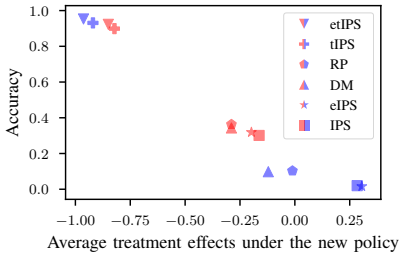


Fig. 6. Correlation between the accuracy and ATENP

Although the policy is evaluated in a deterministic way, ATENP shows a consistent correlation with the accuracy of different policies. In addition, the sample size in different groups serves as a good indicator for the propensity score overfitting problem. For the row-by-row MNIST, there are only 4.6 ± 3.83 samples in group one for the policy learned with IPS, while the number is 4406.8 ± 373.43 for tIPS, which corresponds to the low treatment matching factors as discussed in Sec. V-B3.

In Fig. 7, IPS/eIPS approaches have the smallest estimated risk, which indicates that the IPS estimator is strongly biased if the propensity score overfitting problem occurs. Taking advantage of an additional outcome prediction model, the DR estimator corrects the risk estimation and is therefore more reliable.

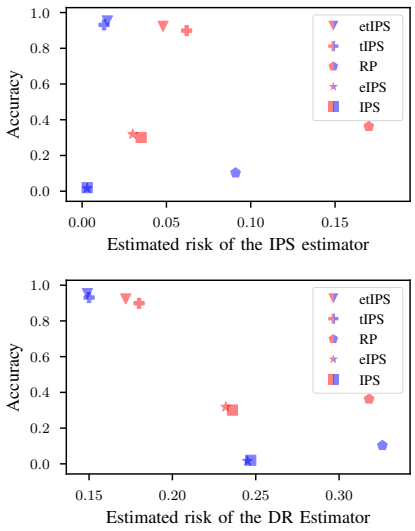


Fig. 7. Comparison of the risk estimation with IPS/DR estimator

Chapter 3

Uncertainty-Aware Regression Models in Medical Image Analysis

Quantifying Predictive Uncertainty in Medical Image Analysis with Deep Kernel Learning

1st Zhiliang Wu
Siemens AG
LMU Munich
Munich, Germany
zhiliang.wu@siemens.com

2nd Yinchong Yang
Siemens AG
Munich, Germany
yinchong.yang@siemens.com

3rd Jindong Gu
Siemens AG
LMU Munich
Munich, Germany
jindong.gu@siemens.com

4th Volker Tresp
Siemens AG
LMU Munich
Munich, Germany
volker.tresp@siemens.com

***Abstract*—Deep neural networks are increasingly being used for the analysis of medical images. However, most works neglect the uncertainty in the model's prediction. We propose an uncertainty-aware deep kernel learning model which permits the estimation of the uncertainty in the prediction by a pipeline of a Convolutional Neural Network and a sparse Gaussian Process. Furthermore, we adapt different pre-training methods to investigate their impacts on the proposed model. We apply our approach to Bone Age Prediction and Lesion Localization. In most cases, the proposed model shows better performance compared to common architectures. More importantly, our model expresses systematically higher confidence in more accurate predictions and less confidence in less accurate ones. Our model can also be used to detect challenging and controversial test samples. Compared to related methods such as Monte-Carlo Dropout, our approach derives the uncertainty information in a purely analytical fashion and is thus computationally more efficient.**

***Index Terms*—uncertainty quantification, medical imaging, sparse Gaussian Process approximation, deep Convolutional Neural Networks,**

I. Introduction

Various machine learning methods have been developed to support patient care to deal with the exploding amount of healthcare data [1], [2]. An important example is medical imaging. In classical image analysis, the standard machine learning methods make predictions based on sophisticated handcrafted features extracted from the medical images. With the introduction of deep neural networks (DNNs), especially Convolutional Neural Networks (CNNs), manual feature engineering is replaced by self-organized supervised representation learning.

Although DNNs now define the state-of-the-art in many applications, an often encountered problem is that they fail to provide reasonable confidence estimates for their predictions [3]. In a classification task, this can result in over-confident predictions for misclassified samples [4]. This observation has encouraged the development of various calibration methods such as temperature scaling [5] and isotonic regression [6]. In a regression task, however, the modeling of input-dependent predictive uncertainty is rarely considered. Whereas in classification, a well-calibrated probabilistic prediction can sometimes be used to derive confidence values, in regression one needs to estimate confidence considering predictive distributions [7].

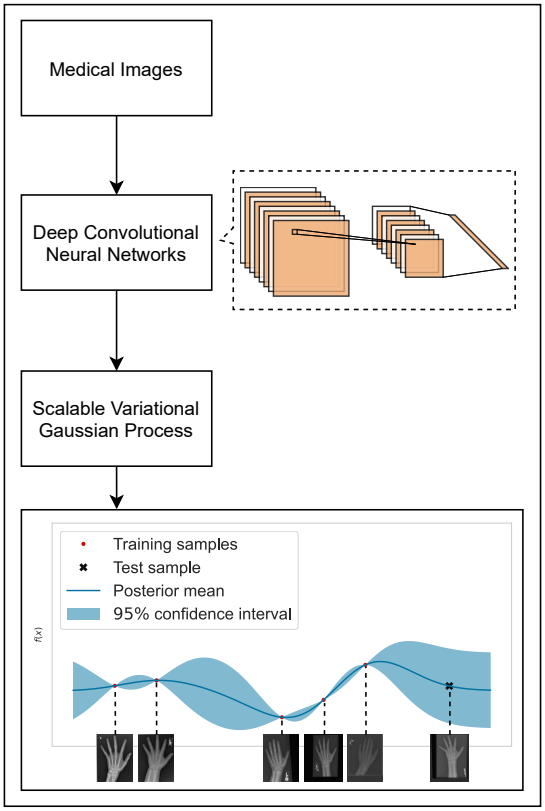


Fig. 1. An illustration of our proposed model: Combining a deep Convolutional Neural Network with the Scalable Variational Gaussian Process. Latent features of a medical image are extracted by the Convolutional Neural Network and then consumed by the Scalable Variational Gaussian Process. The proposed model outputs a predictive distribution of the Gaussian Process posterior, which can be interpreted as a mean estimate and a predictive variance. For instance, with a localized kernel the prediction of a rare test sample (the rightmost image) with few similar training samples demonstrates a larger variance.

The problem of quantifying the predictive uncertainty in deep learning models limits their applicability to safety-critical domains such as healthcare [8], [9]. In most cases, existing clinical decision support systems that rely on deep learning

can only provide a point estimate, e.g., for a continuous severity score, progression-free survival time, or length-of-stay. Physicians, who are supposed to interpret the output of a DNN, face the challenge of not knowing how much they could trust the prediction. The goal of this paper is to provide a quantified uncertainty estimate for each prediction in a principled, mathematically sound manner. An unusually high uncertainty estimate would encourage the physician to investigate a case more closely since it is more likely to deviate from the "normal" ones. A high uncertainty typically means that there are few similar cases in the training data.

In a frequentist analysis of linear regression models, the predicted variance is derived from that of the model parameters; due to its high-dimensional nonlinear nature, this cannot easily be applied to DNNs. Currently, leading approaches to reason on the uncertainty in DNNs include MC dropout methods [10], [11], Bayesian neural networks [12], [13] and deep ensembles [14]. These methods could become computationally expensive when large and deep neural networks are necessary. In this paper, we explore another direction to quantify predictive uncertainty with Gaussian Processes (GPs), a well-known class of Bayesian machine learning methods [15]. A GP implicitly ties the predictive uncertainty with the similarity between samples, which does not model ensembles and can produce a predictive distribution with only one forward pass.

More specifically, when localized kernels are being used, e.g., Radial Basis Function (RBF) kernels or the more general Matérn kernels, a GP model would be confident in its mean estimate if there have been training samples observed in the "neighborhood" of the test input. Otherwise, the model would tend to output high variances for the predictions. Thus, for RBF-kernels, the Euclidean distance of inputs must be meaningful in the application, which often is not the case in high-dimensional problems, a typical example being raw images described by their pixel values.

Another known challenge in GP is the fact that computational complexity scales as $\mathcal{O}(n^3)$ and storage complexity as $\mathcal{O}(n^2)$, where n denotes the number of data samples [15]. In recent years, significant progress has been made to address these scalability issues [16], which has motivated work on combining DNNs with GPs, a.k.a. *Deep Kernel Learning* [17]. The pipeline architecture we propose in this paper is shown in Fig. 1, where we apply a state-of-the-art sparse GP on top of a CNN for predictions on medical images.

Our contribution can be summarized as follows:

- We present a novel deep kernel learning model for regression on medical images based on the latest developments in sparse GPs and CNNs.
- We enhance the proposed model by introducing different pre-training methods for the CNNs and initialization methods to optimize the inducing points in sparse GPs.
- We apply the proposed model to the tasks of univariate bone age prediction and multivariate lesion localization, and provide a thorough comparison of different pre-training and initialization methods in terms of both point estimate and predictive uncertainty.

II. Related work

Deep Convolutional Neural Networks for Medical Image Analysis The analysis of medical images is arguably one of the areas where deep learning methods have been demonstrating the most promising performances, including diagnostics, dermatology, radiology, ophthalmology, and pathology [18]. Deep learning-based solutions can offer physicians second opinions by, e.g., annotating the regions of interest. More specifically, CNN-based solutions have achieved physician-level accuracy, e.g., with CheXNet [19] for pneumonia detection, which is a 121-layer Dense Convolutional Network (DenseNet) [20] trained on the ChestX-ray 14 dataset [21]. One factor limiting the progress is the relatively small size of labeled datasets available for specific clinical tasks when compared to large visual databases on nonmedical images, like the ImageNet dataset [22]. Therefore, methods like transfer learning are commonly used to take advantage of models trained on more or less unrelated datasets. However, many of these solutions focus only on improving the point estimate performance, ignoring the importance of the predictions' uncertainty. In this work, we focus on addressing the problem of providing meaningful uncertainty estimates.

Scalable Variational Gaussian Process with Neural Networks Efforts from earlier times include the Bayesian committe machine [23], Nyström methods [24], [25], the Fully Independent Training Conditional (FITC) Approximation [26], and Variational Free Energy (VFE) [27]. Recently, [28] proposed the Scalable Variational Gaussian Process (SVGP), which reduces the computational complexity to $\mathcal{O}(m^3)$, where m denotes the number of *inducing points* (more details see Sec. III-A). In addition, SVGP enables the training with stochastic gradient descent (SGD)-based methods. Afterward, [29] combined SVGP with DNNs for classification tasks. The approach is called Gaussian Process hybrid deep networks (GPDNN). [17] combined neural networks with a KISS-GP covariance matrix, which takes advantage of a sparse matrix with inducing points lying on some grid structure [30]. The proposed method is called Deep Kernel Learning (DKL). More recently, [31] proposed the Parametric Predictive Gaussian Process (PPGP) regressor to improve the predictive variances in SVGP-based models, which shows promising performance in various applications. Inspired by the idea of DKL, we propose in this paper a model to train a state-of-the-art sparse GP model with deep CNNs in a more cohesive way.

Pre-training Techniques for Deep Convolutional Neural networks Pre-training techniques have been developed for Deep Belief Networks [32] and stacked auto-encoders [33], where unsupervised pre-training was used for initialization, followed by supervised fine-tuning. Meanwhile, transfer learning techniques received increasing attention due to their ability to derive good representations for instances also from domains not considered in training [34]. After the introduction of the ImageNet challenge [22], it has been common practice to pre-train models on the ImageNet dataset as an initialization for other downstream tasks. More recently, self-supervised

learning methods, e.g., contrastive learning [35], have gained much attention as a powerful learning paradigm, which bridges the performance gap between supervised learning methods and unsupervised ones significantly. From a pre-training perspective, self-supervised learning can be considered as an example in deep metric learning (DML). The goal of DML is to map data to a latent space where data points with similar labels are located close together, and data with dissimilar labels are far apart [36]. In this paper, we adapt different pre-training methods under the setting of DKL.

III. Method

In this section, we provide a detailed introduction to our method. Our proposed model consists of two consecutive parts: a trainable feature extractor based on deep CNNs and an uncertainty-aware prediction model in the form of sparse GPs. The feature extractor is also commonly known as the *backbone*, because it refers to the parts of the network excluding the final classification layers in, e.g., DenseNets [20] or ResNets [37]. The output of such backbones, namely the feature maps, a.k.a. latent representations of the raw input, serves as input to the predictive GP regression. In the following, we first discuss how sparse GP models can scale to large datasets. Afterward, we introduce our initialization and pre-training techniques. Finally, we summarize the complete algorithm from the network initialization to the GP fine-tuning.

Notations: We denote the training dataset as $\{\boldsymbol{X}_i, y_i\}_{i=1}^n$, where $\boldsymbol{X}_i \in \mathbb{R}^{n_\mathrm{H} \times n_\mathrm{W} \times n_\mathrm{C}}$ is an image of size $n_\mathrm{H} \times n_\mathrm{W}$ with n_C color dimensions, $y_i \in \mathbb{R}$ is the target variable in a univariate regression task, and n is the number of data samples. With the CNN backbones, we extract a latent representation from $\boldsymbol{X}_i$ and denote it as $\boldsymbol{h}_i$.

A. Scalable Variational Gaussian Processes as Output Layers

A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint zero-mean Gaussian distribution [15]. Formally, if we denote all target variables y_i in the column vector as $\boldsymbol{y} \in \mathbb{R}^n$ in a univariate regression problem, it follows

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K} + \sigma_\mathrm{obs}^2 \boldsymbol{I}),$$

where the covariance matrix $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ is parametrized by the respective inputs as

$$k_{ij} := k(\boldsymbol{h}_i, \boldsymbol{h}_j)$$

and σ_obs^2 is the noise variance. Note that in a standard setup of GP, the input to the kernel function $k(\cdot, \cdot)$ is a pair of feature vectors. In the scope of our work, we feed the latent representations generated by CNN backbones to the kernel function.

To find optimal hyperparameters in the kernel function (e.g., the scaling parameter in an RBF-kernel), the training of the GP involves maximizing the log marginal likelihood

$$\mathcal{L}_\mathrm{GP} = -\frac{1}{2}\boldsymbol{y}^\top \left(\boldsymbol{K} + \sigma_\mathrm{obs}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} - \frac{1}{2}\log\left|\boldsymbol{K} + \sigma_\mathrm{obs}^2 \boldsymbol{I}\right| - \frac{n}{2}\log 2\pi.$$

Given a new input sample $\boldsymbol{h}_*$, the GP model provides a predictive distribution as

$$f_* \sim \mathcal{N}(\boldsymbol{k}_*^\top \left(\boldsymbol{K} + \sigma_\mathrm{obs}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}, k_{**} - \boldsymbol{k}_*^\top \left(\boldsymbol{K} + \sigma_\mathrm{obs}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_*),$$

where $\boldsymbol{k}_* = [k(\boldsymbol{h}_1, \boldsymbol{h}_*), \ldots, k(\boldsymbol{h}_n, \boldsymbol{h}_*)]^\top \in \mathbb{R}^n$.

However, the complexity from the inverse operation of the large matrices in $\mathcal{L}_\mathrm{GP}$ and f_* hinders the application of GP models to large-scale datasets, which was the motivation for works on different approximation methods.

The key idea of [26]–[28] is to learn a number of so-called inducing points by variational methods, which can be viewed as a learnable pseudo dataset $\{\boldsymbol{z}_i, u_i\}_{i=1}^m =: (\boldsymbol{Z}, \boldsymbol{u})$ to summarize the original large dataset, where $m \ll n$. The approximation follows these steps: 1) The original dataset is augmented with the inducing points; 2) Based on different assumptions, the log marginal likelihood $\log p(\boldsymbol{y})$ is approximated as a function only of inducing points; 3) Optimization is done by maximizing either the approximated log marginal likelihood [26], or the lower bound of it, which is also known as the Evidence Lower BOund (ELBO) [27], [28]; 4) The predictions for new samples are based on the optimized inducing points instead of the original dataset. Among all approximation methods, SVGP turns out to be the most popular one, possibly thanks to its largely reduced computational and storage complexity as well as the natural integration of SGD-based methods [28], [38]. Therefore, we take advantage of SVGP as one of the sparse GP models in this paper.

In SVGP [28], a multivariate Normal distribution $\mathcal{N}(\boldsymbol{m}, \boldsymbol{S})$ is introduced to the variational distribution $q(\boldsymbol{u})$. [31] maximizes the ELBO

$$\begin{aligned}\mathcal{L}_\mathrm{SVGP} = &\sum_{i=1}^n \left\{ \log \mathcal{N}\left(y_i \mid \mu_{\boldsymbol{f}}\left(\boldsymbol{h}_i\right), \sigma_\mathrm{obs}^2\right) - \frac{\sigma_{\boldsymbol{f}}^2\left(\boldsymbol{h}_i\right)}{2\sigma_\mathrm{obs}^2} \right\} \\ &- \mathrm{KL}(q(\boldsymbol{u}) \| p(\boldsymbol{u})),\end{aligned} \tag{1}$$

where we have the predictive mean $\mu_{\boldsymbol{f}}(\boldsymbol{h}_i) = \boldsymbol{k}_i^\top \boldsymbol{K}_{\boldsymbol{uu}}^{-1} \boldsymbol{m}$, the predictive variance $\sigma_{\boldsymbol{f}}^2(\boldsymbol{h}_i) = k_{ii} - \boldsymbol{k}_i^\top \boldsymbol{K}_{\boldsymbol{uu}}^{-1} \boldsymbol{k}_i + \boldsymbol{k}_i^\top \boldsymbol{K}_{\boldsymbol{uu}}^{-1} \boldsymbol{S} \boldsymbol{K}_{\boldsymbol{uu}}^{-1} \boldsymbol{k}_i$, $p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{uu}}), \boldsymbol{k}_i \in \mathbb{R}^m, \boldsymbol{K}_{\boldsymbol{uu}} \in \mathbb{R}^{m \times m}$, and $\mathrm{KL}(\cdot||\cdot)$ denotes the Kullback–Leibler divergence between two distributions. We use $\boldsymbol{\Theta}$ to denote all trainable parameters and adapt them with SGD-based methods, including $\boldsymbol{m}, \boldsymbol{S}$ for the variational distribution $q(\boldsymbol{u})$, inducing points $\boldsymbol{Z}, \boldsymbol{u}$ for the covariance matrices like $\boldsymbol{K}_{\boldsymbol{uu}}$ or $\boldsymbol{k}_i$, σ_obs in the likelihood model and various hyperparameters in the kernel function, e.g., length scale in an RBF kernel.

[31] points out that the predictive distribution in SVGP tends to be dominated by the observational noise and underestimates the input-dependent uncertainty. As a solution, they proposed the PPGP Regressor, which takes advantage of the formulation of the predictive distributions in SVGP but restores the symmetry of the function variance $\mu_{\boldsymbol{f}}(\boldsymbol{h}_i)$ in the training objective through the maximum likelihood estimation (MLE) methods. Formally, the objective in PPGP is

$$\begin{aligned}\mathcal{L}_\mathrm{PPGP} = &\sum_{i=1}^n \log \mathcal{N}\left(y_i \mid \mu_{\boldsymbol{f}}\left(\boldsymbol{h}_i\right), \sigma_\mathrm{obs}^2 + \sigma_{\boldsymbol{f}}^2\left(\boldsymbol{h}_i\right)\right) \\ &- \mathrm{KL}(q(\boldsymbol{u}) \| p(\boldsymbol{u})).\end{aligned} \tag{2}$$

In our experiments, we report results of both SVGP and PPGP methods, which only differ in their respective ELBO objectives.

Although the scalability problem in large datasets is nicely addressed in the SVGP-based models, the commonly used GP kernels, such as RBF and Matérn, cannot directly handle high dimensional data such as images. Therefore, there are many efforts to combine the inductive bias in neural networks and the non-parametric nature of GP-based models, including GPDNN [29] and DKL [17]. In [29] and [17], the training is initiated with a standard neural network with a linear predictive model fit on the target variable. Afterward, the linear model is replaced with a GP to enable uncertainty-aware prediction. In our experiments with these approaches, we do not observe performance improvement in terms of point estimates. Therefore, we explore other pre-training methods that do not directly require the target variables, including Convolutional Autoencoders and Deep Metric Learning.

Fig. 2 illustrates the basic idea of integrating DKL in our proposed model. The image sample $\boldsymbol{X}_i$ is embedded in some latent space defined by the backbone as $\boldsymbol{h}_i$. The SVGP-based model then consumes $\boldsymbol{h}_i$ as the input to produce the predictive distribution $\mathcal{N}(\mu_i, \sigma_i^2)$ for the target variable y_i. In other words, we propose to use SVGP-based models as output layers to replace the final linear layers found in common CNN architectures. The trainable parameters $\boldsymbol{\Phi}$ in the backbone and $\boldsymbol{\Theta}$ in the SVGP-based output layers are optimized together w.r.t. the ELBO objective.

Based on our observations, we realized that two modifications turn out to be critical for training the proposed model. First, we find it necessary for the model architecture to add one more linear layer after the backbone to *further* reduce the dimension of the latent space. In ResNet18 and DenseNet121, the dimensions of the extracted latent spaces are 512 and 1024, respectively. These dimensions prove to be too large for RBF kernel functions that rely on l^2 norm, presumably due to the fact that in high dimensional space, the Euclidean norm becomes irrelevant as a distance measure [39], [40]. With a thorough hyper-parameter search, we find that a dimension reduction to 50 always shows a stable performance.

Second, the initialization of the inducing points plays an important role. If the inducing inputs are initialized from random vectors, the training never converges to meaningful results in our experiments. One explanation would be that with purely random initialization of inducing points, the covariance function value between each pair of samples is also random since it is defined via all inducing points (cf. Equation (1) in [27]). To this end, the GP has no chance of modeling the target based on these random distances with a multivariate Gaussian. As a solution, we initialize inducing inputs by the latent representations produced by the backbone from the image samples. Formally, we initialize the inducing inputs by

$$\boldsymbol{z}_i^{\text{init}} = \boldsymbol{h}_i = f_{\boldsymbol{\Phi}}(\boldsymbol{X}_i),$$

where the parameters $\boldsymbol{\Phi}$ in the backbone can be initialized from scratch, transferred from other models, or pre-trained in auxiliary tasks. Such a procedure is similar to using a subset of the dataset as the initial inducing inputs in a vanilla SVGP model, where raw features are fed to the model directly.

So far, we focused on models for univariate regression problems. For the multivariate case, we propose to use the same backbone to generate the latent representations but feed it as input to multiple independent SVGP-based models. The number of involved SVGPs equals the dimension of the target variables.[1] For the cases where there are correlations between the target variables, more advanced methods like Linear Model of Coregionalization (LMC) can be used [41], which we leave to our future work.

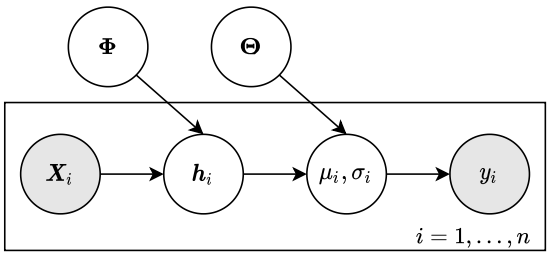


Fig. 2. Graphical model of deep kernel learning model in plate notation. Nodes are variables where shaded ones are observed, and non-shaded ones are latent variables. Plates indicate the repetition of the subgraph.

B. Pre-training Convolutional Neural Networks

In our proposed architecture, the sparse GP model is defined in a latent space learned by the CNN backbone. The optimization task is correspondingly twofold: the GP is supposed to learn the parameters like the inducing points, and the CNN backbone should adapt its parameters to generate representative latent features. However, in the early phase of training, the CNN backbone may not have learned to extract representative features. In other words, the training samples could be mapped somewhat randomly in the latent space. This could pose a challenging task for the downstream GP model such that – based on our observation of the experiments – its parameters might converge to unfavorable values that are difficult to correct later on. To address this issue, we anticipate that the training quality could be improved by two strategies:

- Initialization with transfer learning in Sec. III-B1
- Pre-training with auxiliary tasks:
  - Convolutional Autoencoder in Sec. III-B2
  - Deep Metric Learning in Sec. III-B3

1) Transfer learning: We regard transfer learning as reusing the knowledge from the models trained on different datasets. In the simplest case, we consider reusing CNN layers that have been trained on the classification task on the ImageNet dataset. It has been shown that in a CNN architecture, the early layers that are close to the input can learn to extract generic, low-level features that may apply across different types of image

[1]This is also the configuration for the model with (multivariate) linear layers, which facilitates a fair comparison in experiments.

data [34], [42]. These generic features typically include edges, basic patterns, and color gradients. We anticipate that such an initialization of the CNN would produce a latent space where the distance between samples better represents the distance in the original feature space, thus providing an advantageous starting point of learning the GP kernel. In the following, the pre-training methods are adapted to be used in settings either with or without transfer learning. With the proposed adaptations, we would be able to investigate the effectiveness of various components through experiments.

2) Convolutional Autoencoder: The autoencoder (AE) is a well-known unsupervised representation learning method for dimensionality reduction. It consists of an encoder and a decoder. The encoder maps the inputs to some lower-dimensional latent space, whereas the decoder reconstructs the inputs from the latent representations, which ensures that the encoder has learned the most relevant features. Convolutional Autoencoders (CAEs) are a special case of AEs in that the convolutional filters are reused among different locations of the input to preserve the spatial locality [43].

Normally, CAEs have several convolutional layers in the encoder and transposed convolutional layers in the decoder. To enable transfer learning in CAEs, we propose to use the CNN backbone as the encoder. And we construct a symmetric decoder using transposed convolutional layers. In such a way, the decoder can have a similar model capacity to that of the encoder. Formally, after getting the latent representations $\boldsymbol{h}_i$ from the encoder $f_{\boldsymbol{\Phi}}(\cdot)$, we feed it into the decoder $g_{\boldsymbol{\Psi}}(\cdot)$ to reconstruct the original images

$$\begin{aligned} g_{\boldsymbol{\Psi}} : \mathbb{R}^h &\to \mathbb{R}^{n_{\mathrm{H}} \times n_{\mathrm{W}} \times n_{\mathrm{C}}} \\ \boldsymbol{h}_i &\mapsto g_{\boldsymbol{\Psi}}(\boldsymbol{h}_i) =: \hat{\boldsymbol{X}}_i, \end{aligned}$$

where $\boldsymbol{\Psi}$ denotes the trainable parameters in the decoder network. The parameters of the encoder $\boldsymbol{\Phi}$ can be initialized from scratch or from models using transfer learning. The training of the CAE involves minimizing the Mean Squared Error (MSE) between the original image sample $\boldsymbol{X}_i$ and the reconstructed image $\hat{\boldsymbol{X}}_i$. Formally, the loss function is defined as

$$\mathcal{J}_{\mathrm{CAE}} = \frac{1}{n} \sum_{i=1}^{n} \| \boldsymbol{X}_i - \hat{\boldsymbol{X}}_i \|_2^2, \tag{3}$$

where $\| \cdot \|_2$ denotes the Euclidean norm.

3) Deep Metric Learning: Taking advantage of the structure of twin neural networks (replications of the same NN), CNN backbones are applied in DML for learning the latent representations so that samples with similar labels would be mapped closer to each other in the latent space. The latent representations from the trained backbones turn out to be effective for tasks like face verification [44], [45] or person re-identification [46].

Given an image sample $\boldsymbol{X}_i$ from the dataset, the backbone defines a function $f_{\boldsymbol{\Phi}}(\cdot)$ to embed it in some latent space. Formally, we have

$$\begin{aligned} f_{\boldsymbol{\Phi}} : \mathbb{R}^{n_{\mathrm{H}} \times n_{\mathrm{W}} \times n_{\mathrm{C}}} &\to \mathbb{R}^h \\ \boldsymbol{X}_i &\mapsto f_{\boldsymbol{\Phi}}(\boldsymbol{X}_i) =: \boldsymbol{h}_i, \end{aligned}$$

where $\boldsymbol{\Phi}$ denotes the trainable parameters in the network and h is the dimension of the latent space. The trainable parameters $\boldsymbol{\Phi}$ can be either initialized from scratch or be transferred from models trained on other large-scaled datasets, e.g., the ImageNet dataset.

With the class label information, a triplet is defined to consist of an anchor sample $\boldsymbol{X}_i^{\mathrm{A}}$, a positive sample $\boldsymbol{X}_i^{\mathrm{P}}$, and a negative sample $\boldsymbol{X}_i^{\mathrm{N}}$, where the anchor is of the same class as the positive and the negative is not. However, in a regression problem, the target variables cannot define the triplets directly since they are continuous values. To mitigate this, we propose to categorize the target variables into classes to generate triplets in DML. It can also be viewed as a coarse pre-training step before the final fine-tuning from a learning perspective. Concretely speaking, we categorize target variables according to their binning in the histogram for univariate regression tasks and apply K-means clustering to find a class label for the target variables in multivariate regression tasks.

During training, minimizing the triplet margin loss makes the anchor-positive distance smaller than the anchor-negative distances by a certain margin [45]. Formally, the loss function is defined as

$$\mathcal{J}_{\text{triplet}} = \sum_{i=1}^{n} \left[\mathrm{d}(\boldsymbol{h}_i^{\mathrm{A}}, \boldsymbol{h}_i^{\mathrm{P}}) - \mathrm{d}(\boldsymbol{h}_i^{\mathrm{A}}, \boldsymbol{h}_i^{\mathrm{N}}) + \alpha \right]_+, \tag{4}$$

where $\mathrm{d}(\cdot, \cdot)$ is a distance metric, e.g., the Euclidean distance, α is the value of a pre-defined margin, and $[\cdot]_+$ only takes the positive part of the variable. Also, triplet selection is an important step to get fast convergence of the training since the network only gets gradients from the triplets having positive values in Equation (4). Within each mini-batch, we pick negative samples whose distance to the anchor is larger than the anchor-positive distance (within a margin of α). That means we have

$$0 < \mathrm{d}(\boldsymbol{h}_i^{\mathrm{A}}, \boldsymbol{h}_i^{\mathrm{N}}) - \mathrm{d}(\boldsymbol{h}_i^{\mathrm{A}}, \boldsymbol{h}_i^{\mathrm{P}}) < \alpha,$$

which are regarded as *semi-hard* examples in [45].

To find an appropriate number of epochs for the pre-training, we take advantage of early stopping methods, which terminate the training automatically by monitoring specific metrics derived from the validation set. Here, we use the metric Mean Average Precision at R (MAP@R), a more informative evaluation metric since it combines the ideas of Mean Average Precision and R-precision [36].

We argue that the training objective of the DML agrees with the paradigm of a GP regression using localized kernels, which is to interconnect the similarity of data samples in the target space to the similarity in their input spaces. Since GP cannot directly operate in the raw pixel space, a mapping function that preserves the similarity from the target space to the latent space would provide the GP with an ideal input space.

C. End-to-end Fine-tuning Deep Kernel Learning

In this section, we elaborate our proposed method in Algorithm 1 by inversely joining the modules that have been introduced in the last two sections.

Algorithm 1: Fine-tuning Deep Kernel Learning

Input: An image dataset of the form $\{\boldsymbol{X}_i, y_i\}_{i=1}^n$.
Output: A fine-tuned DKL model for regression

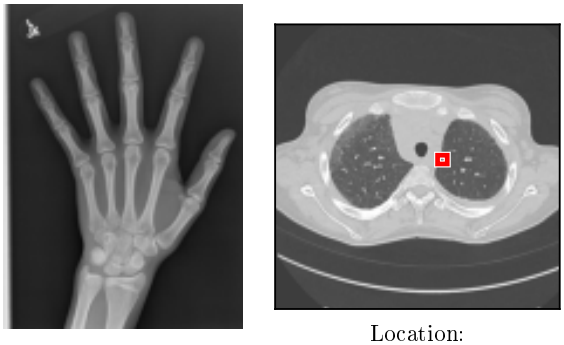
1 **if** *Transfer is True* **then**
2 $\quad \boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi}^{\text{ImageNet}}$
3 **end**
4 **switch** *Pre-training is DML* **do**
5 $\quad$ Generate triplets $\{\boldsymbol{X}_i^{\text{A}}, \boldsymbol{X}_i^{\text{P}}, \boldsymbol{X}_i^{\text{N}}\}$
6 $\quad \boldsymbol{\Phi} \leftarrow \arg\min_{\boldsymbol{\Phi}} \mathcal{J}_{\text{triplet}}(\boldsymbol{\Phi})$
7 **end**
8 **switch** *Pre-training is CAE* **do**
9 $\quad \boldsymbol{\Phi}, \boldsymbol{\Psi} \leftarrow \arg\min_{\boldsymbol{\Phi},\boldsymbol{\Psi}} \mathcal{J}_{\text{CAE}}(\boldsymbol{\Phi}, \boldsymbol{\Psi})$
10 **end**
11 Initialize the inducing points $\{\boldsymbol{Z} | \boldsymbol{z}_i^{\text{init}} = f_{\boldsymbol{\Phi}}(\boldsymbol{X}_i)\}$
12 $\boldsymbol{\Phi}, \boldsymbol{\Theta} \leftarrow \arg\max_{\boldsymbol{\Phi},\boldsymbol{\Theta}} \mathcal{L}_{\text{PPGP}}(\boldsymbol{\Phi}, \boldsymbol{\Theta})$
13 **return** $\boldsymbol{\Phi}, \boldsymbol{\Theta}$

Depending on whether we want to reuse the model trained on the ImageNet dataset, we will initialize the parameters in the backbones from the transferred model or from scratch (line 2). If we use DML as pre-training, we first categorize the target variables to generate triplets with the class information (line 5). Then the backbones are trained with triplet margin loss in Equation (4) (line 6). On the other hand, if we use CAE as pre-training, the parameters in the encoder and decoder are trained jointly against the CAE loss in Equation (3) (line 9), where the encoder will be used as the backbone in later steps. After the pre-training, the backbone is used to initialize the inducing points with a subset of the image samples (line 11). Therefore, it is worth highlighting that the pre-training step affects the parameters in the neural network and the parameters in the SVGP-based output layer. Finally, the fine-tuning step is done w.r.t. the respective ELBO objective, where we use the PPGP objective in Equation (2) as an example (line 12).

IV. Experiments

A. Datasets and Implementation Details

We have conducted experiments with two different datasets to validate our proposed methods. As an example for univariate regression tasks, we included the Bone Age Prediction (BAP) task from the Radiological Society of North America Pediatric Bone Age Machine Learning Challenge [47], [48]. In this dataset, there are $14,236$ hand radiographs, where the target variable is defined as the bone age of pediatric patients under five years old. In addition, we included the lesion localization (LL) task from the DeepLesion dataset [49] as an example for the multivariate regression problem. In the original dataset, there are $32,120$ axial computed tomography (CT) slices from $4,427$ unique patients. Together with the tag information from LesaNet [50], we retrieve $7,310$ slices for the lesion type of lung, where the task is to localize the lesion in a given CT image. The target is in the format of $(x_{1n}, y_{1n}, x_{2n}, y_{2n})$, where $x_{1n}, y_{1n}, x_{2n}, y_{2n}$ denote the normalized x-top-left, y-top-left, x-bottom-right, and y-bottom-right, respectively. Fig. 3 shows examples for the task of BAP and LL.



Month: 180

Location:
(0.57, 0.46, 0.60, 0.49)

Fig. 3. An example for the task of Bone Age Prediction (left) and Lesion Localization (right).

The CNN-related models are built using the *PyTorch* package [51], where the models with GP methods are implemented with the help of the *GPyTorch* package [52]. We conducted cross-validations (CV) for both tasks with 90% samples extracted in the datasets, where hyperparameters are tuned according to the performance on the validation set. The remaining unseen 10% samples constitute the test set, from which the results reported in the following sections are computed. Common data augmentations, including random crop, rotate, and horizontal flip, are applied. Due to the relatively small size of the datasets, the backbone of ResNet18 and DenseNet121 are chosen in all experiments. Related scripts[2] of the work will be published to improve the reproducibility.

B. Evaluation Approaches and Baselines

Due to the probabilistic nature of our proposed model, we considered two lines of evaluation approaches in the experiments: the performance of point estimates and the evaluation of predictive variances. We used the well-known Root Mean Squared Error (RMSE) to reflect the prediction performance for the former, where only the mean predictions of the proposed model are involved in the evaluation. For the latter, we included a novel method to validate the meaningfulness of the predictive uncertainty, namely a quantile performance (QP) plot. Intuitively speaking, a good uncertainty-aware model should demonstrate better performance together with higher confidence in its predictions and vice versa. Any uncertainty-aware regression model that produces point estimates and predictive variances can be evaluated against this criterion.

Given an uncertainty-aware model $f(\cdot)$ and its predictive distribution $f_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, we first sort all predictive vari-

[2]Related scripts see https://github.com/ZhiliangWu/mDKL.

ances in an ascending order $\{\sigma_i^2 \mid \sigma_i^2 \leq \sigma_{i+1}^2, i \in \{1, \ldots, n\}\}$ and then compute K quantiles denoted as $q_1, \ldots, q_K$.

Second, we compute the RMSE evaluation[3] of the subset of predicted point estimates $\hat{y}_i := \mu_i$, whose paired variances σ_i^2 are smaller than or equal to each of the k-th quantile values of $k \in \{1, \ldots, K\}$:

$$\text{Performance}_k := \text{RMSE}(\{(y_i, \hat{y}_i) | \forall \sigma_i^2 \leq k\text{-quantile}\}),$$

where $(y_i, \hat{y}_i)$ denotes the evaluation pair. By plotting the performance value on the y-axis against the corresponding quantile value k on the x-axis, a monotonically increasing line is expected.

To study the point estimate performance of the SVGP-based output layers, models having the same backbone but with a linear layer, which is optimized directly w.r.t. MSE, are included as baselines. Besides, when investigating the effects of pre-training between various representation learning methods, the models without any pre-training serve naturally as baselines. For the performance of predictive variances, we included MC Dropout [10], a popular method for augmenting uncertainty in the neural networks, as a baseline. In MC Dropout, a dropout layer [53] is added before each layer in the network. In our experiments, we used the default dropout setting for DenseNet121 with a dropout rate of 0.2 during training and testing, whereas dropout layers with the same dropout rate are added after each of the four layers of residual blocks in ResNet18. The predicted value and predictive variances are computed by performing 50 stochastic forward passes through the network as suggested in [53].

C. Evaluation on the Bone Age Prediction

1) Results on Point Estimates: Tab. I demonstrates the performance of the proposed method with DenseNets121 on the univariate regression task, Bone Age Prediction. Our proposed models with SVGP-based output layers deliver competitive or, in most cases, even better performances compared to common architectures with linear layers. Overall, the proposed model with SVGP output layers using transfer learning and pre-trained with DML demonstrates the best performance. In addition, significantly superior performance is found on models using the parameters transferred from models trained on the ImageNet dataset, which holds under all pre-training variants. Comparing the models using transfer learning w.r.t. different pre-training methods (upper part in Tab. I), we see an improvement when the model is first pre-trained with DML, whereas the pre-training with CAE does not improve the performance. However, for the models without transfer learning (lower part in Tab. I), both DML and CAE enhance the performance of the models, whereas CAE shows a better pre-training performance than DML. As a reference, [48] reports 10.44 and 7.8 as RMSE values on 200 test samples from human reviewers and model predictions, respectively.

2) Results on Predictive Variances: We include the models with transfer learning but without any pre-training for the QP plot in Fig. 5, where solid lines and error bars denote the means and standard deviations across different CV splits. A clear, monotonically increasing trend is observed in our proposed models with PPGP output layers and the models with MC Dropout. The line of PPGP is located underneath the MC Dropout, indicating better performance. In contrast, the models with SVGP output layers show a monotonically decreasing trend w.r.t. the quantile of the predictive variance. The models with PPGP output layers have an RMSE of 9.476 ± 0.200 (predictive variances at $q = 20\%$) for the samples they are more confident with, which is a relatively large improvement compared to the values reported in Tab. I.

3) Discussion: The superior performance of models with SVGP-based output layers is expected since the ELBO objective is a proxy for the log marginal likelihood objective, which is a generalization to MSE in linear regression. The improvements based on transfer learning conform to its popularity in the computer vision community, which validates the hypothesis that reusing the knowledge from large-scale datasets could help solve a new problem even with domain shift. For pre-training with DML, the models are first trained to embed samples with similar targets into nearby regions, which would be a helpful initialization for the inducing points and non-convex optimization in neural networks. Therefore, we observe a positive contribution from DML to the model performance. For the pre-training with CAE, the goal is to learn a *compressed* representation, which will be recovered in another decoder network. From the experimental results,

[3]This could be any metrics for evaluating point estimates.

TABLE I
BONE AGE PREDICTION WITH DENSENET121

Output Layer	Transfer Learning	RMSE (No pre-training)	RMSE (DML)	RMSE (CAE)
Linear*	Yes	12.118 ± 0.277	11.667 ± 0.231	14.076 ± 0.281
SVGP†	Yes	11.697 ± 0.102	$\mathbf{11.440 \pm 0.132}$	$\mathbf{13.536 \pm 0.279}$
PPGP†	Yes	$\mathbf{11.679 \pm 0.061}$	11.529 ± 0.089	13.694 ± 0.274
Linear*	No	19.934 ± 0.246	$\mathbf{15.805 \pm 0.157}$	15.340 ± 0.390
SVGP†	No	$\mathbf{17.723 \pm 0.298}$	15.832 ± 0.284	$\mathbf{15.323 \pm 0.411}$
PPGP†	No	18.341 ± 0.234	16.084 ± 0.336	15.752 ± 0.352

*Common architectures.
†With our proposed model.

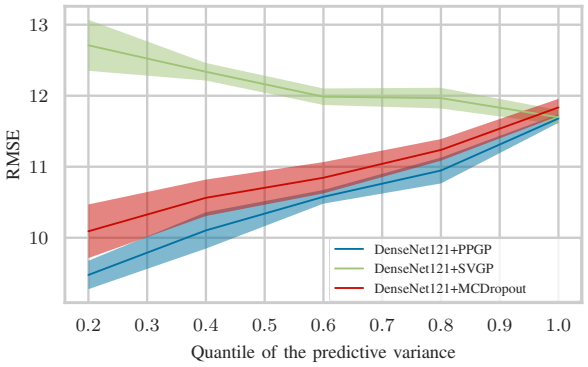


Fig. 4. Quantile Performance for the Bone Age Prediction with DenseNet121

such unsupervised representation learning would improve the performance if the model is trained from scratch but may deteriorate the knowledge transferred from large-scale datasets. It indicates a possibly higher correlation of the current task to the ImageNet classification than the compression task from CAE, which could be attributed to the large number of training samples in the ImageNet dataset. It is also worth mentioning that we also conducted the same experiments with the ResNet18 backbone, where similar results are observed. More details can be found in Appendix A and B.

What makes our proposed method appealing lies in the probabilistic nature of its prediction. The principled predictive variance from PPGP output layers is expected since 1) the inducing points technique facilitates explicit modeling of uncertainty, 2) the symmetric treatment of the predictive variance is restored in the training phase compared with SVGP. However, the good performance from MC Dropout also comes with a considerable computational cost. Roughly speaking, the inference time would be t times as much as our proposed model, where t is the number of stochastic forward passes for the inference. With $t = 50$, our experiment with 1424 test samples requires an inference time of 1777.04 seconds (almost half an hour) for the MC Dropout method, whereas our SVGP-based approach takes only 42.90 seconds. With more samples or more advanced backbone structures, the time cost will be more expensive for the MC Dropout method. These observations indeed motivate the application of our proposed models with PPGP output layers when meaningful predictive variances and low time complexity come to a higher priority in a real-world system.

D. Evaluation on the Lesion Localization

1) Results on Point Estimates: Tab. II shows the performance of our proposed method with DenseNet121 on the multivariate regression task, Lesion Localization. Similar to the task of BAP, our proposed models with SVGP-based output layers have mostly better performance than the common architectures with linear layers, and models with transfer learning outperform the ones without it by a large margin. On the whole, the proposed model with PPGP output layers demonstrates the best results under all settings. The difference lies in the performance with pre-training methods. Both pre-training methods only improve the performance in the settings without transfer learning.

2) Results on Predictive Variances: Due to the multivariate setting in this task, the mean of the predictive variances of different target variables is first computed before quantifying the predictive variances. Like the BAP task, a monotonically increasing trend is observed in models with PPGP and MC Dropout. The line of PPGP overlaps mostly with the one with MC Dropout, indicating relatively similar performance. In contrast, models with SVGP output layers deliver an almost flat trend w.r.t. the quantiles of the predictive variance. The models with the PPGP output layers have RMSE of 0.046 ± 0.012 for the evaluation pairs they are more confident with (predictive variance at $q = 20\%$), which is less than one-half of the ones reported in Tab. II.

TABLE II
LESION LOCALIZATION WITH DENSENET121

Output Layer	Transfer Learning	RMSE (No pre-training)	RMSE (DML)	RMSE (CAE)
Linear*	Yes	0.102 ± 0.002	0.101 ± 0.002	0.102 ± 0.003
SVGP†	Yes	0.099 ± 0.003	0.101 ± 0.003	0.104 ± 0.004
PPGP†	Yes	$\mathbf{0.098 \pm 0.002}$	$\mathbf{0.098 \pm 0.002}$	$\mathbf{0.099 \pm 0.002}$
Linear*	No	0.116 ± 0.001	0.114 ± 0.003	0.114 ± 0.002
SVGP†	No	0.118 ± 0.002	0.114 ± 0.003	0.112 ± 0.003
PPGP†	No	$\mathbf{0.115 \pm 0.002}$	$\mathbf{0.111 \pm 0.005}$	$\mathbf{0.110 \pm 0.002}$

*Common architectures.
†With our proposed model.

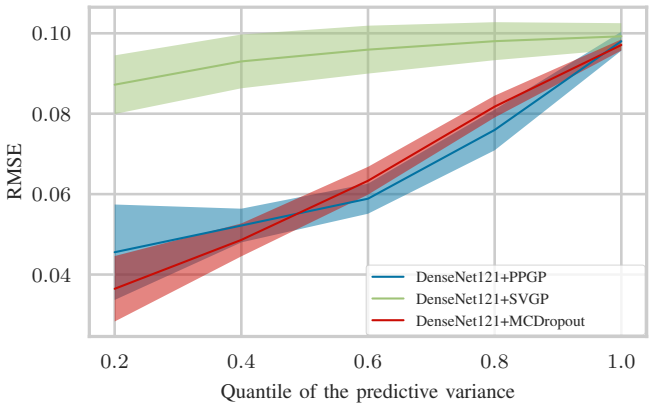


Fig. 5. Quantile Performance for the Lesion Localization with DenseNet121

3) Discussion: Most observations and discussions in the univariate regression still hold in the multivariate task. The only difference lies in the performance of DML, where it does not improve the performance in the setting with transfer learning. The smaller size of the dataset compared to the one in the BAP task could be one possible reason. Another possible explanation is that the task's multivariate nature makes it hard to define a suitable space for DML to facilitate DKL. Further improving the DKL performance in a multivariate setting of DML would be an exciting direction for our future work. Like the BAP task, we also conducted the same experiments with the backbone of the ResNet18. More details can be found in Appendix A and B.

V. CONCLUSIONS

This manuscript addresses the challenge that deep neural networks (DNNs) are often unable to provide uncertainty estimates for their predictions in regression tasks. Especially in the healthcare domain, this issue could prevent the further application of DNNs. We propose a model that consists of a deep Convolutional Neural Network (CNN) and a sparse Gaussian Process (GP). The former part serves as a trainable feature extractor that embeds raw images into a latent space. This enables the latter part to model the similarity of all

sample pairs with localized kernels more effectively in order to produce a predictive distribution for each data sample.

We show that such an architecture can be trained in an end-to-end fashion using stochastic gradient descent (SGD). We also analyzed multiple ways to boost the performance of such a model with different initialization and pre-training methods. Our approach is by no means limited to Convolutional Neural Networks, but could be generalized to other kinds of neural networks that best fit the nature of the data. We also observe a new specific challenge in this setup: We observe that randomly initialized inducing points in a sparse GP cause the prediction to degenerate to its prior when it consumes outputs from CNN backbones. We propose a simple solution that could also encourage further research in the task of jointly learning representations and GPs. Our experiments on the Bone Age Prediction and Lesion Localization tasks show that the proposed model delivers mostly better performance in terms of point estimates if compared to the baselines with a linear output layer. More importantly, we show that our model's prediction performance increases hand-in-hand with its predictive certainty. In other words, given a difficult test sample, our model can realize and communicate that the prediction thereof might be less trustworthy by generating a larger predictive variance. Finally, our model requires significantly less computational cost than popular MC Dropout methods, which motivates its usage in real-world online applications.

As future work, we are interested in studying the integration of multiple sparse GPs to deal with different types of input sources and model the relations between outputs. In addition, a combination of the interpretation using our uncertainty-aware model with the explainability methods like various saliency methods [54], [55] would be an exciting direction for exploration.

ACKNOWLEDGMENT

The authors acknowledge the support from Yan Ke on the Deep Lesion dataset [49] and the support by the German Federal Ministry for Education and Research (BMBF), funding project "MLWin" (grant 01IS18050).

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

REFERENCES

[1] V. Tresp, J. M. Overhage, M. Bundschus, S. Rabizadeh, P. A. Fasching, and S. Yu, "Going digital: a survey on digitalization and large-scale data analytics in healthcare," *Proceedings of the IEEE*, vol. 104, no. 11, pp. 2180–2206, 2016.

[2] C. Xiao, E. Choi, and J. Sun, "Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review," *Journal of the American Medical Informatics Association*, vol. 25, no. 10, pp. 1419–1428, 2018.

[3] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436.

[4] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1321–1330.

[5] J. Zhang, B. Kailkhura, and T. Y.-J. Han, "Mix-n-match: Ensemble and compositional methods for uncertainty calibration in deep learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 117–11 128.

[6] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 694–699.

[7] C. M. Bishop, "Mixture density networks," Aston University, Tech. Rep., 1994.

[8] B. Kompa, J. Snoek, and A. L. Beam, "Second opinion needed: communicating uncertainty in medical machine learning," *NPJ Digital Medicine*, vol. 4, no. 1, pp. 1–6, 2021.

[9] A. L. Beam and I. S. Kohane, "Big data and machine learning in health care," *Jama*, vol. 319, no. 13, pp. 1317–1318, 2018.

[10] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[11] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 5580–5590.

[12] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[13] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2014.

[14] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *NIPS*, 2017.

[15] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, Jan. 2006.

[16] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When gaussian process meets big data: A review of scalable gps," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 11, pp. 4405–4423, 2020.

[17] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Artificial intelligence and statistics*. PMLR, 2016, pp. 370–378.

[18] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, "A guide to deep learning in healthcare," *Nature medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[19] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya *et al.*, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," *arXiv preprint arXiv:1711.05225*, 2017.

[20] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[21] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2097–2106.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[23] V. Tresp, "A bayesian committee machine," *Neural computation*, vol. 12, no. 11, pp. 2719–2741, 2000.

[24] C. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," in *Proceedings of the 14th annual conference on neural information processing systems*, no. CONF, 2001, pp. 682–688.

[25] C. K. Williams, C. E. Rasmussen, A. Scwaighofer, and V. Tresp, "Observations on the nyström method for gaussian process prediction," *University of Edinburgh*, 2002.

[26] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *Advances in neural information processing systems*, vol. 18, pp. 1257–1264, 2005.

[27] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 567–574.
[28] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," in *Uncertainty in Artificial Intelligence*. Citeseer, 2013, p. 282.
[29] J. Bradshaw, A. G. d. G. Matthews, and Z. Ghahramani, "Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks," *arXiv preprint arXiv:1707.02476*, 2017.
[30] A. Wilson and H. Nickisch, "Kernel interpolation for scalable structured gaussian processes (kiss-gp)," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1775–1784.
[31] M. Jankowiak, G. Pleiss, and J. Gardner, "Parametric gaussian process regressors," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4702–4712.
[32] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
[33] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.
[34] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, 2014, pp. 3320–3328.
[35] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.
[36] K. Musgrave, S. Belongie, and S.-N. Lim, "A metric learning reality check," in *European Conference on Computer Vision*. Springer, 2020, pp. 681–699.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[38] J. Hensman, A. Matthews, and Z. Ghahramani, "Scalable variational gaussian process classification," in *Artificial Intelligence and Statistics*. PMLR, 2015, pp. 351–360.
[39] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory*. Springer, 2001, pp. 420–434.
[40] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" in *International conference on database theory*. Springer, 1999, pp. 217–235.
[41] M. A. Álvarez, L. Rosasco, N. D. Lawrence *et al.*, "Kernels for vector-valued functions: A review," *Foundations and Trends® in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2012.
[42] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.
[43] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International conference on artificial neural networks*. Springer, 2011, pp. 52–59.
[44] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.
[45] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.
[46] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.
[47] S. S. e. a. Halabi, "The rsna pediatric bone age machine learning challenge," *Radiology*, vol. 290, no. 2, pp. 498–503, 2019.
[48] D. B. Larson, M. C. Chen, M. P. Lungren, S. S. Halabi, N. V. Stence, and C. P. Langlotz, "Performance of a deep-learning neural network model in assessing skeletal maturity on pediatric hand radiographs," *Radiology*, vol. 287, no. 1, pp. 313–322, 2018.
[49] K. Yan, X. Wang, L. Lu, and R. M. Summers, "Deeplesion: automated mining of large-scale lesion annotations and universal lesion detection with deep learning," *Journal of medical imaging*, vol. 5, no. 3, p. 036501, 2018.
[50] K. Yan, Y. Peng, V. Sandfort, M. Bagheri, Z. Lu, and R. M. Summers, "Holistic and comprehensive annotation of clinically significant findings on diverse ct images: learning from radiology reports and label ontology," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8523–8532.
[51] 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," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.
[52] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," *Advances in Neural Information Processing Systems*, vol. 31, pp. 7576–7586, 2018.
[53] 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*, vol. 15, no. 1, pp. 1929–1958, 2014.
[54] J. Gu, Y. Yang, and V. Tresp, "Understanding individual decisions of cnns via contrastive backpropagation," in *Proceedings of the Asian Conference on Computer Vision*. Springer, 2018, pp. 119–134.
[55] J. Gu, Z. Wu, and V. Tresp, "Introspective learning by distilling knowledge from online self-explanation," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

APPENDIX

A. Results on Point Estimates using ResNets

TABLE III
BONE AGE PREDICTION WITH RESNET18

Output Layer	Transfer Learning	RMSE (No pre-training)	RMSE (DML)	RMSE (CAE)
Linear*	Yes	13.131 ± 0.129	$\mathbf{12.419 \pm 0.098}$	13.842 ± 0.283
SVGP†	Yes	$\mathbf{12.632 \pm 0.149}$	12.567 ± 0.051	$\mathbf{13.375 \pm 0.151}$
PPGP†	Yes	12.899 ± 0.114	12.658 ± 0.048	13.640 ± 0.292
Linear*	No	19.766 ± 0.134	16.533 ± 0.138	16.849 ± 0.275
SVGP†	No	$\mathbf{19.090 \pm 0.286}$	$\mathbf{16.147 \pm 0.264}$	16.641 ± 0.275
PPGP†	No	20.166 ± 0.371	16.986 ± 0.196	$\mathbf{16.469 \pm 0.222}$

TABLE IV
LESION LOCALIZATION WITH RESNET18

Output Layer	Transfer Learning	RMSE (No pre-training)	RMSE (DML)	RMSE (CAE)
Linear*	Yes	0.110 ± 0.003	0.114 ± 0.001	0.111 ± 0.003
SVGP†	Yes	$\mathbf{0.102 \pm 0.002}$	0.107 ± 0.001	0.106 ± 0.003
PPGP†	Yes	0.103 ± 0.001	$\mathbf{0.104 \pm 0.001}$	$\mathbf{0.103 \pm 0.002}$
Linear*	No	0.148 ± 0.003	0.137 ± 0.002	0.123 ± 0.002
SVGP†	No	$\mathbf{0.129 \pm 0.002}$	$\mathbf{0.131 \pm 0.001}$	0.121 ± 0.002
PPGP†	No	0.133 ± 0.002	0.134 ± 0.002	$\mathbf{0.119 \pm 0.002}$

*Common architectures.
†With our proposed model.

B. Results on Predictive Variances using ResNets

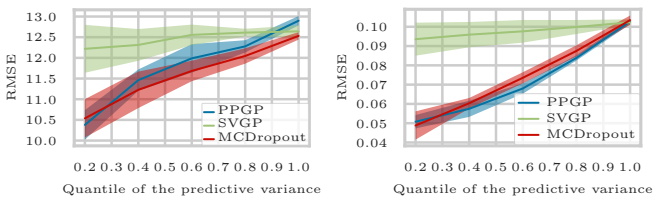


Fig. 6. Quantile Performance for tasks of Bone Age Prediction (left) and Lesion Localization (right) using ResNet18 as the backbone in our model

Chapter 4

Uncertainty-Aware Models in Survival Analysis

Uncertainty-Aware Time-to-Event Prediction using Deep Kernel Accelerated Failure Time Models

Zhiliang Wu ZHILIANG.WU@SIEMENS.COM
Ludwig Maximilians University Munich
Siemens AG, Technology, Munich

Yinchong Yang YINCHONG.YANG@SIEMENS.COM
Siemens AG, Technology, Munich

Peter A. Fasching PETER.FASCHING@UK-ERLANGEN.DE
Department of Gynecology and Obstetrics
University Hospital Erlangen, Erlangen

Volker Tresp VOLKER.TRESP@SIEMENS.COM
Ludwig Maximilians University Munich
Siemens AG, Technology, Munich

Abstract

Recurrent neural network based solutions are increasingly being used in the analysis of longitudinal Electronic Health Record data. However, most works focus on prediction accuracy and neglect prediction uncertainty. We propose Deep Kernel Accelerated Failure Time models for the time-to-event prediction task, enabling uncertainty-awareness of the prediction by a pipeline of a recurrent neural network and a sparse Gaussian Process. Furthermore, a deep metric learning based pre-training step is adapted to enhance the proposed model. Our model shows better point estimate performance than recurrent neural network based baselines in experiments on two real-world datasets. More importantly, the predictive variance from our model can be used to quantify the uncertainty estimates of the time-to-event prediction: Our model delivers better performance when it is more confident in its prediction. Compared to related methods, such as Monte Carlo Dropout, our model offers better uncertainty estimates by leveraging an analytical solution and is more computationally efficient.

1. Introduction

Since the introduction of the Electronic Health Record (EHR), an exploding amount of healthcare-related data has been collected in clinics. The physicians often become overwhelmed by data volume and data complexity and may turn to data-driven clinical decision support systems (Halpern et al., 2016; Tresp et al., 2016; Xiao et al., 2018). These solutions often provide decision support in two ways. A *prescriptive* system generates action recommendations, such as medications and therapy plans, while a *predictive* system provides physicians with a prediction of the outcome, given a decision. Such outcome could be, for instance, the adverse events related to a specific therapy or the progression-free-survival time after treatment. These predictions are often based on modeling the three-way interaction between the outcome, the patients' status, and clinical decisions recorded in historical

data. In this work, we address the prediction of treatment outcome and propose a new class of uncertainty-aware models that can communicate uncertainty to physicians. We argue that such uncertainty estimates add transparency and trustworthiness to the clinical decision support systems and encourage their application on even larger scales.

Due to the high-dimensional, sparse, and sequential nature of EHR data, simpler, more transparent white-box models often fail to capture the complex interactions between the target variable and input features. Meanwhile, recurrent neural network (RNN)-based solutions have proven capable of addressing the longitudinal aspect in EHR data (Esteban et al., 2016; Choi et al., 2017; Yang et al., 2017b; Purushotham et al., 2018; Wu et al., 2020). These models apply RNNs to aggregate historical observations to produce an individual and time-dependent representation of a patient. The last layer is then a linear map from patient representation to the target variable representing, e.g., the predicted outcome for the patient. The advantage of an RNN is twofold. First, it can handle records that vary in length from patient to patient, as in the analysis of texts with varying lengths (Mikolov et al., 2012). Second, the more advanced RNN variants such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Chung et al., 2014) are flexible in memorizing both long-term and short-term input features. Despite the state-of-the-art predictive performance of the neural network (NN)-based methods, these methods fail to provide reasonable uncertainty estimates of the predictions (Nguyen et al., 2015). It is easier to address this issue in classification tasks, as the predicted probability can be interpreted to reflect uncertainty, which leads to various calibration approaches, like temperature scaling (Guo et al., 2017). However, for regression tasks—like the time-to-event prediction task in our case—one has to provide a predictive distribution to address the uncertainty estimates, which is rarely considered in vanilla NN-based solutions. In healthcare applications, we would argue that uncertainty-awareness of the model is as important as point estimate performance, since the uncertainty estimates would assist the physicians in better interpreting the results from a black-box model (Begoli et al., 2019). If the model provides a high uncertainty estimate for its prediction, the physicians would be more careful about it and take a closer look at that case.

It is, however, not a trivial task to augment NNs with reliable uncertainty estimates. Currently, popular choices to do so include MC Dropout methods (Gal and Ghahramani, 2016), Bayesian neural networks (Bishop, 2006) and deep ensembles (Lakshminarayanan et al., 2017). Many variants of them involve repeated sampling procedures either during training or during inference, which could become computationally expensive for large NNs. In this work, we investigate the possibility of quantifying the predictive uncertainty by applying Gaussian Processes (GPs). As a popular class of machine learning methods, GP produces a predictive distribution instead of a single point estimate for each test sample. For small datasets, GPs have proven to be flexible and data-efficient. However, GPs' inclusion of a large training dataset inevitably introduces large storage and computational complexity (Rasmussen and Williams, 2005). With the recent advance of sparse GP techniques, the computational complexity has been largely reduced (Liu et al., 2020). Motivated by the desirable predictive distribution of GPs and the progress of sparse GPs, we propose in this work a novel approach by integrating RNNs as feature extractors into GP-based predictive models. Furthermore, we propose a deep metric learning (DML)-based pre-training method to further improve the performance of the model.

In the context of time-to-event prediction, our proposed model is closely related to the Accelerated Failure Time (AFT) models (Prentice, 1978; Kalbfleisch and Prentice, 2002). As one of the well-known models in survival analysis, the AFT models have shown promising results, especially when, in an application, the direct prediction of survival time is more important than hazard estimation. With our proposed method, we have augmented the uncertainty estimates in the AFT models.

Generalizable Insights about Machine Learning in the Context of Healthcare

Neural networks offer powerful modeling ability to learn from EHR data, but often neglect the uncertainty estimates in the predictions. Leveraging state-of-the-art sparse GPs, we propose a method to integrate the uncertainty into the NN-based solutions for time-to-event prediction tasks. Experiments on two real-world datasets show that the resulting model can 1) scale very well to large datasets; 2) deliver improved performance regarding point estimates; 3) offer reasonable predictive variances, which reflect the confidence of the predictions and enhance the calibration of the model. The uncertainty estimates in our model can help establish a trustworthy relationship with physicians since it expresses higher confidence in more accurate predictions and vice versa.

2. Related Work

Predictive Modeling with EHRs To better capture the time-dependent information of patients, many works have been proposed to model longitudinal EHR data, ranging from earlier statistical methods like *landmarking* (Van Houwelingen, 2007), *Joint Models* (Rizopoulos, 2011; Hickey et al., 2016), to more recent methods like *Bayesian Nonparametric Dynamic Survival* (Bellot and Schaar, 2020). Meanwhile, RNN-based approaches have proved to be very successful both for discrete medical events and continuous time-series data. Esteban et al. (2016) applied sequence-to-sequence RNN models to predict discrete medical events of patients suffering from kidney failure. Yang et al. (2017b) proposed many-to-one RNN models to deal with discrete medical events and predict the therapy decision for breast cancer. At the same time, Choi et al. (2017) applied models of similar structure for the early detection of heart failure onset. For continuous time-series data, multiple readings of individual signals are usually aggregated to reduce the high-resolution patient data so that they can be better consumed by the neural networks, e.g., heart rate and blood pressure in ICU time-series data (Johnson et al., 2016). With the aggregation method on the ICU time series data, Purushotham et al. (2018) provided RNN-based benchmarks for the mortality prediction, length-of-stay prediction, and ICD-9 code group prediction. Following similar data pre-processing steps, Wu et al. (2020) presented RNN-based models to learn the optimal treatment strategies for administering intravenous fluids and vasopressors. In this work, we include EHR data with discrete medical events as well as the dataset with continuous time-series measurements for the time-to-event prediction tasks to validate our proposed model.

Survival Analysis with Neural Networks and Gaussian Processes As most popular approaches in survival analysis are based on generalized linear models, they have been extended to nonlinear models, including NNs and GPs. Saul (2016) proposed chained GPs

to model multiple parameters of the log-logistic likelihood in AFT models through the latent functions of GPs. In addition, many GP-based methods are proposed to enhance the Cox proportional hazards (CPH) model, including the Bayesian semi-parametric model (Fernández et al., 2016) and deep multi-task Gaussian process DMGP (Alaa and van der Schaar, 2017). For NN-based approaches, Yang et al. (2017a) combines tensorized RNN model with the AFT model to predict progression-free survival (PFS) time. Kvamme et al. (2019) proposes an extension of CPH models, *CoxTime*, by parametrizing the relative risk function with NNs as well as modeling interactions between covariates and time. However, most of these works focus on capturing the non-linearity between covariates to improve the performance of point estimates. The uncertainty perspective of the prediction is rarely addressed. More recently, Chen (2020) proposed Deep Kernel Survival Analysis to learn kernel functions for the conditional Kaplan-Meier estimator and enables subject-specific survival time prediction intervals. The uncertainty is quantified by the prediction intervals. With an emphasis on uncertainty-awareness, we explore in this work the time-to-event prediction tasks with the AFT models using a combination of RNNs and GPs.

Exact Gaussian Process and Scalable Variational Gaussian Process with Neural Networks Since the capacity of GPs grows with available training data, many works have proposed possible solutions for both exact GP and sparse GP. Based on the efficient GP inference using Blackbox Matrix-Matrix multiplication from Gardner et al. (2018), Wang et al. (2019) realized exact GP training on over a million training samples by taking advantage of multi-GPU parallelization. Meanwhile, various works have been proposed to approximate the original GPs to save computational cost, including some early efforts, such as the Bayesian committee machine (BCM) (Tresp, 2000), the Nyström methods (Williams and Seeger, 2001), the Fully Independent Training Conditional (FITC) Approximation (Snelson and Ghahramani, 2006), Variational Free Energy (VFE) (Titsias, 2009), the more recent Scalable Variational Gaussian Process (SVGP) (Hensman et al., 2013) and Parametric Predictive Gaussian Process (PPGP) Regressor (Jankowiak et al., 2020) (more details see Sec. 3.2). In addition, the idea of combining sparse GPs with neural networks also received much attention, where the Deep Kernel Learning (DKL) (Wilson et al., 2016) and GP hybrid deep networks (GPDNN) (Bradshaw et al., 2017) are the ones most related to our work.

3. Methods

This section first discusses the RNN-based feature extractors to learn representations from the patients' static and sequential information. The resulting latent representations are used as inputs for the time-to-event prediction task. We will elaborate on our proposed model, which combines the GP-based models with AFT models. Afterward, a DML-based supervised pre-training method is presented to enhance the performance of the proposed model.

3.1. Recurrent Neural Networks as Feature Extractors

EHR data typically consist of *static features* and *sequential features*, both of which are important for the time-to-event prediction tasks. We regard the background information of each patient as static features $\boldsymbol{x}_i^{\text{sta}} \in \mathbb{R}^{\text{n}_{\text{sta}}}$. We use i, n_{sta} to denote the patient sample

index and the number of static features, respectively. In addition, features observed at all time-steps constitute the sequential feature matrix $\boldsymbol{X}_i^{\text{seq}} = [\boldsymbol{x}_i^0, \boldsymbol{x}_i^1, \ldots, \boldsymbol{x}_i^{t_i}]^\top \in \mathbb{R}^{t_i \times \text{n}_{\text{seq}}}$, where t_i is the number of observed time-steps for the i-th patient sample and n_{seq} denotes the number of sequential features. Due to the high sparsity or redundancy in raw features spaces, it has been shown to be beneficial to first apply a (non-linear) embedding layer on the raw features to learn the static hidden representation $\boldsymbol{h}_i^{\text{sta}}$ and sequential feature embeddings $\boldsymbol{X}_i^{\text{seq\_emb}}$ (Esteban et al., 2016). Formally, we have

$$\begin{aligned}\boldsymbol{h}_i^{\text{sta}} &= g_1(\boldsymbol{A}\boldsymbol{x}_i^{\text{sta}}) \in \mathbb{R}^{\text{n}_{\text{sta\_repr}}} \\ \boldsymbol{X}_i^{\text{seq\_emb}} &= g_2(\boldsymbol{X}_i^{\text{seq}}\boldsymbol{B}) \in \mathbb{R}^{t_i \times \text{n}_{\text{seq\_emb}}}\end{aligned}$$

where $\boldsymbol{A} \in \mathbb{R}^{\text{n}_{\text{sta\_repr}} \times \text{n}_{\text{sta}}}$, $\boldsymbol{B} \in \mathbb{R}^{\text{n}_{\text{seq}} \times \text{n}_{\text{seq\_emb}}}$ are embedding matrices, $g_1(\cdot), g_2(\cdot)$ are activation functions like $\tanh(\cdot)$, $\text{n}_{\text{sta\_repr}}, \text{n}_{\text{seq\_emb}}$ denote the dimension of the static hidden representations and the sequential feature embeddings, respectively. Afterward, more advanced variants of RNNs, LSTM or GRU, are used to encode the sequential feature embeddings $\boldsymbol{X}_i^{\text{seq\_emb}}$ into sequential latent representations $\boldsymbol{h}_i^{\text{seq}}$. Since we are mainly interested in modeling the time-to-event, only the last hidden states from LSTM/GRU are involved as inputs in the down-streaming tasks. Formally, we have

$$\boldsymbol{h}_i^{\text{seq}} = \text{RNN}(\boldsymbol{X}_i^{\text{seq\_emb}}) \in \mathbb{R}^{\text{n}_{\text{seq\_repr}}},$$

where $\text{n}_{\text{seq\_repr}}$ is the dimension of sequential hidden representations and $\text{RNN}(\cdot)$ could be an LSTM or GRU.

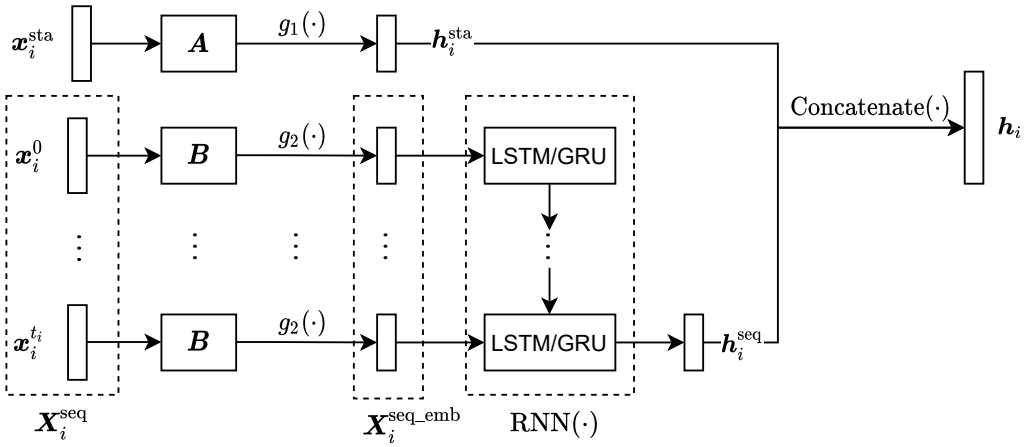


Figure 1: Illustration of the RNN-based feature extractor: An (non-linear) embedding layer is first involved in learning the sequential feature embeddings, which are then fed into RNN-based models to encode the sequential hidden representations. These are then concatenated with the static hidden representations. The complete hidden representation is expected to encode all relevant patient information and serves as abstract covariates for the down-streaming time-to-event prediction task.

The complete structure of the feature extractor is shown in Fig. 1. Similar to the encoder part in Cho et al. (2014), this procedure is especially appealing for the patients with different observed time-steps, as it is theoretically capable of storing all relevant information in the medical events with variable lengths but remains a consistent form of representations.

3.2. Scalable Variational Gaussian Processes for Time-to-Event Prediction

From the last section, we have learned the static hidden representation $\boldsymbol{h}_i^{\text{sta}}$ and the sequential hidden representation $\boldsymbol{h}_i^{\text{seq}}$ through RNN-based feature extractors. By concatenating them, we get the complete hidden representation as

$$\boldsymbol{h}_i = [\boldsymbol{h}_i^{\text{sta}}; \boldsymbol{h}_i^{\text{seq}}] \in \mathbb{R}^{\text{n}_{\text{sta\_repr}}+\text{n}_{\text{seq\_repr}}},$$

which can be viewed as abstract covariates of patients in a latent feature space.

The class of Accelerated Failure Time (AFT) models is a general class of models, where the covariates of the patients are assumed to act multiplicatively on the time-scale (Collett, 2015). Compared to the semi-parametric Cox proportional hazards (CPH) models, the AFT models take advantage of their parametric nature and include a wider range of survival time distributions. Formally, with the time-to-event target variable z_i, the AFT models predict its logarithm as $\log z_i := y_i = \boldsymbol{\beta}^\top \boldsymbol{h}_i + \epsilon_i$, where $\boldsymbol{\beta}^\top \boldsymbol{h}_i$ is the linear predictor with the (trainable) parameter vector $\boldsymbol{\beta}$, $\epsilon_i \overset{i.i.d.}{\sim} \mathcal{D}_\epsilon$ denotes the error term which is specified by a particular probability distribution $\mathcal{D}_\epsilon$. Common choices for $\mathcal{D}_\epsilon$ include Normal, Weibull and Logistic distributions, which correspondingly specifies the target variable z_i to be log-normal, log-weibull and log-logistic distributed, respectively. In this work, we assume our target variable of interest to follow a log-normal distribution. In other words, we have correspondingly $\epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{\text{obs}}^2)$. Furthermore, we propose to replace the linear predictor $\boldsymbol{\beta}^\top \boldsymbol{h}_i$ with Gaussian Process posterior prediction to enable uncertainty-aware predictions. In the following, we shall introduce this approach in detail.

As shown in Fig. 2, after we obtain the hidden representations $\boldsymbol{h}_i$ from the feature extractor $f_{\boldsymbol{\Phi}}(\cdot)$ (details see Sec. 3.1), these are used as abstract patient covariates for the

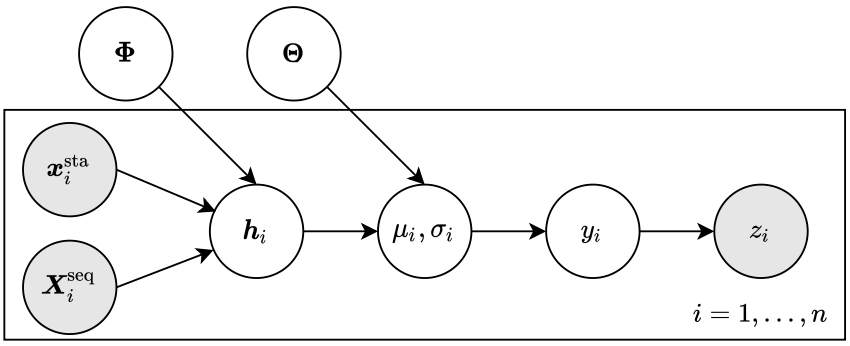


Figure 2: Graphical model of Deep Kernel Accelerated Failure Time models in plate notation. Nodes represent variables, where shaded ones are observable and non-shaded ones are latent. Plates indicate the repetition of the subgraph.

subsequent GP-based models $g_{\boldsymbol{\Theta}}(\cdot)$ to generate predictive distribution $y_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, the logarithm of the target variable. We denote $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ as the trainable parameters in the RNN-based feature extractor and the GP-based predictive model, respectively. Since we take advantage of neural networks with GP-based models as an advanced version of AFT models, we name it Deep Kernel Accelerated Failure Time (DKAFT) models. In the following, we will discuss different GP-based models in our proposed method.

In regression, y_i is a noisy observation of the GP function value $f_i = f(\boldsymbol{h}_i)$, which is assumed to behave a priori according to

$$p(\boldsymbol{f}|\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}),$$

where $\boldsymbol{f} = [f_1, \ldots, f_n]^\top \in \mathbb{R}^n$ is a vector of GP function values and $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ is a covariance matrix, whose entries are given by the covariance function $k_{ij} = k(\boldsymbol{h}_i, \boldsymbol{h}_j)$. The choice of the covariance function reflects the prior knowledge of the generative process of the model, where a Radial Basis Function (RBF) kernel is commonly used. There are some important hyper-parameters in the covariance function, e.g., the length-scale and signal variance in the RBF kernel, which can be learned through maximizing the log marginal likelihood defined as

$$\mathcal{L}_{\text{ExactGP}} = -\frac{1}{2}\boldsymbol{y}^\top \left(\boldsymbol{K} + \sigma_{\text{obs}}^2 I\right)^{-1} \boldsymbol{y} - \frac{1}{2}\log\left|\boldsymbol{K} + \sigma_{\text{obs}}^2 I\right| - \frac{n}{2}\log 2\pi. \tag{1}$$

With the optimized parameters, the prediction of a test sample f_* can be understood as computing the conditional probability of the test location given all values in the training dataset. Formally, the GP model outputs a predictive distribution as

$$f_* \sim \mathcal{N}(\boldsymbol{k}_*^\top \left(\boldsymbol{K} + \sigma_{\text{obs}}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}, k_{**} - \boldsymbol{k}_*^\top \left(\boldsymbol{K} + \sigma_{\text{obs}}^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_*), \tag{2}$$

where $\boldsymbol{k}_* = [k(\boldsymbol{h}_1, \boldsymbol{h}_*), \ldots, k(\boldsymbol{h}_n, \boldsymbol{h}_*)]^\top \in \mathbb{R}^n$ denotes the covariance function values between the training inputs and the test input $\boldsymbol{h}_*$. Please note that, in contrast to the original AFT formulation, the covariates $\boldsymbol{h}_i$ do not influence the logarithm of the target variable directly in GP. Instead, the accelerating effect is realized via the covariance function.

Equation 1 and Equation 2 reveal the training and inference step for our proposed DKAFT model with an *Exact GP output layer*. However, the Exact GP cannot scale well to a large-scale dataset due to the $\mathcal{O}(n^3)$ computational complexity from the inverse operations of the large covariance matrix $\boldsymbol{K}$.

A tremendous amount of work has been proposed to address the scalability issue in the Exact GP, where the techniques of inducing points with variational inference have found most interest (Quinonero-Candela and Rasmussen, 2005). In short, the inducing points constitute a "summary" dataset, which is learned to generalize the original dataset to reduce the $\mathcal{O}(n^3)$ computational complexity. They consist of inducing inputs $\{\boldsymbol{u}_i\}_{i=1}^m =: \boldsymbol{U}$ (corresponding to $\{\boldsymbol{h}_i\}_{i=1}^n$) and inducing variables $\{v_i\}_{i=1}^m =: \boldsymbol{v}$ (corresponding to $\{f_i\}_{i=1}^n$), where $m \ll n$. In the context of our DKAFT model, the inducing inputs refer to a summary of the abstract patient covariates in the latent space. The learning of the inducing points is facilitated by variational methods under different approximation assumptions, e.g., the prior approximation and posterior approximation (Liu et al., 2020).

Among various GP approximations, the Scalable Variational Gaussian Process (SVGP) proposed by Hensman et al. (2013) reduces the computational complexity to $\mathcal{O}(m^3)$ and

makes the training amenable to stochastic gradient descent (SGD)-based methods. More concretely, the variational distribution $q(\boldsymbol{v})$ of the inducing variables is assumed to follow a multivariate Normal distribution $\mathcal{N}(\boldsymbol{m}, \boldsymbol{S})$ in SVGP. Following the notation in Jankowiak et al. (2020), instead of the log marginal likelihood objective in an Exact GP, we optimize the parameters by maximizing the Evidence Lower BOund (ELBO)

$$\mathcal{L}_{\text{SVGP}} = \sum_{i=1}^{n} \left\{ \log \mathcal{N}\left(y_i \mid \mu_{\boldsymbol{f}}\left(\boldsymbol{h}_i\right), \sigma^2_{\text{obs}}\right) - \frac{\sigma^2_{\boldsymbol{f}}\left(\boldsymbol{h}_i\right)}{2\sigma^2_{\text{obs}}} \right\} - \text{KL}(q(\boldsymbol{v}) \| p(\boldsymbol{v})) \tag{3}$$

and the predictive distribution for each sample is

$$f_i \sim \mathcal{N}(\mu_{\boldsymbol{f}}(\boldsymbol{h}_i), \sigma^2_{\boldsymbol{f}}(\boldsymbol{h}_i)) = \mathcal{N}(\boldsymbol{k}_i^\top \boldsymbol{K}_{\boldsymbol{vv}}^{-1} \boldsymbol{m}, k_{ii} - \boldsymbol{k}_i^\top \boldsymbol{K}_{\boldsymbol{vv}}^{-1} \boldsymbol{k}_i + \boldsymbol{k}_i^\top \boldsymbol{K}_{\boldsymbol{vv}}^{-1} \boldsymbol{S} \boldsymbol{K}_{\boldsymbol{vv}}^{-1} \boldsymbol{k}_i). \tag{4}$$

$\boldsymbol{K}_{\boldsymbol{vv}} \in \mathbb{R}^{m \times m}$ is the covariance matrix of the inducing variables, whose entries are computed based on inducing inputs as $k_{ij}^{\boldsymbol{vv}} = k(\boldsymbol{u}_i, \boldsymbol{u}_j)$, $\boldsymbol{k}_i = [k(\boldsymbol{u}_1, \boldsymbol{h}_i), \ldots, k(\boldsymbol{u}_m, \boldsymbol{h}_i)]^\top \in \mathbb{R}^m$ is the covariance function values between all inducing inputs with the sample input $\boldsymbol{h}_i$, and $\text{KL}(\cdot\|\cdot)$ denotes the Kullback–Leibler divergence between two distributions.

Both training and inference of SVGP in Equation 3 and Equation 4 are more computationally tractable, since they only involve the inducing points instead of the whole dataset as in Exact GP. We can therefore take advantage of this formulation for large-scale datasets. With $\mathcal{L}_{\text{SVGP}}$ as an objective, we have our DKAFT model with an *SVGP output layer*.

More recently, Jankowiak et al. (2020) found that the predictive uncertainty from SVGP is dominated by the input-independent observational noise σ^2_{obs}, whereas it is indeed the input-dependent function variance $\sigma^2_{\boldsymbol{f}}(\boldsymbol{h}_i)$ that makes the GP posteriors attractive. Different from the SVGP objective in Equation 3, the Parametric Predictive Gaussian Process (PPGP) Regressor takes advantage of the predictive distribution in Equation 4 and embeds it directly in the objective using Maximum Likelihood Estimation (MLE) methods. Formally, the objective in PPGP is defined as

$$\mathcal{L}_{\text{PPGP}} = \sum_{i=1}^{n} \log \mathcal{N}\left(y_i \mid \mu_{\boldsymbol{f}}\left(\boldsymbol{h}_i\right), \sigma^2_{\text{obs}} + \sigma^2_{\boldsymbol{f}}\left(\boldsymbol{h}_i\right)\right) - \text{KL}(q(\boldsymbol{v}) \| p(\boldsymbol{v})). \tag{5}$$

With $\mathcal{L}_{\text{PPGP}}$ as a training objective, we have our DKAFT model with *an PPGP output layer*.

The objectives introduced above are defined for samples with observed time-to-event. For right-censored cases, we can take advantage of the parametric predictive distribution in Equation 4 to compute the survival function, whose logarithm contributes to the final objective together with the ELBO objective. Such an optimization objective is also used in AFT models, where the non-censored cases contribute to the objective through their respective probability distribution function and the censored ones through the survival function (see Collett, 2015, Equation 5.9). Formally, the survival function of the log-normal distribution in our DKAFT model is

$$S(z|\boldsymbol{h_i}) = 1 - \Phi\left(\frac{\log z - \mu_{\boldsymbol{f}}(\boldsymbol{h}_i)}{\sigma_{\boldsymbol{f}}(\boldsymbol{h}_i) + \sigma_{\text{obs}}}\right),$$

where $\Phi(\cdot)$ is the cumulative distribution function of a standard normal distribution.

3.3. Deep Metric Learning as Supervised Pre-training

The proposed architecture with RNNs as feature encoders and sparse GPs as predictive models is trainable in an end-to-end fashion with gradient descent. The free parameters include parameters in RNNs, the inducing points, and hyper-parameters in the covariance function. In our experiment, we realize that training such architecture from scratch could be challenging. Given an RNN and inducing points that are both randomly initialized, the covariance matrix in GP is also random. This often causes the length scale parameter in the RBF kernel to shrink to extremely small values, and the GP would then degrade to its prior, correspondingly. To alleviate such problems, we find the initialization of the inducing points to be an important step for obtaining good models. More specifically, we find that the training always fails if we initialize the inducing inputs with random vectors. On the other hand, initializing the inducing points with latent representations from the RNN-based feature extractor always shows good performance, even though the parameters in the feature extractor are initialized randomly. Formally, we get the initial inducing inputs as

$$\boldsymbol{u}_i^{\text{init}} = \boldsymbol{h}_i^{\text{init}} = f_{\boldsymbol{\Phi}_{\text{init}}}(\boldsymbol{x}_i^{\text{sta}}, \boldsymbol{X}_i^{\text{seq}}),$$

where a random subset of the training inputs $\{\boldsymbol{x}_i^{\text{sta}}, \boldsymbol{X}_i^{\text{seq}}\}_{i=1}^m$ is involved. To this end, we conjecture that a pre-training step on the feature extractor would boost the performance of our DKAFT model. Since many covariance functions, e.g., RBF kernels, take the distance between samples as input, it would be beneficial if the feature extractor generates abstract covariates in well-clustered latent spaces, where the samples with similar target variables are closer to each other. We propose that one could achieve such a beneficial configuration via Deep Metric Learning (DML), which is initially proposed for vision-related tasks like face verification (Schroff et al., 2015) and person re-identification (Hermans et al., 2017). What DML learns is to represent samples in a latent space that retains the similarity in the target variables.

In DML, pair loss or triplet loss provides the foundation for embedding samples using twin networks, which refers to the replications of the same feature extractor network. Various losses have been proposed to improve the embedding from different perspectives, including contrastive loss (Hadsell et al., 2006), triplet margin loss (Weinberger et al., 2006) or the more recent Signal-To-Noise Ratio loss (Yuan et al., 2019). More specifically, a triplet is defined with the class information to consist of an anchor, a positive, and a negative sample, $\{\boldsymbol{x}_i^{\text{sta}}, \boldsymbol{X}_i^{\text{seq}}\}^{\text{A/P/N}}$, where the anchor is of the same class as the positive and the negative is not. As there are no class labels in time-to-event prediction, we propose to categorize the target variables according to their binnings in the histogram to facilitate the triplet generation. In the context of our DKAFT model, we train the RNN-based feature extractor using, e.g., triplet margin loss (Schroff et al. (2015))

$$\mathcal{J}_{\text{triplet}} = \sum_{i=1}^{n} \left[\mathrm{d}(\boldsymbol{h}_i^{\text{A}}, \boldsymbol{h}_i^{\text{P}}) - \mathrm{d}(\boldsymbol{h}_i^{\text{A}}, \boldsymbol{h}_i^{\text{N}}) + \alpha\right]_+,$$

where $\mathrm{d}(\cdot, \cdot)$ is a distance metric, like Euclidean distance, $\boldsymbol{h}_i^{\text{A}}, \boldsymbol{h}_i^{\text{P}}, \boldsymbol{h}_i^{\text{N}}$ are the abstract covariates of anchor, positive, and negative samples, α is a predefined margin value, and $[\cdot]_+$ takes the positive part of the variable. From the GP perspective, it is the covariance function

that defines the "similarity" between samples, the choice of a specific loss in DML should therefore take it into consideration.

To find a suitable training epoch for the pre-training, we use an early stopping technique, which terminates the training automatically if the monitored metric does not improve over a given number of epochs. Mean Average Precision at R (MAP@R) proposed in Musgrave et al. (2020) is used as the monitored metric on the validation set, which combines the metrics of Mean Average Precision and R-precision.

To conclude, we propose to apply an RNN-based feature extractor to learn fix-sized latent representations from patient trajectories of variable lengths. The feature extractor can be randomly initialized or pre-trained with our proposed DML-based approach. The DKAFT model is trained end-to-end against (sparse) GP objectives using SGD-based methods and produces predictive distributions.

4. Cohort

4.1. Data Extraction

We have included two datasets to validate the effectiveness of our proposed method. In both datasets, we treat observed time-to-event as our target variables.

The first dataset is provided by the PRAEGNANT study network (Fasching et al., 2015), which focuses on patients suffering from metastatic and incurable breast cancer. Based on a patient's background information and medical history, we attempt to predict the Progression-Free Survival time (PFS-PRAEGNANT), i.e., the number of days till the next recorded progression. The raw data are hosted in a relational database system, secuTrial®, and can be accessed under restrictions. After querying and preprocessing, we retrieved a dataset of 1336 patient cases.

The second dataset comes from the Medical Information Mart for Intensive Care database (MIMIC-III), a freely accessible database, which contains data including $53,423$ distinct Intensive Care Unit (ICU) admissions of adult patients between 2001 and 2012 (Johnson et al., 2016). In this work, we consider a cohort of patients from MIMIC-III v1.4, who are older than 15 years at the time of ICU admission. Besides, only the first admission of these patients is included to prevent potential information leakage in the analysis. Based on the data collected during the first 48 hours, we attempt to predict the length-of-stay (LoS-MIMIC) for each admission, i.e., the number of days between hospital admission and discharge from the hospital. More specifically, we followed the scripts[1] provided by Purushotham et al. (2018) and extracted a dataset with $31,986$ patient admissions.

In both extracted datasets, all cases are with observed time-to-event. In case of the MIMIC dataset, the patients were always supposed to leave the ICU and the PRAEGNANT patients all have metastasis and were expecting multiple progressions.

4.2. Feature Processing

In the PRAEGNANT dataset, the static information includes 1) basic patient information, e.g., age and height 2) information on the primary tumor, and 3) history of metastasis before entering the study. In total, there are 26 features of binary, categorical, or numerical

[1] https://github.com/USC-Melady/Benchmarking_DL_MIMICIII

type. After performing one-hot encoding on the binary and categorical features, we obtain a feature vector $\boldsymbol{x}_i^{\text{sta}} \in \mathbb{R}^{114}$ for the static information with an average sparsity of 0.871. The sequential information includes 4) local recurrence 5) metastasis 6) clinical visits 7) radio-therapies 8) systemic therapies, and 9) surgeries. These events are observed with a timestamp, and multiple events could happen at the same timestamp. After performing binary-encoding on 26 sequential features, we extract a feature matrix $\boldsymbol{X}_i^{\text{seq}} \in \mathbb{R}^{t_i \times 188}$ for the sequential information of each patient case, where t_i denotes the length of the sequence before the progression. The length of the sequences t_i varies from 1 to 22 and is on average 6.42. The average sparsity of the sequential feature matrix is 0.973.

In the MIMIC-III dataset, the static information refers to the basic information during the admission, e.g., age and admission type. After performing binary encoding on five static features, we obtain a feature vector for each admission $\boldsymbol{x}_i^{\text{sta}} \in \mathbb{R}^{10}$. Moreover, the sequential information refers to the continuously monitored measurements or prescriptions in the ICU environment. They are sampled or aggregated every one hour to represent the patient status at different time steps. 136 sequential features have been selected. Those features are available for most patients. As a result, we extract a feature matrix $\boldsymbol{X}_i^{\text{seq}} \in \mathbb{R}^{t_i \times 136}$ for the sequential information, where t_i is 48 for all patient admissions. A complete list of the chosen features can be found in Appendix A.

5. Experiments

5.1. Experimental Details and Evaluation Approaches

We conducted cross-validations (CV) for the PFS-PRAEGNANT and LoS-MIMIC prediction tasks with 90% samples in the dataset. The validation set is used for tuning hyper-parameters like, e.g., the dimension of the latent representations, weight decay, and training epochs. As a result, we have $4, 32, 128$ and $4, 64, 64$ for the size of the static latent representations $\text{n}_{\text{sta\_repr}}$, sequential feature embeddings $\text{n}_{\text{seq\_emb}}$, and sequential latent representations $\text{n}_{\text{seq\_repr}}$ in the PFS-PRAEGNANT and LoS-MIMIC prediction tasks, respectively. The evaluation metrics reported in the following are all computed based on the remaining unseen 10% samples of the dataset.

All our NN-based models are built with the PyTorch package (Paszke et al., 2019). The GP-related methods are implemented with the help of the GPyTorch package (Gardner et al., 2018). Related scripts[2] are published to ensure the reproducibility of the work.

Since the target variable z_i is assumed to follow a log-normal distribution, it is not appropriate to measure the results using Root Mean Square Error (RMSE) in its original scale. Therefore, we report the more robust metric of Median Absolute Deviation (MAD) defined as $\text{MAD} = \text{median}_i(|z_i - \hat{z}_i|)$. In addition, we report the RMSE in the logarithmic scale of the target variable, which is defined as $\text{RMSE} = \sqrt{\frac{1}{n}\sum_i^n (y_i - \hat{y}_i)^2}$.

In addition to the point estimate performance, we also evaluate the meaningfulness of the predictive variance σ_i of our proposed model as well as other uncertainty-aware baselines using a *Quantile Performance (QP) plot* (Wu et al., 2021; Yang and Buettner, 2021). Intuitively, the predictive confidence generated by a model is only systematically meaningful if the model assigns higher confidence to the more accurate predictions and lower

[2] https://github.com/ZhiliangWu/DKAFT

confidence to the less confident ones. In the scope of our work, we interpret the predictive variance as a form of confidence estimation, where smaller values correspond to higher confidence. Therefore, the predictive variance from our model enables a formal evaluation of such expected behavior. Concretely speaking, we extract the evaluation pairs $\{y_i, \hat{y}_i\}_{i=1}^n$ (or $\{z_i, \hat{z}_i\}_{i=1}^n$) for each quantile of the predictive variance of the model $q \in \{\frac{1}{Q}, \frac{2}{Q}, \ldots, 1\}$, where the corresponding predictive variance σ_i^2 is smaller than or equal to the q-th quantile. Formally, we have the performance in each quantile as

$$\text{Performance}_\text{q} := \text{Metric}(\{(y_i, \hat{y}_i) \text{ or } (z_i, \hat{z}_i) \mid \forall \sigma_i^2 \leq q\text{-th quantile}\}),$$

where Metric$(\cdot)$ could be MAD for z_i or RMSE for y_i. We plot the performance of each quantile on the y-axis against the corresponding q values on the x-axis. For a model with meaningful uncertain-awareness, a monotonically increasing line is expected in the QP plot. Furthermore, a stronger correlation between the metric and confidence across the quantiles suggest a better quantification of the predictive uncertainty.

Finally, we plot the (empirical) Cumulative Distribution Function (CDF) of the normalized residuals to show how well our model is calibrated as a further evaluation metrics. According to our normal distribution assumption on the logarithmic scale of the target variable, it is expected to be close to the CDF of a standard normal distribution $\mathcal{N}(0, 1)$. In addition, the Continuous Ranking Probability Score (CRPS), a popular calibration metric for regression (Gneiting and Raftery, 2007; Jankowiak et al., 2020), is reported to have a quantitative comparison.

5.2. Evaluation of the PFS-PRAEGNANT and LoS-MIMIC prediction

As weak baselines, we report the performance of standard Cox and AFT regression using the R package *survival* (Terry M. Therneau and Patricia M. Grambsch, 2000; Therneau, 2020) with raw features aggregated w.r.t. the time axis. Such aggregation has been used in Esteban et al. (2015) and Yang et al. (2017a), which turns out to be a reasonable solution to deal with features with time-stamps by ignoring the order of the events.

To investigate the performance of different output layers as we discussed in Sec. 3, we train all models with the same RNN-based feature extractor. Using a linear output layer as our strong baseline (*RNN+AFT*) (Yang et al., 2017a), we include the SVGP and PPGP output layers for both prediction tasks, which are denoted as *DKAFT (SVGP)* and *DKAFT (PPGP)*, respectively. Thanks to the moderate size of the PRAEGNANT dataset, we also have an ExactGP output layer for the PFS-PRAEGNANT prediction task, which we denote as *DKAFT (ExactGP)*. To validate our proposed initialization method, we pre-trained the feature extractor using DML and then fine-tune the proposed model. In such a case, the performance of models without pre-training naturally serves as the baselines. Note that, for our DKAFT models, only the mean predictions are involved in the evaluation of point estimates. Results are summarized in Tab. 1. For NN-based CPH baselines (Kvamme et al., 2019), we report the results in Tab. 3 in Appendix C.

From Tab. 1 we can see that our DKAFT models demonstrate much stronger performance compared to the Cox Regression and AFT Regression with aggregated features. For the evaluation w.r.t. RMSE, our DKAFT models all outperform the corresponding strong baselines, the RNN+AFT models. Tab. 2 shows the p-values of paired t-tests quantifying

Table 1: Experimental results: MAD in the original scale (days) and RMSE in the logarithmic scale for PFS-PRAEGNANT and LoS-MIMIC prediction tasks. Our DKAFT models outperform the baselines, including Cox Regression, AFT regression, and RNN+AFT models. More baselines in Appendix. C.

Method	Pre-training	Progression-Free Survival		Length-of-Stay	
		MAD	RMSE	MAD	RMSE
Cox Regression	–*	200.800 ± 16.984	1.609 ± 0.054	2.727 ± 0.007	0.638 ± 0.0002
AFT Regression	–*	206.065 ± 8.988	1.685 ± 0.080	2.742 ± 0.014	0.630 ± 0.0001
RNN+AFT	None	150.918 ± 3.009	1.273 ± 0.019	2.476 ± 0.040	0.575 ± 0.003
DKAFT (ExactGP)	None	144.622 ± 8.689	1.225 ± 0.022	–†	–†
DKAFT (SVGP)	None	154.237 ± 13.490	1.211 ± 0.020	2.428 ± 0.056	0.572 ± 0.003
DKAFT (PPGP)	None	147.108 ± 6.284	1.220 ± 0.019	2.351 ± 0.021	0.563 ± 0.001
RNN+AFT	DML	138.155 ± 7.496	1.267 ± 0.007	2.452 ± 0.057	0.568 ± 0.001
DKAFT (ExactGP)	DML	$\mathbf{134.422 \pm 7.255}$	1.202 ± 0.012	–†	–†
DKAFT (SVGP)	DML	151.852 ± 11.305	1.221 ± 0.007	2.438 ± 0.079	0.567 ± 0.005
DKAFT (PPGP)	DML	146.616 ± 17.109	$\mathbf{1.195 \pm 0.008}$	$\mathbf{2.346 \pm 0.042}$	$\mathbf{0.557 \pm 0.002}$

*Not applicable to the method.
†Not possible due to the $\mathcal{O}(n^3)$ computational complexity.

the performance improvement of our DKAFT models. In the task of PFS-PRAEGNANT prediction, we can see our DKAFT models outperform RNN+AFT models significantly (with a significant level $\alpha = 0.05$). In contrast, only DKAFT (PPGP) models show significantly better RMSE performance in the LoS-MIMIC prediction task. For the evaluation regarding MAD, our DKAFT (ExactGP) model performs best in the PFS-PRAEGNANT prediction task, while it is DKAFT (PPGP) for the LoS-MIMIC prediction task. Meanwhile, we can observe a moderate improvement for most models pre-trained with DML compared to those trained from scratch.

Table 2: p-values from paired t-tests to assess the significance of improvement achieved by our DKAFT models, based on the RMSEs collected from multiple cross validations. A two factor ANOVA on the effect of different model setups can be found in the Appendix.B

Comparison candidates	No pre-training		DML	
	PFS	LoS	PFS	LoS
DKAFT (ExactGP) vs. RNN+AFT	0.021	–*	2e-4	–*
DKAFT (SVGP) vs. RNN+AFT	0.010	0.282	0.001	0.682
DKAFT (PPGP) vs. RNN+AFT	0.002	0.001	3e-4	3e-4

*Not possible due to the $\mathcal{O}(n^3)$ computational complexity.

As discussed in Sec. 3, DKAFT models are trained by either optimizing the log marginal likelihood objective or the variational approximation of it, which is essentially a generalization to mean square error in linear regression. Therefore, improved performance is expected as the GP-based output layers include inducing points during the training and inference time, which implicitly facilitates integrating all possible linear models. Meanwhile, since MAD is a more robust metric than RMSE regarding outliers, there are some performance differences of the same model between these two metrics.

The superior performance of the DKAFT (SVGP) and DKAFT (PPGP) compared to DKAFT (ExactGP) w.r.t. RMSE is a bit surprising since these models are essentially approximations of the latter. We speculate that such phenomenon comes from the sparse and high-dimensional features in the PRAEGNANT dataset, which results in redundant or even repetitive patient covariates from the feature extractor. This makes the GP model struggle to capture the correlation correctly. On the contrary, the inducing points in SVGP or PPGP are not constrained by the raw input features and could avoid this coupling during the optimization.

For the initialization with pre-training methods, DML helps attain a feature extractor, which clusters samples with similar targets in nearby regions. Models with all output layers, including linear layers, achieve moderate improvement. This can be attributed to 1) the non-convexity nature of the log marginal likelihood objective or the ELBO objective with deep neural networks, where a good starting point could offer advantages for the following optimization procedure; 2) the good initialization of the inducing inputs generated by the pre-trained feature extractor.

5.3. Evaluation of the predictive variances

Apart from improved performance for point estimates, the main advantage of the proposed method lies in the uncertainty-aware nature of the model. As a baseline, we include MC Dropout (Gal and Ghahramani, 2016), a well-known method for enabling uncertainty estimates in neural networks. By adding a dropout layer (Srivastava et al., 2014) before each weight layer, the resulting model is proved to be mathematically equivalent to a probabilistic deep Gaussian Process. In our experiments, we followed the proposed method with the suggested dropout rate of 0.2. The mean prediction and function variance are computed from performing 50 stochastic forward passes through the network. Since QP plots demonstrate unstable performance if the number of evaluation pairs in each quantile is too small (like the test set for the PFS-PRAEGNANT prediction task with only 134 samples in total), we only report the evaluation on the LoS-MIMIC prediction task in Fig. 3.

In Fig. 3 we visualize the quantile performance for MAD and RMSE, where the solid line represents an average performance and the error bar for the standard deviation across CV splits. We observe a strong monotonically increasing line in both metrics from our DKAFT (PPGP) model. For the evaluation pairs that the model is more confident with, e.g., ones corresponding to the predictive variances at quantile 10%, the MAD and RMSE are 1.163 ± 0.030 and 0.316 ± 0.005, respectively. Such results correspond to only half of the values reported in Tab. 1, indicating a significant improvement. Meanwhile, the DKAFT (SVGP) model shows an increasing dependency only in MAD. For the models with MC Dropout, there seems to be no performance difference between quantiles.

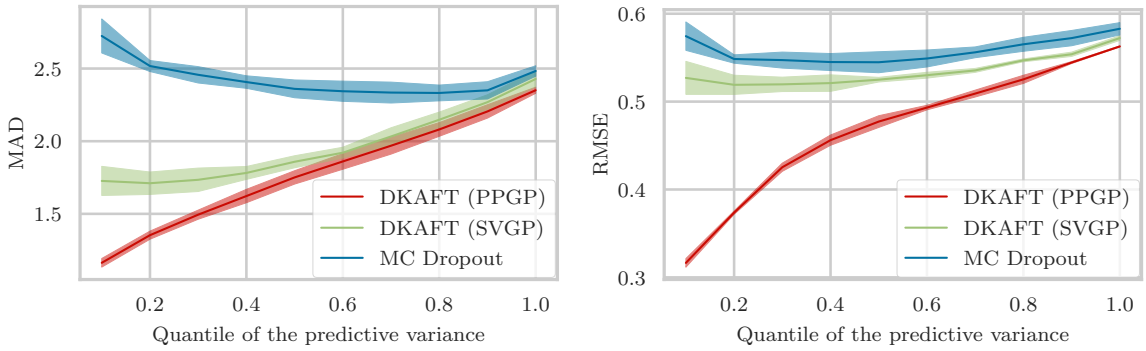


Figure 3: Quantile MAD (left) and Quantile RMSE (right) in the y-axis against quantile predictive uncertainty in the x-axis for the LoS-MIMIC prediction task. Our DKAFT model with a PPGP output layer (red) shows the strongest increasing trend in both MAD and RMSE. This indicates that the model is monotonically more confident in predictions that are indeed closer to the ground-truths. We emphasize that this is a desirable feature to expect from an uncertainty-aware prediction model.

Compared with MC Dropout, our DKAFT models deliver more meaningful uncertainty estimates since the inducing point technique realizes explicit modeling of the predictive variances. As expected, the most meaningful uncertainty estimates are visible from the models with a PPGP output layer, since the function variance is restored explicitly in the training objective compared to those with an SVGP output layer. These observations indeed motivated the application of our DKAFT (PPGP) model when meaningful uncertainty estimates become a higher priority.

5.4. Calibration of the Model

Calibration of a predictive model refers to the statistical consistency between the predictive distribution from the model and the observations (Gneiting et al., 2007), which is arguably also an important aspect for healthcare applications. In GPs, the predictive variance incorporates both the modeling uncertainty (function variance) and data uncertainty (observational noise). Even for data-points lying far from the training data, the resulting predictive distribution tends to be well-calibrated (Rasmussen and Williams, 2005). In this section, we demonstrate that our DKAFT models inherit this nice property from GPs. Meanwhile, as a popular method for calibrating neural networks, MC Dropout is included in our experiment as a baseline. We visualize the empirical CDF of the normalized residuals with the predictive variances in Fig. 4. Besides, the CRPS score is computed to have a quantitative comparison. The CRPS score generalizes the Mean Absolute Error (MAE) to probabilistic predictions.

In Fig. 4, we visualize the CDF plots for both time-to-event prediction tasks. Graphically speaking, all methods demonstrate well-calibrated behavior based on their closeness to the best possible calibrated CDF. The ECDF of our DKAFT models is closer to the ideal CDF

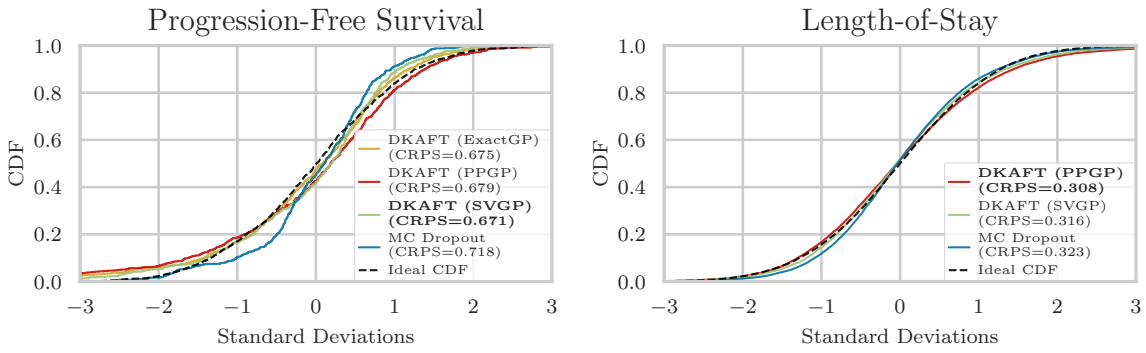


Figure 4: Empirical CDF of the normalized residual, $(y_i - \mu_i)/\sigma_i$, from different models against an "ideal CDF" from a standard normal distribution. Continuous Ranking Probability Score is reported to provide a quantitative comparison. All models show well-calibrated behavior, where our DKAFT models are better calibrated than the baseline method, MC Dropout.

than MC Dropout. Such observation is also verified by the lower values of the respective CRPS score. Within DKAFT models, the ones with an SVGP output layer perform slightly better than those with PPGP and ExactGP output layers in the PFS-PRAEGNANT prediction task. In contrast, the DKAFT models with a PPGP output layer outperform the ones with an SVGP output layer in the LoS-MIMIC prediction task.

It is worth highlighting that the inference in the MC Dropout requires multiple stochastic forward passes through the sampled network, which results in significantly slower processing than in our DKAFT models. This would further motivate the application of the analytical solutions from the proposed DKAFT models when computational efficiency plays an essential role in real-world applications, like in real-time response systems.

6. Conclusion and Future Works

In this work, we propose the Deep Kernel Accelerated Failure Time (DKAFT) model to address the lack of uncertainty estimates in recurrent neural network (RNN) based solutions for time-to-event prediction tasks. Our DKAFT model consists of an RNN encoder and a sparse GP as the prediction model. The former serves as a trainable feature extractor to embed the patient features into a latent space of abstract covariates. The GP-based output layer consumes the abstract covariates of the patients, and outputs a predictive distribution for the time-to-event prediction.

We show that the proposed model can be trained in an end-to-end fashion, like typical neural networks, using stochastic gradient descent-based methods. In addition, a deep metric learning-based pre-training method is proposed to further improve the performance of the proposed model. Through experiments on two real-world datasets, the DKAFT models show better performance in terms of the point estimates than the RNN-based models with linear output layers. More importantly, the predictive variances from our DKAFT model

reflect the confidence of the predictions. It produces better metrics evaluation in terms of RMSE and MAD with monotonically higher confidence about the predictions. Such uncertainty estimates would improve the trustworthiness of the provided model when it interacts with the physicians. Furthermore, the predictive variance also serves to improve the calibration of the model. Compared to MC Dropout, a popular method to augment the uncertainty in the neural networks, our DKAFT model shows better performance and enjoys lower computational cost, which motivates its usage in real-world applications.

As future work, we would like to further study the interpretation of the uncertainty in the proposed model. From a machine learning's perspective, it refers to checking whether a test sample lies far from the manifold constituted by the training samples. From a decision support's perspective—since our proposed model offers predictive variances based on the "neighboring" training samples in the feature spaces—it would offer practical help if it also fits physicians' understanding.

Limitations As shown in Sec. 3.2, in our DKAFT models, (right) censored observations will contribute to the training objective through its survival function instead of the probability density function. However, due to the nature of the time-to-event prediction tasks we focus on in this work, administrative censoring is not included in the experiments. Besides, the evaluation of censoring cases is also beyond the scope of this manuscript. We leave these perspectives as part of our future work.

Acknowledgement The authors acknowledge the support by the German Federal Ministry for Education and Research (BMBF), funding project "MLWin" (grant 01IS18050).

GEFÖRDERT VOM

References

Ahmed M Alaa and Mihaela van der Schaar. Deep multi-task gaussian processes for survival analysis with competing risks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2326–2334, 2017.

Edmon Begoli, Tanmoy Bhattacharya, and Dimitri Kusnezov. The need for uncertainty quantification in machine-assisted medical decision making. *Nature Machine Intelligence*, 1(1):20–23, 2019.

Alexis Bellot and Mihaela Van Der Schaar. Flexible modelling of longitudinal medical data: A bayesian nonparametric approach. *ACM Transactions on Computing for Healthcare*, 1 (1):1–15, 2020.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

George H Chen. Deep kernel survival analysis and subject-specific survival time prediction intervals. In *Machine Learning for Healthcare Conference*, pages 537–565. PMLR, 2020.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

Edward Choi, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Using recurrent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association*, 24(2):361–370, 2017.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

David Collett. *Modelling survival data in medical research*. CRC press, 2015.

Cristóbal Esteban, Danilo Schmidt, Denis Krompaß, and Volker Tresp. Predicting sequences of clinical events by using a personalized temporal latent embedding model. In *2015 International Conference on Healthcare Informatics*, pages 130–139. IEEE, 2015.

Cristóbal Esteban, Oliver Staeck, Stephan Baier, Yinchong Yang, and Volker Tresp. Predicting clinical events by combining static and dynamic information using recurrent neural networks. In *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 93–101. IEEE, 2016.

P.A. Fasching, S.Y. Brucker, T.N. Fehm, F. Overkamp, W. Janni, M. Wallwiener, P. Hadji, E. Belleville, L. Häberle, F.A. Taran, D. Luftner, M.P. Lux, J. Ettl, V. Muller, H. Tesch, D. Wallwiener, and A. Schneeweiss. Biomarkers in patients with metastatic breast cancer and the praegnant study network. *Geburtshilfe Frauenheilkunde*, 75(01):41–50, 2015. URL http://www.praegnant.org/.

Tamara Fernández, Nicolás Rivera, and Yee Whye Teh. Gaussian processes for survival analysis. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 5021–5029, 2016.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in Neural Information Processing Systems*, 31:7576–7586, 2018.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

Yoni Halpern, Steven Horng, Youngduck Choi, and David Sontag. Electronic medical record phenotyping using the anchor and learn framework. *Journal of the American Medical Informatics Association*, 23(4):731–740, 2016.

Frank E Harrell, Robert M Califf, David B Pryor, Kerry L Lee, and Robert A Rosati. Evaluating the yield of medical tests. *Jama*, 247(18):2543–2546, 1982.

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*, page 282. Citeseer, 2013.

Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

Graeme L Hickey, Pete Philipson, Andrea Jorgensen, and Ruwanthi Kolamunnage-Dona. Joint modelling of time-to-event and multivariate longitudinal outcomes: recent developments and issues. *BMC medical research methodology*, 16(1):1–15, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Martin Jankowiak, Geoff Pleiss, and Jacob Gardner. Parametric gaussian process regressors. In *International Conference on Machine Learning*, pages 4702–4712. PMLR, 2020.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

John D Kalbfleisch and Ross L Prentice. *The Statistical Analysis of Failure Time Data*, volume 360. John Wiley & Sons, 2002.

Håvard Kvamme, Ørnulf Borgan, and Ida Scheel. Time-to-event prediction with neural networks and cox regression. *Journal of machine learning research*, 20(129):1–30, 2019.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, 2017.

Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When gaussian process meets big data: A review of scalable gps. *IEEE transactions on neural networks and learning systems*, 31(11):4405–4423, 2020.

Tomáš Mikolov et al. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, 80:26, 2012.

Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *European Conference on Computer Vision*, pages 681–699. Springer, 2020.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Ross L Prentice. Linear rank tests with right censored data. *Biometrika*, 65(1):167–179, 1978.

Sanjay Purushotham, Chuizheng Meng, Zhengping Che, and Yan Liu. Benchmarking deep learning models on large healthcare datasets. *Journal of biomedical informatics*, 83:112–134, 2018.

Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6: 1939–1959, 2005.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Dimitris Rizopoulos. Dynamic predictions and prospective accuracy in joint models for longitudinal and time-to-event data. *Biometrics*, 67(3):819–829, 2011.

Alan D Saul. *Gaussian Process Based Approaches for Survival Analysis*. PhD thesis, University of Sheffield, 2016.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems*, 18:1259–1266, 2006.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Terry M. Therneau and Patricia M. Grambsch. *Modeling Survival Data: Extending the Cox Model*. Springer, New York, 2000. ISBN 0-387-98784-3.

Terry M Therneau. *A Package for Survival Analysis in R*, 2020. URL https://CRAN.R-project.org/package=survival. R package version 3.2-7.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR, 2009.

Volker Tresp. A bayesian committee machine. *Neural computation*, 12(11):2719–2741, 2000.

Volker Tresp, J Marc Overhage, Markus Bundschus, Shahrooz Rabizadeh, Peter A Fasching, and Shipeng Yu. Going digital: a survey on digitalization and large-scale data analytics in healthcare. *Proceedings of the IEEE*, 104(11):2180–2206, 2016.

Hans C Van Houwelingen. Dynamic prediction by landmarking in event history analysis. *Scandinavian Journal of Statistics*, 34(1):70–85, 2007.

Ke Alexander Wang, Geoff Pleiss, Jacob R Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact gaussian processes on a million data points. *Advances in Neural Information Processing Systems*, 32, 2019.

Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

Chris Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems 13*, 2001.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

Zhiliang Wu, Yinchong Yang, Yunpu Ma, Yushan Liu, Rui Zhao, Michael Moor, and Volker Tresp. Learning individualized treatment rules with estimated translated inverse propensity score. In *2020 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–11, 2020. doi: 10.1109/ICHI48887.2020.9374397.

Zhiliang Wu, Yinchong Yang, Jindong Gu, and Volker Tresp. Quantifying predictive uncertainty in medical image analysis with deep kernel learning. *arXiv preprint arXiv:2106.00638*, 2021.

Cao Xiao, Edward Choi, and Jimeng Sun. Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. *Journal of the American Medical Informatics Association*, 25(10):1419–1428, 2018.

Yinchong Yang and Florian Buettner. Multi-output gaussian processes for uncertainty-aware recommender systems. *arXiv preprint arXiv:2106.04221*, 2021.

Yinchong Yang, Peter A Fasching, and Volker Tresp. Modeling progression free survival in breast cancer with tensorized recurrent neural networks and accelerated failure time models. In *Machine Learning for Healthcare Conference*, pages 164–176. PMLR, 2017a.

Yinchong Yang, Peter A Fasching, and Volker Tresp. Predictive modeling of therapy decisions in metastatic breast cancer with recurrent neural network encoder and multinomial hierarchical regression decoder. In *2017 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 46–55. IEEE, 2017b.

Tongtong Yuan, Weihong Deng, Jian Tang, Yinan Tang, and Binghui Chen. Signal-to-noise ratio: A robust distance metric for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4815–4824, 2019.

Appendix A. Feature Set in MIMIC-III

To best represent the clinical status, we extracted both static and sequential information from the MIMIC-III. The included features are the same as the *Feature Set C* defined in Purushotham et al. (2018). In the chosen features, most have continuous values except for acquired immunodeficiency syndrome, hematologic malignancy, metastatic cancer, and admission type. The missing rates of each feature can be found in Table A.26 in Purushotham et al. (2018).

Static Information: age, acquired immunodeficiency syndrome, hematologic malignancy, metastatic cancer, admission type

Sequential Information: Gastric Tube, Stool Out Stool, Urine Out Incontinent, Ultrafiltrate, Fecal Bag, Chest Tube 1, Chest Tube 2, Jackson Pratt 1, OR EBL, Pre-Admission, TF Residual, Albumin 5%, Fresh Frozen Plasma, Lorazepam (Ativan), Calcium Gluconate, Midazolam (Versed), Phenylephrine, Furosemide (Lasix), Hydralazine, Norepinephrine, Magnesium Sulfate, Nitroglycerin, Insulin Regular, Morphine Sulfate, Potassium Chloride, Packed Red Blood Cells, Gastric Meds, D5 1/2NS, LR, Solution, Sterile Water, Piggyback, OR Crystalloid Intake, PO Intake, GT Flush, KCL (Bolus), Magnesium Sulfate (Bolus), Hematocrit, Platelet count, Hemoglobin, MCHC, MCH, MCV, Red blood cells, RDW, Chloride, Anion gap, Creatinine, Glucose, Magnesium, Calcium total, Phosphate, INR(PT), PT, PTT, Lymphocytes, Monocytes, Neutrophils, Basophils, Eosinophils, PH, Base excess, Calculated total CO2, PCO2, Specific gravity, Lactate, Alanine aminotransferase (ALT), Asparate aminotransferase (AST), Alkaline phosphatase, ALBUMIN, Aspirin, Bisacodyl, Docusate Sodium, Humulin-R Insulin, Metoprolol Tartrate, Pantoprazolel, Arterial Blood Pressure diastolic, Arterial Blood Pressure mean, Respiratory Rate, Alarms On, Minute Volume Alarm-Low, Peakinsp.Pressure, PEEP set, Minute Volume, Tidal Volume (observed), Minute Volume Alarm High, Mean Airway Pressure, Central Venous Pressure, Respiratory Rate (Set), Pulmonary Artery Pressure mean, O2Flow, Glucose fingerstick, Heart Rate Alarm Low, Pulmonary Artery Pressure systolic, Tidal Volume (set), Pulmonary Artery Pressure diastolic, SpO2 Desat Limit, Resp Alarm High, Skin Care, gcsverbal, gcsmotor, gcseyes, systolic blood pressure abp mean, heart rate, body temperature, pao2, fiO2, urinary output sum, serum urea nitrogen level, white blood cells count mean, serum bicarbonate level mean, sodium level mean, potassium level mean, bilirubin level, ie ratio mean, diastolic blood pressure mean, arterial pressure mean, respiratory rate, SpO2 peripheral, glucose, weight, height, hgb, platelet, chloride, creatinine, norepinephrine, epinephrine, phenylephrine, vasopressin, dopamine, midazolam, fentanyl, propofol, peep, ph.

Appendix B. ANOVA as an ablation study

We perform ANOVA to further verify the improvements reported in Tab. 1 in terms of RMSE. In case of the progression-free survival (PFS-PRAEGNANT) prediction task (with Fasching et al. (2015) dataset), we report a p-value of 0.064 w.r.t. the four choices of the prediction model, and p-value of 5.9e-6 w.r.t. applying deep metric learning as pre-training or not. In case of the length-of-stay (LoS-MIMIC) prediction task (with Johnson et al. (2016) dataset), we report a p-value of 1.5e-6 w.r.t. the three choices of the prediction model, and p-value of 2.0e-9 w.r.t. applying deep metric learning as pre-training or not.

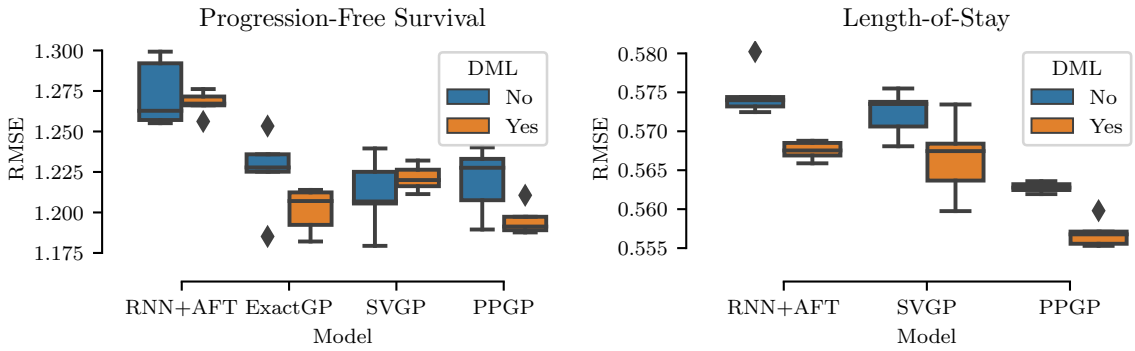


Figure 5: Grouped Box plots visualizing a comparison between different models, RNN+AFT and ExactGP, SVGP, PPGP of our DKAFT models. A similar conclusion to the reported p-values from ANOVA could be drawn from these plots.

It appears that only in the case of PFS-PRAEGNANT, the choice of the prediction model does not have a significant impact on the performance, presumably due to the relatively small number of patient samples. In Fig. 5 we visualize the effects of these two factors as grouped box plots.

Appendix C. Experimental Results with More Baselines and Metrics

We conducted experiments for NN-based CPH models using the PyCox package[3]. Both continuous-time models (*DeepSurv, CoxTime, CoxCC, and PCHazard*) and discrete-time models (*LogisticHazard, PMF, DeepHit, MTLR, and BCESurv*) are included, where the latter perform much worse w.r.t. the metrics we are interested in this manuscript. Therefore, we only report the results for continuous-time models. Besides, we include two neural network architectures for these models, where a Multiple-Layer Perceptron (MLP) receives aggregated features like Cox Regression and a Recurrent Neural Network (RNN) refers to the same base network we used for our DKAFT models. In addition, we report the concordance index (C-Index) (Harrell et al., 1982) of all methods to show their respective discriminative performance. From Tab. 3 we can see, that these continuous-time models only show comparable performance to our strong baseline (RNN+AFT). Meanwhile, the RNN variants of the same model always improve the performance w.r.t. RMSE and C-Index as they take the time-dependency of the patient information into consideration.

[3] https://github.com/havakv/pycox

Table 3: Experimental results: MAD in the original scale (days) and RMSE in the logarithmic scale for PFS-PRAEGNANT and LoS-MIMIC prediction tasks. Our DKAFT models outperform the baselines, including Cox Regression, AFT regression, NN-based CPH models, and RNN+AFT models.

		Progression-Free Survival			Length-of-Stay		
Method	Pretraining	MAD	RMSE	C-Index	MAD	RMSE	C-Index
Cox Regression	–*	200.800 ± 16.984	1.609 ± 0.054	0.676 ± 0.004	2.727 ± 0.007	0.638 ± 0.0002	0.683 ± 0.001
AFT Regression	–*	206.065 ± 8.988	1.685 ± 0.080	0.656 ± 0.008	2.742 ± 0.014	0.630 ± 0.0001	0.684 ± 0.001
MLP+DeepSurv	–*	153.900 ± 7.151	1.293 ± 0.024	0.731 ± 0.006	2.486 ± 0.028	0.593 ± 0.005	0.715 ± 0.003
RNN+DeepSurv	–*	166.700 ± 6.565	1.252 ± 0.016	0.733 ± 0.005	2.456 ± 0.039	0.579 ± 0.003	0.721 ± 0.003
MLP+CoxTime	–*	153.900 ± 6.924	1.387 ± 0.018	0.697 ± 0.004	2.599 ± 0.034	0.606 ± 0.008	0.703 ± 0.005
RNN+CoxTime	–*	146.000 ± 16.634	1.278 ± 0.010	0.723 ± 0.002	2.515 ± 0.051	0.585 ± 0.004	0.718 ± 0.003
MLP+CoxCC	–*	153.100 ± 9.557	1.305 ± 0.010	0.729 ± 0.003	2.463 ± 0.015	0.585 ± 0.005	0.719 ± 0.003
RNN+CoxCC	–*	153.800 ± 6.022	1.262 ± 0.009	0.728 ± 0.001	2.479 ± 0.018	0.578 ± 0.002	0.724 ± 0.002
MLP+PCHazard	–*	167.580 ± 17.490	1.577 ± 0.189	0.694 ± 0.018	3.036 ± 0.019	1.117 ± 0.020	0.665 ± 0.006
RNN+PCHazard	–*	141.041 ± 8.821	1.271 ± 0.017	0.710 ± 0.007	3.055 ± 0.021	0.593 ± 0.009	0.723 ± 0.002
RNN+AFT	None	150.918 ± 3.009	1.273 ± 0.019	0.729 ± 0.008	2.476 ± 0.040	0.575 ± 0.003	0.725 ± 0.002
DKAFT (ExactGP)	None	144.622 ± 8.689	1.225 ± 0.022	0.738 ± 0.008	–†	–†	–†
DKAFT (SVGP)	None	154.237 ± 13.490	1.211 ± 0.020	0.747 ± 0.007	2.428 ± 0.056	0.572 ± 0.003	0.727 ± 0.002
DKAFT (PPGP)	None	147.108 ± 6.284	1.220 ± 0.019	0.745 ± 0.006	2.351 ± 0.021	0.563 ± 0.001	0.734 ± 0.001
RNN+AFT	DML	138.155 ± 7.496	1.267 ± 0.007	0.732 ± 0.003	2.452 ± 0.057	0.568 ± 0.001	0.732 ± 0.001
DKAFT (ExactGP)	DML	$\mathbf{134.422 \pm 7.255}$	1.202 ± 0.012	$\mathbf{0.752 \pm 0.003}$	–†	–†	–†
DKAFT (SVGP)	DML	151.852 ± 11.305	1.221 ± 0.007	0.747 ± 0.002	2.438 ± 0.079	0.567 ± 0.005	0.733 ± 0.003
DKAFT (PPGP)	DML	146.616 ± 17.109	$\mathbf{1.195 \pm 0.008}$	$\mathbf{0.752 \pm 0.006}$	$\mathbf{2.346 \pm 0.042}$	$\mathbf{0.557 \pm 0.002}$	$\mathbf{0.738 \pm 0.001}$

*Not applicable to the method.
†Not possible due to the $\mathcal{O}(n^3)$ computational complexity.

Chapter 5

GAN-Based Models in Categorical EHR Imputation

Categorical EHR Imputation with Generative Adversarial Nets

1st Yinchong Yang
Siemens AG
Munich, Germany
yinchong.yang@siemens.com

2nd Zhiliang Wu
Siemens AG
Ludwig-Maximilians University of Munich
Munich, Germany
zhiliang.wu@siemens.com

2nd Volker Tresp
Siemens AG
Ludwig-Maximilians University of Munich
Munich, Germany
volker.tresp@siemens.com

3rd Peter A. Fasching
University Clinics Erlangen
Department of Gynecology and Obstetrics
Erlangen Germany
peter.fasching@uk-erlangen.de

***Abstract*—Electronic Health Records often suffer from missing data, which poses a major problem in clinical practice and clinical studies. A novel approach for dealing with missing data are Generative Adversarial Nets (GANs), which have been generating huge research interest in image generation and transformation. Recently, researchers have attempted to apply GANs to missing data generation and imputation for EHR data: a major challenge here is the categorical nature of the data. State-of-the-art solutions to the GAN-based generation of categorical data involve either reinforcement learning, or learning a bidirectional mapping between the categorical and the a real latent feature space, so that the GANs only need to generate real-valued features. However, these methods are designed to generate complete feature vectors instead of imputing only the subsets of missing features. In this paper we propose a simple and yet effective approach that is based on previous work on GANs for data imputation. We first motivate our solution by discussing the reason why adversarial training often fails in case of categorical features. Then we derive a novel way to re-code the categorical features to stabilize the adversarial training. Based on experiments on two real-world EHR data with multiple settings, we show that our imputation approach largely improves the prediction accuracy, compared to more traditional data imputation approaches.**

***Index Terms*—Data Imputation, Multiple Imputation, Generative Adversarial Nets**

I. Introduction

The increasing importance of data quality in healthcare: Electronic Health Records (EHR) present a rich data source and are, e.g., used for intra- and inter-departmental information exchange, for documentation purposes, and, most recently, as the basis for many analytic studies. Typically, data involving critical clinical decision paths are of good quality, but less critical data are often incomplete, e.g., due the huge workload of clinical personnel; this poses a significant problem for the secondary use of EHR data. In particular the value of a clinical study greatly depends on data completeness and correctness. Although the prime solution would be to enhance the EHR quality by improving the EHR system design and the data collection process, the missing data problem is not likely to completely disappear. [13] provides an overview on missing data approaches in statistics and [22] presents solutions to the neural network setting. When using nonlinear models, data imputation is often used [14], [23], which is also the approach pursued in this paper. Data imputation is often based on parametric or nonparametric probability density estimation. In this paper we investigate a recently developed GAN architecture. It imputes data without calculating a probability density first, and might become an important method of choice in the future.

Multiple instead of single imputation: More specifically, we discuss a novel realization of the well-known multiple imputation approach [13], [19], [21]. By embedding certain randomness into the imputation method and performing imputation multiple times, one can achieve more flexibility and reliability than with single imputation. This allows for —in contrast to an averaged point estimate of each missing value— estimating the statistical reliability of the imputation methods [4]. Multiple Imputation by Chained Equations (MICE) [16] fits one regression model for each feature that contains missing values, conditioned on all complete features. This method can model the dependency between features but the number of necessary regression models increases quadratically with the number of features. One could also simply assume a multivariate Gaussian distribution for the missing features and draw multiple random samples as imputation. The covariance matrix represents the dependency between features but the Gaussian distribution cannot handle categorical features, which are often present in EHR data. In this paper we investigate a novel approach that takes into account the categorical nature of the features while modeling inter-feature dependency in an efficient way.

GAN as multiple imputation: In recent years, a new class of neural networks, called Generative Adversarial Nets (GANs), have been developed and have generated huge interest in the research community. The original paper [8] proposes to train a network that can learn the underlying distribution of the data, allowing for generating unlimited amount of

data instances. When applied to images, the generated images often appear quite real. Since their initial introduction, a large variety of exciting improvements and modifications of the GAN framework have been proposed to solve different and yet related tasks, such as generating labeled data [15], image translation [30], deriving super-resolution [11] and image augmentation [20]. [26] proposes a new variant, the Generative Adversarial Imputation Nets (GAINs), to perform data imputation and shows promising results on multiple benchmark datasets. This method presents in fact a novel realization of the multiple imputation concept, and it is also related to the MICE algorithm. Instead of trying all possible orders to build the regression chain, it exploits the expressiveness of deep neural networks to model all features with missing values simultaneously. However, this approach cannot immediately be applied to EHR data, as we discuss now.

Challenge in EHR data for GAN: Most of the GAN models have been designed for image data, where the features, i.e., the pixel values, are real numbers. This enables error back-propagation within the GAN framework. In EHR data, however, a large proportion of patient features are categorical. In order to generate categorical data, even when binary coded, requires operations that are not differentiable, meaning that standard adversarial training is not possible. Due to the same reason, GANs have not seen many successful applications in NLP data [17]. A few approaches to generate discrete data with GANs have recently been proposed; most promising are approaches involving reinforcement learning [27], more specifically policy gradients [12]. Another proposed solution is to learn a mapping function from the discrete space of words to a latent real space as well as a reverse mapping [29]. [2] applies this idea to handle categorical features in EHR data and develops auto-encoders to function as the mapping. In such cases, the GAN model only needs to generate real valued vectors that represent the originally discrete data instances, allowing for the gradient propagation from discriminator and generator. In our related works section, we will review this approach in more detail. It is to note that the mapping functions between discrete and real spaces serve as pre- and post-processing steps, and are crucial for the quality of the translation between the discrete and real spaces. Such mapping functions are often trained in an Auto-encoder fashion and thus rely on the completeness of the input features. In a data imputation setting, however, the input features are by definition incomplete and the learned mappings must learn to map incomplete data. That is to say, this proposed approach is only applicable to generating complete feature vector instead of subsets of features.

Our contributions: The Generative Adversarial Imputation Nets framework [26] has been proposed to apply adversarial training to impute missing data of real values. In this work, we adjust this framework so that it can also perform data imputation for categorical features. We hypothesize that the reason that adversarial training often fails with softmax activation in the generator is that, while the true data features contain exclusively 0s and 1s, the softmax function can only produce a probability value between 0 and 1. On one hand, within a couple of epochs, the discriminator with sufficient expressiveness can learn to discriminate the generated values from real data exploiting this fact. On the other hand, it typically takes more epochs of training before the generator can produce real values close to 0 and 1. This phenomenon, i.e., that the discriminator always makes correct decisions and the generator always receives negative feedback from the very beginning, results in the divergence of the adversarial training [1]. In other words, the generator fails to learn anything useful to improve itself.

One of our major contributions is to propose a small but very effective modification to the data processing step. We perform a fuzzy binary coding of categorical features, i.e., we encode the binary values using real numbers between 0 and 1, while retaining the category information. In this way we guarantee that from the very beginning of the adversarial training, values produced by the generator already resemble the real values in their domain. To this end, the discriminator can not "cheat" and exploit the simple fact that real data are all binary while generated data are all real. The discriminator can only focus on the true and informative characteristics of the real and generated data, such as the dependency between features. Thus, the generator can receive more useful gradient updates from the discriminator, which improves the data generation process.

The rest of this paper is organized as follows. In section II we provide an overview of related works in three research fields of GANs: generation of categorical data, application GANs for EHR data and for data imputation. After a brief introduction to the GAN framework in section III, we elaborate the methods we propose in detail, including the fuzzy binary coding and the GAN for categorical data generation in section IV. In section V we present our experimental results on two EHR datasets and show that imputation based on the GAN framework with fuzzy binary coding can be quite effective in dealing with missing categorical data in EHRs.

II. Related Works

GANs that generate categorical features: There are currently three different approaches for generating categorical features with GANs. The first approach modifies the output activation function in the generator so that the gradient can flow from the discriminator to the generator while the latter generates pseudo-discrete features. Examples are so-called Gumbel Softmax [9], [10] or a soft argmax function [28]. The second approach modifies the training objectives. [12], [27] apply REINFORCE [24] algorithm for adversarial training. The third approach, including [2], [29], learns a mapping from the raw discrete feature space to a latent real space, as well as the reverse mapping. These mapping functions are, e.g., realized as an auto-encoder. With the first mapping one transforms all training data that are originally categorical into real representations. Then, the GANs framework only has to operate in this real space, learning to generate real feature vectors. As a post processing step, the generated vectors are

transformed back into the discrete space using the second mapping function.

GANs in EHR data analysis: GANs have already found various interesting applications in healthcare. [2] and [5] aim at generating pseudo-synthetic EHR data for the purpose of de-identification. The former focuses on the challenge of generating categorical features by applying an auto-encoder that can map between the discrete feature space and a real latent space. It is pointed out that applying differentiable Gumbel softmax or soft argmax functions does not completely solve the categorical problem, because patient features could be multinomial (i.e., multiclass) as well as multiple Bernoulli distributed (i.e., multi-label). The latter paper develops GANs that are based on Recurrent Neural Networks (RNN) to generate high dimensional time series EHR data.

Missing Data Imputation using Generative Adversarial Nets: [26] adjusted the GAN framework for the specific task of data imputation. It can be interpreted as a special case of conditional GAN, in the sense that both discriminator and generator take as input a mask vector indicating the missingness of feature values. It is shown that this novel training framework can efficiently impute real-valued features, especially in case where the missing rate is relatively high. Our method is largely inspired by this work, but we focus on the specific techniques to perform adversarial imputation of categorical features.

III. Preliminary: The Generative Adversarial Nets Framework

In its simplest case, a GAN framework [8] consists of two neural networks. The first one is often referred to as the *generator*, which consumes as input some random seeds $\boldsymbol{r}$ and generate data instances $\boldsymbol{g}$ that are supposed to resemble real data $\boldsymbol{x}$. The generator can be seen as a function of

$$\boldsymbol{g} = G(\boldsymbol{r}|\Theta_G). \tag{1}$$

Each generated sample is provided to the second neural network, the *discriminator*, i.e., $D(\boldsymbol{g}|\Theta_D)$. The discriminator also consumes as input the real data samples as $D(\boldsymbol{x}|\Theta_D)$. The training of the generator and the discriminator is adversarial, in that, while the discriminator is trained to correctly classify a sample to be either real or generated, the generator learns to fool the discriminator so that it classify generated samples to be real. More specifically, in term of the log-loss function $\mathcal{H}(a,b) = b\cdot\log(a) + (1-b)\cdot\log(1-a)$, we can write the discriminator loss and the generator loss as

$$\begin{aligned} loss_D =& -\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}_{\text{real}}}\mathcal{H}(D(\boldsymbol{x}|\Theta_D),1) \\ & -\mathbb{E}_{\boldsymbol{r}\sim\mathcal{P}_{\text{seed}}}\mathcal{H}(D(\boldsymbol{g}|\Theta_D),0) \\ =& -\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}_{\text{real}}}\log(D(\boldsymbol{x}|\Theta_D)) \\ & -\mathbb{E}_{\boldsymbol{r}\sim\mathcal{P}_{\text{seed}}}\log(1-D(\boldsymbol{g}|\Theta_D)) \\ loss_G =& -\mathbb{E}_{\boldsymbol{r}\sim\mathcal{P}_{\text{seed}}}\mathcal{H}(D(\boldsymbol{g}|\Theta_D),1) \\ =& -\mathbb{E}_{\boldsymbol{r}\sim\mathcal{P}_{\text{seed}}}\log(D(\boldsymbol{g}|\Theta_D)) \end{aligned} \tag{2}$$

respectively. With sufficient training, D will not be able to differentiate between real and generated samples by assigning neutral values to both cases. G will learn to map random seeds from an arbitrary distribution $\mathcal{P}_{\text{seed}}$ to the underlying distribution of the real data $\mathcal{P}_{\text{real}}$. Denoted as $\hat{\mathcal{P}}_{\text{real}}$, this estimate of the real data distribution allows for unlimited sampling.

IV. Method

In this section we give a detailed introduction to our method, which consists of two major components: the fuzzy binary coding and a modification of the Generative Adversarial Imputation Nets [26] for categorical feature generation.

A. Fuzzy binary coding

It is important to distinguish between *multinomial* and *multi-Bernoulli* distributed categorical features. In the former case, the random variable is realized by taking only one single category, i.e., the categories are mutually exclusive. For instance, the estrogen-receptor status of a patient could only be either positive, negative or unknown. In machine learning, especially if such features appear as targets, they are often referred to as *multiclass* features and modelled with the softmax function. In the *multi-bernoulli* case, a categorical feature can realize more than one categories, such as the location of metastasis, which could be multiple organs at the same time, or multiple (serious) adverse events (AE/SAE) could be triggered by certain treatment. For such a feature with non-mutual exclusive categories, one often use the term *multilabel*. For a concise terminology, we adopt the convention from machine learning and refer to these two cases as multiclass and multilabel features for the rest of the paper.

Assume that we observe p categorical features on one data instance and the j-th feature is a multiclass one, denoted as

$$\xi_j \in \Omega_j \text{ where } |\Omega_j| = q_j \tag{3}$$

As the first step, we perform regular binary coding $\xi_j \rightarrow \boldsymbol{z}_j \in \{0,1\}^{q_j}$. We use the term *inactive category* to refer to a category that is represented by 0; and an *active category* is represented by a 1. It is easy to see that the sum of all elements in $\boldsymbol{z}_j$ is strictly 1 if ξ_j is of multiclass, and could be $\mathbb{N}_0$ if ξ_j is a multilabel feature. These binary codings are also known as one-hot and multi-hot encodings, respectively.

In the second step, we transform the binary coded variable $\boldsymbol{z}_j$ in its fuzzy representation.

a) Multiclass case: We propose a transformation denoted as $f(\cdot)$ of $\boldsymbol{z}_j$ as:

$$\boldsymbol{x}_j(k) = f(\boldsymbol{z}_j(k)) = \begin{cases} \mathcal{U}[0, \frac{1}{q_j}) & \forall k : \boldsymbol{z}_j(k) = 0, \\ 1 - \sum_k \boldsymbol{x}_j(k) & \text{for } k : \boldsymbol{z}_j(k) = 1. \end{cases} \tag{4}$$

Please note that we use $\boldsymbol{x}_j(k)$ to denote the k-th element in the vector $\boldsymbol{x}_j$, in order to avoid double subscripts; $\boldsymbol{\mathcal{U}}[a,b)$ denotes a continuous uniform distribution in the interval of $[a,b)$. Assuming any active category to be k^*, then each of the $q_j - 1$ inactive categories is represented by a fraction $\boldsymbol{x}_j(k)$ which is uniformly sampled from $[0, \frac{1}{q_j})$. With this smoothing,

we can retain exactly the same information encoded in $\boldsymbol{z}_j$. It is easy to see that,

$$1 - \sum_{\forall k \neq k^*} \boldsymbol{x}_j(k) > \frac{1}{q_j}. \tag{5}$$

In other words, the left side of the inequation (5), which represents the active category k^*, is guaranteed to be larger than any fraction encoding an inactive category. Operations such as max, min, $argmax$ and $argmin$ applied on the fuzzy $\boldsymbol{x}_j$ are always able to decode the same information in $\boldsymbol{z}_j$.

b) Multilabel case: Since the categories are no more mutual exclusive, we can derive a fuzzy binary coding by simply taking 0.5 instead of $\frac{1}{q_j}$ as the upper bound of uniform sampling:

$$\boldsymbol{x}_j(k) = f(\boldsymbol{z}_j(k)) = \begin{cases} \mathcal{U}[0, 0.5) \text{ for } \boldsymbol{z}_j(k) = 0, \\ \mathcal{U}[0.5, 1] \text{ for } \boldsymbol{z}_j(k) = 1. \end{cases} \tag{6}$$

It is also guaranteed that the category information in $\boldsymbol{z}_j$ remains intact, since we can always recover $\boldsymbol{z}_j$ applying $I(\boldsymbol{x}_j \geq 0.5)$, where $I(\cdot)$ denotes the indicator function.

Transforming the binary codes into fuzzy binary codes prevents the discriminator from exploiting the fact that the generated values are all fractions and the real values only contain 0's and 1's. This fuzzy binary coding, especially the samplings in Eq. (4) and (6), can be performed only once as pre-processing step, or alternatively, prior to each training epoch. In our experiments, we implement the first variant.

In Fig. 3 we provide some empirical results based on our experiments, demonstrating that without the fuzzy binary coding trick, the adversarial training tends to diverge, i.e., the discriminator keeps improving itself by exploiting the obvious difference between the generated and real data. The generator, therefore, receives no gradients from the discriminator for improvement.

Applying the fuzzy encoding, the discriminator can be forced to focus on discovering the true difference between the real and generated data in term of their distributions and dependencies instead of their different domains. These discoveries in turn shall encourage the generator to approximate the real data distribution.

Lastly, we concatenate the feature vectors of all categorical features as

$$\bar{\boldsymbol{x}} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_p] = [\boldsymbol{x}_j]_{j=1}^{p} \in [0, 1]^{\sum_{j=1}^{p} q_j}, \tag{7}$$

which form the inputs to the generative adversarial imputation network. Note here that we do not use another subscript denoting the data instance in $\boldsymbol{x}$, and simply assume that they are all i.i.d. samples.

B. Categorical Generative Adversarial Imputation Nets (Categorical GAINs)

a) Data notation: In order to represent the missingness of data, [26] introduced a binary mask vector $\boldsymbol{m}$ indicating which features are missing in a data instance represented by a real vector $\boldsymbol{\xi}$. Here $\boldsymbol{m}$ and $\boldsymbol{\xi}$ have exactly the same size and each element $\boldsymbol{m}(k)$ is 1 if $\boldsymbol{\xi}(k)$ is not missing, and 0 otherwise.

In case of categorical features, however, we introduce two masking mechanisms. Firstly, we use $\boldsymbol{\mu}$ to denote the missingness of $\boldsymbol{\xi}$, i.e.,

$$\mu_j = \begin{cases} 0 & \text{if } \xi_j \text{ is missing,} \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

Once the features are binary and fuzzy coded, we construct another mask vector as:

$$\boldsymbol{m}_j = \begin{cases} \mathbf{0} & \text{if } \xi_j \text{ is missing,} \\ \mathbf{1} & \text{otherwise} \end{cases} \in [0, 1]^{q_j}, \tag{9}$$

where we denote a *vector* of 0's and 1's using $\mathbf{0}$ and $\mathbf{1}$, respectively. It can be interpreted as simply repeating μ_j for q_j times for the j-th feature. The rationale for these two kinds of masking is that the discriminator's prediction is in fact equivalent to the missingness of the data. For real-valued features discussed in [26], one could simply reuse the masking vector as the target of the discriminator. But for categorical features that are coded as binary or fuzzy binary, doing so would imply making a prediction for each single *category* instead of each *feature*. In the following introduction to the generator and discriminator, we shall give a more detailed explanation.

Analogously to the construction of $\bar{\boldsymbol{x}}$, we have the concatenation of $\boldsymbol{m}_j$'s:

$$\bar{\boldsymbol{m}} = [\boldsymbol{m}_1, \boldsymbol{m}_2, ..., \boldsymbol{m}_p] = [\boldsymbol{m}_j]_{j=1}^{p} \in [0, 1]^{\sum_{j=1}^{p} q_j} \tag{10}$$

b) The generator: The generator takes as input i) the fuzzy binary coded feature vector $\bar{\boldsymbol{x}}$ that is expected to contain missing values, ii) the equally sized mask vector $\bar{\boldsymbol{m}}$ and iii) a random vector $\bar{\boldsymbol{r}} = [\boldsymbol{r}_j]_{j=1}^{p}$ functioning as seeds. The generator produces as output a single vector denoted $\bar{\boldsymbol{g}}$ that is supposed to contain imputed missing values in $\bar{\boldsymbol{x}}$:

$$\bar{\boldsymbol{g}} = G(\bar{\boldsymbol{x}}, \bar{\boldsymbol{m}}, \bar{\boldsymbol{r}}). \tag{11}$$

Specifically in our implementation, we build as generator a neural network with 3 hidden layers:

$$\boldsymbol{h}_1^G = \text{relu}(\boldsymbol{W}_1^G \cdot [\bar{\boldsymbol{x}} + (1 - \bar{\boldsymbol{m}}) \circ \bar{\boldsymbol{r}},\ \bar{\boldsymbol{m}}] + \boldsymbol{b}_1^G) \tag{12}$$

$$\boldsymbol{h}_2^G = \text{relu}(\boldsymbol{W}_2^G \cdot \boldsymbol{h}_1^G + \boldsymbol{b}_2^G) \tag{13}$$

$$\boldsymbol{h}_3^G = \text{relu}(\boldsymbol{W}_3^G \cdot \boldsymbol{h}_2^G + \boldsymbol{b}_3^G) \tag{14}$$

$$\boldsymbol{g}_j = \sigma(\boldsymbol{W}_o^G(j) \cdot \boldsymbol{h}_3^G + \boldsymbol{b}_o^G(j))\ \forall j \in [1, p] \tag{15}$$

$$\bar{\boldsymbol{g}} = \bar{\boldsymbol{m}} \circ \bar{\boldsymbol{x}} + (1 - \bar{\boldsymbol{m}}) \circ [\boldsymbol{g}_j]_{j=1}^{p} \tag{16}$$

As proposed by [26], the operation carried out in Eq. (12) first fills the missing values in $\bar{\boldsymbol{x}}$ with random seeds $\bar{\boldsymbol{r}}$, before feeding it to the neural network. The hidden layers $\boldsymbol{h}_1^G, \boldsymbol{h}_2^G, \boldsymbol{h}_3^G$ extract hierarchically the global context information from the input. In the last layer, we define for each categorical feature j a specific classification model. Depending on the distribution assumption of the feature, the activation function can be either sigmoid or softmax, both of which are denoted using σ for the sake of simplicity. In Eq. (16), the outputs from all p

activation functions are concatenated as $[\boldsymbol{g}_j]_{j=1}^p$. And if a specific feature is in fact not missing, the generated values are replaced by the real values. Similar to the Multiple Imputation by Chained Equations [21], this generator in fact attempts to approximate the real distribution $\boldsymbol{\pi}_{j^*}$ of each missing variable $\boldsymbol{X}_{j^*}$ conditioned on all other observed features $\boldsymbol{x}_j$, i.e.,

$$\hat{\boldsymbol{\pi}}_{j^*} = \boldsymbol{g}_{j^*} = \mathbb{P}(\boldsymbol{m}_{j^*} = \mathbf{1} \mid \{\boldsymbol{X}_j = \boldsymbol{x}_j\}_{\forall j:\mu_j=1}) \tag{17}$$

This architecture is illustrated in Fig. 1 with only two categorical features as examples.

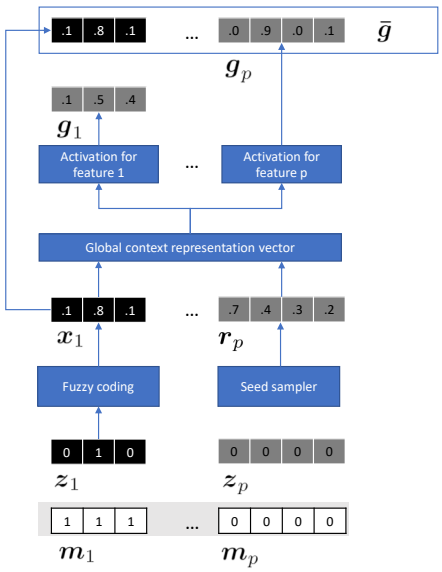


Fig. 1. A detailed illustration of the generator in *categorical GAIN* architecture. As an example, we visualize two categorical features that are binary coded as $\boldsymbol{z}_1$ and $\boldsymbol{z}_p$. The first is labeled as observed while the second as missing in $\boldsymbol{m}_1$ and $\boldsymbol{m}_p$. Therefore, the observed binary input of $\boldsymbol{z}_1 = [0, 1, 0]$ is transformed into a fuzzy representation of $\boldsymbol{x}_1 = [0.1, 0.8, 0.1]$. And the seed sampler fills these positions with random values in $\boldsymbol{r}_p$. On the output side, the generated values for the first feature are replaced by the original, fuzzy codes, since the true values are observed, i.e., $\boldsymbol{g}_1 := \boldsymbol{x}_1$. Only the generated values for the other feature $\boldsymbol{g}_p$ are exposed to the discriminator. The concatenation of the overall generator output is denoted as $\bar{\boldsymbol{g}}$.

c) The discriminator: Like the generator, the discriminator also consumes two input vectors. The first input is the concatenated output of the generator $\bar{\boldsymbol{g}}$. The second input is a hint vector as proposed by [26], which can be interpreted as a *masked mask vector*: For each data instance, one randomly samples a predefined portion of features and sets the corresponding entries in the mask vector $\bar{\boldsymbol{m}}$ to be 0.5. On the output side of the discriminator, we have again an concatenated vector $\hat{\boldsymbol{\mu}} = [\hat{\mu}]_{j=1}^p$. Each $\hat{\mu}_j$ is a point estimate of μ_j as defined in Eq. (8), indicating whether the j-th feature in the input, denoted as $\boldsymbol{g}_j$ is generated or real,

$$\hat{\boldsymbol{\mu}}_{j^*} = \mathbb{P}(\boldsymbol{g}_{j^*}\text{is real} \mid \{\boldsymbol{g}_j\}_{\forall j:j\neq j^*}). \tag{18}$$

Generally, we can describe the discriminator as

$$\hat{\boldsymbol{\mu}} = D(\bar{\boldsymbol{g}}, \bar{\boldsymbol{h}}). \tag{19}$$

Specifically for our experiments, we have a neural network with two hidden layers:

$$\boldsymbol{h}_1^D = \text{relu}(\boldsymbol{W}_1^D \cdot [\bar{\boldsymbol{g}}, \bar{\boldsymbol{h}}] + \boldsymbol{b}_1^G) \tag{20}$$

$$\boldsymbol{h}_2^D = \text{relu}(\boldsymbol{W}_2^D \cdot \boldsymbol{h}_1^D + \boldsymbol{b}_2^D) \tag{21}$$

$$\hat{\mu}_j = \sigma(\boldsymbol{w}_o^D(j)^T \cdot \boldsymbol{h}_3^G + b_o^D(j)) \ \forall j \in [1, p] \tag{22}$$

In parallel to the architecture of the generator, the first two hidden layers represent the global context information, while the last layer contains p logistic regression models. Each of them attempts to predict whether the j-th feature in the input $\boldsymbol{g}_j$ is generated. In the original GANs, each input vector to the discriminator is typically either generated or real. In GAIN, however, one input vector to the discriminator may contain generated and real data simultaneously, and the discriminator performs multiple predictions correspondingly.

In the original setting in [26], where the features are of real values, the training target of the discriminator is in fact identical with the mask vector. In case of categorical features, however, one should not directly utilize the mask vector $\bar{\boldsymbol{m}}$ as training target. Because in order to mask a (fuzzy) binary coded vector $\boldsymbol{x}_j$ completely, we have to define a same sized vector $\boldsymbol{m}_j$. Training a discriminator that attempts to recover every element in $\boldsymbol{m}_j$ is in fact a prediction for each *category* instead of *feature*. To this end, we propose to train the discriminator so that each $\hat{\mu}_j$ would approximate μ_j as in Eq. (8) for all real data. The generator, on the other hand, should make the discriminator assign a $\hat{\mu}_j$ that is close to $1 - \mu_j$ to all generated values.

The hint mechanism is also a crucial component in training the discriminator. Once a subset of entries in the mask vector is set to a neutral value of 0.5, the discriminator is enforced to predict whether the corresponding values in $\bar{\boldsymbol{g}}$ are real or generated. Such prediction is supposed to rely on other entries in $\bar{\boldsymbol{g}}$ that are provided to discriminator. The proportion of features that are neutralized in the hint vector therefore controls the amount of information from which the discriminator is supposed to learn the decision. In order to see that one could consider two extreme cases: With the proportion close to 1, the discriminator would attempt to perform prediction for a large amount of features in $\bar{\boldsymbol{g}}$, based on very few features that are denoted as either real or generated. This could be a challenging task for the discriminator and, more importantly, the discriminator may not learn to build the prediction based on the dependency between features. With a proportion that is close to 0, the hint vector becomes almost identical to the mask vector. In the original setting in [26], where features are of real values and the mask vector is in fact the prediction target of the discriminator, having two almost identical vectors as input and output of a neural network would cause the discriminator to simply learn an identity function, not being able to tell the difference between real and generated data. This is slightly less of a problem in case of categorical features, because as stated above, our mask vector as input to the discriminator and training target are not exactly identical, although they contain the same information on the missingness of the data.

To this end, for experiments, we include the hint mechanism and use a relatively small proportion of 0.1. This reveals 90% of available information of the data missingness to the discriminator, which is encouraged to build its prediction based on the dependency among features.

The hint vector also has to be adjusted for categorical features. Similar to the mask vectors, we define for each categorical feature j a hint vector $\boldsymbol{h}_j$ that consists of exclusively either 0 or 1, and denote the concatenation of all hint vectors as $\bar{\boldsymbol{h}} = [\boldsymbol{h}_j]_{j=1}^{p}$. The proposed approach in [26] would imply masking the missingness information for each *category* instead of each *feature*. To this end, we propose to first sample a subset out of the p features, and set the entire corresponding hint vectors to be 0.5, i.e. $\boldsymbol{h}_j = \mathbf{0.5}$, as can be seen in the illustration in Fig. 2

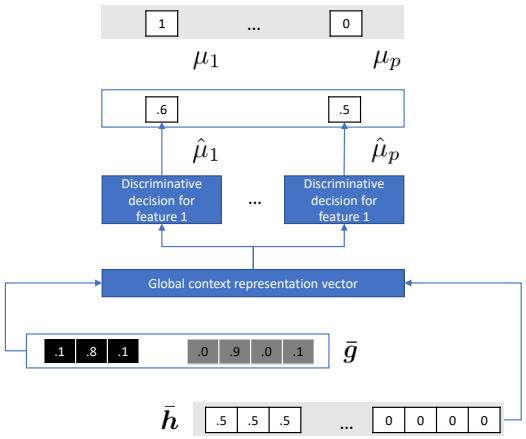


Fig. 2. A detailed illustration of the discriminator in categorical GAIN architecture. The input to the discriminator is the concatenated output $\bar{\boldsymbol{g}}$ from the generator, as well as the hint vector $\bar{\boldsymbol{h}}$. The output of the discriminator here consists of 2 scalars of $\hat{\mu}_1$ and $\hat{\mu}_p$. They are trained against the scalars μ_1 and μ_p, encoding the missingness of both features respectively.

d) Loss functions: Similar to the original GANs framework as in Eq. (2), GAIN also contains two adversarial loss functions $loss_D$ and $loss_G$:

$$loss_D = -\Sigma_j \left(\mu_j \cdot \log(\hat{\mu}_j) + (1-\mu_j) \cdot \log(1-\hat{\mu}_j)\right) \quad (23)$$

$$loss_G = -\Sigma_j (1-\mu_j) \cdot \log(\hat{\mu}_j) \quad (24)$$

The discriminator adjusts itself to make correct classification by minimizing the $loss_D$ in Eq. (23). This objective forces the discriminator to produce large $\hat{\mu}_j$ if $\mu_j = 1$, indicating that $\boldsymbol{x}_j$ is real. The generator learns to fool the discriminator by minimizing $loss_G$ in Eq. (24). This loss is adversarial to the second additive term in the discriminator loss. The generator encourages the discriminator to assign large probability $\hat{\mu}_j$ to features where $\mu_j = 0$, implying that the generated data should be classified as real.

As defined in Eq. (16), once a feature j is observed instead of missing, whatever is generated by the generator gets replaced by the actually observed values. The weight parameters responsible for these features will not get gradient signals for this specific training sample. Therefore, [26] proposes a new loss function that measures the similarity between generated and the observed feature values. In case of real-valued features this loss could be realized as mean-squared error. In our case, we apply the log-loss to measure the distance between probabilities and binary codes:

$$loss_{sim} = \Sigma_j \boldsymbol{m}_j^T (-\boldsymbol{x}_j) \log(\boldsymbol{g}_j). \quad (25)$$

This loss mechanism implies that, in case a feature is observed, the generator should learn to reproduce it based on all other observed features; and in case a feature is missing, the adversarial training forces the generator to produce values that the discriminator would believe to be real.

In comparison to the original GAIN architecture in [26], there are three adjustments that we propose for categorical features. First, the output activation function in the generator: in order to take into account the discrete distribution of the data features, we apply softmax or sigmoid activation functions instead of linear activation. Second, the target variable of the discriminator: In case of real valued features, the discriminator only needs to predict the mask vector $\bar{\boldsymbol{m}}$ which has the same shape as the feature vector $\bar{\boldsymbol{x}}$. This is because each element in the mask vector can represent the missingness of the corresponding feature. However, in order to encode the missingness of a feature containing multiple categories $\boldsymbol{x}_j$, it is unnecessary for the discriminator to recover the corresponding mask vector $\boldsymbol{m}_j$, since all values in this vector are either all 0's or all 1's. Instead, it is much more efficient to train the discriminator to predict the scalar μ_j. Thirdly, due to the same reason, the hint mechanism also has to be defined on the level of feature instead of categories. In other words, for a feature j we initialize a vector $\boldsymbol{h}_j$ from $\boldsymbol{m}_j$, and set all elements to be 0.5 if necessary.

V. Experiments

In this section, we provide experiments conducted on two datasets. The first dataset is publicly available and a well known benchmark for breast cancer classification based on categorical features. The second dataset is provided by the PRAEGNANT study [7], a Germany-wide clinical study for breast cancer research.

Please recall that we perform fuzzy binary coding of the categorical features and our generator produces values that range between 0 and 1. We recover the binary codes applying $I(\boldsymbol{x}_j \geq 0.5)$ for multilabel and $I(\boldsymbol{x}_j = \max(\boldsymbol{x}_j))$ for multiclass features as a post processing step. Because, as discussed in subsection IV-A, the encoded categorical information is always retained after the fuzzy binary coding and can be recovered completely.

A. Experiments on a public dataset

The breast cancer dataset is available on UCI data repository [3]. It contains 9 multiclass features (Tab. I) observed on 286 patient cases. The prediction target is to differentiate between 201 recurrence and 85 no-recurrence cases of the cancer.

Feature	#Categories
age	6
menopause	3
tumor-size	11
inv-nodes	7
node-caps	2
deg-malig	3
breast	2
breast-quad	5
irradiat	2

TABLE I
PATIENT FEATURES FROM THE UCI BREAST CANCER DATASET

We perform 5-fold cross-validation on the complete dataset, applying logistic regressions with ridge regularization (Tab. II) and report the prediction accuracy and AUROC scores. As sanity check we also provide these scores produced by random and most popular predictions, the latter of which constantly produces the frequency of the label class in the training set.

Methods	Accuracy	AUROC
Random prediction	0.516 ± 0.051	0.484 ± 0.053
Most popular prediction	0.707 ± 0.051	0.500 ± 0
Prediction on complete data	0.737 ± 0.056	0.721 ± 0.051

TABLE II
SANITY CHECKS FOR THE PREDICTION TASK ON THE UCI BREAST CANCER DATASET

For each cross-validation split, we randomly mask 10%, 20%, 30% 40% and 50% of the features. We then apply different imputation approaches to recover the masked values. Note that the imputation model is only trained on the training set, and applied on both training and test sets, in order to simulate a realistic setting. The predictive model is then trained on *imputed* training set and validated on the *imputed* test set.

As the first baseline method we implement a low-rank reconstruction model using SVD. Assuming $\boldsymbol{X}_{tr}$ and $\boldsymbol{X}_{te}$ as training and test sets containing missing values, we compose the former as $\boldsymbol{X}_{tr} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$, and impute the training and test sets as $\tilde{\boldsymbol{X}}_{tr} = \boldsymbol{U}_r\boldsymbol{D}_r\boldsymbol{V}_r^T$ and $\tilde{\boldsymbol{X}}_{te} = \boldsymbol{X}_{te}(\boldsymbol{D}_r\boldsymbol{V}_r^T)^\dagger(\boldsymbol{D}_r\boldsymbol{V}_r^T)$, respectively. Here we denote the the low rank representation of $\boldsymbol{U}, \boldsymbol{D}$ and $\boldsymbol{V}^T$ using $\boldsymbol{U}_r, \boldsymbol{D}_r$ and $\boldsymbol{V}_r^T$ with a specific rank r. Please note that we do not perform any *argmax* to the reconstructed values.

The same ranks also apply to the second baseline model, which is an auto-encoder with non-linear $\tanh$ activation for the hidden layer. We summarize all prediction performances in term of accuracy and AUROC scores in Tab. III as average and standard deviation of the 5 cross-validation splits. For both baseline models we conduct experiments using 4 different ranks, i.e., the size of the hidden layer in AE, of 4, 8, 16 and 32, and report the best results. For the categorical GAIN model we perform 100-fold multiple imputation.

In term of accuracy, SVD reconstruction turns out to be more effective for this dataset, achieving the best accuracy in 4 out of 5 settings of masking proportions. In term of AUROC, categorical GAIN achieves 4 out of 5 cases. It is therefore interesting to note that for this dataset, the SVD decomposition

	Methods	Accuracy	AUROC
10%	No imputation	0.718 ± 0.067	0.66 ± 0.11
	Avg imputation	0.744 ± 0.05	0.639 ± 0.088
	SVD reconstruction	**0.776** ± 0.062	0.689 ± 0.114
	Auto-encoder	0.751 ± 0.047	0.652 ± 0.089
	Categorical GAIN	0.739 ± 0.066	**0.697** ± 0.098
20%	No imputation	0.711 ± 0.039	0.634 ± 0.07
	Avg imputation	0.707 ± 0.036	0.671 ± 0.065
	SVD reconstruction	**0.747** ± 0.046	0.664 ± 0.082
	Auto-encoder	0.729 ± 0.051	0.636 ± 0.038
	Categorical GAIN	0.71 ± 0.046	**0.697** ± 0.087
30%	No imputation	0.726 ± 0.031	0.644 ± 0.086
	Avg imputation	0.729 ± 0.04	0.665 ± 0.071
	SVD reconstruction	0.726 ± 0.053	0.689 ± 0.083
	Auto-encoder	0.726 ± 0.038	0.641 ± 0.053
	Categorical GAIN	**0.737** ± 0.032	**0.704** ± 0.042
40%	No imputation	0.678 ± 0.052	0.686 ± 0.058
	Avg imputation	0.708 ± 0.027	0.54 ± 0.066
	SVD reconstruction	**0.751** ± 0.026	**0.709** ± 0.059
	Auto-encoder	0.737 ± 0.033	0.638 ± 0.054
	Categorical GAIN	0.7 ± 0.017	0.686 ± 0.051
50%	No imputation	0.701 ± 0.044	0.607 ± 0.091
	Avg imputation	0.704 ± 0.044	0.632 ± 0.057
	SVD reconstruction	**0.747** ± 0.029	0.665 ± 0.063
	Auto-encoder	0.74 ± 0.034	0.635 ± 0.041
	Categorical GAIN	0.713 ± 0.025	**0.72** ± 0.044

TABLE III
PREDICTION PERFORMANCES ON IMPUTED UCI BREAST CANCER DATASET USING DIFFERENT APPROACHES

does not take into account the the fact that the feature values are in fact binary And yet the SVD reconstruction achieves comparable performances as categorical GAIN. This relatively simple technique, as well as many approaches that it has inspired, are widely applied in recommender systems and knowledge graph, where the most essential task is the completion of matrices and tensors. Therefore, it could very well present a simple and effective solution for data imputation as well. However, one should also note that the label distribution in this dataset is relatively unbalanced (201:85). Consequently, the most popular prediction as in Tab. II can already reach 70% accuracy. And even with complete data the prediction model cannot improve beyond 73.7%. The AUROC, in contrast, seems to be a more informative and convincing measurement, because the prediction on complete data achieves 72% while the most popular prediction 50%. Therefore, for this dataset, ROC seems to be a more reliable means to measure the prediction quality.

B. Experiments on the PRAEGNANT dataset

1) Cohort and Features: For our experiment, we extract EHR data on 1234 patients with metastatic breast cancer who have met the first line of treatment from the PRAEGNANT study network [7]. We build our predictive models based on features that are clinically relevant, as well as those that are based on an earlier study [25] aiming at automatically inferring the feature relevance in EHR data. The features included are listed in Tab. IV. We have 10 multiclass features and 9 multi-label featues, both of which are fuzzy-binary coded. The one

numeric feature is normalized between 0 and 1. Features such as current metastasis, metastasis estrogen receptor, metastasis progesterone receptor, AE/SAE and ECOG life status were originally temporal features. We aggregate and normalize these w.r.t the time dimension as in [6]. For these patients it is especially important for the physicians to decide, whether they should receive antihormone therapy or chemo therapy. The recorded clinical decision serve as ground truth, i.e. the target of our prediction. 750 of the 1234 patients have received antihormone, and the rest chemo therapy.

Multiclass features	#Categories
Staging at breast	15
Staging at axilla	8
Ever received antihormone therapy	8
Ever received chemo therapy	8
Metastasis by diagnostics	5
Tumor estrogen receptor status	4
Tumor progesterone receptor	4
Immunohistochemistry for HER2	6
Tumor grading	5
KI67	3
Multilabel features	**#Categories**
Staging of metastasis	10
Location of earlier metastasis	14
Current metastasis	4
Metastasis estrogen receptor	3
Metastasis progesterone receptor	3
HER2 IHC	5
Metastasis grading	4
AE/SAE	20
ECOG life status	4
Numerical features	**#Dimension**
Age	1

TABLE IV
PATIENT FEATURES FROM THE PRAEGNANT STUDY.

Here we apply almost exactly the same experimental setting as with the public dataset, except that, considering the feature space of higher dimension, we train the SVD and auto-encoder imputation models with an additional rank of 64.

Methods	Accuracy	AUROC
Random prediction	0.516 ± 0.029	0.526 ± 0.041
Most popular prediction	0.607 ± 0.046	0.500 ± 0
Prediction on complete data	0.710 ± 0.029	0.774 ± 0.039

TABLE V
SANITY CHECKS FOR THE PREDICTION TASK ON THE PRAEGNANT DATASET

2) Experimental Results: In Tab. V we could see there is a large improvement from most popular prediction to the prediction on complete data in term of both accuracy and AUROC.

In Tab. VI we could see that, the advantage of categorical GAIN only becomes visible as the masking proportion increases. For smaller proportion like 10% and 20%, simpler methods such as average imputation and SVD shows superior performances. With a proportion larger than 30%, categorical GAIN outperforms all other methods and the improvement grows with masking proportion. In term of AUROC, for instance, categorical GAIN can always achieve a score above

	Methods	Accuracy	AUROC
10%	No imputation	0.674 ± 0.017	0.718 ± 0.016
	Avg imputation	0.689 ± 0.008	$\mathbf{0.727} \pm 0.024$
	SVD reconstruction	$\mathbf{0.7} \pm 0.015$	0.645 ± 0.011
	Auto-encoder	0.609 ± 0.023	0.506 ± 0.022
	Categorical GAIN	0.645 ± 0.012	0.725 ± 0.024
20%	No imputation	0.669 ± 0.014	0.69 ± 0.011
	Avg imputation	$\mathbf{0.684} \pm 0.015$	0.707 ± 0.014
	SVD reconstruction	0.663 ± 0.025	0.621 ± 0.032
	Auto-encoder	0.609 ± 0.021	0.496 ± 0.03
	Categorical GAIN	0.649 ± 0.016	$\mathbf{0.716} \pm 0.022$
30%	No imputation	0.645 ± 0.03	0.696 ± 0.018
	Avg imputation	0.658 ± 0.039	0.695 ± 0.021
	SVD reconstruction	0.662 ± 0.04	0.599 ± 0.018
	Auto-encoder	0.609 ± 0.043	0.528 ± 0.017
	Categorical GAIN	$\mathbf{0.665} \pm 0.018$	$\mathbf{0.723} \pm 0.01$
40%	No imputation	0.652 ± 0.008	0.663 ± 0.009
	Avg imputation	0.643 ± 0.012	0.66 ± 0.014
	SVD reconstruction	0.658 ± 0.01	0.6 ± 0.017
	Auto-encoder	0.608 ± 0.017	0.494 ± 0.034
	Categorical GAIN	$\mathbf{0.666} \pm 0.017$	$\mathbf{0.711} \pm 0.015$
50%	No imputation	0.635 ± 0.027	0.646 ± 0.029
	Avg imputation	0.649 ± 0.041	0.643 ± 0.038
	SVD reconstruction	0.644 ± 0.015	0.566 ± 0.018
	Auto-encoder	0.608 ± 0.013	0.509 ± 0.038
	Categorical GAIN	$\mathbf{0.654} \pm 0.05$	$\mathbf{0.705} \pm 0.029$

TABLE VI
PREDICTION PERFORMANCES ON IMPUTED PRAEGNANT DATASET USING DIFFERENT APPROACHES

70%, while the other performance of other methods drop much faster as the proportion of missing data increases. This agrees with findings in [26], that it is especially advantageous to apply GAIN to impute data in case of a relatively higher missing rate.

One might also hypothesize that the GAIN framework, consisting of relatively complex neural networks, profit from increasing number of training samples. For a smaller dataset such as the public breast cancer dataset, it seems more reasonable to first experiment with simpler methods such as SVD reconstruction. The GAIN approach, on the other hand, turns out to be more appropriate in case of large number of training samples and more complex feature dependencies.

We also present in Fig. 3 the development of the losses of discriminator (top) and generator (bottom), trained on binary (left) and fuzzy coded (right) features. In case of plain binary coded features, it is clear that the adversarial training fails since the generator loss increases, while the discriminator loss decreases constantly. This implies that the discriminator can always tell the real data from generated ones. Consequently, the generator cannot improve itself by learning to generate important characteristics in the feature distribution. When we apply the fuzzy binary coding, in contrast, the generator can improve itself by lowering its loss, i.e., it gets harder and harder for the discriminator to make the decision. In addition, as expected, varying proportion of missing data (masking) has impact on the adversarial training losses. With larger proportion of missing data, the imputation task becomes more

challenging and both discriminator loss and generator loss are expected to increase with larger proportion. This verifies empirically our hypothesis, that, if one applies softmax as the final activation in the generator to generate categorical data, the adversarial training fails as the discriminator can learn to exploit the huge difference in the generated and real data. This typically results in divergence of the adversarial training. By re-coding the binary features in a fuzzy way while retaining the information, we enforce the real data to resemble what softmax would produce. Thus we can make both discriminator and generator converge in training.

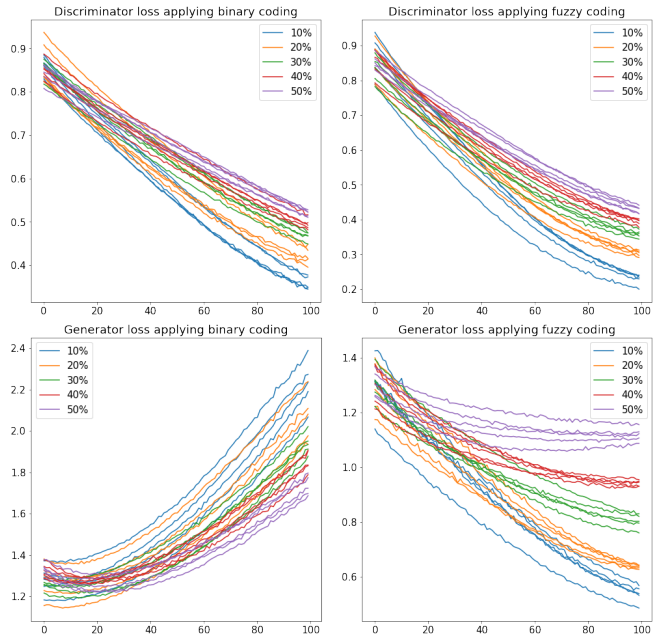


Fig. 3. Losses in adversarial training on the PREAGNANT dataset. X-axis: training epochs; Y-axis: adversarial loss. Above: Discriminator losses with binary coding (left) and fuzzy binary coding (right). Bottom: Generator losses with binary coding (left) and fuzzy binary coding (right).

VI. Summary

In this paper, we have proposed a Categorical Generative Adversarial Nets (*Categorical GAIN*) for EHR data imputation, based on a framework that is originally designed for real values. First, we have hypothesized that applying softmax functions as output activation in the generator directly often results in the discriminator exploiting the obvious difference between generated and real values. And the adversarial training typically ends up in divergence. We have proposed to perform fuzzy coding of the binary values so that they resemble generated values while retaining the encoded information. Secondly, we have performed multiple modifications in the architectures of both generator and discriminator, in order to handle the fuzzy binary coded features.

We have compared our methods with a variety of benchmark methods on two EHR datasets. We have simulated different proportions of missing data by masking out known values and then attempting to perform prediction tasks based on imputed data. We could show that the more complex method of generative adversarial nets turned out to be advantageous in case of relatively higher missing rate and larger training data set, while the simpler methods such as SVD reconstruction and average imputation are more reliable to impute smaller proportion of missing data.

Acknowledgment

The authors acknowledge support by the German Federal Ministry for Education and Research (BMBF), funding project MLWin (grant 01IS18050).

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. *arXiv preprint arXiv:1703.06490*, 2017.

[3] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[4] A Rogier T Donders, Geert JMG Van Der Heijden, Theo Stijnen, and Karel GM Moons. A gentle introduction to imputation of missing values. *Journal of clinical epidemiology*, 59(10):1087–1091, 2006.

[5] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[6] Cristóbal Esteban, Danilo Schmidt, Denis Krompaß, and Volker Tresp. Predicting sequences of clinical events by using a personalized temporal latent embedding model. In *Healthcare Informatics (ICHI), 2015 International Conference on*, pages 130–139. IEEE, 2015.

[7] PA Fasching, SY Brucker, TN Fehm, F Overkamp, W Janni, M Wallwiener, P Hadji, E Belleville, L Häberle, F-A Taran, et al. Biomarkers in patients with metastatic breast cancer and the praegnant study network. *Geburtshilfe und Frauenheilkunde*, 75(01):41–50, 2015.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[9] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[10] Matt J Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.

[11] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017.

[12] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.

[13] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 333. John Wiley & Sons, 2014.

[14] Eugenij Moiseevich Mirkes, Timothy J Coats, Jeremy Levesley, and Alexander N Gorban. Handling missing data in large healthcare dataset: A case study of unknown trauma outcomes. *Computers in biology and medicine*, 75:203–216, 2016.

[15] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[16] Trivellore E Raghunathan, James M Lepkowski, John Van Hoewyk, and Peter Solenberger. A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey methodology*, 27(1):85–96, 2001.

[17] Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*, 2017.

[18] Donald B Rubin. Multiple imputation after 18+ years. *Journal of the American statistical Association*, 91(434):473–489, 1996.

[19] Donald B Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

[20] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, volume 2, page 5, 2017.

[21] Elizabeth A Stuart, Melissa Azur, Constantine Frangakis, and Philip Leaf. Multiple imputation with large data sets: a case study of the children's mental health initiative. *American journal of epidemiology*, 169(9):1133–1139, 2009.

[22] Volker Tresp, Ralph Neuneier, and Subutai Ahmad. Efficient methods for dealing with missing data in supervised learning. In *Advances in neural information processing systems*, pages 689–696, 1995.

[23] Brian J Wells, Kevin M Chagin, Amy S Nowacki, and Michael W Kattan. Strategies for handling missing data in electronic health record derived data. *eGEMs*, 1(3), 2013.

[24] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[25] Yinchong Yang, Volker Tresp, Marius Wunderle, and Peter A Fasching. Explaining therapy predictions with layer-wise relevance propagation in neural networks. In *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 152–162. IEEE, 2018.

[26] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. Gain: Missing data imputation using generative adversarial nets. *arXiv preprint arXiv:1806.02920*, 2018.

[27] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.

[28] Yizhe Zhang, Zhe Gan, and Lawrence Carin. Generating text via adversarial training. In *NIPS workshop on Adversarial Training*, volume 21, 2016.

[29] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*, 2017.

[30] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

Chapter 6

Conclusion

In this dissertation, we have leveraged representation learning techniques to advance clinical decision support from the perspective of prescriptive, diagnostic, predictive, and descriptive analytics. By taking advantage of state-of-the-art neural network architectures in deep learning, the proposed models can consume different medical data sources, such as electronic health records (EHRs) and medical images. More specifically, we included the sequential EHRs of patients who suffered breast cancer in the PRAEGNANT study (Fasching et al., 2015) as well as admissions to intensive care units in the Medical Information Mart for Intensive Care database (MIMIC-III) (Johnson et al., 2019).

In Chapter 2, we proposed a framework, estimated translated Inverse Propensity Score (etIPS), in prescriptive analytics to provide treatment recommendations. We formulated the learning of individualized treatment rules as a contextual bandit problem, where we focused on the offline setting, batch learning from logged bandit feedback (BLBF). By taking advantage of state-of-the-art BLBF algorithms, we trained a potentially better policy with data consisting of context (patient covariates), action (treatment decision), propensity score, and loss (outcome information). Since propensity scores were unavailable in observational studies, we proposed to use state-of-the-art predictive modeling algorithms to estimate them. Experiments were conducted both in a simulation study and based on a real-world dataset. With the simulation study, we validated the effectiveness of using the estimated propensity score to replace the true propensity score. Based on various offline evaluation methods on the real-world dataset, we showed that the policy derived in our framework could demonstrate better performance compared to both the physicians and other baselines, including a simple treatment prediction approach. For future works, it would be thrilling to augment the proposed framework with interpretability and explain-

ability. The new policy can be better understood and trusted by the physicians who will interact with it.

In Chapter 3, we focused on the perspective of uncertainty-awareness in diagnostic medical image analysis. In deep learning, most proposed regression models were found to concentrate purely on prediction accuracy but neglect the uncertainty in their predictions. We proposed an uncertainty-aware deep kernel learning model consisting of a Convolutional Neural Network and a sparse Gaussian Process. Besides, we adopted various pre-training methods from representation learning to investigate their impacts on the proposed model, including transfer learning, convolutional autoencoder, and deep metric learning. In most cases, our model showed better performance than common architectures. To better evaluate the uncertainty-awareness of a regression model, we proposed a visualization method called quantile-performance (QP) plot. We showed that our proposed model could express higher confidence in more accurate predictions with the QP plot. In addition, our proposed model computed the predictive distributions in a purely analytical fashion and was thus computationally more efficient than sampling-based methods like Monte-Carlo Dropout. Future works on combining deep kernel learning with state-of-the-art self-supervised learning have the potential to boost performance.

In Chapter 4, the uncertainty-aware regression models from the previous section were further developed in the context of predictive analytics for forecasting and planning clinical events in the future. The time-to-event prediction tasks were proposed to be solved by our Deep Kernel Accelerated Failure Time (DKAFT) models. More concretely, we tackled two specific challenges in this section. The first challenge lies in the sequential EHR data, which is high dimensional due to many patient features, while the length of each sequential input varies from patient to patient. To conquer this challenge, we applied a state-of-the-art recurrent neural network-based model to learn a fixed-size latent representation for each patient, which could presumably be better consumed by downstream prediction models. The second challenge lies in time-to-event target variables being positively skewed with possible administrative censoring. We generalized the linear class of accelerated failure time models to nonlinear Gaussian Process-based models, which simultaneously enabled the uncertainty-awareness of the prediction. In addition, a deep metric learning-based pre-training method was adapted to enhance the proposed models. From the experiments on two real-world datasets, our approach showed better point estimate performance than various baselines, including recurrent neural network-based ones. With the QP plot discussed in the last section, the proposed model was proven to deliver better performance

when more confident in its predictions. Future works investigating the interpretability of the predictive uncertainty will offer more insights to assist the physicians better.

In Chapter 5, we addressed the missing data imputation problems in descriptive analytics. The imputed data could provide substantial evidence for consequent decision-making processes, which in our case referred to the downstream machine learning models. More specifically, we proposed Categorical Generative Adversarial Imputation Nets (Categorical GAINs) for EHR data imputation with categorical patient features. The Generative Adversarial Imputation Nets (GAINs) were initially proposed for data imputation with continuous values, where we found the adversarial training tended to fail for discrete values like categorical features. Based on such observations, we hypothesized the failure comes from the apparent difference between the softmax outputs (continuous) and ground-truth values (binary), of which the discriminator could take advantage so that the generator could only get negative feedback for updating its parameters. To tackle this problem, we proposed performing fuzzy coding of categorical features to prevent the discriminator from exploiting the apparent differences to stabilize the adversarial training. Together with some other modifications on the architecture of GAINs, we have empirically shown that the post-imputation prediction accuracy with our Categorical GAINs was higher than the ones with more traditional methods, especially when the missing rate was relatively high. For future works, it would be interesting to see whether we can further develop the approach to generate new patient samples with categorical features to address data privacy problems better.

We believe the representation learning techniques will play an increasingly important role in future healthcare analytics. The contributions in this dissertation have made several steps ahead to more advanced clinical decision support. Physicians can presumably benefit from these advanced analytical models to deliver better patient care instead of being overwhelmed by the ever-increasing patient data.

Bibliography

Naoki Abe, Alan W Biermann, and Philip M Long. Reinforcement learning with immediate rewards and linear hypotheses. *Algorithmica*, 37(4):263–293, 2003.

Onur Asan, Alparslan Emrah Bayrak, Avishek Choudhury, et al. Artificial intelligence and human trust in healthcare: focus on clinicians. *Journal of medical Internet research*, 22 (6):e15154, 2020.

Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse gaussian process approximations. In *Advances in neural information processing systems*, pages 1533–1541, 2016.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Stan Benjamens, Pranavsingh Dhunnoo, and Bertalan Meskó. The state of artificial intelligence-based fda-approved medical devices and algorithms: an online database. *NPJ digital medicine*, 3(1):1–8, 2020.

Alina Beygelzimer and John Langford. The offset tree for learning with partial labels. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 129–138, 2009.

Ioana Bica, Ahmed M Alaa, Craig Lambert, and Mihaela Van Der Schaar. From real-world patient data to individualized treatment effects using machine learning: Current and

future methods to address underlying challenges. *Clinical Pharmacology & Therapeutics*, 109(1):87–100, 2021.

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14(11), 2013.

Léon Bottou et al. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

Leo Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–215, 2001.

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine learning for healthcare conference*, pages 301–318. PMLR, 2016a.

Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. *Advances in Neural Information Processing Systems*, 29: 3504–3512, 2016b.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. *Advances in neural information processing systems*, 24:666–674, 2011.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35 (5-6):352–359, 2002.

Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 1097–1104, 2011.

Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

Christo El Morr and Hossam Ali-Hassan. *Analytics in Healthcare: A Practical Introduction*. Springer, 2019.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Cristóbal Esteban, Oliver Staeck, Stephan Baier, Yinchong Yang, and Volker Tresp. Predicting clinical events by combining static and dynamic information using recurrent neural networks. In *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 93–101. IEEE, 2016.

Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

P.A. Fasching, S.Y. Brucker, T.N. Fehm, F. Overkamp, W. Janni, M. Wallwiener, P. Hadji, E. Belleville, L. Häberle, F.A. Taran, D. Luftner, M.P. Lux, J. Ettl, V. Muller, H. Tesch, D. Wallwiener, and A. Schneeweiss. Biomarkers in patients with metastatic breast cancer and the praegnant study network. *Geburtshilfe Frauenheilkunde*, 75(01):41–50, 2015. URL `http://www.praegnant.org/`.

Malte Feucht, **Zhiliang Wu**, Sophia Althammer, and Volker Tresp. Description-based Label Attention Classifier for Explainable ICD-9 Classification. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 62–66. Association for Computational Linguistics, November 2021. doi: 10.18653/v1/2021.wnut-1.8.

Andrew Gelman and Aki Vehtari. What are the most important statistical ideas of the past 50 years? *Journal of the American Statistical Association*, 0(0):1–11, 2021.

Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521 (7553):452–459, 2015.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

Jindong Gu, **Zhiliang Wu**, and Volker Tresp. Introspective Learning by Distilling Knowledge from Online Self-explanation. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020. doi: 10.1007/978-3-030-69538-5_3.

Yoni Halpern, Steven Horng, Youngduck Choi, and David Sontag. Electronic medical record phenotyping using the anchor and learn framework. *Journal of the American Medical Informatics Association*, 23(4):731–740, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

James Hensman, Nicolò Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 282–290, 2013.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

Martin Jankowiak, Geoff Pleiss, and Jacob Gardner. Parametric gaussian process regressors. In *International Conference on Machine Learning*, pages 4702–4712. PMLR, 2020.

Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. Deep learning with logged bandit feedback. In *International Conference on Learning Representations*, 2018.

Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *International conference on machine learning*, pages 3020–3029. PMLR, 2016.

Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Mark Roger. Mimic-iv (version 1.0). *PhysioNet*, 2021.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

Alistair EW Johnson, Tom J Pollard, Seth J Berkowitz, Nathaniel R Greenbaum, Matthew P Lungren, Chih-ying Deng, Roger G Mark, and Steven Horng. Mimic-cxr, a de-identified publicly available database of chest radiographs with free-text reports. *Scientific data*, 6(1):1–8, 2019.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems*, 30:5574–5584, 2017.

Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Matthieu Komorowski, Leo A Celi, Omar Badawi, Anthony C Gordon, and A Aldo Faisal. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature medicine*, 24(11):1716–1720, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.

Igor Kuzmanovski and Slobotka Aleksovska. Optimization of artificial neural networks for prediction of the unit cell parameters in orthorhombic perovskites. comparison with multiple linear regression. *Chemometrics and Intelligent Laboratory Systems*, 67(2):167–174, 2003.

John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. *Advances in neural information processing systems*, 20(1):96–1, 2007.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, 19:143–155, 1989.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

Hanxiao Liu, Zihang Dai, David R So, and Quoc V Le. Pay attention to mlps. *arXiv preprint arXiv:2105.08050*, 2021.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. *Advances in Neural Information Processing Systems*, 31, 2018.

Chris McIntosh, Leigh Conroy, Michael C Tjong, Tim Craig, Andrew Bayley, Charles Catton, Mary Gospodarowicz, Joelle Helou, Naghmeh Isfahanian, Vickie Kong, et al. Clinical integration of machine learning for curative-intent radiation treatment of patients with prostate cancer. *Nature Medicine*, 27(6):999–1005, 2021.

Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997.

Abdel-rahman Mohamed, George Dahl, Geoffrey Hinton, et al. Deep belief networks for phone recognition. In *NIPS workshop on deep learning for speech recognition and related applications*, volume 1, page 39. Vancouver, Canada, 2009.

George B Moody and Roger G Mark. A database to support development and evaluation of intelligent intensive care monitoring. In *Computers in Cardiology 1996*, pages 657–660. IEEE, 1996.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Inbal Nahum-Shani, Shawna N Smith, Bonnie J Spring, Linda M Collins, Katie Witkiewitz, Ambuj Tewari, and Susan A Murphy. Just-in-time adaptive interventions (jitais) in mobile health: key components and design principles for ongoing health behavior support. *Annals of Behavioral Medicine*, 52(6):446–462, 2018.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 807–814, 2010.

Ju Gang Nam, Sunggyun Park, Eui Jin Hwang, Jong Hyuk Lee, Kwang-Nam Jin, Kun Young Lim, Thienkai Huy Vu, Jae Ho Sohn, Sangheum Hwang, Jin Mo Goo, et al. Development and validation of deep learning–based automatic detection algorithm for malignant pulmonary nodules on chest radiographs. *Radiology*, 290(1):218–228, 2019.

Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6: 1939–1959, 2005.

Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):1–10, 2018.

CE. Rasmussen and CKI. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, January 2006.

Philippe Rigollet and Assaf Zeevi. Nonparametric bandits with covariates. *COLT 2010*, page 54, 2010.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688, 1974.

Donald B Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009. ISBN 0136042597.

Mohammed Saeed, Mauricio Villarroel, Andrew T Reisner, Gari Clifford, Li-Wei Lehman, George Moody, Thomas Heldt, Tin H Kyaw, Benjamin Moody, and Roger G Mark. Multiparameter intelligent monitoring in intensive care ii (mimic-ii): a public-access intensive care unit database. *Critical care medicine*, 39(5):952, 2011.

Matthias W Seeger, Christopher KI Williams, and Neil D Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *International Workshop on Artificial Intelligence and Statistics*, pages 254–261. PMLR, 2003.

Terrence J Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, 117(48):30033–30038, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems*, 18:1257–1264, 2005.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Adith Swaminathan and Thorsten Joachims. The self-normalized estimator for counterfactual learning. *advances in neural information processing systems*, 28, 2015.

Zhiliang Wu, Yinchong Yang, Yunpu Ma, Yushan Liu, Rui Zhao, Michael Moor, and Volker Tresp. Learning Individualized Treatment Rules with Estimated Translated Inverse Propensity Score. In *2020 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–11, 2020. doi: 10.1109/ICHI48887.2020.9374397.

Zhiliang Wu, Yinchong Yang, Peter A Fashing, and Volker Tresp. Uncertainty-Aware Time-to-Event Prediction using Deep Kernel Accelerated Failure Time Models. In *Proceedings of the 6th Machine Learning for Healthcare Conference*, volume 149 of *Proceedings of Machine Learning Research*, pages 54–79. PMLR, 06–07 Aug 2021a.

Zhiliang Wu, Yinchong Yang, Jindong Gu, and Volker Tresp. Quantifying Predictive Uncertainty in Medical Image Analysis with Deep Kernel Learning. In *2021 IEEE 9th International Conference on Healthcare Informatics (ICHI)*, pages 63–72, 2021b. doi: 10.1109/ICHI52183.2021.00022.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR, 2009.

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

Eric J Topol. High-performance medicine: the convergence of human and artificial intelligence. *Nature medicine*, 25(1):44–56, 2019.

Volker Tresp, J Marc Overhage, Markus Bundschus, Shahrooz Rabizadeh, Peter A Fasching, and Shipeng Yu. Going digital: a survey on digitalization and large-scale data analytics in healthcare. *Proceedings of the IEEE*, 104(11):2180–2206, 2016.

Chih-Chun Wang, Sanjeev R Kulkarni, and H Vincent Poor. Bandit problems with side observations. *IEEE Transactions on Automatic Control*, 50(3):338–355, 2005.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems*, 13, 2000.

Yinchong Yang, Peter A Fasching, and Volker Tresp. Predictive modeling of therapy decisions in metastatic breast cancer with recurrent neural network encoder and multinomial hierarchical regression decoder. In *2017 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 46–55. IEEE, 2017.

Yinchong Yang, **Zhiliang Wu**, Volker Tresp, and Peter A. Fasching. Categorical EHR Imputation with Generative Adversarial Nets. In *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 1–10. IEEE, 2019. doi: 10.1109/ICHI.2019.8904717.

Gal Yarin. Uncertainty in deep learning. *University of Cambridge*, 1(3), 2016.

Jinsung Yoon, James Jordon, and Mihaela Schaar. Gain: Missing data imputation using generative adversarial nets. In *International Conference on Machine Learning*, pages 5689–5698. PMLR, 2018a.

Jinsung Yoon, James Jordon, and Mihaela Van Der Schaar. Ganite: Estimation of individualized treatment effects using generative adversarial nets. In *International Conference on Learning Representations*, 2018b.

Kun-Hsing Yu, Andrew L Beam, and Isaac S Kohane. Artificial intelligence in healthcare. *Nature biomedical engineering*, 2(10):719–731, 2018.

Elcin Zan, David M Yousem, Marco Carone, and Jonathan S Lewin. Second-opinion consultations in neuroradiology. *Radiology*, 255(1):135–141, 2010.

Yingqi Zhao, Donglin Zeng, A John Rush, and Michael R Kosorok. Estimating individualized treatment rules using outcome weighted learning. *Journal of the American Statistical Association*, 107(499):1106–1118, 2012.

Xin Zhou, Nicole Mayer-Hamblett, Umer Khan, and Michael R Kosorok. Residual weighted learning for estimating individualized treatment rules. *Journal of the American Statistical Association*, 112(517):169–187, 2017.