9-13-2023

# BETTER MODELS FOR HIGH-STAKES TASKS

Jacob Ryan Epifano
*Rowan University*

# BETTER MODELS FOR HIGH-STAKES TASKS

by

Jacob Epifano

A Dissertation

Submitted to the
Department of Electrical & Computer Engineering
College of Engineering
In partial fulfillment of the requirement
For the degree of
Doctor of Philosophy in Electrical & Computer Engineering
at
Rowan University
September 13, 2023

Thesis Chair: Ravi Ramachandran, Ph.D., Professor, Department of Electrical & Computer Engineering

Committee Members:
Ghulam Rasool Ph.D., Assistant Member, Machine Learning Department, Moffitt Cancer Center
John Schmalzel Ph.D., Professor and Founding Chair, Department of Electrical & Computer Engineering
Nidhal Bouaynaya, Ph.D., Professor & Assoc. Dean for Research, Department of Electrical & Computer Engineering
Umashanger Thayasivam, Ph.D., Professor & Department Chair, Department of Mathematics
Aaron Masino, Ph.D., Adjunct Assistant Professor , Department of Biostatistics, Epidemiology, Informatics, University of Pennsylvania Perelman School of Medicine

## Dedication

I dedicate this dissertation to my family whose unwavering love and support have been my anchor throughout my time at Rowan University. To my parents, who instilled in me the value of hard work and the pursuit of knowledge, thank you. This journey was made possible because of your collective belief in me.

**Acknowledgements**

I would like to extend my deepest gratitude to my colleagues, Sean McGuire, Glenn Dawson, and several members of my committee: Dr. Ravi, Dr. Rasool, and Dr. Masino, whose time, expertise, and dedication significantly contributed to the completion of this project.

*This work was supported by the U.S. Department of Education through the Graduate Assistance in Areas of National Need (GAANN) program, Award Number P200A180055.*

## Abstract

Jacob Epifano
BETTER MODELS FOR HIGH-STAKES TASKS
2023-2024
Ravi Ramachandran, Ph.D.
Doctor of Philosophy in Electrical & Computer Engineering

The intersection of machine learning and healthcare has the potential to transform medical diagnosis, treatment, and research. Machine learning models can analyze vast amounts of medical data and identify patterns that may be too complex for human analysis. However, one of the major challenges in this field is building trust between users and the model. Due to things like high false alarm rate and the black box nature of machine learning models, patients and medical professionals need to understand how the model arrives at its recommendations. In this work, we present several methods that aim to improve machine learning models in high-stakes environments like healthcare. Our work unifies two sub-fields of machine learning, explainable AI, and uncertainty quantification. First we develop a model-agnostic approach to deliver instance-level explanations using influence functions. Next, we show that these influence functions function are fairly robust across domains. Then, we develop an efficient method that reduces model uncertainty while modeling data uncertainty via Bayesian Neural Networks. Finally, we show that when combined our methods deliver significant utility beyond traditional methods while retaining a high level of performance via a real world deployment. Overall, the integration of uncertainty quantification and explainable AI can help overcome some of the major challenges of machine learning in healthcare. Together, they can provide healthcare professionals with powerful tools for improving patient outcomes and advancing medical research.

## Table of Contents

**Table of Contents (Continued)**

**Table of Contents (Continued)**

**Table of Contents (Continued)**

**Table of Contents (Continued)**

## List of Figures

## List of Tables

# Chapter 1

## Introduction

I began working on my first machine learning related publication in 2018 where my team acquired a novel dataset to predict levels of cognitive impairment. The dataset consisted of discrete measurements and statistics collected from a digital pen. Participants were asked to draw an analog clock with the time "ten minutes after eleven". The manner in which the clock was drawn can be a factor in determining whether or not a patient has Alzheimer's disease. While the models that we developed achieved a high level of performance, the part that struck me was that once published, these models were discarded and our work concluded [1].

The next time I was exposed to this phenomena was during an internship at the Children's Hospital of Philadelphia. My team had just concluded research on a sepsis prediction model for infants. This model achieved a high level of performance and can detect sepsis up to approximately 4 hours before clinical recognition [2]. In general, early detection of sepsis is a key in treating it as the earlier one can start treatment, the outcome is often better. Designing a model to detect sepsis prior to clinical recognition can lead to a high false alarm rate [3]. A common fix for this is to increase the complexity of the model thereby increasing its performance. In healthcare and high risk fields in general, complex models are undesirable due to their uninterpretable outputs [4]. These factors make deploying models in sensitive areas quite difficult as even highly accurate machine learning models have not been widely accepted or endorsed by clinical staff [5, 6]. At the time, it was clear that several considerations needed to be made before prediction models would be readily accepted in these environments.

## 1.1 Motivation

Artificial Intelligence (AI) and Machine Learning (ML) have started to gain traction in medical domains [7]. However, due to the high-stakes nature of medicine, AI/ML models rarely make it to real-world deployments [8]. While a few models have successfully transitioned to deployment [2, 9, 10], user feedback is often negative [5, 6]. Factors such as high false alarm rates and uninterpretable outputs [3, 4] demand several considerations for AI/ML to flourish in high-risk environments like healthcare. [11]

## 1.2 Problem Statement

AI models have the potential to greatly enhance healthcare delivery by improving diagnostic accuracy, predicting patient outcomes, personalizing treatment plans, and increasing overall efficiency. However, their reception in the healthcare sector has been characterized by significant resistance and skepticism. This is largely due to concerns around lack of interpretability and transparency in AI decision-making, potential for bias in algorithms, and lack of appropriate technical understanding and skills in the healthcare community. The ”black box” phenomenon is a prevalent problem in AI. This refers to the situation where only the inputs and outputs of a learning model can be observed. It is not precisely known how the parameters of the model interact to arrive at the final output.

Therefore, we need to devise strategies and solutions that can improve the perception and acceptance of AI models in healthcare. This requires comprehensive efforts to address the aforementioned issues and to enhance the understanding, trust, and usability of AI technologies among healthcare practitioners and patients. In the pursuit of integrating AI in healthcare, it is essential to ensure that the use of these models is transparent, and

interpretable, while also providing tangible benefits to patient care and outcomes.

## 1.3 Outline of the Dissertation

Chapter 1 provides the introduction and motivation for the problem of improving AI/ML research in the healthcare space. Chapter 2 provides relevant background information in the areas where we have innovated including feature importance and uncertainty quantification. Chapter 3 introduces our first crack at addressing this problem, mainly improving the mortality prediction task with influence functions as the method of choice for providing explanations. Chapter 4 addresses the problem of feature selection for the mortality prediction task. We provide several approaches to choose a minimum set of features that provides the highest performing model. Chapter 5 introduces a simplification to a recent Bayesian framework for training neural networks with built-in robustness and uncertainty quantification. Chapter 6 addresses claims from a recent work that shows influence function fragility. We reproduce their work and come up with different conclusions. Chapter 7 unifies our prior work. We deploy a mortality prediction model with explanations and uncertainty quantification and observe its performance during the COVID-19 pandemic and afterwards. Finally, Chapter 8 summarizes our contributions and outlines future efforts for this field.

# Chapter 2

# Background

## 2.1 Motivation

Tonekaboni *et al.* [12] have provided the majority of the motivation for our work in ML for healthcare by contextualizing the requirements to successfully deploy ML models in clinical environments. Those requirements can be boiled down to a few components which form the basis of our work: **feature importance, instance level explanation, and Uncertainty Quantification (UQ)**. Feature importance and explanations in general attempt to attribute the prediction to one or several input features. Explanations can often be categorized into global, where we try to measure the overall importance of each input feature, and local, where we try to measure how the input features of one individual sample affected the model output. The comcept of UQ refers to measuring the confidence or certainty of a given prediction. This has become more important with the advent of deep learning, where softmax outputs have drifted away from approximations of the true posterior probability and often make models appear overconfident [13]. Feature importance and instance level explanations typically go together. However, since the advent of deep learning, these topics have been drifting apart. Due to the black-box nature of Deep Neural Networks (DNNs), implementing a convincing solution for these components has proven to be quite difficult.

Prior to our work in this field, less consideration has been given to the utility and explainability of machine learning models in healthcare in favor of high performance and self-indulgent publications. Tonekaboni *et al.* [12] have given us a framework to design machine learning models for real world deployments. Using this framework as a guide,

our objective was to reduce alarm fatigue, provide actionable insights and build user-model trust through a combination of explainable predictions and uncertainty prediction.

## 2.2 Supervised Learning

For all inference problems in this thesis, we perform supervised learning. Supervised learning assumes that the available data consists of data-label pairs wherein we try to find a function that maps the data to the label. More formally, consider an inference problem where $\mathcal{X}$ represents the input space and $\mathcal{Y}$ represents the output space. Given training points $z_1, z_2, ..., z_n$, where $z_i(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, we find a set of parameters, $\theta \in \Theta$ that minimizes the empirical risk $R(\theta) = \frac{1}{n} \sum_{i=0}^{n} L(z, \theta)$ where $L$ is the loss function. Let the training set $X$ and $Y$ be sampled from $\mathcal{X}$ (Equation 2.1) and $\mathcal{Y}$ (Equation 2.2), where $n$ is the number of samples, $k$ is the number of features and $f_1, f_2, \ldots, f_k$ are the input features.

$$X = \begin{bmatrix} x_1 : & [f_1 & f_2 & \ldots & f_k] \\ x_2 : & [f_1 & f_2 & \ldots & f_k] \\ \vdots & & & & \\ x_n : & [f_1 & f_2 & \ldots & f_k] \end{bmatrix} \tag{2.1}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2.2}$$

## 2.3 Feature Importance

In machine learning, the explainability of a model and its performance are generally competing factors [14]. Explainable models, (logistic regression, trees, etc.) are often outperformed by black-box models (neural networks (NNs)). Early work has been done by Ribeiro, et. al who explain the predictions of any classifier using Local Interpretable Model-Agnostic Explanations (LIME) and High Precision Model-Agnostic Explanations (Anchors) [15, 16]. While LIME's local explanation breaks down immediately upon changing the point under test, Anchors resolves this by optimizing for the whole test set. These model agnostic approaches leave much to be desired in terms of explainability, and, therefore, model-specific approaches are preferred especially for NNs. A lot of work in the field of explainable machine learning has been done for image datasets. Specifically, saliency maps are used to propagate classification information from the last layers of Neural Networks to their inputs. For image data specifically, this results in a 2D map that is the size of the original input where each pixel's intensity represents how important that pixel is to the classification [17–22] [11].

In this thesis, we investigated analytical techniques known as influence functions, which originate from robust statistics, as our primary method of explainability. Influence functions allow one to approximate the change that a leave one out (LOO) training scheme would have on the parameters [23]. Recently, Koh and Liang showed that influence functions can be used to approximate which training points most effected the loss of a test point and what features were most important for each training point [24]. Since this algorithm is model-specific, it can be leveraged to extract multiple statistics about how the model

interprets the data. This makes influence functions an ideal candidate for use in the medical field. One of the limitations of their work is that the gradient is not propagated through the multiple layers of the network [25]. Koh and Liang use features from the last layer as an input to a logistic regression model and approximate influence functions [24]. The algorithm cannot provide any information on how the original features affected functions of the test loss [24] [11].

## 2.4 Influence Function Derivation

This derivation was taken directly from Koh *et al.* [24] and has been reproduced below:

Influence functions are considered one of the classic techniques from robust statistics that can quantify the change in model parameters attributed to up-weighting a training point by an infinitesimal amount. In the following, we derive mathematical relationships for influence functions, examine their underlying assumptions, and attempt to explain these in the context of large neural networks. We start by defining an optimization problem where $\hat{\theta}$ minimizes the empirical risk as defined by Koh *et al.* [24]:

$$R(\theta) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) \tag{2.3}$$

where (as mentioned earlier) $L$ is the loss function, $z_i$ are the training data, $n$ is the total number of training points, and $\theta$ are the network parameters.

A fundamental assumption for influence functions is that $R$ is twice-differentiable and **strongly** convex in $\theta$. That is, the Hessian, as defined by:

$$H_{\hat{\theta}} \stackrel{\text{def}}{=} \nabla^2 R(\hat{\theta}) = \frac{1}{n}\nabla^2_\theta L(z_i, \hat{\theta}), \tag{2.4}$$

exists and is positive definite. The convexity assumption guarantees the existence of $H^{-1}_{\hat{\theta}}$. Recallthat we approximate the removal of a training point by up-weighting the parameters by a small quantity $\epsilon \approx -\frac{1}{n}$. The perturbed parameters, $\hat{\theta}_{\epsilon,z}$ can be written as [24]:

$$\hat{\theta}_{\epsilon,z} = \arg\min_{\theta\in\Theta} \left[R(\theta) + \epsilon L(z,\theta)\right]. \tag{2.5}$$

The total parameter change by up-weighting a training example can be defined as $\Delta_\epsilon = \hat{\theta}_{\epsilon,z} - \hat{\theta}$. Differentiating with respect to $\epsilon$ and noting that $\hat{\theta}$ doesn't depend on $\epsilon$, we can write:

$$\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} = \frac{d\Delta_\epsilon}{d\epsilon}. \tag{2.6}$$

We note that for the optimal parameters $\hat{\theta}_{\epsilon,z}$, we can rewrite Eq. 2.5 as:

$$0 = \nabla R(\hat{\theta}_{\epsilon,z}) + \epsilon \nabla L(z, \hat{\theta}_{\epsilon,z}). \tag{2.7}$$

Since $\hat{\theta}_{\epsilon,z} \to \hat{\theta}$ as $\epsilon \to 0$, the **first-order** Taylor expansion of the right-hand side produces:

$$0 \approx \left[\nabla R(\hat{\theta}) + \epsilon \nabla L(z,\hat{\theta})\right] + \left[\nabla^2 R(\hat{\theta}) + \epsilon \nabla^2 L(z,\hat{\theta})\right]\Delta_\epsilon. \tag{2.8}$$

Solving for $\Delta_\epsilon$,

$$\Delta_\epsilon = -\left[\nabla^2 R(\hat{\theta}) + \epsilon \nabla^2 L(z, \hat{\theta})\right]^{-1} \left[\nabla R(\hat{\theta}) + \epsilon \nabla L(z, \hat{\theta})\right] \tag{2.9}$$

Since $\hat{\theta}$ minimizes $R$, then $\nabla R(\hat{\theta}) = 0$. Neglecting higher order $\epsilon$ terms,

$$\Delta_\epsilon \approx -\nabla^2 R(\hat{\theta})^{-1} \nabla L(z, \hat{\theta}) \epsilon. \tag{2.10}$$

When we substitute Equations 2.4 and 2.6, we have:

$$\left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla L(z, \hat{\theta}) \stackrel{\text{def}}{=} \mathcal{I}_{\text{up,params}}(z). \tag{2.11}$$

To study how upweighting $z$ changes functions of $\hat{\theta}$ we can apply the chain rule to Equation 2.11. The influence of upweighting $z$ on the loss of a test point $z_{\text{test}}$ is:

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}). \tag{2.12}$$

Finally, we can use the influence function to study the effect that perturbing $z$ ($z \to z_\delta$), where $z_\delta = (x + \delta, y)$ has on the features of $z_{\text{test}}$. This is given by

$$\mathcal{I}_{\text{pert,loss}}(z, z_{\text{test}}) = -\nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_x \nabla_\theta L(z, \hat{\theta}). \tag{2.13}$$

### 2.5 Uncertainty Quantification

The total uncertainty in a machine learning model can be characterized by two distinct components. The first is epistemic uncertainty, which arises from the parameters of the model. The second is aleatoric uncertainty, which stems from the data or environment. Uncertainty quantification is becoming increasingly important in high-stakes decision-making, as establishing trust between users and models is crucial for the widespread adoption of such models [26]. To date, the implementation of machine learning models in high-stakes

domains, such as medicine, has not achieved substantial user acceptance and confidence [5, 6]. This is primarily due to high false alarm rates [4] and suboptimal test characteristics [27]. Variational inference has been demonstrated to alleviate the impact of epistemic uncertainty by effectively averaging predictions during inference [28–30]. Furthermore, the output variance associated with a single prediction obtained from a Bayesian Neural Network (BNN) can quantify the level of aleatoric uncertainty in that prediction [31, 32].

# Chapter 3

## Towards an Explainable Mortality Prediction Model

In our preliminary work [33], we sought to approach the first two suggestions provided by Tonekaboni *et al.* [12], feature importance and instance-level explanations, on a medically relevant task. For this work, we chose to tackle mortality prediction for patients in the Intensive Care Unit (ICU) using first-day (acquired within 24 hours of admittance) lab values [33].

The current state-of-the-art for mortality prediction exists in the form of evaluation scores. Acute Physiology and Chronic Health Evaluation (APACHE), and Simplified Acute Physiology Score (SAPS) are examples of mortality predictors [34, 35]. A simple logistic regression model is fitted to these scores and a binary classifier is built. These methods have been shown to have an area under the curve (AUC) of the receiver operating characteristic (ROC) range from 0.6-0.7 depending on the time of prediction [36]. The prediction time has a huge impact on the utility of these models. Predicting mortality at 48 hours from admittance is not as useful as predicting at the time of admission or after 24 hours. In this work, we focused on predicting mortality at 24 hours after admittance in the ICU [33].

### 3.1 Extended Influence Functions

Recall from Section 2.4, that the original influence function implementation utilizes the bottleneck features of a deep learning model as input to a logistic regression model [24]. This simplification reduces the complexity of the algorithm since the gradient only has to propagate back through one layer. This simplification works when calculating the influence on the model parameters (Equation 2.11) or the influence on the loss of a test

point (Equation 2.12). While this may be good enough for most applications, if one would want to estimate the influence that perturbing the input features would have on the loss of a test point (Equation 2.13), this will not work due to the requirement of having to compute the gradient with respect to the input features, $\nabla_x$. We remove this restriction of using the extracted features by keeping the graph intact and providing a methodology for layer-by-layer calculation of the influence functions.

In our settings, we compute the influence functions using Eqs. (3.1) and (3.2) with respect to the output layer only rather than with respect to each layer of the Neural Net. This enables an efficient computation of the influence functions. Our approach is inspired by the techniques used in saliency maps [17–22] [33]

We perform stochastic estimation of the inverse of the Hessian using the LiSSA algorithm [37]. As our approach involves the computation of the Hessian, all operations in the Neural Network must be twice differentiable. Therefore, we used Scaled Exponential Linear Units (SELU) [38] activation functions (as opposed to Rectified Linear Units (ReLU)) and the cross-entropy loss function [33].

### *3.1.1 Feature Importance - Local and Global*

We define *feature importances* by taking the average of Equation (2.13) across all training points in the dataset. Later, when examining the loss of one test point $z_{\text{test}}$, we refer to this as the local feature importance given by

$$FI_{\text{local}} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{I}_{\text{pert, loss}}(z_i, z_{\text{test, point}}), \tag{3.1}$$

where $\mathcal{I}_{\text{pert, loss}}$ is given in Equation 2.13. The variable N denotes the number of training points in the dataset. When using the average loss across all test points, we refer to this as the global feature importance:

$$FI_{\text{global}} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{I}_{\text{pert, loss}}(z_i, z_{\text{test, set}}). \tag{3.2}$$

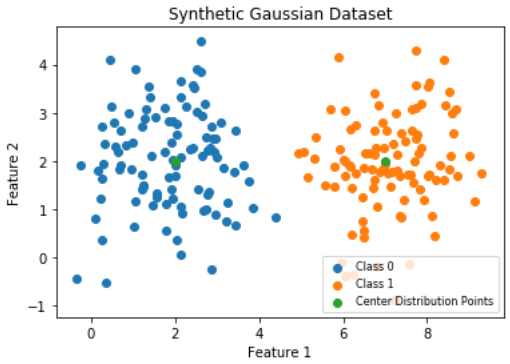
[33]

## 3.2 Synthetic Gaussian Dataset

In order to evaluate our implementation, we tested our extended influence functions (given in Equations (3.1) and (3.2)) using simulated data. We generated two multivariate Gaussian distributions, each having two dimensions. Both distributions shared the same mean in the first dimension and had different means in the second dimension. Therefore, it should be easy to quantify the feature importance. We split the data into train and test sets and trained two classifiers, i.e., a logistic regression and a neural network with one hidden layer. We evaluated the correlation between the logistic regression coefficients (ground truth) and feature importance from Equation (2.13) [33].

**Figure 1**

*Synthetic Multi-Variate Gaussian Distributions Data*



We evaluated how the feature importance values can change across test samples. When evaluating the center points of each distribution as our test point (Figure 1), we noticed very different average feature importances that were also different from the logistic regression coefficients (Equation 3.1). However, when the entire test set is used to compute the loss (Equation 3.2), the feature importance values approach the logistic regression coefficients (the ground truth) with correlation values above 0.95 and with $p < 0.05$, where $p$ is the p-value of the correlation statistic [33].

This correlation analysis shows that, at least for small models, the explanations produced by influence functions align well with the logistic regression coefficients. These results also validates our formulation for global feature importance (Equation 3.2).

## 3.3 Model Selection

### *3.3.1 eICU Collaborative Research Database*

The eICU database is a collection of datasets from multiple intensive care units (ICUs) across the United States [39]. We chose the dataset that included first-day laboratory test results as input features ($x$) and patient survival as labels ($y$). Furthermore, we split the dataset into two groups, (1) septic: dataset of patients who were diagnosed for sepsis only, and (2) all-comers: dataset of patients with all types of diagnosis. The septic patient data is a sub-set of the all-comers dataset. The decision to split the dataset was based on the assumption that the features indicating mortality may differ depending on the diagnosis of the patient, i.e., septic or non-septic. For the septic dataset, we have 19379 instances and 28 input features. For the all-comers dataset, we have 148532 instances and 20 input features [33].

### *3.3.2 Data Pre-Processing*

We start with a correlation analysis of all features in the datasets. Subsequently, all features with a correlation coefficient greater than 0.9 were dropped from the dataset. The missing data were assumed to be missing at random and features with the number of missing data > 50% of the total data were dropped from the dataset. For the rest of the missing data, we used multiple imputation with chained equations via iterative imputer method available in the ski-kit learn Python package [40] [33].

### *3.3.3 Training, Validation, and Testing*

We used k-fold cross-validation with $k = 5$ for the performance evaluation of all models. For each split, the training and testing data were standardized using the mean and the standard deviation of the training data. We observed a large class imbalance in our dataset (90%-10% in septic and 95%-5% in all-comers). To explore further and address the potential problem of class imbalance, we trained and tested two sets of models, (1) without introducing any class balancing technique, and (2) performing minority class oversampling using Synthetic Minority Oversampling Technique (SMOTE) [41]. SMOTE performs synthetic sampling using interpolation along the space between a minority instance's k-nearest-neighbors until the number of points in each class are equal [33].

In addition to Simplified Acute Physiology Score (SAPS) and Acute Physiology and Chronic Health Evaluation (APACHE), which were available in the database, we trained three models, i.e., logistic regression, XGBoost, and a Neural Net with one hidden layer using SELU activations. We used nested grid search to find optimal hyperparameters for

the logistic regression (C, Penalty, etc.) and XGBoost (max depth, learning rate, etc.) models. However, for the Neural Net, we used Bayesian optimization which is available in Weights and Biases Python package [42]. For each patient, we extracted the corresponding APACHE and SAPS scores and fit a logistic regression model using the same training and testing scheme as described above for the other three models. We evaluated all five models using the Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) metric. The models scoring highest on the ROC AUC metric were selected to evaluate the test set for each cross-validation loop [33].

**Table 1**

*AUC Scores or Both Test Datasets*

| Septic Patients: ROC AUC | | | | | |
|---|---|---|---|---|---|
| SMOTE Oversampling | SAPS | APACHE IVa | Logistic Regression | XGBoost | Neural Network |
| No | 0.7717 | 0.7849 | 0.7985 | 0.6901 | 0.7969 |
| Yes | 0.7736 | 0.7864 | 0.8001 | 0.7199 | **0.8046** |
| All-Comers Patients: ROC AUC | | | | | |
| No | 0.8325 | 0.8451 | 0.8463 | 0.6892 | 0.8326 |
| Yes | 0.8330 | 0.8457 | 0.8474 | 0.8124 | **0.8564** |

### *3.3.4 Extraction and Comparison of Important Features*

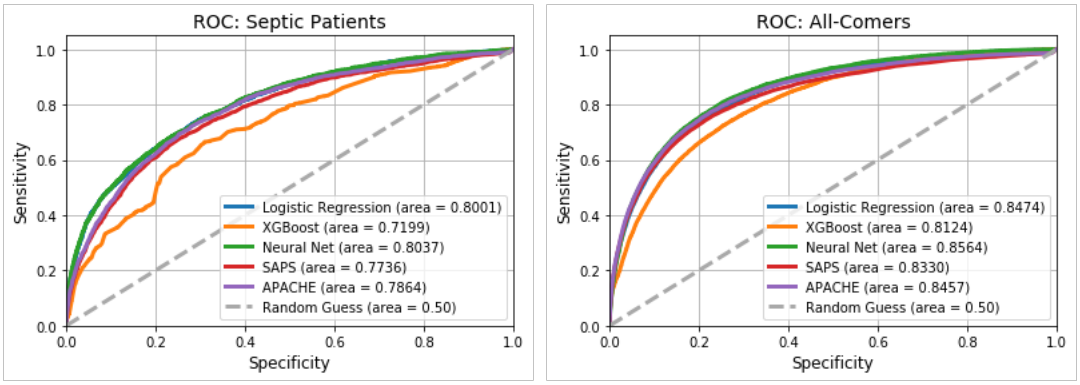
We extracted important features from each model using various techniques and compared these with features manually selected by domain experts. We created a survey

that included all features from both datasets and asked ICU clinicians (n=50 for the septic dataset and n=20 for the all-comers dataset) to pick the top the 10 features that they believed were the best indicators for patient mortality [33].

For the logistic regression model, we selected features with the highest absolute value. For XGBoost, we calculated SHapley Additive exPlanations (SHAP) values for each feature and selected those with the highest values [43]. For the Neural Net, we used SHAP (via DeepLIFT [44] which is another feature attribution method for neural networks) and extended influence functions to select important features. To evaluate how each of these methods performed, we found 10 features most often picked by the domain experts and counted how many features were common in the top 10 ranked by the above methods [33].

**Figure 2**

*ROC Curves for Different Models for Each Test Dataset*




### 3.4 Results and Discussion

In Figure 2, we present ROC curves for both test datasets (septic and all-comers) and for all five models, Neural Net, logistic regression, XGBoost, SAPS, and APACHE. In Table 1, we present AUC scores for both test datasets (Septic and All-Comers) and all

five models for two cases, i.e., with and without SMOTE oversampling. We note that for both datasets, the Neural Network outperforms all other models and XGBoost is the worst performing model [33].

**Table 2**

*Feature Ranking (FR) of Septic and All-Comers Datasets*

| FR | Survey | LR - no SMOTE | LR - SMOTE | XG SHAP - no SMOTE | XG SHAP - SMOTE | NN SHAP - no SMOTE | NN SHAP - SMOTE | NN IFs - no SMOTE | NN IFs - SMOTE |
|---|---|---|---|---|---|---|---|---|---|
| Septic Dataset | | | | | | | | | |
| 1 | hepaticfailure | CHLORIDE_max | CHLORIDE_max | PT_max | LACTATE_max | gender | SODIUM_max | LACTATE_max | CHLORIDE_max |
| 2 | LACTATE_max | BICARBONATE_min | BICARBONATE_min | LACTATE_max | BICARBONATE_min | day1motor | BICARBONATE_min | age | LACTATE_max |
| 3 | age | LACTATE_max | LACTATE_max | BICARBONATE_min | day1motor | age | diabetes | PT_max | INR_max |
| 4 | immunosuppression | ALBUMIN_min | ALBUMIN_min | BUN_max | BUN_max | BILIRUBIN_max | WBC_max | SODIUM_min | BICARBONATE_min |
| 5 | metastaticcancer | SODIUM_max | SODIUM_max | INR_max | INR_max | diabetes | SODIUM_min | SODIUM_max | PT_max |
| 6 | CREATININE | day1motor | ANIONGAP_max | PLATELET_min | ALBUMIN_min | patientunitstayid | ANIONGAP_max | BUN_max | SODIUM_max |
| 7 | INR | age | age | ALBUMIN_min | age | BICARBONATE_min | day1motor | HEMATOCRIT_min | day1motor |
| 8 | GCS-motor | ANIONGAP_max | day1motor | day1motor | BILIRUBIN_max | hepaticfailure | ALBUMIN_min | metastaticcancer | BUN_max |
| 9 | BILIRUBIN | BUN_max | SODIUM_min | HEMATOCRIT_min | patientunitstayid | SODIUM_min | BUN_max | INR_max | hepaticfailure |
| 10 | leukemia | SODIUM_min | BUN_max | GLUCOSE_min | aids | SODIUM_max | CHLORIDE_max | lymphoma | diabetes |
| All-Comers Dataset | | | | | | | | | |
| 1 | age | day1motor | day1motor | LACTATE_max | day1motor | ANIONGAP_max | CHLORIDE_max | day1motor | day1motor |
| 2 | immunosuppression | BICARBONATE_min | CHLORIDE_max | day1motor | LACTATE_max | gender | PLATELET_min | BICARBONATE_min | LACTATE_max |
| 3 | hepaticfailure | ALBUMIN_min | ALBUMIN_min | CREATININE_max | BUN_max | HEMATOCRIT_min | GLUCOSE_min | PT_max | INR_max |
| 4 | LACTATE_max | CHLORIDE_max | BICARBONATE_min | ALBUMIN_min | CREATININE_max | GLUCOSE_min | day1motor | ALBUMIN_min | PT_max |
| 5 | metastaticcancer | LACTATE_max | age | PT_max | ALBUMIN_min | LACTATE_max | CREATININE_max | CHLORIDE_max | ALBUMIN_min |
| 6 | CREATININE | age | LACTATE_max | BUN_max | PT_max | ALBUMIN_min | SODIUM_max | ANIONGAP_max | BICARBONATE_min |
| 7 | leukemia | SODIUM_max | SODIUM_max | INR_max | INR_max | age | BICARBONATE_min | CHLORIDE_min | PTT_max |
| 8 | PLATELET | ANIONGAP_max | ANIONGAP_max | PLATELET_min | age | BICARBONATE_min | INR_max | gender | CREATININE_max |
| 9 | BILIRUBIN | BUN_max | BUN_max | PTT_max | gender | SODIUM_min | PTT_max | SODIUM_min | ANIONGAP_max |
| 10 | aids | HEMATOCRIT_min | CREATININE_max | HEMATOCRIT_min | ANIONGAP_max | PTT_max | LACTATE_max | PTT_max | SODIUM_max |

If we look at each dataset by itself (Table 1), the difference in performance between the mortality scores and our models is much better in the septic dataset. It appears that a larger performance boost can be attained by creating models for specific diagnosis groups rather than one model for all diagnosis groups. XGBoost consistently performed worse than all other models in all scenarios. Logistic Regression performed about as well as the

traditional mortality scores which is expected due to the predictions of these scores being based on the same raw variables. Neural Nets consistently outperformed all other models when the minority class was oversampled [33].

The 10 features selected for each experiment as well as the clinician opinion can be found in Table 2 [33]. We note that there were significant differences between the top ten features chosen by the clinician as compared to those by the algorithms. The features selected by the clinician included comorbidities such as metastatic cancer, AIDS, and need for immunosuppression but these were less likely to be in the algorithm feature selection lists. The differences are stark but not necessarily surprising. Physicians being humans will always have biases and blind spots, both of which are more likely with the inherent stressors of the ICU. This study is limited due to the small size of our surveys as well as the known variability in clinician decision making for end of life care [45].

In Table 3, we present the number of common features selected by the clinicians and the various models for both datasets. We note that XGBoost with SHAP had more in common with the features selected by the clinicians. However, XGBoost was the worst predictor of mortality as compared to other models. It is important to note that both SHAP algorithms (XGBoost and the Neural Net) extracted the control variable (patient ID) in their top 10 features. However, this was not the case for the Neural Net with influence functions, which was also the best predictor of patient mortality [33].

On the sepsis dataset, the runtime for the Neural Net extended influence functions was around 1.5 minutes to calculate global feature importance. However, the runtime for SHAP with Neural Net was around 220 minutes. The computational efficiency of the proposed extended influence functions was even more evident in the all-comers dataset,

e.g., SHAP calculations took more than one week on a NVIDIA TITAN V GPU [33].

**Table 3**

*Common Feature Count*

| Model with algorithm Type | Septic | All-comers |
|---|---|---|
| Logistic regression coefficients | 3 | 2 |
| XGBoost with SHAP | 5 | 3 |
| NN with SHAP | 4 | 2 |
| NN with influence functions | 4 | 2 |

# Chapter 4

## Feature Selection Techniques for Mortality Prediction in the ICU

In a follow up work, we examined the 194 features from our previous dataset in an effort to find the smallest subset that allows us to maintain State of The Art (SoTA) performance. Our primary reason in seeking the smallest subset is to find a method that has high performance while causing little burden to the users. The main user of such an ICU prediction model would be ICU clinicians who don't have the time to enter in 15+ variables for each patient at a time. This is the case with the current tools available to clinicians [46].

### 4.1 Feature Selection

Feature selection is the process of reducing the number of input signals such that the most relevant attributes are used to create a predictive model [47]. There are a total of 194 features within the eICU database [48]. Therefore, it is imperative that this number is greatly reduced to not burden the user. Having excess tests or features not only slows down the time it takes to collect and enter the data, but also can have a negative impact on the results of the prediction [49]. Having the ability to decrease the number of features used while maintaining a relative measurement of success is crucial for this experimentation and is the motivation behind using feature selection for this research [46].

Feature selection can be further classified as either supervised or unsupervised. In an unsupervised method, the function is provided inputs, but no target function. The goal is to identify the most relevant features in the input dataset. This can be achieved by calculating the correlation with the target variable, or by applying feature-specific statistical tests (such as t-tests). Supervised methods, on the other hand, have the goal of identifying

the most relevant features in order to improve the performance of a target function. This can be done by using a training dataset, where the target output is known. The features that correlate with the target variable are identified and used to optimize the target function. In this work, the filter, wrapper, and embedded methods under supervised feature selection will be utilized [46].

### *4.1.1 Filter Methods*

Filter methods are those that are typically performed during the preprocessing stage of feature selection. For this method, the features are chosen based on certain characteristics or scores that they achieve in various statistical tests. Some of the most popular examples of filter methods include correlation, Chi-Square tests, and analysis of variance (also known as ANOVA) [50]. The features are then sorted based on their scores and a threshold is chosen either by heuristic or statistics to choose the final set of features [46].

### *4.1.2 Wrapper Methods*

Wrapper methods are those that evaluate various subsets and combinations of features in order to choose the one that produces the best result for the given algorithm. These methods are often referred to as greedy due to the fact that they search through all subsets and therefore can become computationally expensive and take a long time to execute. The benefit of this method is that it is guaranteed to provide the optimal set of features. The process of utilizing a wrapper method starts with choosing a search method or technique in order to select an available subset of features. Once the subset is identified, the desired machine learning algorithm is trained on the chosen subset. The model is then evaluated

and the process is repeated with various other subsets of features until the best model is identified. Popular search methods include forward selection, backward elimination, and bi-directional searches [51] [46].

### *4.1.3 Embedded Methods*

For embedded methods, as the name suggests, the feature selection is embedded within the training of the model. In the previous two methods, feature selection was performed before the model was trained. The process of the embedded method includes training a machine learning model and then deriving the feature importance from the model. As the model is being trained, the features that have the least impact on the prediction are discarded. Examples of embedded methods include lasso/ridge regression and decision trees [51] [46].

## 4.2 Methods

Our analysis consists of three components. First, an evaluation of the dataset with all 194 features is done. Second, a 1-1 comparison is made with our prior work by selecting the 20 features using each feature selection technique. Lastly, a quantitative and qualitative study of the performance of each feature selection method is compared. The following techniques were evaluated: variance threshold, Analysis of Variance (ANOVA), Mutual Information (MI), Recursive Feature Elimination (RFE), Elastic Net, and Principle Component Analysis (PCA) [46].

### *4.2.1 Variance Threshold*

The variance threshold method is a simple, baseline approach to feature selection. It removes any feature that has a variance less than a chosen threshold. By default, this technique eliminates features with zero variance, but can be changed to any desired threshold value. Features with little to no variance may often not contain much significant information and can in some instances reduce the overall performance of a model. Since this method is dependent on the statistics of the feature, this method must be performed before standardizing the data [40] [46].

### *4.2.2 Analysis of Variance*

The Analysis of Variance method, often referred to as ANOVA, is a widely popular filter method. For this research, the ANOVA f-test statistic was utilized. The ANOVA technique is used to determine how similar or different the means of two or more variables are to each other. It can also be utilized to realize the correlation between an independent variable and the dependent variable. Using the ANOVA method, f-scores are assigned to each feature where the higher f-score indicates a higher correlation between that particular feature and its impact on the output [50] [46].

### *4.2.3 Mutual Information*

Shannon Mutual Information between two random variables is defined by conditional entropy. The entropy of the class variable, Y, is desired to be very low in order to maximize classification performance. For a given feature X, the Mutual Information

between X and Y (denoted as $I(X;Y)$) is a measure of the change in entropy of Y due to the presence of X as defined in Equation 4.1. In this equation, $p(xy)$ is the joint probability of $x$ and $y$ and $p(x)$ and $p(y)$ are the marginal probabilities. Therefore, a high Mutual Information between a feature and the class label indicates that the feature is a good predictor of the class label [52] [46].

$$I(X;Y) = \sum_{x \epsilon X} \sum_{y \epsilon Y} p(xy) \log \left( \frac{p(xy)}{p(x)p(y)} \right) \tag{4.1}$$

### *4.2.4 Recursive Feature Elimination*

Recursive Feature Elimination (RFE) is a wrapper-type feature selection algorithm. It operates by first building a model on a dataset containing all features and then computing an importance score for the features using the model (ROC AUC for our case). Next, a set of features are removed, the model is retrained and the outputs are an updated importance score on the reduced feature set. This process is then repeated until all the least important features, determined by the importance score from the model, are eliminated. RFE requires two inputs to operate; number of $k$ features to keep and an estimator model with a built-in importance score [40] [46].

To find the $k$ number of features to keep, RFE with cross validation (RFECV) can be utilized. RFECV keeps track of a score computed from the model for a given number of features. Using logistic regression as the model for RFECV and tracking the ROC AUC score, the number of features to keep was based on the specifications from previous work [53] [46].

The following parameters were used; a five-fold stratified for cross-validation, lo-

gistic regression as the estimator and the area under the receiver operating characteristic curve (ROC AUC) for scoring the performance of the estimator. The dataset was divided into 33% testing (for scoring) and the remaining was used for training the estimator [46].

**Table 4**

*Statistics for Classifier Trained on all 194 Features*

| Measurement | Mean | Standard Error |
|---|---|---|
| Accuracy | 0.855106 | 0.002542 |
| Precision | 0.260902 | 0.003493 |
| Sensitivity | 0.819577 | 0.008339 |
| Specificity | 0.857289 | 0.002884 |
| **ROC AUC** | **0.918066** | **0.002748** |
| PRC AUC | 0.507141 | 0.009412 |
| Balanced Accuracy | 0.838433 | 0.003765 |

### *4.2.5 Elastic Net*

Elastic net is an embedded type feature selection algorithm that combines two types of regression. The first is Least Absolute Shrinkage and Selection Operator (LASSO). The second is Ridge regression. Both LASSO and Ridge Regression use regularization to avoid overfitting. Regularization achieves this by penalizing complex models by adding a penalty to the following cost function, $W$ in Equation 4.2. In this equation, $y_i$ is the target class for instance $i$, $x_{ij}$ is the feature value for instance $i$ and feature number $j$ and $w_j$ is the

corresponding weight value [46].

$$W = \sum_{i=1}^{N} \left( y_i - \sum_{j=0}^{M} w_j x_{ij} \right)^2 \tag{4.2}$$

Ridge regression uses L2 regularization, which modifies the cost function by adding a penalty term to the residual sum of squares. Equation 4.3 shows the added penalty term of lambda times the sum of square of weights, where $N$ is the total number of training instances and $M$ is the total number of input features [46].

$$W = \sum_{i=1}^{N} \left( y_i - \sum_{j=0}^{M} w_j x_{ij} \right)^2 + \lambda \sum_{j=0}^{M} w_j^2 \tag{4.3}$$

For LASSO regression, L1 regularization is used. This modifies the cost function by adding a penalty term of lambda times the sum of absolute value of weights to the residual sum of squares. This is shown in Equation 4.4 [46].

$$W = \sum_{i=1}^{N} \left( y_i - \sum_{j=0}^{M} w_j x_{ij} \right)^2 + \lambda \sum_{j=0}^{M} |w_j| \tag{4.4}$$

Finally, when combining the two penalty terms from LASSO and Ridge in Elastic-Net, an additional $\alpha$ term is added that determines the ratio of L1 to L2 regularization. The Elastic-Net cost function is given in Equation 4.5 [46].

$$W = \sum_{i=1}^{N} \left( y_i - \sum_{j=0}^{M} w_j x_{ij} \right)^2 + \alpha\lambda \sum_{j=0}^{M} |w_j| + (1-\alpha)\,\lambda \sum_{j=0}^{M} w_j^2 \tag{4.5}$$

The values of $\alpha$ and $\lambda$ for Equation 4.5 can be computed using elastic net with a cross validation (ElasticNetCV) function. A list of $\alpha$ values was provided and the value of $\lambda$ was picked automatically from the ElasticNetCV function [40]. The value of $\alpha$ was then

chosen by performing a grid search on the interval $[0, 1]$. The following parameter values were found to be optimal: $\lambda$ = 4.317e-4, $\alpha$ = 0.995 [46].

**Table 5**

*Comparison of Feature Selection Techniques*

| Method | ROC AUC |
|---|---|
| Clinician Chosen [33] | 0.8564 ± 0.002748 |
| Variance Threshold | 0.7661 ± 0.00747 |
| ANOVA | 0.8901 ± 0.00309 |
| Mutual Information | 0.8365 ± 0.00554 |
| PCA | 0.8980 ± 0.00302 |
| ElasticNet | **0.9056 ± 0.00336** |
| Recursive Feature Elimination | 0.8512 ± 0.00415 |

### *4.2.6 Principal Component Analysis*

Principal Component Analysis, more commonly known as PCA, is an unsupervised method that can be used for dimensionality reduction while having a maximum variability. The overall goal is to make sure all of the of the transformed features are linearly independent, as well as, finding components in order of highest importance. The PCA approach can be defined as the eigendecomposition of the covariance matrix $X^TX$ [46].

There are five steps to complete the process of principal component analysis. To begin with, the data must be standardized using Equation 4.6, where $\mu$ is the mean and $\sigma$

is the standard deviation of all features [46].

$$x_{\text{stand}} = \frac{x - \mu}{\sigma} \tag{4.6}$$

The standardized data would then have a mean of 0 for each feature while having a standard deviation of 1. Doing this will scale all features properly and prevent skewing in the results. Step 2 involves finding the covariance matrix of the standardized data. Equation 4.7 was used to find the covariance matrix where $\bar{x}_i$ is the mean of the $i$th column, $\bar{x}_j$ is the mean of the $j$th column, $x_{im}$ is the $i$th column and $x_{jm}$ is the $j$th column [46].

**Table 6**

*Minimum Number of Features Required*

| Method | Number of features |
|---|---|
| Variance Threshold | 102 |
| ANOVA | 13 |
| Mutual Information | 30 |
| PCA | 3 |
| ElasticNet | 6 |
| Recursive Feature Elimination | 26 |

$$\text{Cov}\,(i, j) = \frac{1}{n-1} \sum_{m=1}^{n} \left(x_{\text{im}} - \bar{x_i}\right) \left(x_{\text{jm}} - \bar{x}_j\right) \tag{4.7}$$

The resultant covariance matrix will be a square matrix $\mathrm{X^T X}$. The next step was to find the eigenvectors and eigenvalues using eigendecomposition. After finding the

eigenvalues, the fourth step is to sort them and its corresponding eigenvectors in descending order. The fifth and final step is to choose the number of components to keep from the highest importance to then transform the standardized data into a transformed matrix using Equation 4.8 [46].

**Table 7**

*Top 10 Features Selected*

| FR | Clinician Survey [33] | Variance Threshold | ANOVA | Mutual Information | Recursive Feature Elimination | Elastic Net |
|---|---|---|---|---|---|---|
| 1 | Age | CPK | Lactate | Lactate | Potassium | Lactate |
| 2 | Immunosuppresion | AST (SGOT) | GCS Motor | Mechanical Ventilation | paCO2 | GCS Motor |
| 3 | HepaticFailure | Lymphs | GCS Eyes | GCS Eyes | pH | paCO2 |
| 4 | Lactate | ALT (SGPT) | GCS Verbal | GCS Motor | Myglobin | Bicarbonate |
| 5 | Metastatic Cancer | FiO2 | Intubation | GCS Verbal | Lactate | Fio2 |
| 6 | Creatinine | Glucose | Ventilator | Albumin | Glucose | BUN |
| 7 | Leukemia | Platelets | AST (SGOT) | Age | Respiratory Rate | WBC |
| 8 | Platelet | Akaline | PT-INR | Creatinine | Chloride | Mechanical Ventilation |
| 9 | Bilirubin | PaO2 | Anion Gap | BUN | Albumin | Sodium |
| 10 | Aids | Glucose | ALT (SGPT) | INR | TV | Age |

$$T_R = XW_R \tag{4.8}$$

In Equation 4.8, $T_R$ is the transformed matrix, $W_R$ are the loadings or eigenvectors, $X$ is the standardized data, and $R$ itself is the number of components chosen [46].

## 4.3 Model Selection

For our first and second analyses, missing data was assumed to be missing at random and features with missingness at 70% or greater were dropped from the dataset. Next, features with pair-wise Pearson correlations greater than 0.9 were also discarded.

The data was then standardized and imputed with multiple imputation [54]. To correct for the 95%-5% class imbalance, we apply the Synthetic Minority Oversampling (SMOTE) technique [55]. For each 20-feature set, we perform a 10-fold cross validation and determine the mean ROC AUC and 95% confidence interval. The classifier is a multi-layer perceptron (MLP) with 2 hidden layers and SELU activations [38]. The model is trained for 127 epochs using full-batch stochastic gradient descent with a learning rate of 0.03104 and Nesterov momentum of 0.4204 [56]. The hyperparameter values were obtained by performing Bayesian Optimization using ROC AUC as the target metric [33] [46].

For our third analysis, we want to consider all features. Therefore we do not drop features based on missingness and correlation. For each method, we choose an increasing number of features and perform 10-fold cross validation to collect a ROC AUC and 95% confidence interval. The rest of the preprocessing procedure remains the same as the previous analyses. This process results in 194 ROC AUC means and 95% confidence intervals. To quantitatively measure which method performs the best, we compute the area under the ROC AUC vs number of features curve. In addition, for each method, we determine the minimum number of features that can provide a statistically significant result over the work in [33]. Lastly, we provide a qualitative measure of performance by comparing the top selected features from each method compared to clinician opinion [33]. Our baseline measure of performance is to consider using all features. The performance metrics for this baseline can be found in Table 4 [46].

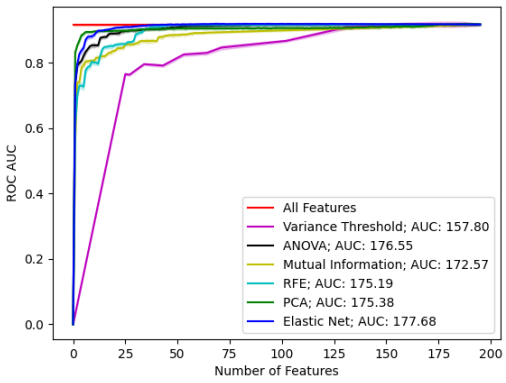
## 4.4 Results and Discussion

In Table 4, we show the target performance by using all 194 features. Next, we reduce the number of features to what was used in the previous state-of-the-art [33], which is 20 features. The results for each method is shown in Table 5. Here, we observe that the top methods are ANOVA, PCA and ElasticNet. When looking at the number of features required to outperform the opinion of the clinicians (as in Table 6), we show that PCA and Elastic Net vastly do better than other methods. This trend continues in Figure 3. The PCA approach is quick to rise to a ROC AUC value of 0.89 but has trouble increasing as more components are added. The best performing method overall is Elastic Net due to it having the highest ROC AUC value as features are increased as well as consistent performance across the other experiments [46].

In Table 7, we show the top 10 features chosen by each method and compare them to what is selected by ICU clinicians. In the clinician column, we observe a large amount of indicator variables as opposed to the columns selected by feature selection. There are also many common features/feature sets that are selected by multiple methods. These include: lactate, GCS scores (Eyes/Motor/Verbal), mechanical ventilation, and age. We postulate that this difference can be attributed to clinician focus on comorbidities with our machine learning models having no knowledge of diagnosis and patient history. It is possible that the optimal feature set lies somewhere in the combination of these methods. We defer the use of a meta set of features selected by multiple methods for future work [46].

**Figure 3**

*10 Fold Cross Validated ROC AUC*



# Chapter 5

## Efficient Scaling of Bayesian Neural Networks

The state of machine learning research has seen tremendous growth, with increasingly complex and large-scale models being developed [57]. This increase in size and complexity, however, has led to a notable concern regarding overfitting [58]. As models become larger, they may inadvertently memorize training data instead of learning to generalize, resulting in poor performance on unseen or novel data samples. A guiding principle against complexity increase has been the The Minimum Description Length Principle which states that the best model is one which minimizes the distance between the model and data and the models description of itself [59]. This principle is the foundation of stochastic modeling and a benefit of using Bayesian inference [60].

Bayesian inference provides a principled approach for dealing with uncertainty by combining prior knowledge and observed data to update beliefs about model parameters [61]. In the context of machine learning, Bayesian Neural Networks (BNNs) extend this framework by incorporating Bayesian principles into the architecture and training of neural networks [62]. BNNs estimate the posterior distribution over model parameters, allowing them to capture and quantify uncertainties in both predictions and model structures. By leveraging Bayesian inference, BNNs offer a more robust and adaptive learning approach, which can potentially alleviate overfitting and improve generalization performance in complex machine learning tasks [28] [60].

BNNs have demonstrated successful implementations across a wide range of tasks, including computer vision [29, 31, 63], speech [64], and natural language processing [65]. BNNs have shown particular promise in critical decision-making tasks [66–68],

due to their ability to capture and model uncertainty in both parameters and predictions [29]. Several classical techniques have been used to approximate Bayesian inference for neural networks, such as Laplacian approximation [69], Hamiltonian Monte Carlo [70], and Variational Inference (VI) [28]. Specifically, estimating the parameters of the variational distribution that approximates the posterior of the latent variables given an observation is analogous to averaging model parameters via Monte Carlo sampling [71]. This model averaging inherently reduces variance in computed parameters, thus mitigating overfitting [30]. Moreover, the regularization term of the VI algorithm can lead to extreme self-compression of the model's parameters [72], further decreasing overfitting [60].

A major drawback of BNNs is their limited scalability in terms of both model and data size [73]. A high-quality implementation of a VI-based training scheme called Variational Density Propagation (VDP) has emerged, offering state-of-the-art performance on various computer vision tasks, reduced epistemic uncertainty, and an estimate of aleatoric uncertainty [31, 63]. However, the primary limitation of the VDP algorithm is the substantial computational load due to the propagation of the full covariance matrix through layers of the neural network. Propagating these large matrices quickly becomes infeasible in terms of space and time for all but the smallest ResNets, significantly limiting the applicability of the VDP method for problems that require training on datasets larger than CIFAR-10 [74] [60].

In this chapter, we propose a modification to the VDP method, in which we disregard the off-diagonal terms of the covariance matrix and only propagate the diagonal terms. This simplification enables efficient scaling of these models to larger and more complex datasets, with a theoretical memory requirement only twice that of a traditional deep neural network,

while retaining all the benefits of BNNs [60].

## 5.1 Related Work

### *5.1.1 Self-Compression*

Since their inception in the early 1990s, BNNs have been developed with the goal of minimizing the information content in the weights of a neural network. This approach is based on the principle that smaller model sizes often lead to more generalizable models [28, 69]. However, contemporary BNN methods such as Bayes by Backprop (BBB) [29] and Dropout CNNs [75] appear to lack this property. In the case of BBB, parameter histograms suggest a less efficient use of parameters compared to traditional deterministic neural networks [29]. To the best of our knowledge, the only modern framework that exhibits this self-compression property is Variational Density Propagation (VDP) [31, 63, 72]. The self-compression characteristic of VDP can be attributed to the propagation of variance information through the network layers [72]. By quantifying parameter uncertainty, the model can selectively target less important parameters during training. In the absence of uncertainty quantification, the network's utilization of its parameter space is less efficient [60].

### *5.1.2 Uncertainty Quantification*

The total uncertainty in a machine learning model can be characterized by two distinct components. The first is epistemic uncertainty, which arises from the parameters of the model. The second is aleatoric uncertainty, which stems from the data or environment. Uncertainty quantification is becoming increasingly important in high-stakes decision-making,

as establishing trust between users and models is crucial for the widespread adoption of such models [26]. To date, the implementation of machine learning models in high-stakes domains, such as medicine, has not achieved substantial user acceptance and confidence [5, 6]. This is primarily due to high false alarm rates [4] and suboptimal test characteristics [27]. VI has been demonstrated to alleviate the impact of epistemic uncertainty by effectively averaging predictions during inference [28–30]. Furthermore, the output variance associated with a single prediction obtained from a BNN can quantify the level of aleatoric uncertainty in that prediction [31, 32] [60].

### *5.1.3 Explainability*

A common approach to interpreting predictions in computer vision models involves calculating sensitivity maps. These gradient-based methods are used to determine the contribution of each image pixel to the classification output [17]. However, interpretations derived from deep neural networks, particularly sensitivity maps, are often fragile—meaning they are sensitive to small perturbations in the input or model [76–78]. One potential solution to this issue involves sampling the surrounding input space, adding Gaussian noise, and averaging the resulting interpretations [18]. This averaging technique for reducing uncertainty aligns with our hypothesis on BNNs and weight averaging. It has been demonstrated that the explanations generated by the VDP model outperform those from traditional models [79] [60].

## 5.2 Variance-Only Variational Density Propagation (VDP++)

Our work presents a novel simplification of the VDP algorithm developed in [31]. This method utilizes VI and assumes Tensor Normal Distributions (TNDs) defined over the parameters of the neural network in order to propagate the first two moments (mean and covariance) of these TNDs through the layers of a neural network [31]. However, in an experimental setting, propagating covariance proves to be computationally expensive in terms of both time and space. As these networks scale, calculating such large matrices becomes increasingly burdensome, rendering the method impractical. In this work, we have streamlined the VDP algorithm by propagating only the diagonal elements of the covariance matrix, i.e., the variance. This simplification, originally discussed by Hinton [28], is akin to the assumptions of the Naive Bayes classifier [80] [60].

For the sake of brevity, we have omitted the original derivation of the VDP algorithm and only provide the modifications needed to compute the variance-only version [31]. To describe our method, we use a 2-layer fully-connected neural network as an example. In a traditional or deterministic neural network, Equation 5.1 describes the forward pass of the model [60].

$$\begin{aligned} z &= Wx + b^{[1]}, \\ a &= f(z), \\ \hat{y} &= g(Va + b^{[2]}), \end{aligned} \tag{5.1}$$

where $W \in \mathbb{R}^{j \times k}$ is the weight matrix of layer 1, $x \in \mathbb{R}^{k \times 1}$ is the input vector, $b^{[1]} \in \mathbb{R}^{j \times 1}$ is the bias vector in layer 1, $z \in \mathbb{R}^{j \times 1}$ is the result of the linear operation in layer 1, $f$ is an arbitrary element-wise non-linear function, $g$ is an arbitrary non-linear activation function

that does not operate element-wise, $a \in \mathbb{R}^{j\times 1}$ is the result after applying the non-linear activation function in layer 1, $V \in \mathbb{R}^{l\times j}$ is the weight matrix in layer 2, $b^{[2]} \in \mathbb{R}^{j\times 1}$ is the bias vector in layer 2, $\hat{y} \in \mathbb{R}^{l\times 1}$ is the predicted output, $k$ is the dimensionality of the input vector, $j$ is the number of nodes in layer 1 and $l$ is the number of classes to predict [60].

To propagate the first two moments, several assumptions must be made. First, let us consider $w_m^\top = m^{\text{th}}$ row of $W$, $m = 1, 2, ..., j$ and $z_m = w_m^\top x + b_m^{[1]}$, $m = 1, 2, ..., j$. Next, consider the following assumptions: the input vector $x$ is deterministic, $a \sim \mathcal{N}(\mu_a, \Sigma_a)$, $w_m \sim \mathcal{N}(\mu_{w_m}, \Sigma_{w_m})$, $m = 1, 2, ..., j$, $b_m^{[1]} \sim \mathcal{N}(\mu_{b_m^{[1]}}, \sigma^2_{b_m^{[1]}})$, $m = 1, 2, ..., j$, $v_n \sim \mathcal{N}(\mu_{v_n}, \Sigma_{v_n})$, $n = 1, 2, ..., l$, $b_n^{[2]} \sim \mathcal{N}(\mu_{b_n}^{[2]}, \sigma^2_{b_n^{[2]}})$, $n = 1, 2, ..., l$ and the weight vectors $w_m$, $a$, bias $b^{[1]}$ and $b^{[2]}$ are mutually uncorrelated with each other for $m = 1, 2, ..., j$. Based on these assumptions, the elements of $\mu_z$ and $\sigma_z^2$ are given in Equations 5.2 and 5.3 [60].

$$\begin{aligned} \mu_{z_m} &= \mathbb{E}[w_m^\top x + b_m^{[1]}], \\ &= \mathbb{E}[w_m^\top]x + \mathbb{E}[b_m^{[1]}], \\ &= \mu_{w_m}^\top x + \mu_{b_m^{[1]}}. \end{aligned} \tag{5.2}$$

$$\begin{aligned} \sigma^2_{z_m} &= \text{Var}[w_m^\top x + b_m^{[1]}], \\ &= x_m^\top \Sigma^2_{w_m} x_m + \sigma^2_{b_m^{[1]}}, \\ &= x_m^2 [\sigma^2_{w_m}]^\top + \sigma^2_{b_m^{[1]}}. \end{aligned} \tag{5.3}$$

Since we have assumed the weight vectors and the elements of the bias vector to be uncorrelated, $\Sigma_{w_p w_q} = 0$ and $\sigma_{b_p b_q} = 0$ for $p \neq q$, where $p, q = 1, 2, ..., j$. Hence, the covariance is zero. To propagate the first two moments through an arbitrary element-wise non-linear function, we utilize the first-order Taylor series approximation shown in

Equations 5.4 and 5.5 [60].

$$\begin{aligned} a &= f(z), \\ &= f(\mu_z) + f'(\mu_z)(z - \mu_z) + \ldots \\ &\approx f(\mu_z) + f'(\mu_z)(z - \mu_z), \\ \mathbb{E}[a] &= \mu_a \approx f(\mu_z), \\ \mu_a &= f(\mu_z). \end{aligned} \tag{5.4}$$

$$\sigma_a^2 = \sigma_z^2 \odot (f'(\mu_z))^2. \tag{5.5}$$

For the second layer of the network, we can no longer consider the incoming vector, $\mu_a$, to be deterministic. Additionally, we now need to propagate the incoming variance, $\sigma_a^2$. Again, we assume the off-diagonal elements of the covariance $\Sigma_a$ to be 0 and choose to only propagate the variance. The mean and variance propagated through the second fully-connected layer are given in Equations 5.6 and 5.7 [60].

$$\begin{aligned} \mu_{\tilde{y}} &= \mathbb{E}[v_n^\top a + b_n^{[2]}], \\ &= \mathbb{E}[v_n^\top]\mathbb{E}[a] + \mathbb{E}[b_n^{[2]}], \\ &= \mu_{v_n}^\top \mu_a + \mu_{b_n^{[2]}}. \end{aligned} \tag{5.6}$$

$$\begin{aligned} \sigma_{\tilde{y}}^2 &= \mathrm{Var}[v_n^\top a + b_n^{[2]}], \\ &= \mathrm{Var}[v_n^\top a] + \mathrm{Var}[b_n^{[2]}], \\ &= \mathrm{Tr}(\Sigma_v^2 \Sigma_a^2) + \mu_v^\top \Sigma_a^2 \mu_v + \mu_a^\top \Sigma_v^2 \mu_a + \Sigma_{b_n^{[2]}}^2, \\ &= \sigma_v^2 [\sigma_a^2]^\top + \mu_v^2 [\sigma_a^2]^\top + \mu_a^2 [\sigma_v^2]^\top + \sigma_{b_n^{[2]}}^2. \end{aligned} \tag{5.7}$$

Since the diagonals of the covariance matrix are along the rows of the variance matrix, the trace of the product of two vectors is equivalent to their inner product [60].

Next, we use a non-linear activation function such as softmax to obtain the predictions of our model. Since $g$ does not operate element-wise on our mean and variance, we use a slightly different Taylor-series to obtain Equations 5.8 and 5.9 [81] [60].

$$\mu_{\hat{y}} \approx g(\mu_{\tilde{y}}), \tag{5.8}$$

$$\sigma^2_{\hat{y}} \approx J_g^2 \sigma^2_{\tilde{y}}, \tag{5.9}$$

where $J_g$ is the Jacobian matrix of the softmax function $g$ with respect to $\tilde{y}$ and calculated at $\mu_{\tilde{y}}$ [60].

Finally, we use the Evidence Lower Bound (ELBO) function, $\mathcal{L}(\phi, D)$, which consists of two parts: the expected log-likelihood of the training data given the weights, and a regularization term, $D = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$ given by Equation 5.10 [60].

$$\mathcal{L}(\phi, D) = \mathbb{E}_{q(\phi)}[\log p(D|\phi)] - \mathrm{KL}[q(\phi)|p(\phi)]. \tag{5.10}$$

In Equation 5.10. $\phi$ represents two aspects, namely, the (1) weights $W$ and $V$ and (2) biases $b^{[1]}$ and $b^{[2]}$. The expected log-likelihood is given in Equation 5.11 and the KL term is given in Equation 5.12 [60].

$$
\begin{aligned}
E_{q(\phi)}[\log p(D|\phi)] &\approx \frac{1}{M}\sum_{m=1}^{M}\log p(D|\phi), \\
&\approx -\frac{Nl}{2}\log(2\pi) - \frac{1}{M}\sum_{m=1}^{M}\Bigg[\frac{N}{2}\log(|\Sigma_{\hat{y}}|) \\
&\quad + \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \mu_{\hat{y}}^{(m)})^{\top}(\Sigma_{\hat{y}}^{(m)})^{-1}(y^{(i)} - \mu_{\hat{y}}^{(m)})\Bigg], \\
&\approx -\frac{Nl}{2}\log(2\pi) - \frac{1}{M}\sum_{m=1}^{M}\Bigg[\sum_{k=1}^{l}\log\sigma_{\hat{y}_k}^2 \\
&\quad + \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - \mu_{\hat{y}}^{(m)}\right)^2 \mathrm{diag}(\sigma_{\hat{y}}^2)^{-1}\Bigg].
\end{aligned}
\tag{5.11}
$$

In Equation 5.11, $y^{(i)}$ is the true label of the $i^{\text{th}}$ data point, $N$ is the number of data points and $M$ is the number of Monte Carlo samples needed to approximate the expectation by summation [60].

Recall that we have assumed the weight vectors and the elements of the bias vector to be uncorrelated, $\Sigma_{w_p w_q} = 0$ and $\sigma_{b_p b_q} = 0$ for $p \neq q$, where $p, q = 1, 2, ..., j$. It follows that $\Sigma_{\hat{y}}$ is a diagonal matrix with elements $\sigma_{\hat{y}}^2$. The log determinant of a diagonal matrix simplifies to the product of the elements that can be implemented as a sum of the log of the elements, $\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{k=1}^{l}\log\sigma_{\hat{y}_k}^2\right)$ to prevent numerical overflow. Additionally, the inverse of a diagonal matrix is equivalent to the reciprocal of each element, $\mathrm{diag}(\sigma_{\hat{y}}^2)^{-1}$ [60].

$$
\mathrm{KL}[q(\phi)|p(\phi)] = -\frac{1}{2}\sum_{n=1}^{l}(j\log\sigma_{v_n}^2 - ||\mu_{v_n}||_F^2 - j\sigma_{v_n}^2). \tag{5.12}
$$

### *5.2.1 VDP++ for Convolutional Kernels*

In practice, convolution operations are implemented as matrix multiplications. Therefore, no additional derivations are needed to propagate the first two moments through

convolutional kernels. For max-pooling, we cannot take the maximum variance. Instead, we utilize the co-pooling operation. The co-pooling operation is the same for the first moment: the maximum of the means in the kernel is passed forward. For the variance, we keep only the elements of the variance that correspond to the maximum means [31] [60].

### *5.2.2 VDP++ for Residual Connections*

Residual connections are a module used in deep architectures to resolve the issue of vanishing gradients. These connections propagate parameters from previous layers in the neural network by concatenating the output of one layer to the input of the next layer [82]. The residual function is effectively a non-elementwise, non-linear function. Therefore, we can use the same formulation for Softmax as in Equation 5.9 [63] [60].

$$\begin{aligned} \mu_{\mathbf{x}_{l+1}} &\approx \mu_{\mathbf{x}_l} + \mathcal{F}(\mu_{\mathbf{x}_l}), \\ \sigma^2_{\mathbf{x}_{l+1}} &\approx J^2 \sigma^2_{\mathbf{x}_l}. \end{aligned} \tag{5.13}$$

In Equation 5.13, $J$ is the Jacobian of $x_{l+1}$ with respect to $x_l$ and $\mathcal{F}$ is the residual function [63] [60].

### *5.2.3 VDP for Vision Transformers*

A novel contribution in this work, we extend the VDP framework to Vision Transformers (ViTs). Given the simple nature of the ViT architecture, to implement models like ViT-S/16 and ViT-B/16 [83, 84], we need only implement the layer normalization operation [85] for VDP. Similar to the batch normalization formulation for VDP [63], layer normalization for VDP++ is given in Equation 5.14 where $x$ is the input and $y$ is the output of the

layer [60].

$$\begin{aligned} \mu_y &= \frac{\mu_x - \mu_{LN}}{\sqrt{\text{Var}(\mu_x) + \epsilon}} \odot \gamma + \beta, \\ \sigma_y^2 &= \left( \frac{\mu_{LN}}{\sqrt{\text{Var}(\mu_x) + \epsilon}} \right)^2 \odot \sigma_x^2, \end{aligned} \tag{5.14}$$

Also in Equation 5.14, $\mu_x$ and $\sigma_x^2$ are the incoming mean and variance, $\mu_{LN}$ is the mean of the layer norm layer, $\gamma$, $\beta$, and $\epsilon$ are hyperparameters of the layer norm layer and $\odot$ denotes element-wise multiplication.

### 5.3 Validation and Robustness

To validate our approach, we will assess the VDP++ method on the MNIST dataset using various back-ends: a deterministic CNN, VDP [31], Bayes by Backprop (BBB) [29], and VDP++. On the CIFAR-10 dataset, we will evaluate the deterministic model and VDP++. For the MNIST dataset, we employ a LeNet architecture, while for CIFAR-10, we use ResNet-18. In order to determine whether any of the backends result in significantly different test accuracy, we will repeat training five times and perform a one-way Analysis of Variance (ANOVA) on the results. As our approach aims to simplify the original method while preserving its properties, we would like to show no significant differences in performance among the methods, i.e., $p > \alpha$ [60].

To evaluate the robustness properties of VDP, we will replicate an experiment from the original implementation, which entails adding zero-mean Gaussian noise to the test set at varying magnitudes and examining the pattern of the normalized output variance [31]. We will use the Spearman rank-order correlation measure to compare the two implementations. Additionally, we will compare the maximum GPU memory allocation and training time for

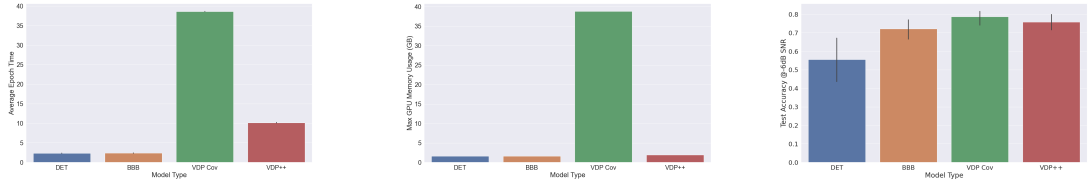
each formulation at a fixed batch size of 512 to benchmark each method's performance [60].

To examine the self-compression properties of VDP++ [72], we will evaluate the performance of each backend on the MNIST test set as each model undergoes global pruning by weight magnitude (L1 unstructured pruning) [60].

Lastly, we will assess the explainability of saliency maps produced by the deterministic model and VDP++. The metrics we will utilize to evaluate the quality of the explanations are Infidelity and Sensitivity Max [86]. We will employ pair-wise t-tests to determine statistical significance, if any. For all statistical analyses, we choose $\alpha = 0.01$ [60].

**Figure 4**

*Epoch Time (a), GPU Memory (b) and Test Accuracy (c)*




**(a)** **(b)** **(c)**

## 5.4 Scaling to Large Datasets

Remember that the primary limitation of using Bayesian neural networks lies in the original formulation's inability to scale to model sizes larger than ResNet-18. Overcoming this drawback would enable the application of BNNs to more complex problems and larger datasets, which is essential for advancing the state-of-the-art in machine learning. To that end, we will evaluate the performance of VDP++ on the ImageNet-1k dataset using Vision

Transformers (ViTs). This analysis involves examining the ViT-B/16 [84] and ViT-S/16 [83] architectures. To our knowledge, this is the first analysis of BNNs on ImageNet-1k using the ViT architecture [60].

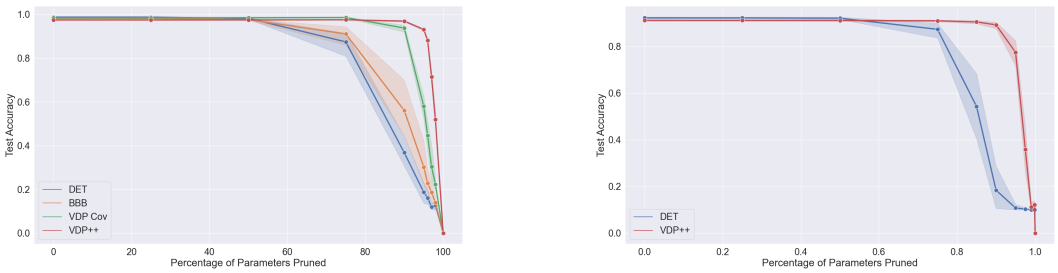
## 5.5 Results

### *5.5.1 Time-Space Improvements*

We began first by examining the advantages of our method in both time and space. On the MNIST dataset, Figure 4 shows a comparison of average epoch time and maximum GPU allocation for each method. The original VDP method [31] denoted "VDP Cov" takes nearly 10 times longer to complete one epoch of training as well as 10 times the GPU memory requirement as compared to traditional networks (DET) and Bayes-by-Backprop (BBB) [29]. Training on MNIST with a batch size of 512 requires nearly 40GB of GPU memory. Our proposed method, VDP++, reduces the average epoch time by about 4 times over the prior method and reduces the GPU memory requirement by a factor of 10 [60].

**Figure 5**

*Global Unstructured Pruning on (a) MNIST and (b) CIFAR-10*



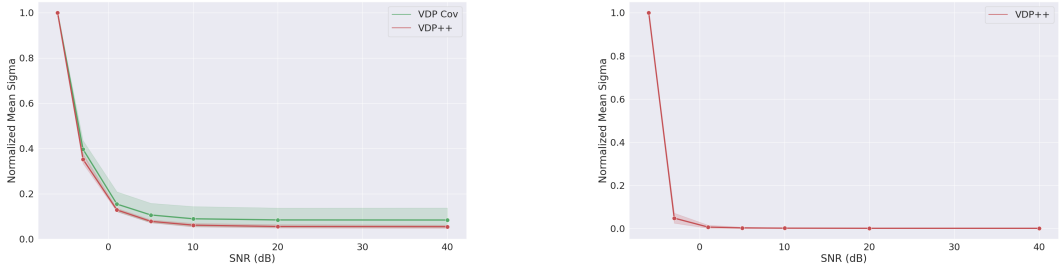

**(a)** **(b)**

### *5.5.2 Self-Compression*

The first of the inherent properties of VDP that we investigated was self-compression. We began by training 5 models from each method on MNIST (DET, BBB, VDP Cov, VDP++) and CIFAR-10 (DET, VDP++). A one-way ANOVA statistic was computed on the test statistics of each method and it was found that the performances were not significantly different from one another ($p > 0.01$). Next, we iteratively pruned each model using global L1 unstructured pruning and computed test statistics. Figure 5 shows the result of this procedure. The VDP Cov and VDP++ approaches perform similarly on MNIST (Figure 5a). However, VDP++ is able to prune $> 90\%$ of its parameters before the performance begins to drop significantly. This is verified by using a t-test of the statistics that compares $< 90\%$ pruning and $> 90\%$ pruning. This trend continues to hold for the CIFAR-10 experiments (Figure 5b) [60].

**Figure 6**

*Model's Noise Robustness*



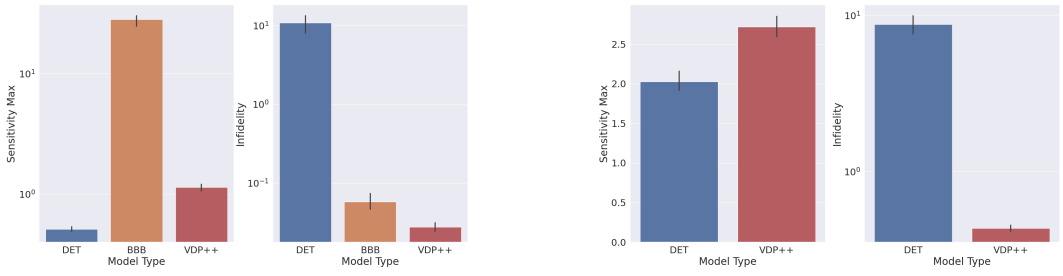

**(a)** **(b)**

### *5.5.3 Noise Robustness and Uncertainty Quantification*

Figure 4c shows the result of adding zero-mean Gaussian noise to the test set of MNIST and examining the test accuracy at the highest level of noise (SNR of -6 dB). Here, the Bayesian models (BBB, VDP Cov, VDP++) all significantly outperform the deterministic network (DET). Pair-wise t-tests were performed on deterministic versus each Bayesian method at this level of noise and all p-values were significant ($p < 0.01$). Figure 6 shows the output variance as a function of SNR normalized by the -6 dB value for both MNIST (Figure 6a) and CIFAR-10 (Figure 6b). When comparing the behavior of VDP-Cov and VDP++ in the presence of varying amount of noise (Figure 6a), it was found that there was no significant difference (Spearman correlation $p < 0.01$) between the methods [60].

**Figure 7**

*Explanation Quality of (a) MNIST and (b) CIFAR-10*



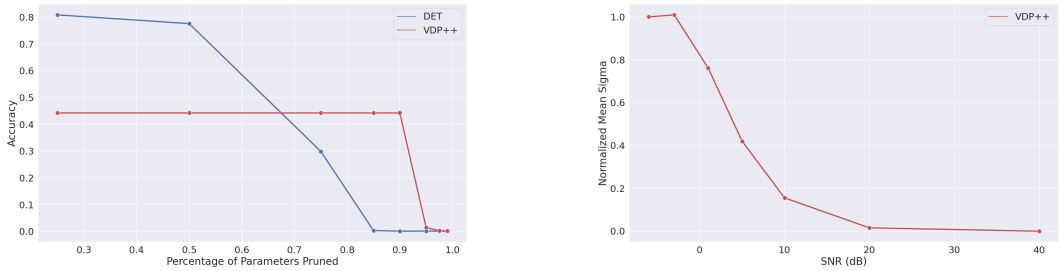

**(a)** **(b)**

### *5.5.4 Explanation Sensitivity*

Due to the high regularization of the network, we postulated that explanations via saliency maps from Bayesian Networks would be less sensitive to perturbations. Figure 7 shows the average Sensitivity Max and Infidelity on MNIST (Figure 7a) and CIFAR-10 (Figure 7b) (lower is better). Pair-wise t-tests indicate significant differences between methods for all experiments ($p < 0.01$). The deterministic method narrowly outperformed VDP++ by Sensitivity Max. However, VDP++ significantly outperformed all methods by Infidelity [60].

**Figure 8**

*Pruning Experiments (a), and Robustness Experiments (b) ImageNet*




**(a)** **(b)**

### *5.5.5 ImageNet and VDP-ViT*

To first validate our implementation of VDP-ViT, we trained a small ViT on MNIST and confirmed noise robustness using the variance vs SNR experiment. We then began parameter sweeps for ViT-B/16 and ViT-S/16 but ultimately decided on only optimizing

ViT-S/16 due to long training times. The training was split across 6 Quadro RTX 8000 GPUs following the regime outlined by Beyer *et al.* [83]. Ultimately, we were only able to obtain a model with 45% top-1 training accuracy given the long training times and large parameter space to search. Figure 8 shows the same analyses as above using the underfitted ImageNet-1k model. Even in this state, the VDP++ model still retains noise robustness and self-compressing properties, remaining consistent with the smaller models. The results for explanation sensitivity were not consistent with our prior results most likely due to the low performance of the model [60].

### 5.6 Discussion

Our study aimed to investigate the benefits and properties of the proposed novel VDP++ method compared to the original VDP method and other traditional networks. The results show that VDP++ offers several key advantages, including significant time-space improvements, self-compression, noise robustness, and uncertainty quantification. The time-space improvements demonstrated by VDP++ enable more efficient training and reduced GPU memory requirements, which are crucial in large-scale applications. This addresses the primary limitations of the original VDP method, which required much longer training times and higher memory consumption [60].

In terms of self-compression and noise robustness, VDP++ performs similarly to the original VDP method. Additionally, we show that it is able to prune more than 90% of its parameters before performance significantly declines. This ability to maintain performance despite substantial pruning suggests that VDP++ learns more efficient representations and could lead to more compact models that may be beneficial for edge computing [60].

The evaluation of the explanations produced by the various backends revealed unexpected results. Our original hypothesis was that due to the averaging effect of BNNs, we would have expected to see an improvement in Sensitivity Max (measures the stability and robustness of the explanation with respect to small perturbations in the input data) over deterministic models. An improvement in infidelity would suggest that the explanation approximates the underlying model's decision-making process better in BNNs than in deterministic models. The difference in Sensitivity Max, while significant ($p < 0.01$), was quite small. According to the original work from which it was derived, this score can be reduced by modifying the formulation of the saliency map, while infidelity cannot [86]. This yields promising results for the use of BNNs in areas where interpretability and explainability are desirable [60].

During our testing, we found that the balance between the two terms of the loss function in Equation 5.10 had the largest effect on the convergence and subsequent properties of our models. Since the KL term of Equation 5.12 is largely dominated by matrix norms, in larger models especially, this term tends to dominate. When this happens, the model compresses itself at the cost of performance. Conversely, when the KL term is scaled down too much (i.e. $\mathbb{E}_{q(\phi)}[\log p(D|\phi)] \gg \mathrm{KL}[q(\phi)|p(\phi)]$), the model overfits and the resulting models lack the inherent properties like self-compression and noise robustness, effectively becoming a deterministic model. To mitigate this behavior, we provided scaling terms to our hyperparameter optimizer and searched outright. This proved to be the fastest and most reliable method for training VDP++ models with high performance while retaining their inherent properties [60].

For classification problems such as ImageNet-1k, the Negative Log-Likelihood

(NLL) term in the ELBO loss function became a bottleneck. In practice, the NLL term uses one-hot encoded labels. This led to vanishing gradients and low model performance. To alleviate this problem, we changed the NLL term to categorical cross-entropy for our ImageNet-1k experiments. We confirmed that this change did not negatively affect our prior results [60].

## 5.7 Limitations

Our limitation in this study was a lack of computing resources. Given the increased parameter space we needed to search for our ImageNet-1k experiment, it became infeasible to expect high model performance in a short period of time. It is also worth noting that the original ViT formulation made use of the JFT-300M dataset for pretraining, which is not publicly available. We have substantially reduced the computational burden of the original VDP method while preserving its desirable properties. In tandem with our preliminary results on ImageNet-1k, scaling this approach to larger datasets is both possible and feasible given adequate computing resources and time. The robustness and improved explanatory capabilities of the VDP framework make it highly attractive for high-risk domains such as healthcare. Furthermore, the self-compressing properties render this technique suitable for applications with varying resource constraints, such as edge computing and large language models where computational resources are highly valuable [60].

# Chapter 6

## Revisiting the Fragility of Influence Functions

Influence functions were originally proposed to diagnose and debug linear models by predicting the parameter or loss change due to removing a training instance [87]. Their extension to deep learning models, however, did not occur until recently [24]. Influence functions and their applications have been well studied since their reemergence and have since been adopted as a mainstream tool for the interpretation of deep models in a variety of data modalities [88–91], including high-risk areas such as mortality prediction for patients in the Intensive Care Unit [33]. Due to the diversity of the use cases for influence functions, understanding their limitations is imperative if they are to be used to explain model behavior. Without key validation procedures, we run the risk of providing misleading or incorrect information to the model users [92].

To validate these methods, we must first agree on a metric to rate explanations. Spearman correlation between the approximate and true loss differences has been used as a metric to determine the accuracy of influence estimates. The approximate loss differences are given by the influence functions and the true loss differences are obtained by retraining an already trained network after removing a specific training sample [24]. Recent works have used this metric to study the effects that increases in model and dataset size have on the influence functions. It has been found that influence functions are extremely sensitive to these increases [93] [92].

It is well known that increases in model and dataset size affect the curvature of the loss function [94–97]. Convexity of the loss function is a critical assumption of influence functions as they heavily rely on the approximation of the inverse Hessian-vector product.

The stochastic estimation algorithm used to compute the inverse Hessian-vector product assumes that the Hessian is positive semidefinite [37, 98]. Preliminary work has been done to try to remedy these problems via higher-order approximations [99] and group influences [100], i.e., computing loss differences for more than one training instance at a time [92].

In this chapter, we examine the cases where influence functions seemingly fail, i.e. have low Spearman correlation between approximate and true loss differences. We obtain the operands for the correlation using the retraining procedure introduced in [24], where the approximate loss differences are computed for the test point with the maximal loss using influence functions. Each training point is removed one at a time and the neural network is retrained from the optimal parameters until convergence in order to obtain the true difference in the loss function values. We determined that this training procedure is not valid for most applications of deep learning and present evidence for these cases [92].

### 6.1 Influence Function Guidance

Ideally, a model must be trained until the optimal parameters $\hat{\theta}$ are obtained in order to compute the influence functions. For a single test instance, $z_{\text{test}}$, we would then compute the inverse Hessian-vector product, $\nabla_\theta L(z_{\text{test}}, \hat{\theta})^T H_{\hat{\theta}}^{-1}$, using stochastic estimation [98]. In reality, due to non-linearities in our networks, our objective function may become non-convex and we obtain our parameters $\tilde{\theta}$ via SGD, where $\tilde{\theta} \neq \hat{\theta}$. In this case, the Hessian may have negative eigenvalues which would cause the stochastic estimation algorithm to not converge. To address this, we adopt a regularization scheme similar to L2 regularization discussed by Koh *et al.*[24]. We regularize the computation of the Hessian-vector product using a damping term of $\lambda = 0.01$. We can then compute the gradient of the loss as

$\nabla_\theta L(z, \hat{\theta})$. The inner product of the Hessian-vector product and the gradient of the training instance results in a scalar value that tells us the approximate change in loss to expect on $z_{\text{test}}$ if we were to remove the training instance $z$. Note that we compute the gradient of the loss function with respect to **only** the parameters of the last layer [24] [92].

### 6.2 Non-Convexity and Eigenvalues of the Hessian

Due to the importance of the Hessian in the computation of influence functions, the convexity of the loss function and its effects on the Hessian are important. Recall that influence functions assume the Hessian is positive definite such that it is invertible. Koh *et al.* [24] have shown that even with negative Hessian eigenvalues it is still possible to obtain good influence estimates . It is understood that large overparameterized networks affect the convexity of the loss function [95, 96], which we observe via the eigenvalues of the Hessian. Basu *et al.* [93] have shown that larger eigenvalues are correlated with decreases in the Spearman correlation metric when network depth (i.e., the number of hidden layers in the network) and width (i.e., the number of nodes in each hidden layer) are increased. This contradicts the literature where the long tail of the Hessian Eigen Spectral Density (ESD) has been well studied for large DNNs and it has been shown that the largest eigenvalue does not tend to increase as the width of the network increases [94]. In this paper, we utilize a method developed by Yao *et al.* [101] to compute the eigenvalues of the Hessian in an effort to quantify the effect if any, of non-convexity and non-convergence on Influence functions [92].

## 6.3 Bayesian Deep Neural Networks

The current state of the art for influence functions, suggests that by applying L2 regularization to our networks during training, we can reduce the negative effects that are associated with overparameterization [93]. Variational Bayesian Learning has been shown to result in superior regularization, better model averaging and built-in uncertainty prediction [29]. We select this method specifically for its regularization strength. In this chapter we utilize the VDP++ algorithm presented in Section 5.2 [92].

## 6.4 Experiments

### *6.4.1 Iris Dataset*

To study the effect of random initialization on influence function estimates, we reproduced an experiment from Basu *et al.* [93] using the Iris dataset. The Iris dataset consists of 150 instances with 4 features and 3 classes. The decision to use this dataset as a benchmark is due to its simplicity. To make our models more robust to random initialization, we considered weight decay as well as Stochastic Weight Averaging (SWA) and Bayesian Neural Networks (BNNs) as novel additions to this experiment [31, 102, 103] [92].

This experiment was repeated for two types of DNNs: (1) DNNs with constant width (number of nodes in a hidden layer) and variable depth (number of hidden layers), and (2) DNNs with constant depth and variable width. In the experiments with variable depth, the number of nodes per hidden layer was held constant at 5 as in Basu *et. al.* [93]. In the variable width experiments, the depth of the network was held constant at 1, i.e., one hidden layer only. We used the Adam optimizer with an initial learning rate of 0.001 as in Basu *et. al.* [93]. A learning rate scheduler was used to decrease the learning rate by a

factor of 10 if the loss did not decrease for 100 epochs. For the experiments with weight decay, we used a constant value of 0.005 as in Basu *et. al.* [93]. Each experiment was repeated 50 times [92].

Koh *et al.*[24] showed that fine-tuning a trained DNN from the optimal parameters is approximately equal to retraining the same network with a training instance removed. Therefore, to obtain the true differences in loss when removing a test point, we replicate the training procedure outlined by Basu *et al.* [93]. The models are initially trained for 60k epochs of full-batch gradient descent instead of SGD. The training instances are then sorted by their loss and the 40 training instances with the maximal loss are identified. We then fine-tune only the top layer for 7.5k epochs when individually removing each of the training points with the highest loss. Finally, we compute the influence function estimates for those training instances with respect to the test instance with the highest loss. The Spearman correlation between the true and approximate differences in loss are then computed. The eigenvalues of the Hessian for each network were computed via power iteration using the PyHessian Python package [101] [92].

### *6.4.2 MNIST and CIFAR10*

We drastically increase the model and dataset size to study the performance of influence functions in non-convex settings. Similar to the experiment described in Basu *et al.* [93], we chose to look at a small fully connected network, LeNet, and VGG13. Each model was trained in a similar manner as our previous experiment. The Adam optimizer was used with an initial learning rate of 0.001 and weight decay of 0.001. The learning rate was reduced by a factor of 10 if the loss did not decrease after 2 epochs. The test

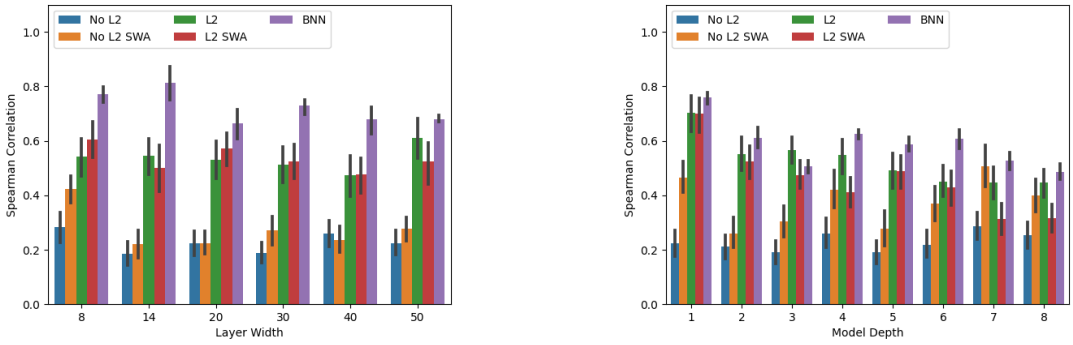
instance with the maximal loss was used to compute the influence functions. Also, the influence functions were computed for all training instances. We deviate from our previous experiment when choosing the training instances to remove and retrain. The true loss difference was computed for the top 40 most influential training points (highest absolute value) using the re-train from optimal approximation. The Spearman correlation between the true and estimated differences in loss was computed [92].

### *6.4.3 Statistical Analysis*

We use one-way analysis of variance (ANOVA) to compare various dependent variables and establish statistical significance in various experiments described above [92].

**Figure 9**

*Influence Function Performance Evaluation on Iris Dataset*




## 6.5 Results and Discussion

### *6.5.1 Effect of Model Size on the Influence Function Estimates*

In Figure 9, we present the Spearman's rank correlation coefficient ($\rho$) between the true and estimated loss differences for the Iris dataset for a variety of model types and

sizes. We present four different types of models, including a model with L2 regularization, a model without L2 regularization, a model with SWA, and a BNN. The figure presents models trained using an increasing number of neurons in one layer (Figure 9-A) and increasing number of layers with fixed number of neurons in each layer (Figure 9-B). The true loss difference is found using the re-training strategy and the estimated loss difference is found using equation 2.12. The error bars represent the 95% confidence intervals obtained by repeating the experiment 50 times. It is evident from both sub-figures that for any type of model (L2, No-L2, SWA, and BNN), there is a minimal effect of increasing the number of neurons or number of layers on the quality of the estimate (of the influence of a training point on the selected test data point) provided by influence functions (using equation 2.12). A statistical analysis performed using ANOVA did not reveal any significant effect of the number of neurons or layers on the Spearman correlation ($p > 0.5$ for all cases). Previously, Basu *et al.* [93] had reported increasing model size (depth and width) degrades influence function estimates. We believe that the discrepancy between the reported results is linked to statistical rigor as no statistical tests or analyses were reported by Basu *et al.* [93] to establish the effect of model size on the quality of estimates produced by influence functions [92].

We also observe that the estimates provided by influence function are more accurate for models with regularization, as shown in Figure 9. In particular, the Bayesian models (BNNs) outperform all other methods. We consider that the observed behavior is linked to (1) the "ensemble" or "average" effect introduced by Bayesian approaches in the model training, and (2) the type of regularization present in the ELBO loss function which has been shown to give these models superior self-compression properties [72]. This performance

increase, however does not seem to carry over to our experiments with larger datasets (Figure 10), where all models were trained with regularization. This is congruent with the results obtained by Basu *et al.* [93] on the same datasets [92].

**Figure 10**

*Spearman Correlation Between True and Approximate Loss Differences*

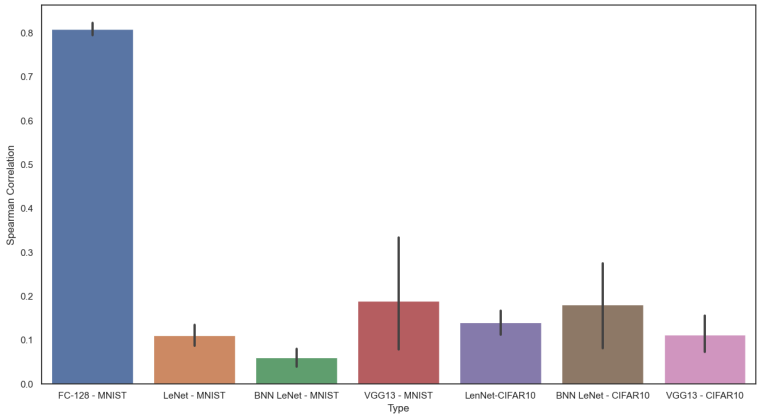
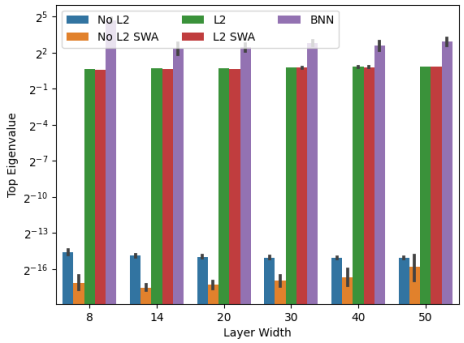
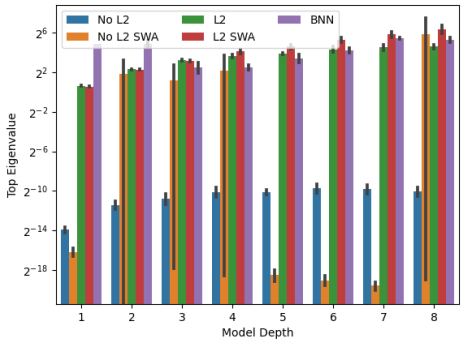



**Figure 11**

*Iris Dataset: Largest Eigenvalue*

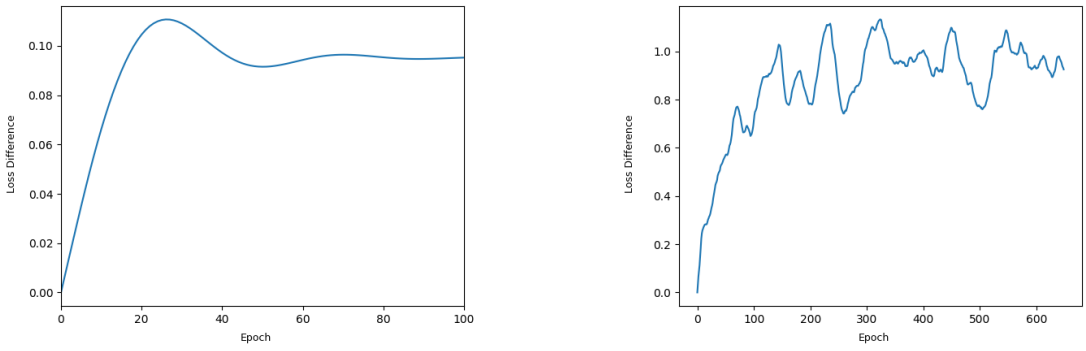


**6.5.1.a The Largest Eigenvalue.** In Figure 11, we observe the same trend that Basu *et. al.* [93] found in the Iris experiment, e.g., the eigenvalues of the Hessian increase with model width and depth (ANOVA $p < 0.05$). We do not however, relate the supposed decrease in influence function estimates to the increasing top eigenvalue as a proxy for curvature of the loss function given that our statistical results from Figure 9 show that there are no significant differences between model sizes and influence function performance. Given that Koh *et al.* [24] have shown that even when most assumptions about convexity of the loss function have been broken, i.e., the optimal parameters have not been obtained ($\tilde{\theta} \neq \hat{\theta}$) and the Hessian has negative eigenvalues (Hessian not PD), we can still obtain "good" influence estimates. We postulate that the problem lies with the methods that have been used to evaluate influence functions [92].

**Figure 12**

*The Loss Trajectories Followed During Re-Training*

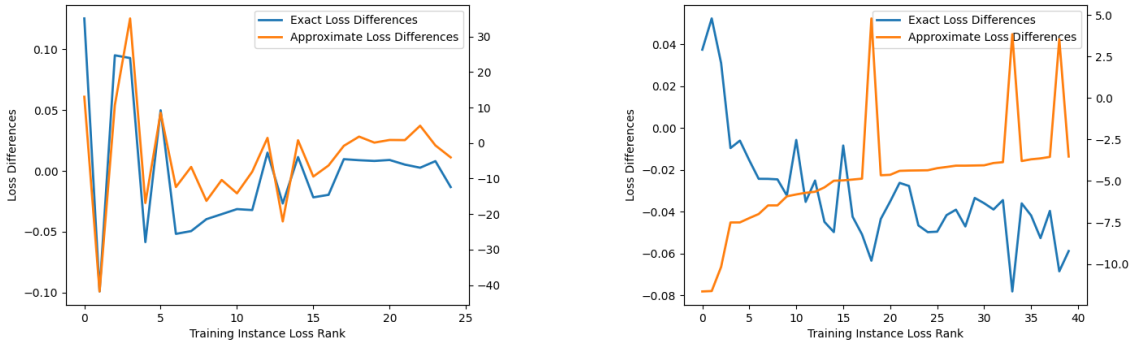


### *6.5.2 The (In-)validity of Spearman's Rank Correlation Coefficient*

Spearman's rank correlation coefficient is an established metric for determining the accuracy of influence function estimates [24, 93, 100]. We note that the output of Equation 2.12 is the difference in the loss function value for the test instance if the training instance is removed. This loss difference can be positive or negative. For a training instance to be influential, it needs to have a large magnitude [92].

**Figure 13**

*Example of Miss-Relation*




In Figure 13, we provide an example where the Spearman's correlation coefficient is unable to capture the underlying relationship between the true loss difference and estimated loss difference, where the estimate is being calculated using Equation 2.12. The horizontal axis in both sub-figures (Figure 13 **left** and **right**) corresponds to the rank of the training point, where the rank is determined by the approximate loss difference. Thus, we should expect to see the exact loss differences (blue points) move from a large magnitude towards zero as we move from left to right on the horizontal axis. In Figure 13 (a), the estimated and true values (after ignoring the scale) are close to each other. In Figure 13 (b), the values of

true and estimated loss differences are significantly different from each other. However, the value of the Spearman's correlation coefficient for both cases is approximately 0.85 [92].

We consider that since the relationship between the estimated and true loss function difference values may not always be **monotonically** decreasing or increasing, the Spearman's correlation can lead to misleading results [92].

### *6.5.3 Re-Training for Optimal Parameters*

To compute the Spearman's correlation coefficient, we need to know the true difference in the loss function value. This requires retraining models for every training instance that we want to analyze. This is a very costly operation in time. The re-training from optimal parameters has been shown to be an approximately equivalent alternative to re-training from scratch [24]. Previous works have not proven that this approximation is valid for large datasets [24, 93]. It has been well established that increasing model and dataset complexity increases the largest eigenvalue of the Hessian of the loss function [95, 96]. While we have demonstrated that the increasing curvature does not affect estimates made by influence functions with small datasets and models, with large datasets and models the extreme curvature of the loss function makes us question the validity of the re-training approximation [104]. To study this, we looked at the loss of the test instance at each epoch during re-training in both the Iris and MNIST experiments. In Figure 12, the test loss difference is plotted against epochs on the horizontal axis. We note significant differences in the trajectories followed by the gradient descent algorithm for two cases (Iris - Figure 12 left and MNIST 12 right). The Iris model has a well damped convergence whereas the the MNIST model is underdamped and does not seem to converge as smoothly as did Iris [92].

### *6.5.4 The Effect of Large Networks*

We consider large neural networks as having more parameters, more non-linear operations owing to their depth, and reluctantly requiring large datasets for training. We note that originally Cook and Weisberg derived influence functions for regression models (which can be considered as neural networks with one layer) and the mean-square error loss function [87]. Recently, Koh *et al.* [24] extended the idea of using influence functions in deep neural networks by treating all but the last layer of the deep neural network as a feature extractor. The influence functions were computed with respect to only the last layer. This practice seems to work in some cases and produce promising results [24]. However, it does not account for the large dimensionality of the final layer of modern neural networks. This proves to be a problem when these large parameter matrices become ill-conditioned [105]. This problem was captured by Basu *et al.* [93] in their analysis of large datasets like CIFAR-100 and ImageNet where true differences in losses from removing training instances resulted in very noisy results [92].

Large neural networks may have more layers (depth) and/or more operations per layer (width). This results in increasing the number of non-linear operations which are performed on the data for calculating the loss function. The most popular implementation of influence functions, as defined by Koh *et al.* [24], relies on only a first-order Taylor series approximation to efficiently compute influence (Eq. 2.8). We argue that the increasing number of non-linear operations strongly affects the convexity assumption of the loss function $R(\theta)$ (Eq. 2.3) as used in the mathematical relationships derived for influence functions (Eq. 2.4). There is evidence suggesting that adding the second term of the Taylor

series in the influence function approximation improves the estimates [99, 100] [92].

Finally, large networks typically go hand-in-hand with large datasets. From Equation 2.11, it is evident that removing a training instance is equivalent to up-weighting it by $\epsilon = -\frac{1}{n}$. In the Iris dataset, $|\epsilon| \approx 6.6e-3$ compared to MNIST where $|\epsilon| \approx 1.6e-5$. Any larger datasets will lead to smaller epsilon, that is:

$$\hat{\boldsymbol{\theta}}_{-z} - \hat{\boldsymbol{\theta}} \approx 0 \text{ or } \hat{\boldsymbol{\theta}}_{-z} \approx \hat{\boldsymbol{\theta}}. \tag{6.1}$$

In other words, owing to the large dataset, the influence of a single training point on a test sample is asymptotically approaching zero. Perhaps the first-order Taylor series approximation of the influence functions does not provide enough resolution to predict loss differences when predicting on only one training instance. If one wanted to use influence functions in large datasets like CIFAR-100 and ImageNet, one would have to turn to higher-order approximations of influence functions as well as group influences. In Basu *et al.* [100], promising results were obtained by examining the effect of a second-order approximation as well as group influence. These solutions of course have an associated cost. Adding a second-order term increases the cost and complexity of the analysis. Optimal group selection is also a non-trivial and expensive operation [92].

While there is theoretical evidence to suggest that the first-order implementation of influence functions is fragile, due to the difficulty of finding robust ways to empirically evaluate them in difficult settings, their supposed fragility remains unclear [92].

## 6.6 Limitations

While we have established that the procedures used to measure the accuracy of influence functions are flawed in multiple ways, we have not been able to ascertain exactly where

or why these procedures break down. It appears that the answer lies with increasing model and dataset size. To precisely define the boundaries on where violating the approximations that Koh *et al.* [24] have established is valid, we would need to exhaustively search the space of increasing complexity of the model and dataset [92].

# Chapter 7

## Deployment of a Robust and Explainable Mortality Prediction Model

In this chapter, we tie together the contributions made in previous chapters and unite the methods under one application. Again, we revisit first-day mortality prediction in the ICU. However, this time we utilize Bayesian Neural Networks combined with influence function based explanations to deliver one application that satisfies the requirements outlined in [12] to achieve widespread acceptance by ICU clinicians. Two components we haven't discussed yet has been dataset shift and algorithmic bias [11]. Dataset shift is a common problem in machine learning where the distribution of the data once deployed may be different or drift over time. Algorithmic bias deals with biases in the training dataset that may affect the performance of the model in the deployed environment. The classic example of algorithmic bias in machine learning is the inclusion of race/ethnicity data in bond price prediction.

### 7.1 Dataset Shift and Algorithmic Bias

Some studies highlight dataset-shift and algorithmic biases as crucial factors to consider when designing AI/ML models for healthcare [106]. Dataset shift, in particular, is often overlooked during the design and development process or is left for manual review and update. It is naive to assume that the independent and identically distribution (i.i.d.) assumption of many AI/ML algorithms will hold in a field like healthcare, where different patient populations can exhibit vastly different distributions [107]. Algorithmic bias is a related concern, as it has been demonstrated that mortality prediction models' performance can vary depending on patient ethnicity [108]. While these requirements can overlap, they

collectively address the full spectrum of failure in previous models [11].

One obvious example of dataset-shift in healthcare came with the advent of COVID-19. A particular issue is mortality prediction in the Intensive Care Unit (ICU). During several periods of the pandemic, ICU mortality rates spiked [109]. Several biomarkers were found to be better predictors of mortality in COVID-19 patients than standard AI/ML models [110]. This same model was replicated and tried in the Netherlands and it was found that the mortality prediction accuracy was much lower [111]. This case study shows that even when a tool is tailor made to serve a specific subset of the population, it may only work well in a specific region. When developing these models, it can become easy to fool oneself about the performance of a model. Several strategies should be employed once the model is deployed. These include periodical testing using data from the deployment, model fine-tuning, and full model retraining [112] [11].

## 7.2 Methods

### *7.2.1 Model Selection*

To train our model, we aggregated mortality data from two publicly available datasets: MIMIC-III [113] and the eICU [39] databases. From these databases, we selected data such as demographic information (age, gender), indicator variables (patient on mechanical ventilation, metastatic cancer, etc.) and lab values (blood lactate, glucose, creatinine, etc.) that are given within 24 hours of patient admittance. This results in nearly 200,000 instances and 200 features with a 92%-8% class imbalance where patient death is the positive class. In order to get these features down to a reasonable number, we adopted an aggressive feature selection procedure. First, pair-wise correlations were computed and all

features with Pearson correlation coefficient greater than 0.9 were dropped from the dataset. Next, any features with greater than 50% missingness were removed from the dataset. We then computed the mutual information between the remaining features and the mortality outcome and selected the top 20. Out of the top 20, a group of clinicians chose 12 based on clinical opinion. We opted to not take a purely data science feature selection technique since some of the top features chosen did not give clinicians any actionable insights. Additionally by taking user opinion into account during the model selection stage, we hoped to foster stronger understanding between model and user prior to deployment. Our final set of features is as follows: Blood lactate, mechanical ventilation (yes/no), all Glasgow Coma Scale parameters (eyes, motor, verbal), albumin, age, creatinine, Prothrombin Time (PT-INR), White Blood cell Count (WBC), Blood Urea Nitrogen (BUN), and Mean Arterial Pressure (MAP) [11].

Model training was performed in a similar manner to [33] where first-day mortality prediction was investigated. Missing data was assumed to be missing at random and multiple imputation with chained equations [114] used as the imputation method. Prior to training, the data was standardized. The learning algorithm chosen here is a shallow neural network. To optimize parameters and capture performance metrics, a 10-fold cross validation scheme was used. We used Weight and biases [42] to optimize the hyper parameters of our neural network. We chose to optimize for positive likelihood ratio (LR+) as this will give our model a good balance between sensitivity and specificity. Additionally, model performance given in likelihood ratios is readily understood by clinicians. Given the large class imbalance, various options to rectify the imbalance were given to the optimizer: Synthetic Minority Oversampling (SMOTE) [41, 115], majority undersampling [115], and

loss function weighting (penalizing minority class predictions). After the hyperparameters are selected, the models were trained using all available data [11].

### *7.2.2 Variational Density Propagation (VDP)*

In addition to our traditional or deterministic neural net, we also trained a Bayesian Neural Network (BNN) in the same manner. We will refer to this as the stochastic neural net. For this study, we used the original formulation of Variational Density Propagation (VDP) as VDP++ (Chapter 5 Section 5.2) was not researched at the time this project began [31, 63, 116] [11]. Specifically, the mortality app project began in May of 2020. The app was deployed and we began collecting results in January of 2021. We began research on VDP++ in March of 2021. It is important to note that it is justifiable to use VDP for this study as there is no significant difference in performance between VDP and VDP++. The main difference is that VDP++ uses much less memory.

To describe the original VDP method by Dera *et al.* [31], let us use a 2-layer fully connected neural network as an example. In a traditional or deterministic neural network, Equation 7.1 describes the forward pass of the model [11].

$$\begin{aligned} z &= Wx + b^{[1]}, \\ a &= f(z), \\ \hat{y} &= g(Va + b^{[2]}), \end{aligned} \tag{7.1}$$

where $W \in \mathbb{R}^{j \times k}$ is the weight matrix of layer 1, $x \in \mathbb{R}^{k \times 1}$ is the input vector, $b^{[1]} \in \mathbb{R}^{j \times 1}$ is the bias vector in layer-1, $z \in \mathbb{R}^{j \times 1}$ is the result of the linear operation in layer-1, $f$ is an arbitrary element-wise non-linear function, $g$ is an arbitrary non-linear activation function that does not operate element-wise, $a \in \mathbb{R}^{j \times 1}$ is the result after applying the non-linear

activation function in layer 1, $V \in \mathbb{R}^{l \times j}$ is the weight matrix in layer 2, $b^{[2]} \in \mathbb{R}^{j \times 1}$ is the bias vector in layer 2, $\hat{y} \in \mathbb{R}^{l \times 1}$ is the predicted output and $k$ is the dimensionality of the input vector, $j$ is the number of nodes in layer 1 and $l$ is the number of classes to predict [11].

To propagate the first two moments, several assumptions must be made. First, let us consider $w_m^\top = m^{\text{th}}$ row of $W$, $m = 1, 2, ..., j$. Then, $z_m = w_m^\top x + b_m^{[1]}$, $m = 1, 2, ..., j$. Next, consider the following assumptions: the input vector $x$ is deterministic, $a \sim \mathcal{N}(\mu_a, \Sigma_a)$, $w_m \sim \mathcal{N}(\mu_{w_m}, \Sigma_{w_m})$, $m = 1, 2, ..., j$, $b_m^{[1]} \sim \mathcal{N}(\mu_{b_m^{[1]}}, \sigma^2_{b_m^{[1]}})$, $m = 1, 2, ..., j$, $v_n \sim \mathcal{N}(\mu_{v_n}, \Sigma_{v_n})$, $n = 1, 2, ..., l$, $b_n^{[2]} \sim \mathcal{N}(\mu_{b_n}^{[2]}, \sigma^2_{b_n^{[2]}})$, $n = 1, 2, ..., l$ and the weight vectors $w_m$, $a$, and bias $b^{[1]}$ and $b^{[2]}$ are uncorrelated to each other for $m = 1, 2, ..., j$. The elements of $\mu_z$ and $\sigma_z^2$ are defined in Equations 7.2 and 7.3 [11].

$$\begin{aligned} \mu_{z_m} &= \mathbb{E}[w_m^\top x + b_m^{[1]}] \\ &= \mathbb{E}[w_m^\top]x + \mathbb{E}[b_m^{[1]}] \\ &= \mu_{w_m}^\top x + \mu_{b_m^{[1]}} \end{aligned} \tag{7.2}$$

$$\begin{aligned} \sigma^2_{z_m} &= \text{Var}[w_m^\top x + b_m^{[1]}] \\ &= \text{Var}[w_m^\top x_m] + \text{Var}[b_m^{[1]}] \\ &= x_m^\top \text{Var}[w_m] x_m + \text{Var}[b_m^{[1]}] \\ &= x_m^\top \Sigma^2_{w_m} x_m + \sigma^2_{b_m^{[1]}} \end{aligned} \tag{7.3}$$

Since we have assumed the weight vectors and elements of the bias $b$ to be uncorrelated, $\Sigma_{w_p w_q} = 0$ and $\sigma_{b_p b_q} = 0$ for $p \neq q$, where $p, q = 1, 2, ..., j$. Hence, the covariance is zero.

To propagate the first two moments through an arbitrary element-wise non-linear function, we utilize the first-order Taylor series approximation shown in Equations 7.4 and 7.5 [11].

$$\begin{aligned} a &= f(z) \\ &= f(\mu_z) + f'(\mu_z)(z - \mu_z) + \ldots \\ &\approx f(\mu_z) + f'(\mu_z)(z - \mu_z) \end{aligned} \tag{7.4}$$

$$\mathbb{E}[a] = \mu_a \approx f(\mu_z)$$

$$\mu_a = f(\mu_z)$$

$$\Sigma_{a_p a_q} \approx \begin{cases} \sigma_{z_p}^2 f'(\mu_z)^2, & p = q, \\ \sigma_{z_p z_q} f'(\mu_{z_p}) f'(\mu_{z_q}), & p \neq q. \end{cases} \tag{7.5}$$

For the second layer of the network, we can no longer consider the incoming vector, $\mu_a$, to be deterministic. Additionally we now need to propagate the incoming variance, $\sigma_a^2$. The mean and covariance propagated through the second fully-connected layer are given in Equations 7.6 and 7.7 [11].

$$\begin{aligned} \mu_{\tilde{y}} &= \mathbb{E}[v_n^\top a + b_n^{[2]}] \\ &= \mathbb{E}[v_n^\top]\mathbb{E}[a] + \mathbb{E}[b_n^{[2]}] \\ &= \mu_{v_n}^\top \mu_a + \mu_{b_n^{[2]}} \end{aligned} \tag{7.6}$$

$$\Sigma_{\tilde{y}_p \tilde{y}_q} = \begin{cases} \mathrm{Tr}(\Sigma_v \Sigma_a) + \mu_v^\top \Sigma_a \mu_v \\ \quad + \mu_a^\top \Sigma_v \mu_a + \Sigma_{b_n^{[2]}} & , \quad p = q, \\ \mu_{v_p}^\top \Sigma_a \mu_{v_q}, & p \neq q. \end{cases} \tag{7.7}$$

Next, we must use a non-linear activation function such as softmax to obtain the predictions of our model. Since $g$ does not operate element-wise on our mean and variance, we use a slightly different Taylor-series to obtain Equations 7.8 and 7.9 [81] [11].

$$\mu_{\hat{y}} \approx g(\mu_{\tilde{y}}) \tag{7.8}$$

$$\Sigma_{\hat{y}} \approx J_g \Sigma_{\tilde{y}} J_g^\top, \tag{7.9}$$

where $J_g$ is the Jacobian matrix of the softmax function $g$ with respect to $\tilde{y}$ and calculated at $\mu_{\tilde{y}}$ [11].

Finally, we use the Evidence Lower Bound (ELBO) function, $\mathcal{L}(\phi, D)$, which consists of two parts: the expected log-likelihood of the training data given the weights, and a regularization term, $D = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$ given by Equation 7.10 [11].

$$\mathcal{L}(\phi, D) = \mathbb{E}_{q(\phi)}[\log p(D|\phi)] - \mathrm{KL}[q(\phi)|p(\phi)], \tag{7.10}$$

where $\phi$ represents the (1) weights $W$ and $V$ and (2) biases $b^{[1]}$, $b^{[2]}$. The expected log-likelihood is given in Equation 7.11 and the KL term is given in Equation 7.12 [11].

$$\begin{aligned} E_{q(\phi)}[\log p(D|\phi)] &\approx \frac{1}{M}\sum_{m=1}^{M} \log p(D|\phi) \\ &\approx -\frac{Nl}{2}\log(2\pi) - \frac{1}{M}\sum_{m=1}^{M}\left[\frac{N}{2}\log(|\Sigma_{\hat{y}}|)\right. \\ &\quad \left. + \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \mu_{\hat{y}}^{(m)})^\top (\Sigma_{\hat{y}}^{(m)})^{-1}(y^{(i)} - \mu_{\hat{y}}^{(m)})\right] \end{aligned} \tag{7.11}$$

where, $y^{(i)}$ is the true label of the $i^{\text{th}}$ data point, $N$ is the number of data points and $M$ is

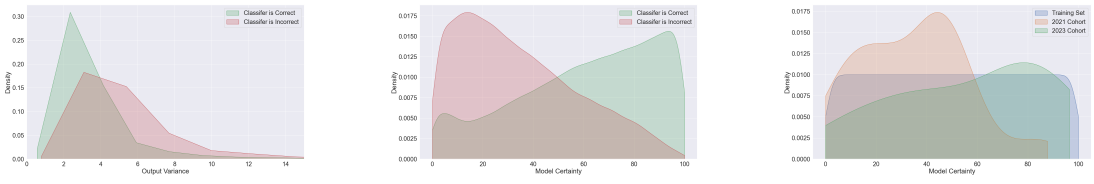
the number of Monte Carlo samples needed to approximate the expectation by summation [11].

$$\mathrm{KL}[q(\phi)|p(\phi)] = -\frac{1}{2}\sum_{n=1}^{l}(j\log\sigma_{v_n}^2 - ||\mu_{v_n}||_F^2 - j\sigma_{v_n}^2)) \tag{7.12}$$

To compute the uncertainty, we have to modify the output of the network. The output of the model contains the mean and covariance. The variance (diagonal elements of the covariance) contain the information about the uncertainty of the prediction. Since this value can range from $[0, \infty)$, it can be difficult to quantify the uncertainty of any one given prediction. What we have done is to plot the distribution of variance outputs when the prediction is wrong and when it is right. Figure 14a shows these distributions. As expected, the variance output is, on average, lower (more confident) when the classifier is right versus when it is wrong. Now that we know this, we can use the joint distribution to determine how confident any given prediction is by computing the CDF along the distribution. Computing $1 - \mathrm{CDF}(\sigma^2)$ will put the confidence in the range $[0, 1]$, where 0 indicates low confidence and 1 indicates high confidence (Figure 14b) [11].

**Figure 14**

*KDE of (a) Variance and (b) Uncertainty, and (c) Scores by Cohort*




**(a)** **(b)** **(c)**

### *7.2.3 Explanations and Interpretability*

To satisfy the interpretability constraints requested by clinicians [12], we provide instance-level explanations via influence functions for each prediction (Equation 3.1). Explanations generated by influence functions are ideal for this application as the loss difference provides a magnitude and direction indicating feature importance and sentiment, indicating which class the feature value favors (in the binary case) [11].

In addition to the explanations, we have taken steps to make the model more interpretable using the app UI. As suggested by clinicians during development, normative ranges for each signal are provided via drop down on the input screen as shown in Figure 15a. We have also computed some first order statistics for each feature in our dataset in an effort to foster understanding between the model and users. These statistics are provided via drop down on the output screen as shown in Figure 15d [11].

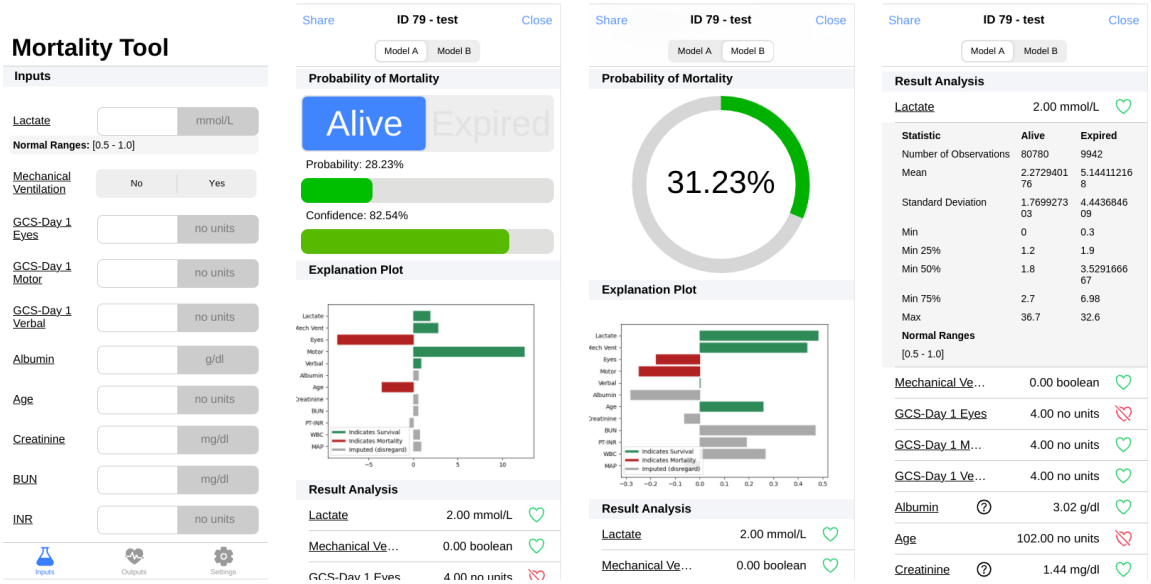
### *7.2.4 App Design for Android/Apple*

The design of the app can be split into three components: front-end client, API server, and ML background worker. The parts are deployed in a monolithic architecture on a Google Compute Engine virtual machine (VM) with a graphical processing unit (GPU). Ionic Framework and React library using TypeScript language was used to develop the front-end UI as a Progressive Web Application. The application can be accessed by a publicly accessible domain on a web browser and optionally installed on the device. With communication to the API server, the UI allows users to create accounts, input data using the interface, and view or update previous records with retrospective analysis [11]. The input

pane is displayed in Figure 15a, The output pane for the Stochastic model and Deterministic models with their explanations are displayed in Figures 15b and 15c respectively.

**Figure 15**

*App UI*




**(a)** **(b)** **(c)** **(d)**

## 7.3 Results

### *7.3.1 Model Selection*

Upon the conclusion of model selection, the optimal parameters for each model were as follows. For the deterministic network, the model had three hidden layers with widths 197, 198, and 112 nodes, respectively. The loss function was weighted to account for the large class imbalance with a positive weight of 14.80. The model was trained for 127 epochs of full-batch gradient descent using stochastic gradient descent (SGD) with

parameters: learning rate $\gamma = 0.03104$, weight-decay $\lambda = 0.0104$, and Nesterov momentum $\mu = 0.4204$ [11].

**Table 8**

*10-fold Cross Validated Training Results*

| **Deterministic Network** | **Precision↑** | **Sensitivity↑** | **Specificity↑** | **ROC AUC↑** | **PRC AUC↑** | **Balanced Accuracy↑** |
|---|---|---|---|---|---|---|
| eICU Only | 0.21±(0.01) | 0.79±(0.02) | 0.82±(0.02) | 0.89±(0.00) | 0.43±(0.01) | 0.81±(0.00) |
| eICU + MIMIC-III | 0.19±(0.00) | 0.80±(0.00) | 0.77±(0.00) | 0.87±(0.00) | 0.39±(0.00) | 0.79±(0.00) |
| **Stochastic Network** | **Precision↑** | **Sensitivity↑** | **Specificity↑** | **ROC AUC↑** | **PRC AUC↑** | **Balanced Accuracy↑** |
| eICU Only | 0.20±(0.01) | 0.79±(0.02) | 0.81±(0.02) | 0.88±(0.00) | 0.37±(0.02) | 0.80±(0.00) |
| eICU + MIMIC-III | 0.19±(0.00) | 0.80±(0.01) | 0.79±(0.00) | 0.87±(0.00) | 0.35±(0.02) | 0.79±(0.00) |

The stochastic model also had three hidden layers with widths 31, 93, and 94 respectively. The class imbalance was handled by majority undersampling. The model was trained for 18 epochs with a batch-size of 1000 using mini-batch SGD with parameters: learning rate $\gamma = 0.0022$, weight-decay $\lambda = 0.0064$, and Nesterov momentum $\mu = 0.7589$. A t-test was performed on the test statistics of the two models and no significant difference was found ($p > 0.01$) [11].

Table 8 shows the results of a 10-fold cross validation using the optimal parameters found for each model. The metric typically used to compare mortality prediction models is the area under the Receiver Operating Characteristic Area Under the Curve (ROC AUC). Both of our models achieve a significantly higher ROC AUC than the currently available and widely used by ICU Clinicians (APACHE [34], SAPS [117]), as well as a recently published Neural Network model [33] ($p < 0.01$ for all pair-wise t-tests). An initially

troublesome result was the low precision obtained by both models. Due to the loss function weighting/majority undersampling, both models predict the positive class (right or wrong) about 40% of the time. This results in the model correctly classifying the positive class about 76% of the time. Using Bayes rule, we can estimate a theoretical positive predictive value of 0.15, given the positive class only occurs 8% of the time in the dataset. Therefore, our models are performing as expected given this imbalance [11].

### *7.3.2 Deployment and Data Collection*

Model deployment commenced in January 2021, during which data from two distinct cohorts were gathered. The first cohort consisted of 59 subjects, with data collected between January 2021 and November 2021 (COVID-19 surge). The second cohort included 25 subjects, and their data were obtained from January 2023 to May 2023 (Post COVID-19). In both cohorts combined, a total of 43 subjects had their mortality outcomes documented (21 from 2021 and 22 from 2023). This was accompanied by predictions from medical professionals prior to viewing model predictions. We chose to collect data in this way in order to study the effects that dataset drift may have on our models due to the COVID-19 pandemic [11].

**Table 9**

*Performance Metrics for Each Model by Cohort*

| | Clinician Prediction | | Deterministic Network | | Stochastic Network | |
|---|---|---|---|---|---|---|
| | **ROC AUC** | **Accuracy** | **ROC AUC** | **Accuracy** | **ROC AUC** | **Accuracy** |
| **Traning Set** | - | - | 0.87±(0.00) | 0.77±(0.00) | 0.87±(0.00) | 0.77±(0.01) |
| **2021 Cohort** | - | **0.95** | **0.99** | 0.90 | **0.99** | **0.95** |
| **2023 Cohort** | - | **0.83** | **0.89** | 0.75 | 0.88 | **0.83** |
| **Combined** | - | **0.90** | **0.93** | 0.84 | 0.90 | **0.90** |

### *7.3.3 Dataset Drift and Model Performance*

A considerable drift in the label distribution between cohorts was observed. To recall, the label distribution of our training set consisted of 92% for negative classes and 8% for positive classes. In contrast, the 2021 and 2023 cohorts exhibited label distributions of 20%-80% and 64%-36%, respectively. A one-way Chi-square test was employed to compare the relative frequencies of mortality outcomes in each cohort against the training distribution. The results revealed that for the 2021 cohort, $p < 0.01$, and for the 2023 cohort, $p > 0.01$ [11].

To investigate the potential impact of dataset drift on performance, we utilized a 2-sample Kolmogorov-Smirnov test with bonferroni correction to compare distributions. For all continuous features in our feature set, we assessed the distribution split by cohort relative to the training set distribution. Blood lactate and blood albumin were identified as significant ($p < 0.01$ for the 2021 cohort and $p > 0.01$ for the 2023 cohort), implying that the distribution of these variables significantly deviated from the training distribution for

the 2021 cohort but not for the 2023 cohort. Figure 16 displays the kernel density estimates of these two features, separated by cohort [11].

**Figure 16**

*Kernel Density Estimates of Significant Features*


**(a)** **(b)**

Turning our attention to model performance, we devoted substantial effort to optimizing the ROC AUC and LR+ during model selection, as these are conventional benchmarks for this problem due to the prevalent class imbalance. Since we also gather clinician predictions during data input, we can directly compare our models to clinicians using accuracy as a metric. Table 9 shows these results. We found that the stochastic model had achieved clinician level performance while the deterministic model fell behind across all cohorts even though the deterministic model had higher ROC AUC across cohorts [11].

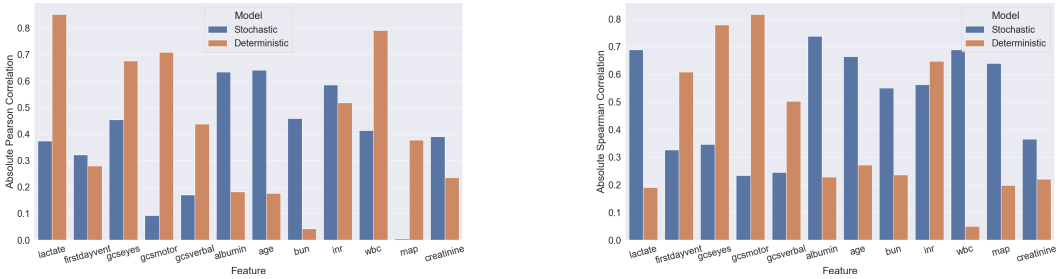
### *7.3.4 Explainability*

To assess the explanations generated by both models, we performed a correlation analysis similar to that which was done in [24]. We computed the correlation between pair-wise feature differences and pair-wise influence differences from our data collected from

deployment by randomly sampling subjects and subject pairs without duplicate (N=500). Figure 17 shows the absolute Pearson (Figure 17a) and absolute Spearman (Figure 17b) correlations. All correlations above 0.05 were significant ($p < 0.01$) with bonferonni correction.

**Figure 17**

*Influence Function Performance*




**(a)** **(b)**

To better understand the kinds of explanations generated by each model, we examined the top three features by absolute magnitude for each subject and compared their frequency across both cohorts. The stochastic model displayed greater variability in its explanations, with nine features appearing in the top three positions for both cohorts. In contrast, the deterministic model only had six features in the top three positions. Higher variability is preferable in this setting as it is more desirable to have an explanation that is sensitive to the particular instance, generating a local explanation, rather than an explanation that is not sensitive, generating what is essentially global feature importance. Additionally, the stochastic model included lactate and albumin in it's top feature sets for both cohorts indicating that the model is picking up on the distribution shift shown in Figure 16 [11].

All models indicated importance in both continuous (e.g.., lactate, albumin, etc.) and categorical/ordinal (e.g., mechanical ventilation, GSC sub-scores) features demonstrating effective explanations for features of all types.

Recall, that in addition to importance scores, the influence function output is signed which can be used to evaluate class sentiment. To evaluate the overall sentiment associated with each feature, we applied the signum function to them. A highly positive/negative score would suggest that the model consistently attributes importance to one class, while a score closer to zero would indicate that the model is equally likely attribute importance to either class. Our findings revealed that the stochastic model had more values near zero compared to the deterministic model. The deterministic model exhibited mostly very positive or very negative values, indicating limited inter-subject variability. The superior performance in explanation quality by the stochastic model is consistent with literature on the explanations generated by BNNs [79] [11].

### *7.3.5 Un(Certainty)*

To assess the effectiveness of our uncertainty metric, we conducted another 2-sample Kolmogorov-Smirnov test, comparing the "Confidence" metric reported by our stochastic model. This test revealed significant ($p < 0.01$) differences between the training set and the 2021 cohort, while no significant ($p > 0.01$) differences were observed between the training set and the 2023 cohort. Figure 14 presents the kernel density estimates of the training distribution, first broken down by classifier correctness (Figure 14b), and then by cohort (Figure 14c). These visualizations demonstrate the utility of our uncertainty metric identifying differences between the cohorts and the training set, providing valuable

insights into the performance of our stochastic model across different conditions [11]. The uncertainty metric may also be used as an outlier detection method. By utilizing the 2-sample KS test on a set of incoming data a significant result would indicate that the incoming data may be considered outliers compared to the training distribution.

## 7.4 Discussion

### *7.4.1 Robust Models and the Effect of COVID-19*

Given the substantial label shift in the data, we should have expected a sharp decline in performance. We attribute the robust performance of our models, in part, to the class-weighting/resampling approach we took during model selection. By resampling or weighting the loss function (instead of changing the decision threshold), we blind our model to the inherent imbalance of the training dataset. Since, in both cohorts, the label shift actually makes the classes more balanced than the training set, we see an increase in performance [11].

Evaluating the explanations highlights the advantages of using stochastic models (such as BNNs) in generating more diverse and nuanced explanations for individual subjects. This increased variability can better capture the unique characteristics of different patients, potentially leading to more accurate and personalized predictions in healthcare settings. As a result, future research should continue to explore and develop AI/ML models that can offer more detailed and individualized insights, ultimately enhancing their utility and applicability in real-world clinical scenarios [11].

Figure 14 offers strong evidence supporting the robustness of BNNs. Our statistical results indicate that the "Confidence" metric may be used to detect dataset shift in small

populations. The ability to effectively quantify uncertainty is crucial for AI/ML models in healthcare, as it allows clinicians to better understand the reliability of a given prediction and make more informed decisions based on the model's outputs. As such, future research should continue to explore and refine methods for estimating uncertainty in AI/ML models, ensuring that these tools can provide healthcare professionals with accurate, reliable, and interpretable information to support their decision-making processes [11].

These findings underscore the importance of considering the deployment environment when developing AI/ML models. By utilizing stochastic models like BNNs that enhance the robustness of AI models, researchers can develop AI solutions that maintain their performance and reliability even in the face of changing conditions. This ultimately improves their utility and applicability in real-world clinical settings [11].

### *7.4.2 Metric Chasing*

The prevailing trend in the majority of AI/ML research for healthcare involves developing models to predict various clinical endpoints. Numerous studies showcase incremental advancements in key performance metrics for these models. However, only a few extend beyond these improvements to address other essential aspects of AI in healthcare. We firmly believe that future efforts should prioritize implementation science over merely enhancing the performance of predictive models [11].

Implementation science plays a crucial role in ensuring the successful deployment, adoption, and sustainability of AI systems in real-world healthcare settings. By focusing on this aspect, researchers can better understand the factors that facilitate the integration of evidence-based interventions and strategies into routine clinical practice. This, in turn, can

lead to the development of AI solutions that not only excel in their predictive capabilities but also effectively address the unique challenges and complexities of healthcare environments [11].

Moreover, emphasizing implementation science can promote interdisciplinary collaboration, user-centered design, and stakeholder engagement, all of which contribute to the practical utility and long-term success of AI solutions in healthcare. By adopting a more comprehensive approach that encompasses these aspects, researchers can help bridge the gap between theoretical advancements and tangible improvements in clinical practice and patient outcomes [11].

### *7.4.3 Limitations*

The most significant limitation of our study was the lack of integration into clinicians' workflow. At the outset, we underestimated the time and effort required for ICU clinicians to interrupt their tasks and enter 12 values on their phones. This challenge was further intensified by the COVID-19 pandemic, as clinicians faced longer working hours and managed more critical patients. If Electronic Health Record (EHR) integration had been possible, we estimate that we could have collected a considerably larger volume of data, potentially an order of magnitude greater than our current dataset [11].

While obtaining more data could have strengthened our results, we believe that it would not have fundamentally altered our findings. Given that our model was trained on 200,000 instances, we anticipate that our models demonstrate satisfactory calibration. Furthermore, the statistical testing we conducted produced positive results that corroborated our hypotheses [11].

In light of these limitations, future research should prioritize seamless integration into clinical workflows to facilitate data collection and ensure that AI models are practical and beneficial for healthcare professionals. Additionally, researchers should continue to explore methods for improving model robustness and generalizability, ensuring that AI solutions remain effective across diverse patient populations and varying medical conditions [11].

# Chapter 8

# Conclusion

In this dissertation, we examined the performance, explainability, and robustness of AI models for predicting clinical endpoints. To our knowledge, this is the first study of its kind to deploy interpretable and explainable models with uncertainty prediction in a real-world setting. Our findings demonstrate the effectiveness of BNNs and the importance of adoption by taking into account robustness during model selection in order to maintain model performance in the face of significant dataset shifts. By addressing these challenges, we were able to develop AI/ML models that matched or outperformed clinician predictions even when confronted with substantial dataset drift. Our investigation into model explainability revealed the advantages of stochastic models in generating more diverse and nuanced explanations as well as offering more personalized insights for individual patients. These findings emphasize the need for future research to focus on AI models that can deliver detailed and individualized information to enhance their utility and applicability in real-world clinical settings. We also highlighted the importance of quantifying uncertainty in AI models for healthcare, as it allows clinicians to better understand the reliability of predictions and make more informed decisions [11].

## 8.1 Future Work

Future research should continue to refine these methods to ensure these tools provide accurate, reliable, and interpretable information for healthcare professionals. Our study reinforces the need to prioritize implementation science in AI research for healthcare, ensuring that AI solutions are practical, beneficial, and sustainable in real-world clinical

environments. By addressing the unique challenges and complexities of healthcare settings, researchers can develop AI models that not only excel in predictive capabilities but also effectively improve clinical practice and patient outcomes [11].

## 8.2 Contributions

The contributions of this work include:

- Extension of the influence function derivation for neural networks provided by Koh *et. al*. [24].
- Derivation of feature importance and instance-level explanations via influence functions (Equations 3.1 and 3.2)
- Development of mortality prediction model using a minimum feature set that maximizes ROC AUC [46]
- Developed a simplified form of Variational Density Propagation (VDP++) that reduces the memory requirements of the original method but retains all of its desirable properties [60]
- Refuted the claims of influence function fragility by pointing out flaws in the preceding work's [93] methodology and replicating their results [92]
- Combined Bayesian neural networks with explanations from influence functions to create an optimal mortality prediction model in both performance and interpretability [11]

- Deployed the optimal mortality prediction model and collected results in a real-world setting [11]

## References


[1] R. Binaco, N. Calzaretto, J. Epifano, S. McGuire, M. Umer, S. Emrani, V. Wasserman, D. J. Libon, and R. Polikar, "Machine learning analysis of digital clock drawing test performance for differential classification of mild cognitive impairment subtypes versus alzheimer's disease," *Journal of the International Neuropsychological Society*, vol. 26, no. 7, pp. 690–700, 2020.

[2] A. J. Masino, M. C. Harris, D. Forsyth, S. Ostapenko, L. Srinivasan, C. P. Bonafide, F. Balamuth, M. Schmatz, and R. W. Grundmeier, "Machine learning models for early sepsis recognition in the neonatal intensive care unit using readily available electronic health record data," *PloS one*, vol. 14, no. 2, p. e0212665, 2019.

[3] M. Dewan, N. Muthu, E. Shelov, C. P. Bonafide, P. Brady, D. Davis, E. S. Kirkendall, D. Niles, R. M. Sutton, D. Traynor *et al.*, "Performance of a clinical decision support tool to identify picu patients at high-risk for clinical deterioration," *Pediatric critical care medicine: a journal of the Society of Critical Care Medicine and the World Federation of Pediatric Intensive and Critical Care Societies*, vol. 21, no. 2, p. 129, 2020.

[4] J. C. Ginestra, H. M. Giannini, W. D. Schweickert, L. Meadows, M. J. Lynch, K. Pavan, C. J. Chivers, M. Draugelis, P. J. Donnelly, B. D. Fuchs *et al.*, "Clinician perception of a machine learning-based early warning system designed to predict severe sepsis and septic shock," *Critical care medicine*, vol. 47, no. 11, p. 1477, 2019.

[5] A. D. Bedoya, M. E. Clement, M. Phelan, R. C. Steorts, C. O'Brien, and B. A. Goldstein, "Minimal impact of implemented early warning score and best practice alert for patient deterioration," *Critical care medicine*, vol. 47, no. 1, p. 49, 2019.

[6] J. L. Guidi, K. Clark, M. T. Upton, H. Faust, C. A. Umscheid, M. B. Lane-Fall, M. E. Mikkelsen, W. D. Schweickert, C. A. Vanzandbergen, J. Betesh *et al.*, "Clinician perception of the effectiveness of an automated early warning and response system for sepsis in an academic medical center," *Annals of the American Thoracic Society*, vol. 12, no. 10, pp. 1514–1519, 2015.

[7] B. Meskó and M. Görög, "A short guide for medical professionals in the era of artificial intelligence," *npj Digital Medicine*, vol. 3, no. 1, pp. 1–8, 2020.

[8] E. Vayena, A. Blasimme, and I. G. Cohen, "Machine learning in medicine: addressing ethical challenges," *PLoS medicine*, vol. 15, no. 11, p. e1002689, 2018.

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

[10] M. C. Elish, "The stakes of uncertainty: developing and integrating machine learning in clinical care," in *Ethnographic Praxis in Industry Conference Proceedings*, vol. 2018. Wiley Online Library, 2018, pp. 364–380.

[11] J. R. Epifano, S. Glass, R. P. Ramachandran, S. Patel, and G. Rasool, "Deployment of a robust and explainable mortality prediction model: The covid-19 pandemic and beyond," in *Proceedings of the Thirty-seventh Conference on Neural Information Processing Systems*, 2023, SUBMITTED UNDER REVIEW.

[12] S. Tonekaboni, S. Joshi, M. D. McCradden, and A. Goldenberg, "What clinicians want: contextualizing explainable machine learning for clinical end use," in *Machine learning for healthcare conference*. PMLR, 2019, pp. 359–380.

[13] S. Ahmed, D. Dera, S. U. Hassan, N. Bouaynaya, and G. Rasool, "Failure detection in deep neural networks for medical imaging," *Frontiers in Medical Technology*, vol. 4, 2022.

[14] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, 2018, pp. 80–89.

[15] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[16] ——, "Anchors: High-precision model-agnostic explanations," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[17] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[18] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," *arXiv preprint arXiv:1706.03825*, 2017.

[19] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[20] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, 2015.

[21] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3145–3153.

[22] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," *34th International Conference on Machine Learning, ICML 2017*, vol. 7, pp. 5109–5118, 2017.

[23] R. D. Cook and S. Weisberg, "Characterizations of an empirical influence function for detecting influential cases in regression," *Technometrics*, vol. 22, no. 4, pp. 495–508, 1980.

[24] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1885–1894.

[25] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 27–38.

[26] E. Begoli, T. Bhattacharya, and D. Kusnezov, "The need for uncertainty quantification in machine-assisted medical decision making," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 20–23, 2019.

[27] M. Dewan, N. Muthu, E. Shelov, C. P. Bonafide, P. Brady, D. Davis, E. S. Kirkendall, D. Niles, R. M. Sutton, D. Traynor *et al.*, "Performance of a clinical decision support tool to identify picu patients at high risk for clinical deterioration," *Pediatric Critical Care Medicine*, 2019.

[28] G. E. Hinton and D. Van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proceedings of the sixth annual conference on Computational learning theory*, 1993, pp. 5–13.

[29] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning*. PMLR, 2015, pp. 1613–1622.

[30] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, no. 3, pp. 457–506, 2021.

[31] D. Dera, G. Rasool, and N. Bouaynaya, "Extended variational inference for propagating uncertainty in convolutional neural networks," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2019, pp. 1–6.

[32] A. Y. Foong, Y. Li, J. M. Hernández-Lobato, and R. E. Turner, "'in-between' uncertainty in bayesian neural networks," *arXiv preprint arXiv:1906.11537*, 2019.

[33] J. R. Epifano, R. P. Ramachandran, S. Patel, and G. Rasool, "Towards an explainable mortality prediction model," in *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2020, pp. 1–6.

[34] J. E. Zimmerman, A. A. Kramer, D. S. McNair, and F. M. Malila, "Acute physiology and chronic health evaluation (apache) iv: hospital mortality assessment for today's critically ill patients," *Critical care medicine*, vol. 34, no. 5, pp. 1297–1310, 2006.

[35] R. P. Moreno, P. G. Metnitz, E. Almeida, B. Jordan, P. Bauer, R. A. Campos, G. Iapichino, D. Edbrooke, M. Capuzzo, J.-R. Le Gall *et al.*, "Saps 3—from evaluation of the patient to evaluation of the intensive care unit. part 2: Development of a prognostic model for hospital mortality at icu admission," *Intensive care medicine*, vol. 31, no. 10, pp. 1345–1355, 2005.

[36] J. Y. Choi, J. H. Jang, Y. S. Lim, J. Y. Jang, G. Lee, H. J. Yang, J. S. Cho, and S. Y. Hyun, "Performance on the apache ii, saps ii, sofa and the ohca score of post-cardiac arrest patients treated with therapeutic hypothermia," *PloS one*, vol. 13, no. 5, 2018.

[37] N. Agarwal, B. Bullins, and E. Hazan, "Second-order stochastic optimization for machine learning in linear time," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 4148–4187, 2017.

[38] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in neural information processing systems*, 2017, pp. 971–980.

[39] T. J. Pollard, A. E. Johnson, J. D. Raffa, L. A. Celi, R. G. Mark, and O. Badawi, "The eicu collaborative research database, a freely available multi-center database for critical care research," *Scientific data*, vol. 5, p. 180178, 2018.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

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

[42] L. Biewald, "Experiment tracking with weights and biases," 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[43] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in neural information processing systems*, 2017, pp. 4765–4774.

[44] J. Li, C. Zhang, J. T. Zhou, H. Fu, S. Xia, and Q. Hu, "Deep-lift: deep label-specific feature learning for image annotation," *IEEE Transactions on Cybernetics*, vol. 52, no. 8, pp. 7732–7741, 2021.

[45] A. G. Randolph, M. B. Zollo, M. J. Egger, G. H. Guyatt, R. M. Nelson, and G. L. Stidham, "Variability in physician opinion on limiting pediatric life support," *Pediatrics*, vol. 103, no. 4, pp. e46–e46, 1999.

[46] J. R. Epifano, A. Silvestri, A. Tripathi, A. Yu, G. Rasool, and R. P. Ramachandran, "A comparison of feature selection techniques for first-day mortality prediction in the icu," in *Proceedings of the 56th International Symposium on Circuits and Systems (ISCAS)*, 2023.

[47] S. Shilaskar and A. Ghatol, "Feature selection for medical diagnosis : Evaluation for cardiovascular diseases," *Expert systems with applications*, vol. 40, no. 10, pp. 4146–4153, 2013.

[48] T. J. Pollard, A. E. W. Johnson, J. D. Raffa, L. A. Celi, R. G. Mark, and O. Badawi, "The eicu collaborative research database, a freely available multi-center database for critical care research," *Scientific Data*, vol. 5, no. 1, 2018.

[49] C. Chu, A.-L. Hsu, K.-H. Chou, P. Bandettini, and C. Lin, "Does feature selection improve classification accuracy? impact of sample size and feature selection on classification using anatomical magnetic resonance images," *NeuroImage (Orlando, Fla.)*, vol. 60, no. 1, pp. 59–70, 2012.

[50] R. G. Miller Jr, *Beyond ANOVA: basics of applied statistics*. CRC press, 1997.

[51] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & electrical engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[52] G. Brown, "A new perspective for information theoretic feature selection," in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, D. van Dyk and M. Welling, Eds., vol. 5. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 49–56. [Online]. Available: http://proceedings.mlr.press/v5/brown09a.html

[53] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1, pp. 389–422, 2002.

[54] I. R. White, P. Royston, and A. M. Wood, "Multiple imputation using chained equations: issues and guidance for practice," *Statistics in medicine*, vol. 30, no. 4, pp. 377–399, 2011.

[55] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *The Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[56] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*. PMLR, 2013, pp. 1139–1147.

[57] OpenAI, "Gpt-4 technical report," 2023.

[58] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models." in *USENIX Security Symposium*, vol. 6, 2021.

[59] J. Rissanen, "Stochastic complexity and modeling," *The annals of statistics*, pp. 1080–1100, 1986.

[60] J. R. Epifano, T. Duong, R. P. Ramachandran, and G. Rasool, "Efficient scaling of bayesian neural networks," in *Proceedings of the Thirty-seventh Conference on Neural Information Processing Systems*, 2023, SUBMITTED UNDER REVIEW.

[61] G. E. Box and G. C. Tiao, *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011.

[62] D. J. MacKay, "The evidence framework applied to classification networks," *Neural computation*, vol. 4, no. 5, pp. 720–736, 1992.

[63] D. Dera, N. C. Bouaynaya, G. Rasool, R. Shterenberg, and H. M. Fathallah-Shaykh, "Premium-cnn: Propagating uncertainty towards robust convolutional neural networks," *IEEE Transactions on Signal Processing*, vol. 69, pp. 4669–4684, 2021.

[64] X. Xie, X. Liu, T. Lee, S. Hu, and L. Wang, "Blhuc: Bayesian learning of hidden unit contributions for deep neural network speaker adaptation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5711–5715.

[65] Y. Xiao and W. Y. Wang, "Quantifying uncertainties in natural language processing tasks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 7322–7329.

[66] A. Bate, M. Lindquist, I. R. Edwards, S. Olsson, R. Orre, A. Lansner, and R. M. De Freitas, "A bayesian neural network method for adverse drug reaction signal generation," *European journal of clinical pharmacology*, vol. 54, no. 4, pp. 315–321, 1998.

[67] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik, "Uncertainty quantification using bayesian neural networks in classification: Application to ischemic stroke lesion segmentation," in *Medical Imaging with Deep Learning*, 2018.

[68] ——, "Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation," *Computational Statistics & Data Analysis*, vol. 142, p. 106816, 2020.

[69] D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[70] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[71] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[72] G. Carannante, D. Dera, G. Rasool, and N. C. Bouaynaya, "Self-compression in bayesian neural networks," in *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2020, pp. 1–6.

[73] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *International conference on machine learning*. PMLR, 2015, pp. 1861–1869.

[74] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[75] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with bernoulli approximate variational inference," *arXiv preprint arXiv:1506.02158*, 2015.

[76] D. Alvarez-Melis and T. S. Jaakkola, "On the robustness of interpretability methods," *arXiv preprint arXiv:1806.08049*, 2018.

[77] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," *Advances in neural information processing systems*, vol. 31, 2018.

[78] A. Ghorbani, A. Abid, and J. Zou, "Interpretation of neural networks is fragile," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 3681–3688.

[79] I. E. Nielsen, R. P. Ramachandran, N. Bouaynaya, H. M. Fathallah-Shaykh, and G. Rasool, "Evalattai: A holistic approach to evaluating attribution maps in robust and non-robust models," *arXiv preprint arXiv:2303.08866*, 2023.

[80] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[81] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[82] 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.

[83] L. Beyer, X. Zhai, and A. Kolesnikov, "Better plain vit baselines for imagenet-1k," *arXiv preprint arXiv:2205.01580*, 2022.

[84] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[85] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[86] C.-K. Yeh, C.-Y. Hsieh, A. Suggala, D. I. Inouye, and P. K. Ravikumar, "On the (in) fidelity and sensitivity of explanations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[87] R. D. Cook and S. Weisberg, *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

[88] G. Cohen, G. Sapiro, and R. Giryes, "Detecting adversarial samples using influence functions and nearest neighbors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 453–14 462.

[89] H. Guo, N. F. Rajani, P. Hase, M. Bansal, and C. Xiong, "Fastif: Scalable influence functions for efficient model interpretation and debugging," *arXiv preprint arXiv:2012.15781*, 2020.

[90] D. Lee, H. Park, T. Pham, and C. D. Yoo, "Learning augmentation network via influence functions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 961–10 970.

[91] X. Han, B. C. Wallace, and Y. Tsvetkov, "Explaining black box predictions and unveiling data artifacts through influence functions," *arXiv preprint arXiv:2005.06676*, 2020.

[92] J. R. Epifano, R. P. Ramachandran, A. J. Masino, and G. Rasool, "Revisiting the fragility of influence functions," *Neural Networks*, vol. 162, pp. 581–588, 2023.

[93] S. Basu, P. Pope, and S. Feizi, "Influence functions in deep learning are fragile," in *International Conference on Learning Representations*, 2020.

[94] L. Sagun, L. Bottou, and Y. LeCun, "Eigenvalues of the hessian in deep learning: Singularity and beyond," *arXiv preprint arXiv:1611.07476*, 2016.

[95] L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou, "Empirical analysis of the hessian of over-parametrized neural networks," *arXiv preprint arXiv:1706.04454*, 2017.

[96] B. Ghorbani, S. Krishnan, and Y. Xiao, "An investigation into neural net optimization via hessian eigenvalue density," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2232–2241.

[97] G. Alain, N. L. Roux, and P.-A. Manzagol, "Negative eigenvalues of the hessian in deep neural networks," *arXiv preprint arXiv:1902.02366*, 2019.

[98] B. A. Pearlmutter, "Fast exact multiplication by the hessian," *Neural computation*, vol. 6, no. 1, pp. 147–160, 1994.

[99] P. W. Koh, K.-S. Ang, H. H. Teo, and P. Liang, "On the accuracy of influence functions for measuring group effects," *arXiv preprint arXiv:1905.13289*, 2019.

[100] S. Basu, X. You, and S. Feizi, "On second-order group influence functions for black-box predictions," in *International Conference on Machine Learning*. PMLR, 2020, pp. 715–724.

[101] Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney, "Pyhessian: Neural networks through the lens of the hessian," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 581–590.

[102] P. Madhyastha and R. Jain, "On model stability as a function of random seed," *arXiv preprint arXiv:1909.10447*, 2019.

[103] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[104] G. Zhang, C. Wang, B. Xu, and R. Grosse, "Three mechanisms of weight decay regularization," *arXiv preprint arXiv:1810.12281*, 2018.

[105] D. A. Belsley, E. Kuh, and R. E. Welsch, *Regression diagnostics: Identifying influential data and sources of collinearity*. John Wiley & Sons, 2005.

[106] C. J. Kelly, A. Karthikesalingam, M. Suleyman, G. Corrado, and D. King, "Key challenges for delivering clinical impact with artificial intelligence," *BMC medicine*, vol. 17, no. 1, pp. 1–9, 2019.

[107] B. Nestor, M. McDermott, G. Chauhan, T. Naumann, M. C. Hughes, A. Goldenberg, and M. Ghassemi, "Rethinking clinical prediction: why machine learning must consider year of care and feature aggregation," *arXiv preprint arXiv:1811.12583*, 2018.

[108] I. Chen, F. D. Johansson, and D. Sontag, "Why is my classifier discriminatory?" *Advances in neural information processing systems*, vol. 31, 2018.

[109] S. C. Auld, K. R. Harrington, M. W. Adelman, C. J. Robichaux, E. C. Overton, M. Caridi-Scheible, C. M. Coopersmith, and D. J. Murphy, "Trends in icu mortality from coronavirus disease 2019: a tale of three surges," *Critical Care Medicine*, vol. 50, no. 2, p. 245, 2022.

[110] L. Yan, H.-T. Zhang, J. Goncalves, Y. Xiao, M. Wang, Y. Guo, C. Sun, X. Tang, L. Jing, M. Zhang *et al.*, "An interpretable mortality prediction model for covid-19 patients," *Nature machine intelligence*, vol. 2, no. 5, pp. 283–288, 2020.

[111] M. J. Quanjel, T. C. van Holten, P. C. Gunst-van der Vliet, J. Wielaard, B. Karakaya, M. Söhne, H. S. Moeniralam, and J. C. Grutters, "Replication of a mortality prediction model in dutch patients with covid-19," *Nature Machine Intelligence*, vol. 3, no. 1, pp. 23–24, 2021.

[112] S. E. Davis, R. A. Greevy Jr, C. Fonnesbeck, T. A. Lasko, C. G. Walsh, and M. E. Matheny, "A nonparametric updating method to correct clinical prediction model drift," *Journal of the American Medical Informatics Association*, vol. 26, no. 12, pp. 1448–1457, 2019.

[113] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[114] S. Van Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in r," *Journal of statistical software*, vol. 45, pp. 1–67, 2011.

[115] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

[116] D. Dera, G. Rasool, N. C. Bouaynaya, A. Eichen, S. Shanko, J. Cammerata, and S. Arnold, "Bayes-SAR net: Robust SAR image classification with uncertainty estimation using bayesian convolutional neural network," in *2020 IEEE International Radar Conference (RADAR)*. IEEE, 2020, pp. 362–367.

[117] J.-R. Le Gall, S. Lemeshow, and F. Saulnier, "A new simplified acute physiology score (saps ii) based on a european/north american multicenter study," *Jama*, vol. 270, no. 24, pp. 2957–2963, 1993.