A Computational System for Detecting the Acute Respiratory Distress Syndrome Using Physiologic Waveform Data from Mechanical Ventilators

By

GREGORY REHM
DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

_____
Chen-Nee Chuah Chair

_____
Jason Y. Adams

_____
Yong-Jae Lee

Committee in Charge

2021

ABSTRACT

Acute respiratory distress syndrome (ARDS) is a severe form of acute hypoxemic respiratory failure affecting 10% of patients admitted to the intensive care unit (ICU) globally. In-hospital mortality of 29.7%-42.9% has been reported across the spectrum of mild-severe ARDS, and one third of patients with initially mild ARDS will progress to moderate or severe ARDS. Over the last 20 years, multiple studies have reported improved outcomes for ARDS patients using ARDS-targeted therapies. However, ARDS remains persistently under-recognized and challenging to diagnose. Only one third of ICU providers correctly identify ARDS on the first day when diagnostic criteria are met, and less than two thirds ever recognize the diagnosis in the ICU. This under recognition of ARDS may prevent some patients from receiving lifesaving therapies necessary for treating the disease. Attempts to automate ARDS diagnosis using rule-based algorithms have seen limited success, and require subjective analysis of infrequently sampled patient data, like chest radiographs, which limit diagnosis automation, timeliness, and study reproducibility.

To improve the current state of the art of ARDS detection technology, we intend to utilize objective and readily available ventilator waveform data (VWD) to improve the recognition of ARDS and develop next generation clinical decision support systems (CDSS) that can function as a pervasive monitoring system for mechanical ventilation management. For this task, we make use of a novel dataset consisting of only VWD from patients receiving mechanical ventilation. We analyze this data using both classical and deep machine learning models and show results that suggest that ARDS can be detected in absence of a chest scan or medical history. This finding highlights that VWD shows promise for use as a future digital biomarker of ARDS pathophysiology.

This dissertation is broadly aimed at showing the research challenges we solved and software that we built to create our ML-based ARDS detection models. Our work encompasses multiple sub-studies including: 1) designing software for annotating large amounts of VWD for our ML datasets, 2) developing and training machine learning models to extract important metadata (such as ventilation mode) from VWD, 3) performing a comprehensive clinical validation for existing, but clinically unvalidated respiratory compliance estimation algorithms, 4) training an ML model for early recognition of ARDS using only VWD, 5) improving our initial ML model for ARDS recognition by applying deep learning, and 6) discussing how to integrate all our work into a comprehensive monitoring system for patient ventilation.

While our work is specifically focused on ARDS detection, this dissertation broadly highlights the utility of applying modern machine learning tools to underutilized high-frequency physiologic waveform data as a surrogate for manually charted EHR data types in disease detection and early health warning. These warnings and alerts can then be continuously monitored and returned to clinicians at the point of care in the form of actionable advice and disease prognostication. Such improvements will enable pervasive clinical decision support systems that promise to reduce the cognitive burden on care providers, improve quality of care, reduce patient suffering, and reduce overall hospital mortality from deadly conditions such as ARDS.

ACKNOWLEDGMENTS

Thank you to the NIH for generously funding this work, and everyone else who helped with support, encouragement, advice, ideas, revisions, and all the other things that are unsung but critical to making a Ph.D. happen. However, most credit goes to my wife Jackie. Your contributions in this endeavor are too numerous to list.

Table of Contents

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Chapter 1: Introduction

1.1 The ICU and ARDS

The intensive care unit (ICU) is one of the most heavily monitored environments in healthcare. In the ICU, patients are continuously monitored by multiple life support machines, and other surveillance devices that are meant to measure important vital signals and alert physicians before or during a major health emergency. However, despite use of sophisticated equipment, many health emergencies can go undetected until a patient experiences a severe health consequence such as a cardiac arrest.(1–4) The reasons for this are complex, but often can be reduced to a distinct lack of advanced analytics, and clinical decision support systems (CDSS) that can ingest the large amounts of data gathered, and then accurately alert physicians to a patient's deteriorating health state.(1,5) The construction of such CDSS that can intelligently monitor patients is acknowledged to be a tremendous opportunity as part of the overall digitalization of healthcare.(6) In theory, CDSS could ensure higher care standards, lower hospitalization costs, automate laborious aspects of healthcare, improve patient mortality, and even perform automated diagnosis and prognostication of complex diseases.(7–10)

One such affliction that may necessitate automated improved diagnostic and screening systems is the acute respiratory distress syndrome (ARDS).(11) ARDS is a respiratory syndrome that occurs because of an acute, diffuse lung injury and is characterized by acute hypoxemia, reduced lung compliance, and increased work of breathing in patients.(12) The causes for ARDS are varied, but most commonly ARDS results from an acute lung injury such as bacterial or viral infection that results in pneumonia or sepsis. Other causes are gastric aspiration, shock, trauma,

and blood transfusion, among others.(13) As a result of injury, air sacs in the lung responsible for transporting oxygen to the bloodstream, called alveoli, become filled with fluid (edema). Edematous alveoli are no longer capable of transporting oxygen to the blood stream. If this process occurs for sufficient alveoli, then a patient will be unable to naturally fulfill the oxygen demands for their body, which will eventually cause hypoxemic respiratory failure. Hypoxemic respiratory failure is a critical medical condition, and when severe, necessitates immediate hospitalization in the ICU.

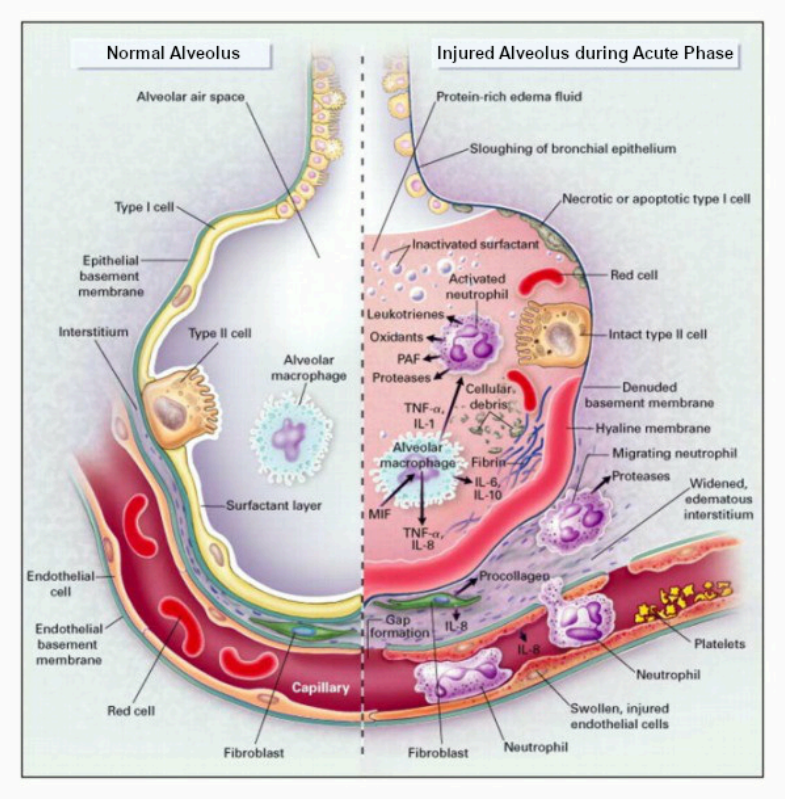


Figure 1.1: ARDS disease process. The image on the left shows a normal alveolus that is functioning correctly and free of fluid (edema). The right shows an alveolus that has been injured due to some physiologic insult, possibly due to pneumonia from bacterial or viral infection or sepsis. As a result of damage, the alveolus becomes edematous. Edema prevents proper function of the alveolus and transport of oxygen to the bloodstream. ARDS manifests when many alveoli have been injured in this manner, and thus severely degrade function of the lungs. The body attempts to fight the damage through activating the immune system, as seen by macrophage, and neutrophil activity, but recovery often takes time. In some cases, cellular damage cannot be completely repaired, and patients experience fibrotic lung damage from ARDS. (Wikimedia

Commons, “File:The-normal-alveolus-Left-Hand-Side-and-the-injured-alveolus-in-the-acute.jpg,” Posted: 2012/08/01, Accessed: 2021/10/10)

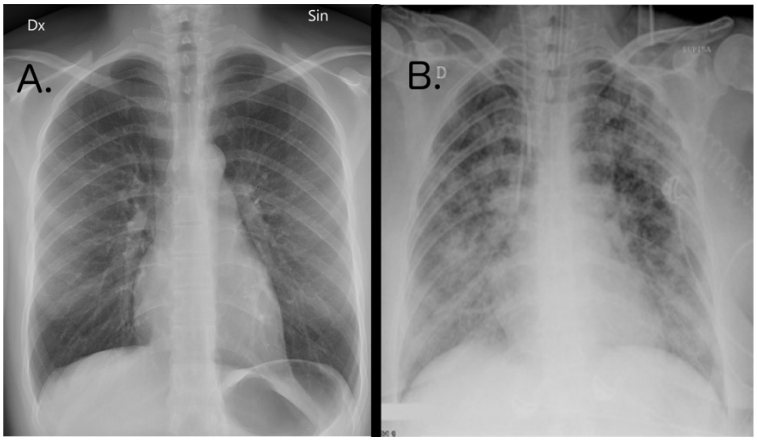


Figure 1.2: Visualization of what occurs to a patient’s lungs in ARDS via chest X-Ray. A) displays a chest x-ray from a normal patient. B) displays a chest X-Ray from a patient with ARDS. Note how areas normally transparent in the x-ray now display diffuse white regions in both right and left lungs. These are termed “bilateral opacities,” and are a primary diagnostic signature of ARDS.

ICU physicians are very familiar with ARDS, and prior to the COVID-19 pandemic, approximately 10% of all ICU admissions were caused by ARDS.(12–15) ARDS leads to the death of over 100,000 Americans each year, with disease mortality rates ranging from 29.7%-42.9% depending upon syndrome severity.(13,16–20) Despite its lethality and prevalence, ARDS was found to be a difficult syndrome to diagnose because of the wide variety of ways it can be represented in patients.(12,13,16) So, to improve diagnostic criteria, scientists accepted a consensus standard for ARDS, the Berlin definition, which defines clinical criteria a patient must meet for ARDS diagnosis, and also sets a severity scale ranging from mild to severe forms of ARDS.(21) The Berlin definition states ARDS can be diagnosed if there is evidence of “bilateral opacities” (Figure 1.2B) on a patient chest X-ray (CXR) or computed tomography (CT) scan. Additionally, patient PaO2/FiO2 (P/F) ratios must be $\leq$ 300 mm Hg, respiratory failure must not be able to be fully explained by potential cardiac failure, and the timing of respiratory failure

must be within 1 week of new or worsening respiratory symptoms. ARDS is then sub-classified by severity using the P/F ratio. Patients with P/F $\leq$ 100 mm Hg are classified as severe ARDS, $100 < P/F \leq 200$ mm Hg is moderate ARDS, and $200 < P/F \leq 300$ mm Hg is determined as mild ARDS.

Despite intensive research into the etiology, diagnosis, and treatment of ARDS, multiple studies have shown that bedside providers continue to under-recognize the syndrome. In the largest multi-national prospective cohort study to date, recognition of ARDS occurred in only 34% of patients on the first day when Berlin diagnostic criteria were present, and ever in only 60% of patients.(13) Under-diagnosis also ranges by levels of severity, and only 51% of patients with mild ARDS and 79% of patients with severe ARDS were correctly diagnosed during their hospitalization.(12) Reasons for delayed or failed diagnosis are not completely understood, but under-recognition is not likely the result of confused clinical findings since 88% of patients already met Berlin criteria on day one of hypoxemic respiratory failure in LUNG SAFE, and in 76% of patients at the time of intubation in the LOTUS-FRUIT study.(13,22)

It is suspected that under-diagnosis is a major cause of the high rates of death with ARDS. Indeed, under-diagnosis has also been associated with suboptimal care delivery, and studies have demonstrated repeatedly that clinicians, operating unassisted by decision support, consistently fail to apply evidence-based therapies.(13,20,22–24) These therapies are often critical for improving an ARDS patient's chance of survival. In a large study, it was found that patients with ARDS who received restricted tidal volumes of air from ventilators had a dramatically higher survival rate than those that received standard ventilation procedures.(25) In another study it was found that using paralytic agents in eliminating patient respiratory effort improved chances of survival.(26) Future clinical care may even include using biomarkers to

more accurately perform phenotyping of ARDS patients so that patients receive more personalized care protocols.(27) However, all of these treatment and diagnostic protocols would not be considered for use in patients without ARDS. Furthermore, the longer that ARDS persists, the more cellular death that may occur in the lungs. In some patients, this cellular death can lead to long term injury and scarring, and can permanently reduce functional air capacity of lungs, thereby reducing quality of life for these patients, while placing them at greater risk for subsequent hospitalization.(16,17) These concerns highlight that prompt and accurate diagnosis is essential for patients with ARDS. To hasten the time to diagnosis, scientists have proposed automated algorithms for diagnosing ARDS.

Early automated ARDS diagnostic systems used rule-based processing of keywords from CXR reports and numeric P/F values extracted from the electronic health record (EHR) (28,29), whereas recent studies have used more modern tools such as natural language processing and machine learning (ML) algorithms (30–32). While these studies have demonstrated that proof-of-concept automated systems can improve the accuracy and timeliness of ARDS diagnosis, these approaches are dependent on availability of laboratory and radiographic data, local radiologist practices, and information technology that may not be present in all healthcare settings. Most importantly however, these systems may suffer from lack of generalizability outside their originating institutions, and when ARDS sniffers were tested in new patient populations, algorithm specificity declined substantially.(33–35) It was found generalizability of sniffer systems is limited because sniffers rely on text of radiologists' reports of chest radiograph findings, and institutional or radiologist-specific practice patterns may vary between healthcare centers.(33,36) Furthermore, infrequent sampling of chest radiographs and P/F measurements, and delays in their interpretation may also limit automation and timeliness of diagnosis.(30,31)

Recent research has attempted use of more advanced analytic algorithms and improved data types, but still, the fundamental problems of differing practice patterns, limited automation, and timeliness of diagnosis still remain unsolved. (30,37–44)

1.2 Avenues of Improvement for ARDS Screening.

To improve generalizability and data timeliness/availability of ARDS screening we hypothesized that we could find alternative sources of physiologic information for future ARDS CDSS from mechanical ventilators. Mechanical ventilators are sophisticated life support devices that support oxygenation, ventilation, and the work of breathing for patients with respiratory failure by providing supplemental oxygen, pressure and air flow.(45,46) This air flow and pressure data can be broadly defined as ventilator waveform data (VWD) (Figure 1.3). VWD is displayed visually and is used by bedside clinicians to diagnose and manage patients in real time. VWD is an appealing source of information for ARDS surveillance because it contains unbiased physiologic data (Figure 1.3) that is available from the start of MV, and has been hypothesized to contain information that is indicative of onset and progression of ARDS.(47). VWD is also a data source that can be utilized in any medical setting because all patients undergoing respiratory failure receive MV.(48) These advantages may mean that VWD is capable of being used either as a standalone ARDS detection data source, or as an element of a more complex detection model.

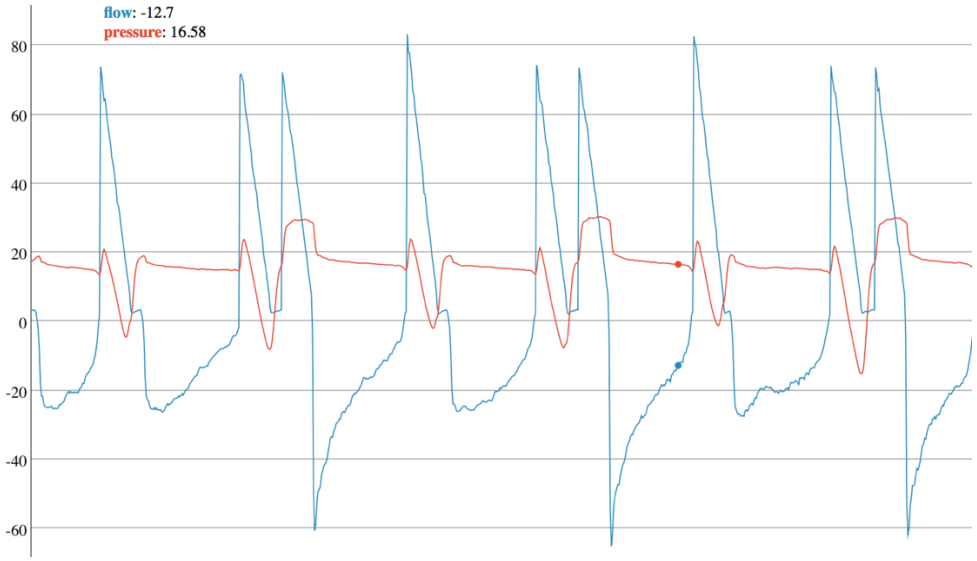


Figure 1.3: Flow (blue) and pressure (red) waveforms captured from the ventilator of a patient with severe ARDS.

Until now there has been no prior research exploring VWD as a mechanism for ARDS detection. So, this dissertation aims to address research challenges posed by this problem. A major issue is that VWD is a heterogeneous data source whose morphology and other characteristics depends heavily on patient physiology and bedside ventilation settings. This data heterogeneity causes difficulty to even expert respiratory care providers when asked to distinguish different types of ventilator waveforms.(49) So, it is critical that ARDS classification models can compensate for varying levels of noise and breathing heterogeneity in VWD. To compensate for this problem, we use machine learning (ML) to build models that can learn complex classification rules based on patterns in the relationships between predictor variables (referred to as features) and outcomes. ML offers advantages over traditional statistical methods including the ability to detect nonlinear relationships, the ability to detect relationships between features in high-dimensionality models, and even the ability to create unique features from patterns in the sample data.(5,50) ML also has been previously applied to VWD to automate the

assessment of patient-ventilator asynchrony so there is literature supporting use of ML in VWD applications.(45,51–56)

We discuss the proposed work of building an ML-based ARDS classifier as follows: First we need to create high-fidelity annotated datasets of breath information for our ML classifiers. Our work in *Chapter 2.2* shows how to build software necessary for annotating large quantities of VWD in a rapid, reproducible, and reliable manner. *Chapters 2.3* and *2.4* then discuss how we build models for better characterizing VWD using these datasets. Characterizing VWD is important because we desire our ARDS detection models to have high accuracy. So, the more information we can gather from it, the more information our models will have to utilize when performing disease detection. In *Chapter 3* we discuss the further work in characterizing VWD by performing reproduction and clinical validation of 15 different algorithms used to extract important lung-physiologic information from VWD for purposes of improving ARDS detection. Next, *Chapter 4* describes the research challenges to build a classifier for detecting ARDS using VWD. Then, in *Chapter 5* we describe how we can improve our ARDS classifier using deep learning. In *Chapters 6.1 and 6.2* we show how we integrated our ARDS detection models into a real-time system for remote patient monitoring and adaptive feedback for clinical providers. Finally, in *Chapter 6.3* we discuss future work that can be undertaken following our research advances.

1.3 Improving the Characterization of Ventilator Waveform Data

Utilizing VWD in algorithmic classifiers in the ICU has the potential to detect ARDS, and other pathologic phenomena.(25,47,52,57–62) To develop and validate high performance ML classifiers, clinicians have historically been required to manually annotate large volumes of

VWD by visually analyzing the shape and magnitude of air flow and pressure data.(52,55,57–59) However, this process is time-consuming, subjective, and has been marked by low inter-rater agreement due to visual classification criteria based on "expert opinion", as well as data management challenges resulting from the high volume of data and the laborious manual annotation process. (49,52,55,57,58) Previous groups have developed waveform annotation software that can handle large volume data and output categorization results, but these efforts lack the specific functionality necessary for overcoming problems inherent with annotating VWD.(63–74) These challenges are: 1) consistent annotation workflows do not exist for VWD annotation, which has led to the creation of datasets that are potentially non-reproducible across medical centers,(49) and 2) large, high-quality, multi-reviewer adjudicated datasets of VWD are highly time-consuming to generate for expert clinicians.(55)

With these challenges in mind, in *Chapter 2.2* we explore and address gaps in the field of VWD annotation by developing the Annotation PipeLine (APL) (Figure 1.4). APL supports expert annotation of ventilator data and builds on concepts from existing platforms for annotating data other than VWD.(75) Utilizing APL, we have been able to annotate and provide dual clinician validation for over 315,000 ventilator recorded breaths across two different datasets. These data comprise some of the largest recorded breath-level, multi-clinician adjudicated datasets for use in developing algorithms to improve the delivery of MV.(54,55,57,58,76) APL's fundamental design features allow for consistent and efficient annotation of VWD and may be broadly applied in future research to improve efficiency, reproducibility, and fidelity of medical waveform annotation and classifier development.

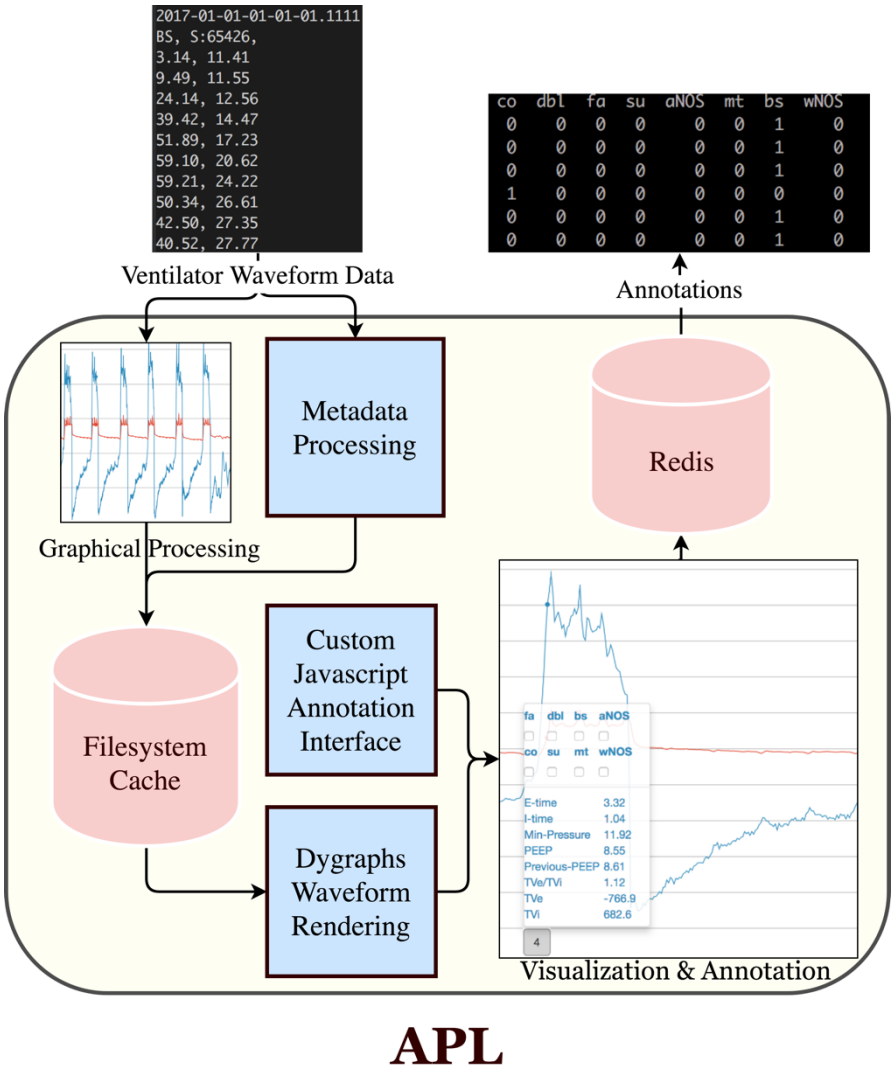


Figure 1.4: Shows the flow of data within APL as it is processed by our software architecture. Raw VWD is uploaded to APL and then processed for graphical rendering, using ventMAP metadata processing to extract clinical metadata from each breath. Results from processing are stored on the filesystem. Next, we visualize both VWD, and perform breath-level annotation by utilizing Dygraphs with our own custom annotation overlay. All annotations are stored in Redis and can be output by the user into CSV format.

Using APL, we can perform per-breath annotations for a variety of phenomena including patient ventilator asynchrony (PVA). PVA is a pathologic phenomenon that occurs because of a mismatch between a patient's desired ventilation needs and a mechanical ventilator's configured settings.(77) This mismatch will manifest clinically by increasing the amount of work a patient

must perform to breathe. Over long periods of time PVA can result in ventilator-induced lung injury (VILI), which is a frustrating phenomenon that can cause additional lung damage, increased length of ICU stays, and even death.(11,26,52,57,78) Clinicians can detect PVAs during bedside examination, but PVA detection can be delayed due to lack of 24/7 access to appropriately trained clinicians, provider inattention, or heavy clinical workloads. PVA detection can be performed with electronic algorithms, but most PVA detection algorithms rely only on expert rules that may not generalize to broader patient populations seen in the ICU. To improve upon existing automated algorithms for detecting PVA, we used a dataset of 9,719 breaths derived from APL to train an ensemble ML model to detect 2 pathological forms of PVA. We discuss this work in *Chapter 2.3* and highlight modeling challenges encountered. This work serves a twofold purpose: 1) it shows how we can leverage APL to create high performance ML models, and 2) it also shows how we can create secondary ML systems to improve our characterization of VWD for later ARDS detection applications.

Another key piece of information that MV CDSS lack is the choice of ventilation mode (VM) that determines the pattern of flow and pressure delivery with each breath (Figure 1.5B-D). This information is generally unavailable to CDSS due to the lack of interoperability and information exchange between CDSS and the ventilator or electronic health record.(76) CDSS knowledge of VM is important because VM can be important for care optimization, (79–82) and can affect how featurization can be performed for ARDS algorithmic classifiers.(83) For example, if we know that our VM is in volume control, then we can utilize algorithms that are optimized for estimating physiologic features from volume control VWD. Such optimizations may help improve accuracy of ARDS classifiers. Another example would be that CDSS could provide alerts to clinicians if patients continually violate safe volumes of air to inhale. This

would be especially important in cases where patients have ARDS and need limited tidal volumes.(25,84) In this case, the CDSS might recommend that patients be placed on a VM that limits tidal volumes such as volume-control.

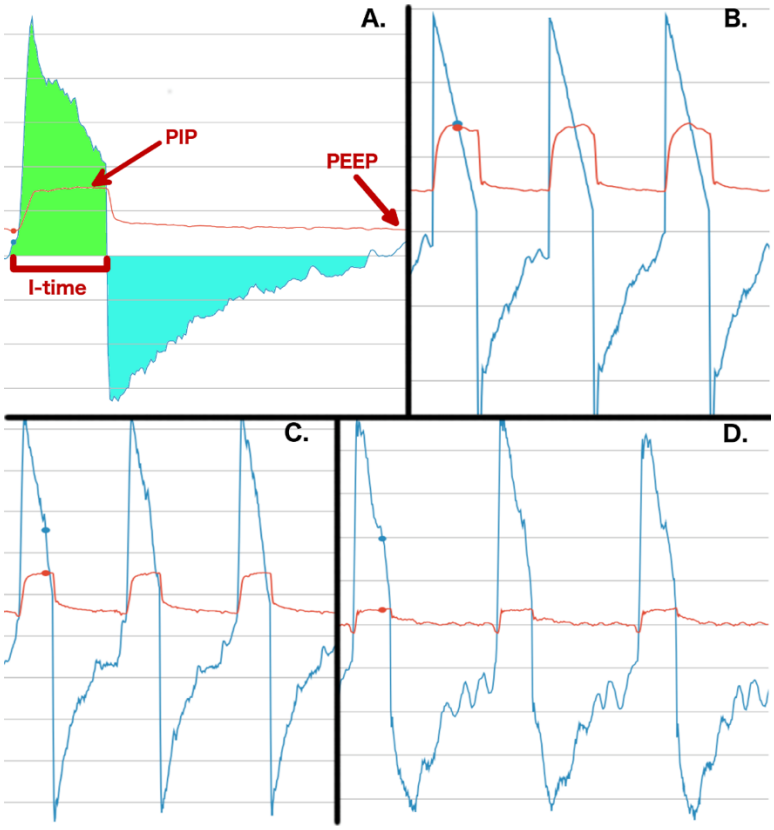


Figure 1.5: Shows visualizations of ventilator waveform data (VWD) across different ventilation modes. Air flow measurements are represented in blue, and pressure in red. **A.)** How to extract information from VWD. Positive End Expiratory Pressure (PEEP) is noted as the minimum pressure supplied by a ventilator, and peak inspiratory pressure (PIP) is the maximum pressure supplied during inhalation. Inspiratory time (I-time) is the amount of time a patient breathes in. Total amount of air breathed in is represented in green, and air exhaled is shown in teal. **B.)** An example of volume control (VC), a mode where a patient receives a fixed volume of air for each breath. It can be recognized by a triangular inspiratory flow. **C.)** Shows an example of pressure control (PC). In PC, pressure is fixed during inhalation. **D.)** An example of continuous positive airway pressure (CPAP). Here minimal pressure support is given, and all breaths are initiated by the patient.

If MV CDSS lacks knowledge of VM from more traditional methods, it may still be able to access it by utilizing information derived from streams of flow and pressure readings that

comprise VWD. To date, at least one previous effort developed a rule-based VM classifier using analysis of VWD, but its use of a closed dataset, its limited temporal resolution, and the accuracy of the model represent potential limitations both for research and decision support.(76,79) Having highly granular temporal resolution of results is important for VM. Previous efforts performed per-hour VM classification, however, in practice providers may change VM frequently based on changes in clinical state or patient tolerance of VM. These changes may cause VM periods to last for as low as minutes of time. To improve upon previous work, we note that machine learning (ML) has proven capable of accounting for the highly variable nature of physiologic data such as VWD on temporally granular time scales.(55,56) So, we created a ML model that could identify different VMs on a per-breath basis, with a freely accessible dataset, using only VWD as input.

So, in *Chapter 2.4*, we detail multiple important considerations for modeling an ML classifier that can classify ventilator mode. First, we discuss how we created one of the largest datasets of per-breath labeled information, the extraction of features from VWD, and the performance of our resulting ML model that can determine 5 of the most widely used ventilation modes in the USA.(83) Second, we discuss experiments of how our model performs in the presence of missing training data. Finally, we discuss experimentation we conducted for reducing the size of our training dataset by nearly 65% while maintaining generalizability of our classifier to our testing set. To allow reproducibility of our work, our code and dataset are

publicly accessible and published on GitHub[1]. Thus, we hope that our work will serve as a catalyst for continuing to improve the capabilities and efficiency of MV CDSS.

1.4 Evaluating Algorithms for Estimating Respiratory Compliance

To train machine learning models for ARDS detection we necessitate a system for extracting respiratory compliance (C_{rs}) from VWD. C_{rs} measures the ease at which the lung and associated thoracic structures are able to stretch when filling with air(85), and is hypothesized to be indicative of ARDS(47,86) because the lungs of ARDS patients are typically very stiff and have low C_{rs} measurements compared to non-ARDS patients (47,87). Despite physiologic rationale for its use, the monitoring of C_{rs} is not reliably performed in the ICU because it involves time consuming procedures(88–91), or use of clinically invasive devices that may adversely affect care(92). These barriers mean that C_{rs} can go unrecorded and thus unused when making important care decisions for a patient, which in turn may affect patient care outcomes(47,93–95).

To offer solution to this problem, a non-invasive and continuous measurement of respiratory mechanics has been proposed by modeling the lung as a single compartment governed by the equation of motion(45,88). This approach makes simplifying assumptions about the behavior of the lung that has allowed researchers to measure C_{rs} on mechanically ventilated patients using a number of different computer algorithms(45,47,89,96–105). Many of these

[1] https://github.com/hahnicity/ventmode

algorithms have been validated in idealized circumstances, however, few of them have been thoroughly tested in scenarios typically seen in the clinical environment(92,96,106).

The clinical environment poses challenges to C_{rs} estimator algorithms because there are various types of breathing and ways that a breath can be delivered to a patient that cause estimation error. One scenario in which algorithms may provide inaccurate or inconsistent estimates is in cases of severe PVA (Figure 1.6B). PVA occurs when patient breathing is mismatched with ventilator settings(77), which can result in breathing patterns that C_{rs} estimators are not designed to handle. Another scenario that C_{rs} estimators may struggle to adapt to is different ventilation modes (VMs)(89). VM determines the way a patient receives a breath from the ventilator. One mode, named volume control (VC) specifies that patients receive a fixed amount of air per breath. Other modes such as pressure control (PC) and pressure regulated volume control (PRVC) are designed so that patients receive a fixed amount of pressure during a breath and can breathe a more flexible amount of air per breath. A number of algorithms have been developed specifically for VC(102,103), while others have been tested and validated under PC or other VMs(89,96,106). However, it is unclear if these algorithms are cross-compatible with other VMs because the research community lacks widely available data sources from a variety of patients, and because of the difficulty in collecting ventilation data during clinical trials. This lack of validation also makes it unclear to clinicians and researchers which algorithm can be used in varying circumstances and may hinder the implementation of model-based approaches in clinical practice.

To address this situation, in *Chapter 3* we explore the performance of a comprehensive number of algorithms for estimating C_{rs} in a retrospective cohort of subjects under varying clinical scenarios. First, we explore general algorithmic performance across all patients, and then

we focus on performance under scenarios of differing VM and types of patient-ventilator synchrony (PVA). In doing so, we highlight several insights. First, we highlight which algorithms are best for calculating C_{rs} under varying scenarios of patient state such as varying VMs and PVAs. Second, we discuss a windowing approach that will help improve using algorithmic results in clinical practice. Third, we show a surprising result that algorithms designed for VC perform equally well when applied to patient breath data collected from mechanical ventilators operating under pressure modes, and vice versa. These contributions and our open dataset will help improve the understanding of advantages and limitations of the model-based approach and may help guide clinicians and researchers in using C_{rs} estimators in their own practice and studies.

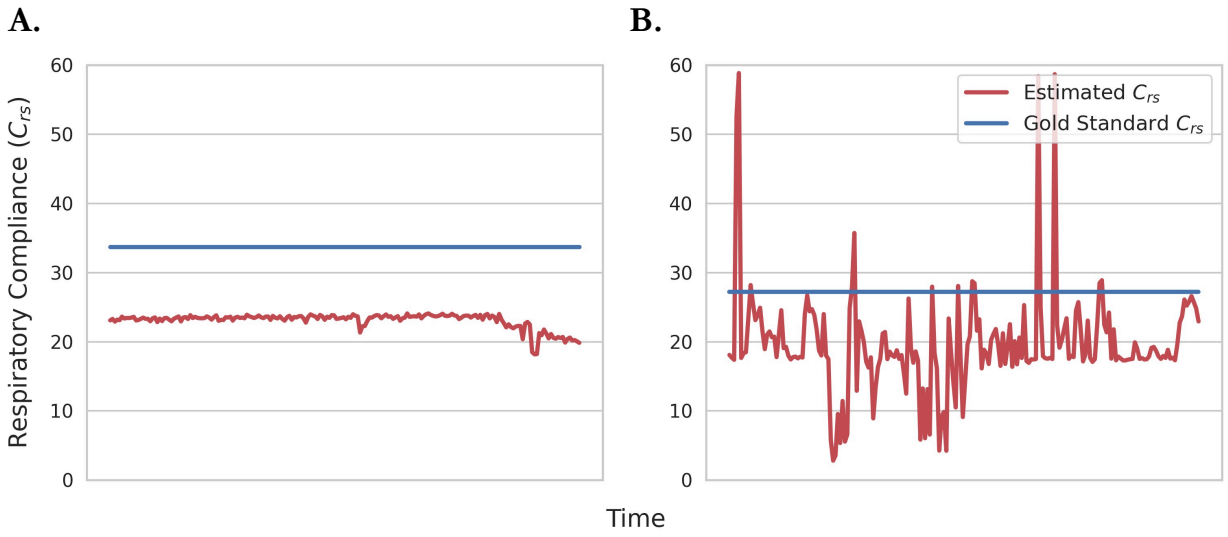


Figure 1.6: We provide 2 samples of data from different patients. In **A.** the patient is breathing synchronously with the ventilator and there is little variation in C_{rs} estimates. In **B.** the C_{rs} estimate is highly variable due to patient ventilator asynchrony or transient waveform anomalies.

1.5 ARDS Detection Using Machine Learning

In *Chapter 4* we explore our primary research question of developing a system that can utilize widely used, and unbiased VWD to develop early-warning models that can alert clinicians for onset of ARDS. This research builds upon the work in *Chapter 2* and *Chapter 3* by extracting important features for use in ARDS ML classification models.(11,25,26,107,108) Using a preliminary cohort of 100 patients, we utilize our extracted features to build ML models that can differentiate between VWD from ARDS patients and patients who were hospitalized for other reasons. Our findings show that VWD shows promise for early warning of ARDS at 6, 12, and 24-hour intervals after a patient first meets Berlin criterion. This promising proof of concept may allow wider clinical studies for utilizing VWD in the ICU and improve diagnostic rates for ARDS.

Although our work in *Chapter 4* is a promising proof of concept, it may be limited by our initial choice of using classical ML algorithms like random forests (RF) with expert-informed waveform featurization.(110) Expert-informed feature engineering is common in traditional machine learning and may aid in understanding model behavior but the use of human-interpretable features may bias models in unanticipated ways and may potentially constrain performance.(50,110)

To solve these limitations, researchers have applied deep learning (DL) approaches to automatically learn high-level features from the input data.(111) This approach has been used widely in image processing and in other time series analytics work.(112–117) So, we hypothesized that DL models would improve upon traditional ML models in discriminating between patients with ARDS and non-ARDS indications for MV. In *Chapter 5* we investigate this hypothesis and develop DL models capable of screening for ARDS using only VWD. We

analyzed performance of two commonly used DL networks used for this task, convolutional neural networks (CNN) and long short-term memory (LSTM) networks and compared DL performance to that of a previously published traditional ML models. Finally, we investigated the mechanisms by which our best-performing DL model was able to learn features from raw VWD that contributed to the differential performance observed between DL and traditional ML models. Our work makes contributions of 1) Showing that DL can improve the overall performance of ML classifiers for task of ARDS detection and shows specific improvements that allow DL to surpass our work in *Chapter 4*. 2) Exploring the performance of our DL models with varying dataset sizes. This experimentation casts doubt on the traditional assumption that deep learning using physiologic data necessitates very large medical datasets to achieve performance gains over traditional machine learning. 3) We highlight potentially novel techniques for better understanding how DL classifiers improve upon traditional ML models using high-frequency VWD.

Chapter 2: Tooling For Improving Ventilator Clinical Decision Support Systems

2.1 Overview

High-performance model development requires well characterized datasets annotated for salient events by expert clinicians through a process of multi-reviewer adjudication.(54) Accurately and reproducibly annotating data requires substantial investment of effort from time-limited, expert reviewers, which to date has limited the size and generalizability of validated datasets.(55,118–120) We attempt to ameliorate this issue by creating APL, a software platform for the fast, reproducible, and large-scale annotation of VWD.

Using APL, we can annotate datasets of ventilator data, and then train ML models to scan for potentially harmful ventilation anomalies called patient ventilator asynchrony (PVA). We create an ML system for PVA detection because there are currently no intelligent/automated systems integrated into mechanical ventilators capable of detecting PVAs and generating alerts to clinicians. Current systems consist of simple threshold-based alarms that are prone to frequent false positive alerts, which cause clinicians to ignore them. The only reliable way to detect PVA is via bedside examination of patients, but this is can only occur during scheduled clinician visits, and even then, studies have shown that even trained clinicians often fail to consistently recognize PVA.(121) To improve the speed and accuracy of PVA detection, we aimed to create a system that could compute upon VWD and automatically classify a breath as normal or PVA (Figure 2.3).

Identification of PVA is not the only method of improving care for ventilated patients. CDSS also promises to reduce chances of delivering improper MV by automating aspects of ventilator configuration. However, a key detriment to these systems is that they often lack access to the configured ventilator mode and therefore lack information that may improve the efficiency of these CDSS.(76) In *Chapter 2.4* we discuss our work towards creating a machine learning system that scan detect ventilator mode through analysis of only VWD. Our work highlights potential improvements in ventilation CDSS and highlights further tooling that we can utilize to improve ARDS detection CDSS.

2.2 Fast and Reproducible Ventilator Waveform Annotation

2.2.1 Methods

APL is a free, publicly accessible, web-based annotation platform developed since 2015 to support studies at University of California Davis (UCD) utilizing VWD collected from mechanically ventilated patients.(122) All VWD were collected under an approved IRB protocol and do not contain protected health information. Data were collected from Puritan Bennet Model 840™ (PB-840) ventilators (Covidien, Dublin, Republic of Ireland) and stored in multiple sequential files up to two hours in length. APL uses an open-source stack including: ventMAP software for extraction of waveform metadata, the Python Flask web framework, the JavaScript library Dygraphs for graphing of VWD, Redis for storage, and Docker for APL deployment.(54,123) APL's source code and install instructions are available on GitHub for modification or use of APL.

APL was built to support objective classification of VWD by displaying clinically relevant quantitative metadata to help improve the consistency of the annotation workflow. After

data collection, VWD is uploaded to APL in CSV format, is preprocessed for graphical display, and automatically split into individual breaths. Each breath is processed to generate clinically relevant metadata pertinent to the classification of VWD using ventMAP (Figure 1.4).(54,56) Examples of metadata generated include the inspiratory tidal volume (the volume delivered by the ventilator with each breath) and the expiratory time (the duration of each exhalation). All derived metadata were determined to be clinically relevant by four critical care clinicians (JYA, BTK, ICP, JN). Metadata are cached in a CSV file and then retrieved by Flask to generate a breath-level drop-up table that displays breath-specific metadata and annotation options. Display of metadata in a drop-up table mimics the user interface of a mechanical ventilator and was designed to facilitate use of APL by expert reviewers (Figure 2.1).

APL also incorporates functionality to support online reconciliation of discordant annotations to improve consistency and reproducibility of the annotation process. This functionality is possible because APL is an online web platform and can store all user annotations centrally in the Redis database. Stored annotations can then be retrieved later by the Python Flask web framework to compare classification decisions by different reviewers. Differing classifications are displayed by Dygraphs as a red label that contains the cause for classification disagreement. Dygraphs displays a green checkbox above a breath if no disagreement for a breath is found.

The annotation view (AV) functionality provides the ability to select subsets of available metadata and breath classifications to display when reviewers are performing annotation. AVs are aimed at reducing information overload for the user and simplifying the annotation process by targeting it to specific project requirements. All AVs are stored in Redis, and new AVs can be

created by a user by selecting the specific metadata and breath classification desired.

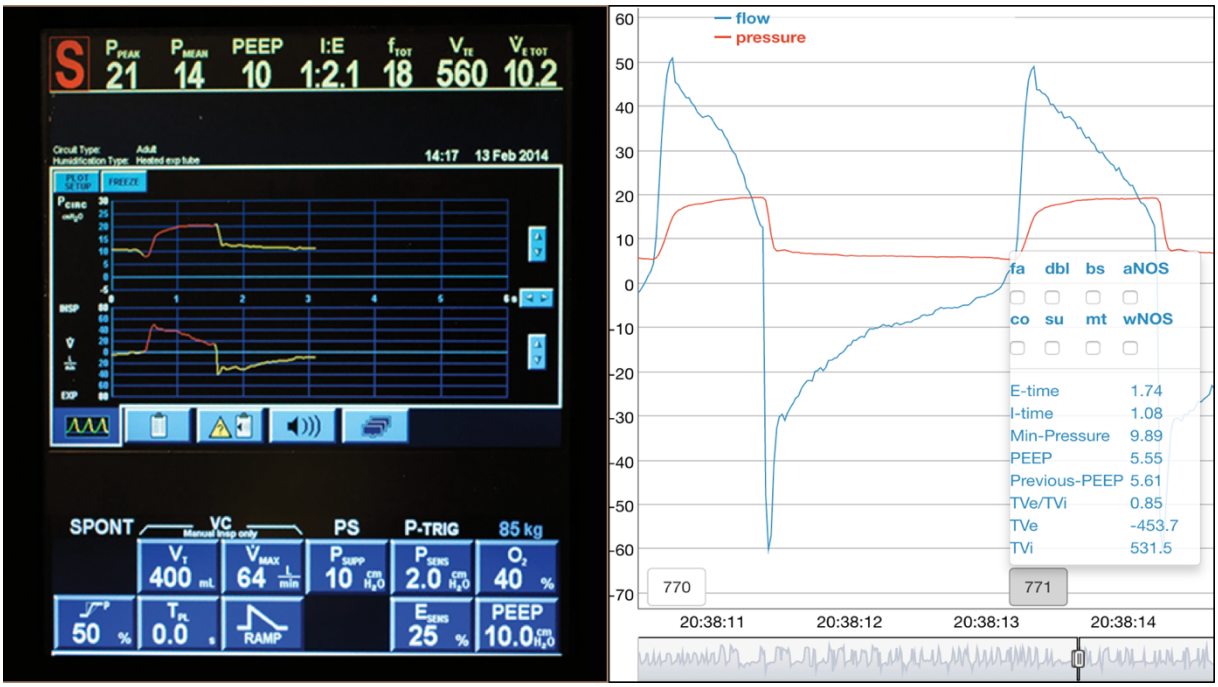


Figure 2.1: We designed APL to be familiar to clinicians, rendering information analogous to the user interface screen of a mechanical ventilator. Left, an image of the graphical user interface of a mechanical ventilator. Right, APL pressure (red) and flow (blue) waveforms with customizable breath-level metadata and point-click annotation. Breath interpretation at the bedside relies on both waveform and metadata analysis. APL presents both waveform morphology and breath-level metadata for identification of threshold-defined events.

To streamline the end-to-end annotation workflow, we designed a unified system of annotation and visual display termed ‘on-screen annotation’ that allows for simultaneous display of waveform morphology, metadata, and annotation choices for every breath (Figure 2.1). On-screen annotation is accomplished by coding customized instructions in JavaScript that render drop-up tables containing metadata, annotation choices, and server callback logic. We then ensure that Dygraphs runs our overlay software after rendering of VWD. Using this overlay interface, we incorporated additional functionality to improve the speed of how annotations are performed, such as the option to perform single or multi-breath classifications.

2.2.2 Results

Before starting the annotation process, users typically visit the Settings page to choose an AV (Figure 2.2). While users may choose a pre-defined AV, project leaders can create an AV, requiring that all other reviewers utilize that view to enforce consistency of annotations and outputs for a dataset. When an AV is created, leaders choose from a variety of annotation options and metadata available for display (Figure 2.2). When finished, the view can be named and saved to be used by other project members. In our initial work, we utilized AVs to create two different gold-standard datasets for PVA and ventilator mode (VM) classification.

Once a view is selected, uploaded VWD files can be graphically displayed for annotation. If reviewers only wish to annotate portions of a file, they can zoom in and out of specific breath regions using Dygraphs to focus on regions of interest. When a specific breath is annotated, the user can click on its breath number, and a drop-up window will appear. Drop-up windows display both quantitative metadata and annotation options allowed by the current AV (Figure 2.2). Users can annotate a breath by selecting the checkbox corresponding to the specific annotation desired. A user can also annotate multiple consecutive breaths using multi-breath functionality where the user annotates the first and the final breaths in a series, and all intermediate annotations will be automatically completed.

After multiple users annotate the same file, reconciliation of discordant annotations can occur. Reconciliation functionality displays VWD using an interface like annotation but highlights disagreements in red and agreements with green checkboxes. Reviewers can evaluate each disagreement simultaneously and collaborate to resolve them. Once finished, users can output a CSV file containing all gold standard breath-level annotations.

Using APL, three critical care clinicians from UCD (BTK, JN, and JYA) annotated breaths for 18 different classifications from 254 VWD patient files and generated two datasets totaling 318,676 multi-clinician adjudicated breaths. To our knowledge, this collection represents one of the largest reported datasets of breath-level, multi-clinician adjudicated VWD (Table 2.1)

Dataset	**Number of Annotated Breaths**	**Number of Breath Classifications**
Pohlman 2008 (124)	391	2
Blanch 2012 (125)	1,024	2
Gholami 2018 (126) *(using ExpertLabeler)*	1,377	3
Sottile 2018 (55)	2,500	4
Colombo 2011 (49)	3,731	4
Adams 2017 (54) *(using APL)*	*9,719*	*8*
Thille 2006 (77)	35,580-58,110	3
Beitler 2016 (53)	143,199	1
Rehm (Ventilator Mode) 2019 (127) *(using APL)*	*308,957*	*10*

Table 2.1*:* Existing multi-clinician adjudicated datasets of VWD. Number of classes corresponds with the number of annotation decisions available to the clinician as they performed annotation, including normal breaths. Exact counts for number of breaths annotated by Thille were not mentioned in the literature, so a range was extrapolated from the number of patients enrolled in their study and the respiratory rate of patients.

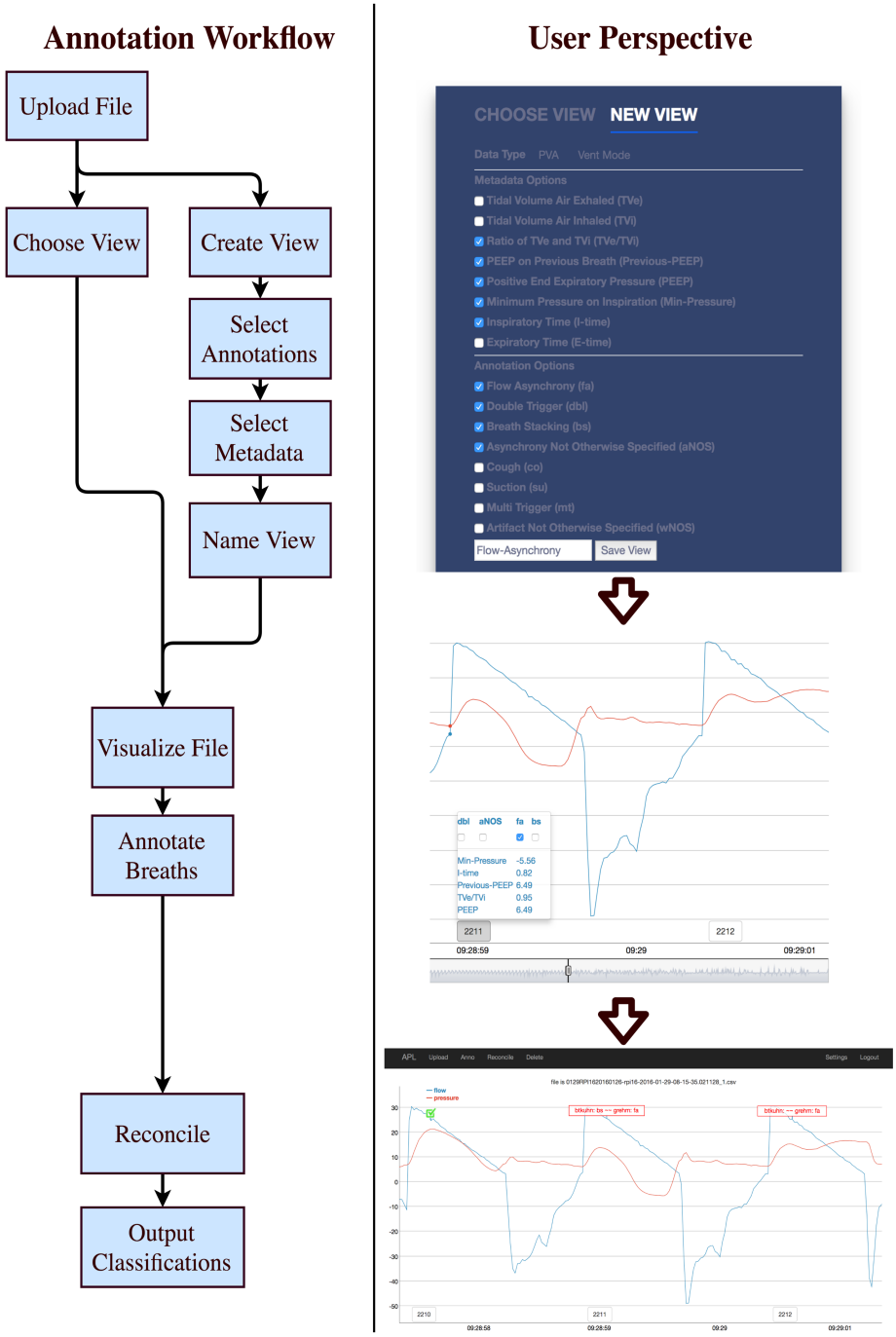


Figure 2.2: We display a typical workflow for a user when using APL, and screenshots from APL that display typical uses and functionality. On the left, annotation workflow discusses the individual steps a user would take to annotate a file. On the right three images are shown: the first showing how a user would create a new annotation view, the second displays how the annotation process occurs on ventilator waveform data. The third shows how reconciliation occurs with discordant classifications highlighted in red, displaying exactly which classifications are mismatched. Agreements are displayed as a checkbox.

2.2.3 Discussion

Creating datasets of large, multi-clinician annotated VWD for MV research is limited by a lack of accessible, efficient, and reproducible multi-reviewer annotation methods. We developed the APL to remedy these issues for VWD annotation and perform structured output of results. APL uses breath metadata to create hybrid visual-quantitative classifications of VWD. Incorporation of quantitative and qualitative methods into a web-based visual annotation interface may reduce reviewer subjectivity in VWD classification and enables efficient resolution of reviewer disagreements to generate gold standard data sets to drive classifier development. Using these methods, our team has annotated 318,676 dual-adjudicated breaths for use in development of VM and PVA detection algorithms, which represents one of the largest existing collections of multi-reviewer annotated VWD.(49,54,55,61,124,125)

Historically, VWD annotation has been performed either by hand, use of electronic spreadsheets, and most recently by the web-based ExpertLabeler system.(126) A major limitation of these methods is that classifications are performed by visual criteria alone which are subject to high inter-rater bias due to reviewer subjectivity and local practice pattern.(49) APL addresses this limitation with the development of methods to support the simultaneous use of waveform morphology and metadata derived from VWD in classification decisions. To our knowledge, no MV-specific annotation package exists can compute breath level metadata, display it on-screen, and allow multi-reviewer annotation and adjudication of events.

Technologically, APL builds on previous work related to annotation platform development and features novel contributions of its own. Our work to display VWD builds on

previous platforms that provide annotation of generic physiologic data such as AcqKnowledge, ChronoViz, and BEDA, which have exemplified how to perform annotation on temporally archived waveform data.(67,128–131) APL's combined morphologic and metadata display functionality also builds upon several previous studies that have derived higher-level metadata from raw input streams for purposes of display and annotation.(72,132–134) For example, Jing et al developed the ability to annotate multiple electroencephalogram (EEG) waveforms simultaneously.(135) Similarly, APL extends existing work developed in natural language processing applications for inter-rater reconciliation to a new domain of streaming physiologic waveform annotation.(75) Finally, APL contributes novel functionality including the use of drop-up windows with combined metadata and annotation capabilities, and the addition of annotation views that enable different metadata and annotation capabilities to be enforced across teams and customized to specific research needs.

Our work is limited by use of data and clinicians from a single center, and APL is presently limited to processing VWD from a single ventilator. Future work will generalize APL's data processing capabilities to other ventilators and medical devices by exploring the use of emerging waveform encoding standards.(136) In addition, APL has only been used by four clinicians to date and further modifications may be required to meet the needs of other researchers. Changes to APL may also enable additional investigation into the reproducibility of waveform classification, and may facilitate the development of consensus guidelines for annotating VWD.(49,137,138) To promote further research, we have freely released our code on GitHub so that the wider research community may modify our work.

2.3 PVA Detection

2.3.1 Methods

From our repository of collected data, we extracted VWD from 35 patients who received ventilation at the University of California Davis Medical Center (UCDMC). For each patient, we selected a period of approximately 300 breaths where PVA was highly prevalent. Two ICU physicians independently annotated 9,719 individual breaths to achieve a ground truth labeled data set. Classification was performed via a combination of clinically guided heuristic rules and visual inspection, and each breath was labeled as one of 4 categories: normal, artifact, double trigger asynchrony (DTA), or breath stacking asynchrony (BSA). We targeted DTA and BSA because they are two of the most common forms of PVA and are thought to contribute to ventilator induced lung injury. Artifact breaths like suction and cough were identified and included in the dataset because they share characteristics with common forms of PVA that can result in false-positive PVA classification. All artifact and normal breaths were then included together and labeled as non-PVA. Any disagreements in breath classification were reconciled between the reviewing clinicians, and a consensus label was chosen. Using this process, we created one of the largest dual-adjudicated datasets devoted to PVA detection reported to date. In total our dataset contains 1,928 BSA breaths, 752 DTA breaths, and 7,039 non-PVA breaths.

To automatically distinguish different types of breathing, PVA detection algorithms must have the ability to extract quantitative features from breaths instead of relying on visually subjective breath characteristics. So, we used the ventMAP software suite to extract clinically relevant features from VWD.(54) In total, we derived 16 different features from each breath (Figure 2.3). After features were extracted from VWD, we evaluated multiple supervised ML

models to perform PVA classification. PVA classification was done on a per-breath basis where each breath is trained and classified based on a corresponding class label of non-PVA, BSA, or DTA. When training our models, we encountered a class imbalance issue because the number of PVA breaths were disproportionate to the number of non-PVA breaths in our dataset. Imbalanced training sets can be an obstacle to training accurate classifiers, resulting in decreased model performance when classifying DTA (56) in our case. We explored multiple methods to correct for class imbalance including random under sampling (RUS) and the synthetic minority over-sampling technique (SMOTE). We found that SMOTE offered the best balance of recall and specificity while RUS offered better recall than SMOTE at the cost of decreased specificity. In our experiments, we found that our models performed best when we used SMOTE to create a 1:1:1 ratio of non-PVA, DTA, and BSA observations for our training set. This ratio created the same number of DTA and BSA observations while keeping non-PVA observations static.

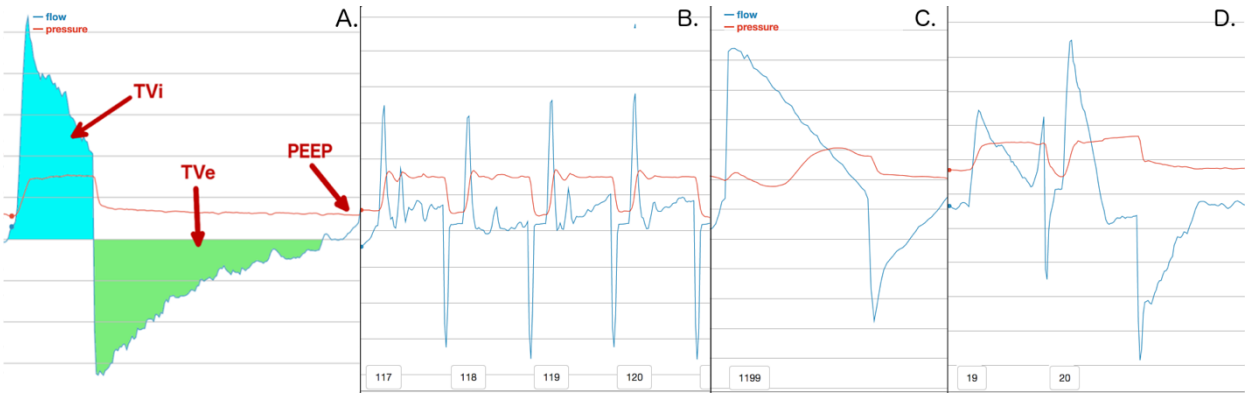


Figure 2.3: A. displays a normal breath and how information can be extracted from breaths in general. We define volume inhaled (TVi) as the amount of air breathed in on a breath. Tidal volume exhaled (TVe) is the amount of air exhaled. Positive end expiratory pressure (PEEP) is the minimum pressure setting for a ventilator. B. shows a series of breaths that occur due to a suctioning procedure. C. shows a breath stacking event, where a patient breathes in significantly more air than they exhale. D. shows a double trigger, which is two breaths that occur in rapid

2.3.2 Results

In our prior work,(56) we evaluated 10 ML algorithms: SVM, extreme learning, naïve bayes, multi-layer perceptron (MLP), and six tree-based approaches, namely decision trees, extra trees classifier, random forest, Adaboost, extremely random trees classifier (ERTC), and gradient boosted classifier (GBC). The performance of these algorithms was evaluated through k-fold validation, where we left one patient's data out for testing, and used the rest for training. This yielded 35 training and testing folds, corresponding with the number of patients in our dataset. The performance metrics of interest are accuracy, recall, and specificity. Precision was not reported because its measurement would be biased because we focused on specifically selected regions of breath data with high PVA occurrence. Our explorations showed that extremely random trees classifier (ERTC), gradient boosted classifier (GBC), and multi-layer perceptron (MLP), achieve the best performance, but each with its own trade-offs.(56) ERTC achieved better accuracy for DTA class, while GBC and MLP performed better for BSA. An ensemble classifier consisting of ERTC, GBC, and MLP outperformed all other classifiers in terms of recall (sensitivity) and specificity, and the results are summarized in Table 2.2. The high accuracy of our ensemble classifier was the result of numerous optimizations and DTA performance was especially assisted using SMOTE. These results suggest that ML-based PVA detection algorithms have potential to be translated into clinical practice where they may improve the quality of care for patients receiving mechanical ventilation.

Algorithm	Class	Accuracy	Sensitivity	Specificity
Ensemble	Non-PVA	0.971	0.9673	0.9806
	DTA	0.9742	0.9601	0.9754
	BSA	0.9793	0.9445	0.9879
ERTC	Non-PVA	0.7245	0.6744	0.856
	DTA	0.8693	0.9934	0.8589
	BSA	0.7683	0.5835	0.814
GBC	Non-PVA	0.9707	0.9692	0.9746
	DTA	0.9745	0.9335	0.9779
	BSA	0.9779	0.9445	0.9861
MLP	Non-PVA	0.954	0.9439	0.9806
	DTA	0.9576	0.9628	0.9572
	BSA	0.9678	0.9155	0.9807

Table 2.2: Descriptive statistics for all classifiers run on the multiclass classification problem using SMOTE. ERTC: Extremely Randomized Trees classifier. GBC: Gradient Boosting classifier. MLP; Multi-layer Perceptron. DTA; Double-Trigger Asynchrony. BSA; Breath Stacking Asynchrony

2.3.3 Discussion

In this chapter, we created an ensemble machine learning model for classifying two common, clinically relevant PVA subtypes, BSA and DTA, thought to be detrimental to patient health. We showed that addressing several issues common to ML model development in other fields resulted in excellent sensitivity and specificity for the classification of PVA in mechanically ventilated patients. In addition to mechanical ventilation, these challenges are broadly relevant to the creation of ML models in other areas of medicine. We addressed the class imbalance problem that resulted from PVA breaths being outnumbered by non-PVA breaths by using SMOTE to equilibrate the number of DTA, BSA, and non-PVA breaths.

Use of SMOTE and ensemble methods may be particularly helpful for clinicians in the future. Our study highlights that mechanical waveform data is highly imbalanced, with patients taking in range of 20,000-40,000 breaths over a given day, where most waveform data is indicative of normal, synchronous breathing. When PVA does occur, it can be used as the basis for a classification model, but episodes of PVA can be brief, and relatively infrequent. SMOTE helps to address class imbalance resulting from data paucity by creating synthetic samples of a minority class, and while it is not the only method to deal with class imbalance, it is well represented in the literature and has proven itself valuable in areas that also deal with data imbalances such as security and networking research. Ensemble models can be useful as well for data that are noisy or have highly complex decision boundaries, and ensemble-based models have been shown to create more accurate classifiers by combining many weaker classifiers. In our use case, our ensemble model yields sensitivity and specificity improvements for DTA classification compared to the MLP classifier, while maintaining equivalent performance for BSA classification compared to a single GBC classifier.

There are several limitations that must be noted with this work. First, this is a single center study, and differing types of patient care in other centers may affect how our model performs. Similarly, our selection of ML algorithms was not exhaustive and use of different algorithms may have similarly affected model performance. Due to time constraints imposed by manual waveform annotation, our classifier uses a relatively small amount of ventilator data (7.63 hours total) based on highly selective ROI's where PVA is far more prevalent than it would be normally. Even though we present promising results, further studies will be necessary to claim generalizability of a ML anomaly detection model. Future work will need to examine additional methods for improving the accuracy of detection around strict classification boundaries, for

example where strict cutoffs in a feature value (e.g., E-time) are necessary for the correct classification of related classes such as DTA and BSA.

2.4 Ventilator Mode Detection

2.4.1 Methods

In this chapter, we used a dataset of VWD collected from 103 subjects (IRB# 647002) within intensive care environments of the University of California Davis Medical Center (UCDMC) consisting of MV flow and pressure measurements sampled at 50 Hz.(54,122) VM was not recorded during VWD data collection. We then randomly selected 2-4 hour epochs of VWD from the 103 subjects. All VWD was stored in data files of 2 hours in length, and approximately 2000 breaths were stored per data file. Each breath in these epochs was then annotated by three expert clinicians (JYA, BTK, JN) for the presence of one of five VMs: volume control (VC), pressure control (PC), pressure support (PS), continuous positive airway pressure (CPAP), and proportional assist ventilation (PAV) (Table 2.3). Many patients had 2-4 hour periods selected where ventilator mode was switched multiple times, other modes such as pressure regulated volume control (PRVC), volume support, and airway pressure release ventilation (APRV) were found, and annotated within these epochs, but were excluded in our final analysis because of their rarity of use at UCDMC.

Because VWD is so heterogeneous it can be difficult for even expert clinicians to make consistent classification of breathing patterns.(49) Thus, in performing classification of VM we ensured that each breath was dual clinician adjudicated, meaning that two clinicians would independently annotate a single breath, and if the classifications disagreed, they would be resolved through communication between the two.(54) To further account for breathing

heterogeneity, we included regions containing pathologic patient-ventilator interactions such as patient-ventilator asynchrony (PVA) and routine clinical events such as suctioning and cough.(54) We also include regions of data that include significant signal noise caused by moisture/blood/mucus in the ventilation tube.

With this dataset, we utilized 55 patients and 140,928 breaths for our training cohort, and 48 patients and 165,988 breaths for our testing cohort. There was no patient overlap between testing and training cohorts. The testing set was chosen to be approximately as large as the training set because initial modeling yielded strong results, and we wished to utilize a large testing set as further validation for our approach. Using both Scikit-learn and PyTorch ML libraries,(139,140) we then evaluated the use of multiple ML algorithms including: support vector machine (SVM),(141) multi-layer perceptron (MLP), long-short term memory recurrent neural network (LSTM RNN),(142) logistic regression, and a random forest (RF) classifier.(143) All models performed classification on a per-breath basis, the highest possible level of granularity possible in VM classification. Based on model investigation, we settled on using an RF with a parameterization of 30 classifier trees for our final model (see online supplemental[2]).

[2] https://github.com/hahnicity/ventmode/blob/master/supplemental

	Volume Control	Pressure Control	Pressure Support	CPAP	PAV
Patients	23	37	55	28	22
Total Breaths	61,662	78,635	92,360	14,795	36,303
PVA Breaths	7,714	4,570	6,924	2,373	7,669
Suction Breaths	750	136	681	350	373
Cough Breaths	229	117	178	56	96

Table 2.3: Descriptive statistics for our dataset for each ventilator mode analyzed. We also analyzed number of patient ventilator asynchrony (PVA), suction, and cough breaths found. While these breaths do not represent normal breathing, they are typical in clinical practice. CPAP - Continuous Positive Airway Pressure, PAV – Proportional Assist Ventilation

Feature	Description
Inspiratory Flow Slope Variance (per breath)	This feature measures the variance of successive, 0.08-second-long slope measurements of the inspiratory flow curve of a single breath. This feature was effective for classifying volume control.
Variance of Pressure (per breath)	This feature takes the variance of all pressure measurements for a single breath. This feature was helpful for classifying CPAP which typically utilizes low pressures relative to PEEP on inspiration.
Variance of Per-Breath Inspiratory Flow Slope	The inspiratory flow slope variance was found on a per breath basis, and this feature takes the variance of the inspiratory flow slope variance across a 10-breath window.
Flow-Based I-time Variance (10 breath window)	The amount of time that a patient inhales for a single breath is called the I-time. This feature calculated the variance of 10 successive breaths.
Pressure-Based I-time Variance (10 breath window)	We defined pressure-based I-time as the amount of time (seconds) that pressure is elevated by [0.4 * (PIP - PEEP)] above PEEP. This was an important variable to measure in pressure control and pressure support, where flow-based I-time can be shorter than the ventilator's set I-time, which may occur in delayed cycling asynchrony. The variance of this statistic was taken over a 10-breath window.
N Plateau Pressures (20 breath window)	A plateau pressure is taken on a ventilator when inspiratory flow is set to 0 for a certain amount of time, during which a ventilator can read the residual pressure in the respiratory system. PAV will repetitively take plateau pressures to adjust ventilation to a patient's needs.

Pressure-Based I-time Variance (100 breath window)	In this feature, the pressure-based I-time statistic is also calculated for a 100-breath window. This feature was necessary to provide capacity for differentiating between pressure control and pressure support in synchronously breathing patients.

Table 2.4: The set of proposed features for our model. Features were segmented into per-breath and multi-breath time frames. I-time - inspiratory time. PEEP - Positive end expiratory pressure. PIP - peak inspiratory pressure. CPAP - Continuous Positive Airway Pressure. PAV - Proportional Assist Ventilation.

Our feature set is composed of 7 items of expert-guided information derived from raw VWD and is described in Table 2.4. Our features are derived from both per breath and multi-breath analytic time frames. Per-breath time frames occur over a single breath, while multi-breath time frames are composed of windows of short, medium, and long periods of breathing. The short window is 10 breaths long, the medium window is 20 breaths, and the long window is 100 breaths. Tuning of features and hyperparameters was guided by performing 10 K-fold validation of our training data. After tuning model hyperparameters during the validation phase, we evaluated our model on our testing set. No additional changes to our feature set, or model hyperparameters was performed after model development was completed in the training set. Model performance is primarily reported through F1-score because it is more representative of class-imbalanced classifier performance than accuracy is. F1-score is calculated as the harmonic mean of precision (PPV) and recall (sensitivity):

$$F1\ score = 2\frac{precision * recall}{precision + recall}$$

A limitation to using RF to classify ventilator mode is the RF classifier assumes that all breaths are independent of each other. However, ventilator mode is a continuous setting that does not vary over time, unless it is manually changed by the provider. Therefore, one breath's mode is often predictive of the next breath's mode. This modeling incongruity causes the RF classifier to sometimes perform an incorrect VM classification even in periods where the classifier correctly

predicts the correct VM for majority of breaths. To smooth these incorrect predictions, we implement an algorithm we term "look-ahead smoothing" which operates as a second pass heuristic on all per breath RF breath predictions. More specifically, once the RF is finished, look-ahead smoothing examines each breath VM classification sequentially, and if it determines a breath's classification is not in accordance with the previous n breaths then it will look ahead at the next n breaths in sequence. The breath will then be re-classified in accordance with the majority x percent of the subsequent n breaths. Both n and x are configurable parameters that we set at $n = 30$ and $x = 60$, parameters which were found via sensitivity analysis. In real-time classification, assuming an average respiratory rate of 20 breaths per minute, this technique results in a latency of at most 1.5 minutes between a given breath and the availability of its final classification.

Finally, we implemented an experiment to test how well our classifier would generalize to a larger dataset if random breaths in our training dataset were missing due to some technical error. So, we conduct an experiment where we ablate (i.e., remove) data observations at random from our training dataset in equal proportion for VC, PC, PS, CPAP, and PAV. We do not perform any ablation on the testing set. We then report results of this experiment by recording F1-score for each class with respect to the percentage of the dataset that simulated as missing.

2.4.2 Results

Using a RF model with the feature set defined in Table 2.4, we initially performed 10-fold cross validation with our training set to test performance of our VM classifier. We found that during cross validation our model consistently performed within 98-99% for F1-score, recall, and specificity for all VMs. We then evaluated our model on the withheld test set and only received

slightly worse performance from our model, with CPAP suffering the largest drop in performance, and VC/PAV suffering no drop in performance (Table 2.5).

Mode	F1-Score	Accuracy	Precision	Recall	Specificity
VC	0.999	1.0	0.999	0.999	0.999
PC	0.986	0.991	0.986	0.987	0.993
PS	0.985	0.989	0.989	0.981	0.993
CPAP	0.955	0.997	0.937	0.974	0.998
PAV	0.994	0.999	0.990	0.998	0.998

Table 2.5: Performance of our Random Forest model when applied to our withheld testing set.

We hypothesized that since the model performed well on both training and testing sets that it would also be robust to scenarios in which breath data went missing due to reason of sensor or software failure. We report results for this experiment in Figure 2.4. We found the model is robust to missing data until approximately 90% of data is removed. After this point PC and PS F1-score performance begins to decrease and other classifications begin to fluctuate. After 99% of data is removed our classifications lose clinical utility.

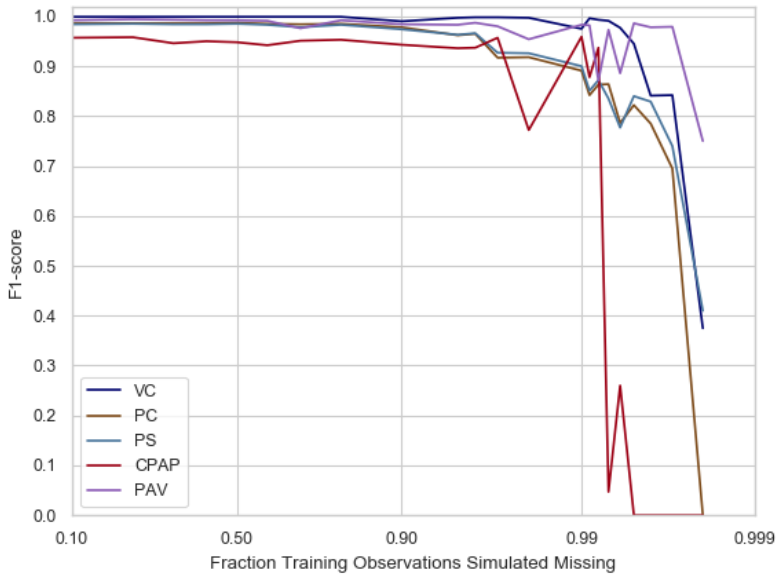


Figure 2.4: Here we simulate the scenario where a percentage of training observations is missing due to software/hardware error.

Given the results of the random ablation experiment we hypothesized that we may have created too large a training set. To reduce the size of our training set in a generalizable, non-random way, we hypothesized we only needed to keep the first of a certain number of contiguous breaths from each VM per data file, and still maintain the performance of our original model. In this respect, we could make recommendations to physicians to only annotate the first m breaths in a series and just leave the rest alone. This could also decrease the amount of time necessary to annotate VM on future patients. So, we performed a sensitivity analysis to determine the optimal number of contiguous observations to keep per ventilator mode. Our analysis (Figure 2.5) showed that it was most optimal to only use the first 450 VC observations, the first 1,000 PC and PS observations, and the first 70 CPAP, and 300 PAV observations in a file. Our investigation also revealed that PC and PS do worse when only utilizing short periods of contiguous observations. Using this methodology, we ablated the overall number of training observations by 64.73% from 140,928 to 49,702 observations, while still maintaining generalizability of our training set to our withheld test set, and slightly improved CPAP performance (Table 2.6).

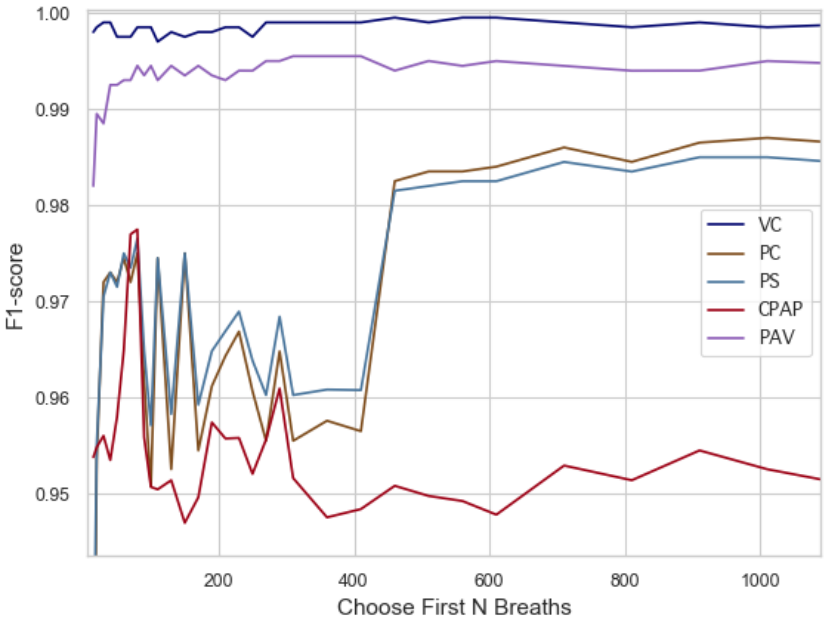


Figure 2.5: Results from our sensitivity analysis for choosing the first N contiguous breaths for a given mode in a data file.

Mode	Training Observations	F1-Score
VC	6,079 (-83.70%)	0.999 (+0)
PC	14,775 (-50.39%)	0.988 (+0.002)
PS	21,453 (-33.25%)	0.988 (+0.003)
CPAP	1,330 (-86.75%)	0.975 (+0.02)
PAV	4,173 (-77.63%)	0.994 (+0.001)

Table 2.6: Results of our ablation experiment where we only keep the first 450 VC observations, the first 1,000 PC and PS observations, and the first 70 CPAP, and 300 PAV observations in a data file. We note the final number of training observations that we kept, and report how much of a reduction that contrasted with the original training set. Performance improvements over results listed in Table 2.5 are bracketed alongside final performance metrics. E.g., a performance increase of 2.0% is denoted as (+.02).

2.4.3 Discussion

In this chapter, we highlighted how we created a dataset of 308,957 breaths annotated for VM on a per-breath basis and how we developed a highly accurate, ML-based VM classification model that only utilizes raw VWD to perform classifications. Our VM classifier was highly performant in detecting five of the most widely used VMs in the USA, even in the presence of signal noise, episodes of PVA, and routine clinical events such as cough and suction.(83) Using our approach, we were able to achieve methodological and performance improvements in VM classification compared to the current state of the art.(76) In this regard, Murias reported 89% accuracy at detecting per-hour VM classification,(76) and we report average accuracy of 99.52% of per-breath VM classification. Finally, we examined how robust our model is to the presence of missing training data, and additional experimental results that suggested how we can decrease the size of our dataset while still maintaining generalizability of our classifier.

We took multiple measures to ensure we were not overtraining our classifier. First, we utilized a relatively large and diverse sampling of patients to create both our testing and training sets. This created one of the largest available datasets of per-breath labeled VWD. 2-4 hour epochs were chosen at random from each of these patients. Our testing set included almost as many patients as our training set and was composed of more breaths than our training set. There was no overlap of patients between training and testing sets. Finally, all model features and hyperparameters were evaluated on the training set using K-fold validation and were frozen after initial evaluation of our testing set.

Our ablation experiments deserve additional consideration. The results of random ablation experiment highlight multiple things: 1) RF is extremely performant with our

featurization approach and is also able to perform VM classification with small amounts of data. 2) Our ablation results also illustrate that it may not be necessary to create very large training datasets of information to create performant ML classifiers for VM. 3) Our size reduction experiment surprisingly showed that the first 70 breaths seem to be most representative of CPAP breathing patterns. We hypothesize this can be explained by the fact that some patients tire quickly when on CPAP, and thus their breathing can become more irregular. In this case, later breaths in CPAP sequences may more closely resemble asynchronous breathing from other ventilator modes instead of CPAP.

This work had limitations where it was confined to a single academic medical center and single ventilator type. Classifier performance may change when introduced to a multi-center or multi-ventilator dataset. There are also additional ventilator modes such as PRVC that we were unable to add to our model due to their paucity of use at UCDMC. We welcome additional collaboration and inclusion of multi-center data and have publicly released our code and dataset.

Chapter 3: Clinical Validation of Algorithms for Estimating Static Compliance

3.1 MATERIAL AND METHODS

This is a retrospective study that was conducted as part of an institutional review board (IRB)-approved research effort collecting ventilator waveform data (VWD) from mechanically ventilated adults at UC Davis Medical Center (UCDMC). Enrolled subjects were considered for analysis if a clinically valid inspiratory pause maneuver(88) was performed. All subjects were ventilated in the following assist control VMs: VC, PC, or PRVC. Subjects had RASS scores ranging from -1 to -5. Subjects were excluded from analysis if they were under 18 years of age, were pregnant or a prisoner, or if all plateau pressure checks were determined invalid.

We used a previously published methodology to collect VWD(122). VWD consists of raw pressure (cm H_2O) and airflow (L/min) observations sampled at 50 Hz. Our dataset is split into two parts: first we collected VWD using a mechanical lung (QuickLung^{TM}) over the course of 8 different experiments, and 8,787 breaths. We used this data to validate all reproduced algorithm code. Next, we collected a dataset of 39,660 breaths from 18 subjects hospitalized at UCDMC between 2015 and 2020. Ground-truth C_{rs} was found for each breath through analysis of end-inspiratory pause maneuvers that were performed during routine course of care(47,144). All inspiratory pause maneuvers were validated by three clinicians (JN, CG, MB) by either performing the procedure at bedside, or through retrospective review of VWD. VMs and RASS

scores were collected via review of both electronic medical record (EMR) and collected VWD(145). All VWD was processed through rule-based algorithms to determine presence of the following subtypes of PVA: double trigger asynchrony, breath stacking asynchrony, flow asynchrony (FA), and delayed cycling asynchrony (DCA)(54). Flow asynchrony is further graded by severity in terms of the level of inspiratory pressure deflection into mild, moderate, and severe grades of asynchrony.

We replicated 15 different computer algorithms used for non-invasively estimating C_{rs} using the single-chamber model of the lung(146,147). The single-chamber model is expressed by:

$$P_{aw}(t) = R_{aw}Q(t) + \frac{V(t)}{C_{rs}} + P_0,$$

where $P_{aw}(t)$ is the airway pressure at a given time t, R_{aw} is the patient's airway resistance, $Q(t)$ is airflow at time t, $V(t)$ is the volume of air in the lung at time t, C_{rs} is the respiratory compliance, and P_0 is the positive end expiratory pressure (PEEP). Although analytic methods exist(96,98), this equation is usually solved by using least squares regression to calculate for C_{rs} and R_{aw} terms using measured $Q(t)$ and $V(t)$ data from either the inspiratory or expiratory limbs of a breath(45,89,104). This method can non-invasively measure lung physiology and can accurately estimate C_{rs} when patients are exerting no muscular effort in breathing.

The single chamber model, however, fails to account for muscular effort during spontaneous breathing. Furthermore, the performance of C_{rs} estimators during spontaneous breathing has received scant validation(92). So, in this chapter, we analyze a comprehensive list of C_{rs} estimation methods including: Al-Rawas method (98,148), flow-targeted inspiratory least squares(89), Howe's least-squares(104), IIPR(102), IIMIPR(149), IIPREDATOR(92),

Kannangara's method(89), Major's method(105), MIPR(149), polynomial method(150), PREDATOR(103), pressure-targeted expiratory least-squares(47), pressure-targeted inspiratory least squares(146,147), constrained optimization(97), and Vicario's non-invasive estimation of alveolar pressure(96). Many of these methods are mode-restricted where they are designed to operate in either VC or PC/PRVC modes (Table 3.1). So, in our VM experiments, we only evaluated algorithms that were designed to work for a specific mode. For instance, in VC mode experiments, we only evaluated VC-specific algorithms, and mode-agnostic methods. Other algorithms, such as Howe's least squares(104) should theoretically work in any VM, but are unvalidated in VC.

VC Algorithms	**PC/PRVC Algorithms**	**Mode-Agnostic Algorithms**	**Mode-Agnostic Algorithms but Unvalidated in VC**
IIPR(102)	Flow-targeted inspiratory least squares(89) (Figure 3.1C)	Al-Rawas method(98,148)	Howe's least-squares(104)
IIMIPR(149)	Kannangara's method(89) (Appendix Figure B.8)	Pressure-targeted expiratory least-squares(47,104) (Figure 3.1B)	Vicario's non-invasive estimation of alveolar pressure(96)
IIPREDATOR(92)		Constrained optimization(92,97)	
Major's method(105)			
MIPR(149)			
Polynomial method(150)			
PREDATOR(103)			
Pressure-targeted inspiratory least squares(47) (Figure 3.1A)			

Table 3.1: Enumerates list of mode-specific, and mode-agnostic algorithms we analyze in this work.

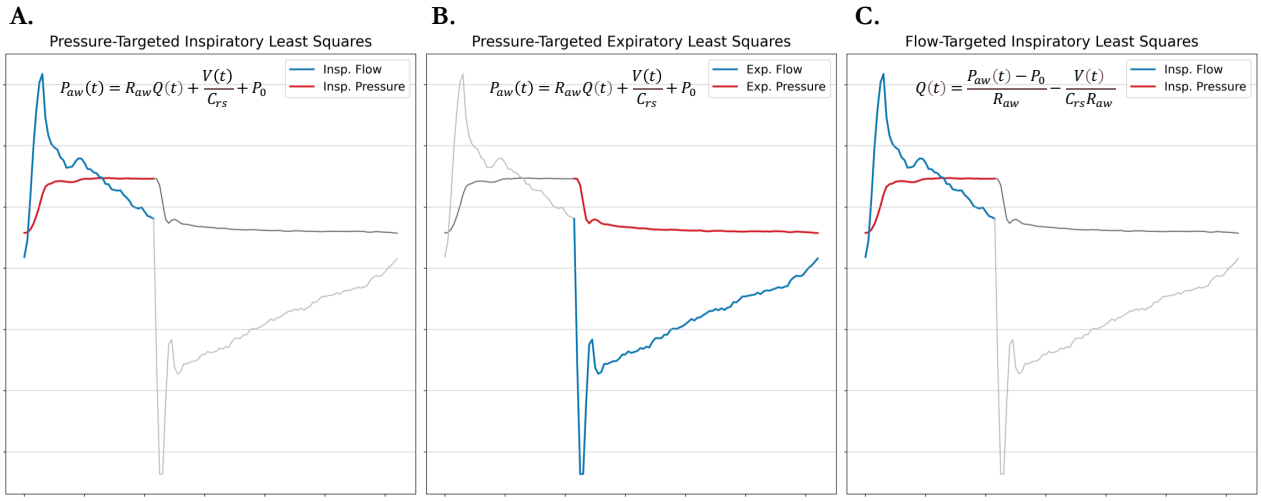


Figure 3.1: Shows different least squares regression methods and what kinds of data they use from different parts of the breath. Also shows which formulation of the single-chamber model is used in calculation.

We define respiratory compliance C_{rs}^k as the measured compliance of breath k, $1 \geq k \geq N$ given N breaths in any given patient sample. We measure C_{rs}^k by performing an inspiratory pause maneuver to find plateau pressure, and then we derive C_{rs}^k from the plateau pressure (P_{plat}^k) (88) using the equation:

$$C_{rs}^k = \frac{V^k}{P_{plat}^k - P_0}$$

where V^k is the inhaled volume of air for breath k. We make the assumption that C_{rs} does not change significantly in a short period of time (92), so we assume that all breaths within 30 minutes of the time of finding P_{plat}^k have equivalent C_{rs}^k. If more than one plateau pressure was measured within 30 minutes of a breath, then breath C_{rs}^k was averaged across the multiple observations.

After our gold-standard compliance C_{rs}^k for each breath is found, we can compute the absolute difference (AD^k), and standard difference (D^k), between C_{rs}^k and the estimated compliance $\hat{C}_{rs}^k$ for each breath from a chosen algorithm.(92,104,105)

$$AD^k = |C_{rs}^k - \hat{C}_{rs}^k|$$

We can then find various statistics, such as median, mean, and median absolute deviation (MAD) (92,104), for all AD^k, $1 \geq k \geq N$ for any given patient. This allows us to determine performance of any algorithm for a given patient. Per-breath AD can be highly variable however, due to periods of PVA and transient abnormality, and may not always be clinically usable by clinicians. To ameliorate this issue, we propose to measure algorithmic performance by using windowing for a given window size s, and then smooth $\hat{C}_{rs}^k$ using the median of $\hat{C}_{rs}^k$ and $s - 1$ prior samples. We define this metric as windowed difference (WD):

$$WD^k = |C_{rs}^k - median(\{\hat{C}_{rs}^{k-s}, \dots, \hat{C}_{rs}^k\})|$$

WD can be calculated using a series of overlapping (i.e. – rolling), or non-overlapping (i.e. – sequential) windows. For this chapter, we use overlapping windows with $s = 100$.

We use the metrics defined above to examine how the algorithms perform across all patients in our dataset. For this we compute the median AD for each patient in our dataset, per algorithm. Then we compute the mean of each patient median across all patients to generate a mean AD across all patients. Mean MAD is computed in a similar way. The MAD of estimated compliance is found per patient, and then the mean MAD is computed across all patients per algorithm. These two computations are motivated by our desire to have an algorithm that on average estimates patient C_{rs} as accurately as possible, while ensuring that C_{rs} estimates are consistent.

There are also algorithm statistics that are best reported on a per-breath basis. For per-breath performance we collect all standard differences D^k between estimated and true C_{rs} across all patients and then plot results using a box and whiskers plot. This allows us to examine

algorithm performance in specific circumstances, such as during asynchronous breathing, when utilizing per-patient results not representative of patient-level clinical realities.

3.2 RESULTS

First, we compute the mean MAD of the best performing algorithms for VC mode and PC/PRVC modes (Figure 3.2, Figure 3.3) on a per-patient basis. We then perform a similar analysis across all VMs using only mode-agnostic algorithms (Appendix Figure B.2). Our results show that most mode-agnostic algorithms underperform mode-specific algorithms. The best mode agnostic algorithm is Vicario's non-invasive estimation of alveolar pressure. However, this algorithm is also highly variable and has the largest mean MAD compared to other algorithms.

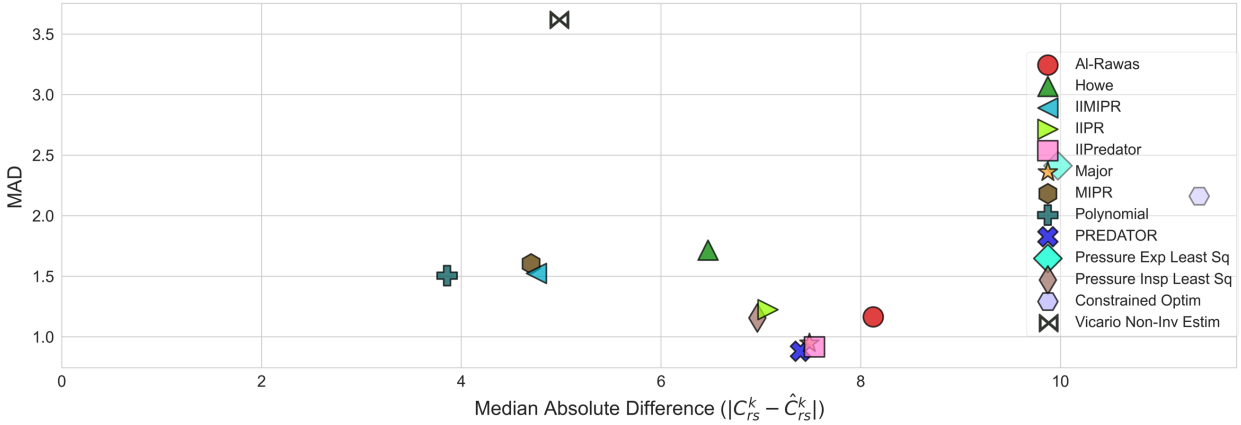


Figure 3.2: Displays which algorithm performs best on a per-patient basis for all volume control (VC) breaths. X-axis is found by computing the mean, median absolute difference between gold standard compliance and algorithm estimated compliance on a per-patient basis. Y-axis shows the mean median absolute deviation of all compliance estimates per patient. Note that VC specific algorithms and mode-agnostic algorithms are included in this analysis.

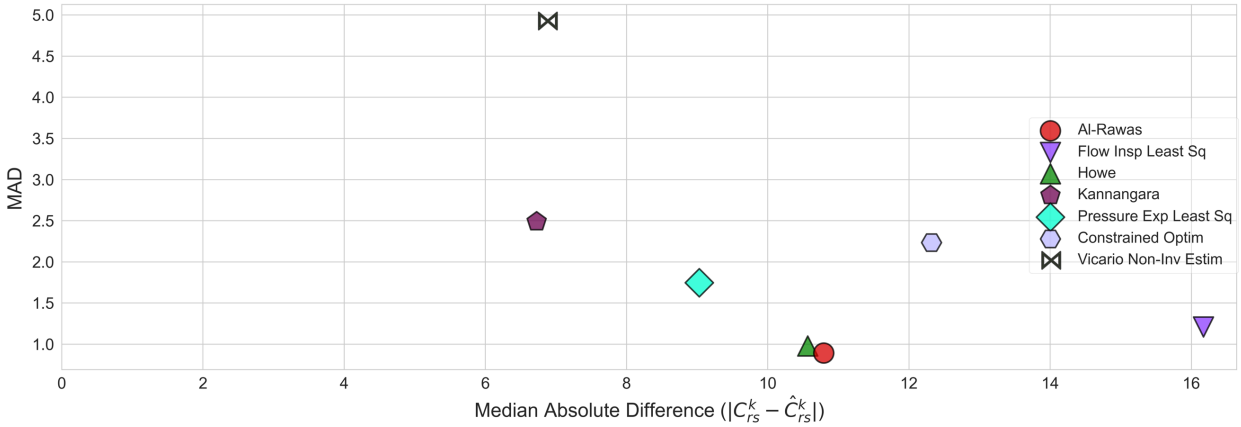


Figure 3.3: Displays which algorithm performs best on a per-patient basis for all pressure control (PC) and pressure regulated volume control (PRVC) breaths. Note that PC/PRVC specific algorithms and mode-agnostic algorithms are included in this analysis.

We also measure algorithmic performance under different PVA scenarios across VMs *using per-breath analysis*. Here we use boxplots to describe the distribution, median, and IQR of standard differences between estimated and true C_{rs}. We do not use absolute difference for boxplots because standard difference allows us to better see distribution of errors around 0. Figure 3.4 shows the performance of VC-capable algorithms during asynchronous breathing excluding mild FA. We exclude mild FA because we found almost all algorithms are not negatively affected by it, and the preponderance of mild-FA in our dataset dampens the effect of other PVA types on our results (Appendix Figure B.3 and Appendix Figure B.4). Our results in Figure 3.4 further highlight our findings from Figure 3.2 that VC VWD is best analyzed by VC specific algorithms, and that mode generic algorithms, with exception of Al-Rawas method, are less performant on VC breathing. Al-Rawas itself has a mechanism for filtering breathing that does not have an ideal exhalation curve, which be part of the reason why Al-Rawas performs well with asynchronous breathing in VC.

In Figure 3.5, we investigate the effect of moderate to severe FA on VC performance. This investigation is based on the hypothesis that VC algorithms work best in this scenario because they can correct for inherent nature of FA. We find that this hypothesis is partially correct, and that methods used to correct for FA, like IIPREDATOR, perform well in FA. Our analysis also finds that algorithms using expiratory data, instead of inspiratory data, like Howe's expiratory least squares are best at estimating C_{rs} during moderate-severe FA. This may be because the expiratory limb is less affected than the inspiratory limb during FA.

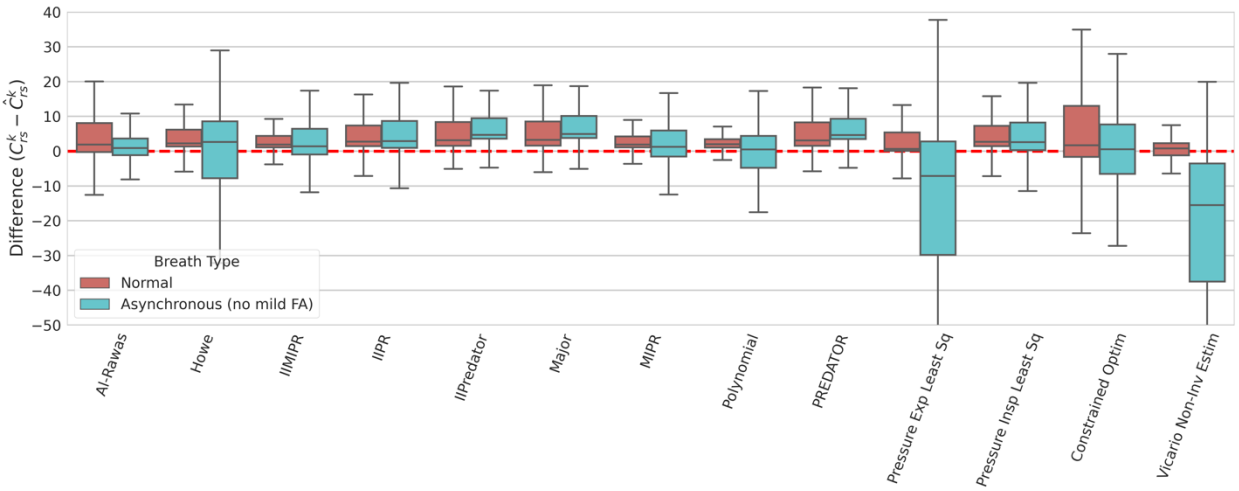


Figure 3.4: Comparison of algorithm performance between non-asynchronous and asynchronous breathing (without mild flow asynchrony) in volume control (VC) mode.

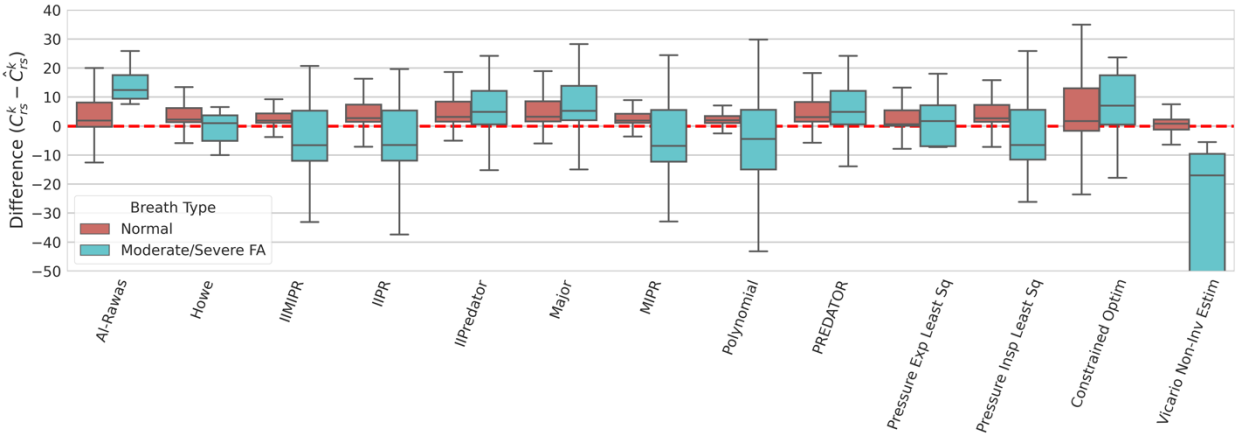


Figure 3.5: Comparison of algorithm performance between non-asynchronous and moderate/severe flow asynchrony (FA) breathing in volume control mode. We cut off results of Vicario's non-invasive estimation of alveolar pressure because the IQR would disrupt visualization of results.

In Figure 3.6 we show investigation of the prior methodology comparing normal and asynchronous breathing but applied to PC/PRVC only. We see that almost all algorithms are relatively unaffected by PVA. However, like VC, this is because milder effects from DCA mutes the effects of more severe asynchronies (Appendix Figure B.5). When we remove DCA we find all algorithms perform significantly worse at estimating C_{rs} than during normal breathing (Figure 3.7). We also find the Kannangara method tends to yield median differences closest to 0 of all algorithms surveyed. This is not surprising because the Kannangara had the best per-patient results for PC/PRVC breaths (Figure 3.3) and it was designed to correct for PVA.

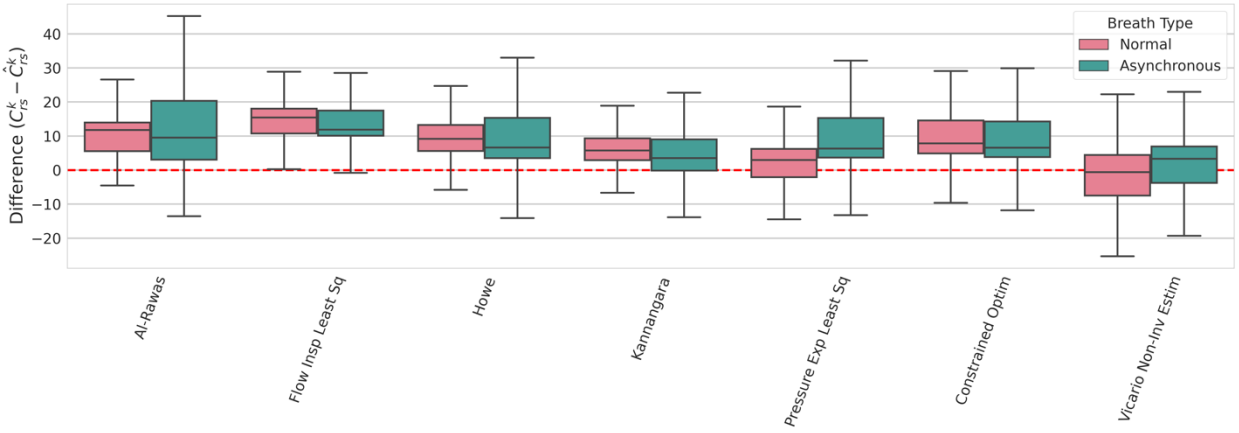


Figure 3.6: Boxplot comparison of algorithm performance between normal and asynchronous breathing in pressure modes (pressure control / pressure-regulated volume control).

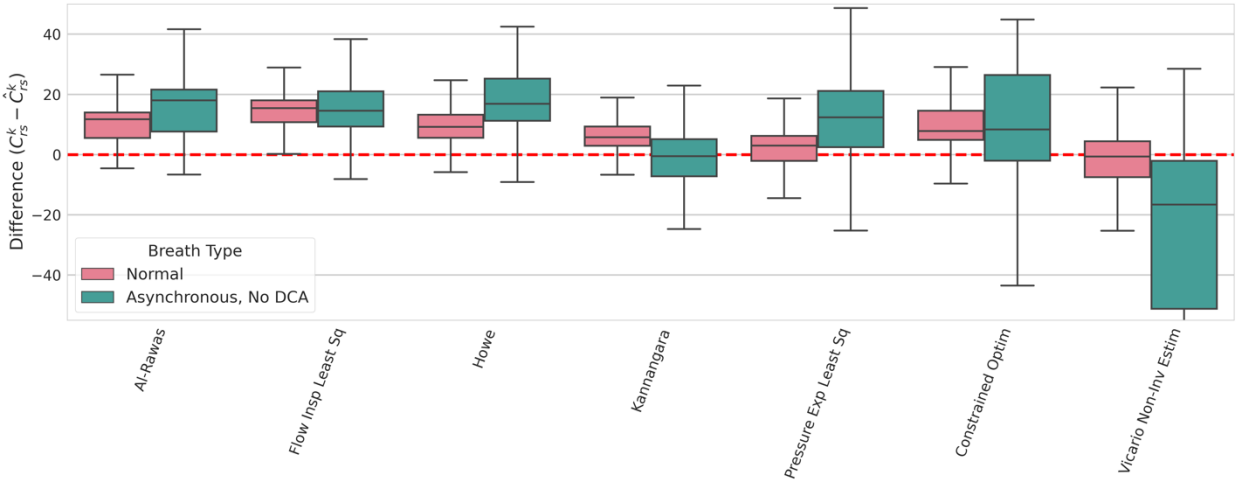


Figure 3.7: Boxplot comparison of algorithm performance between normal and asynchronous breathing without delayed cycling asynchrony (DCA) in pressure modes.

We note that prior results (Figure 3.4-Figure 3.7) show per-breath IQR increases significantly during PVA. This finding has motivated development of algorithms that are resistant to PVA(89,102,103,150), and has motivated our hypothesis that windowed difference would function to improve algorithm resistance to PVA and other transient anomalies in breathing. To investigate, we first analyze results of windowing on a per-patient basis. We found that for all algorithms, windowing did not yield any significant change in AD (Figure 3.8A), however, windowing does improve per-patient algorithm MAD for a majority of surveyed

algorithms (Figure 3.8B). We also investigate effect of windowing on asynchronous breathing. Like previously, we only analyze asynchronous breaths, and then compare the estimated C_{rs} of our algorithms without windowing and with rolling median windowing (Figure 3.9). Our findings show that windowing can reduce impact of asynchronous breathing by utilizing prior synchronous breaths to smooth estimated C_{rs}. Combined with results of (Figure 3.8), we see windowing is most effectively used on a per-breath basis during asynchronous breathing. This may mean that windowing is best used during real-time estimation of C_{rs} in clinical practice.

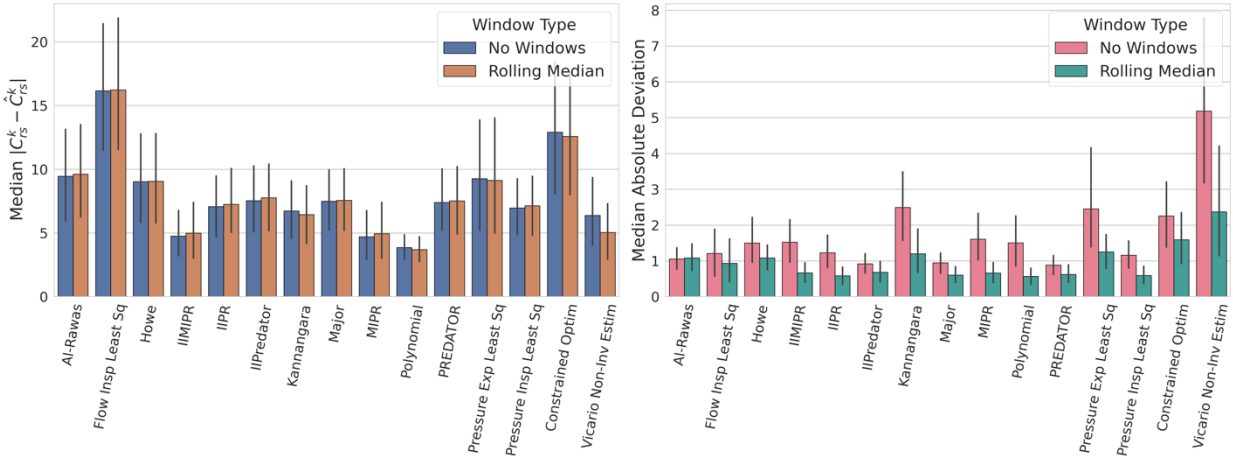


Figure 3.8: **A.** Shows the effect of windowing on absolute difference (AD) between estimated and true compliance. We were unable to find an algorithm that was statistically improved using windowed difference. **B.** Shows effect of windowing on median absolute deviation. Note: 95% CI is shown in bar plot whiskers

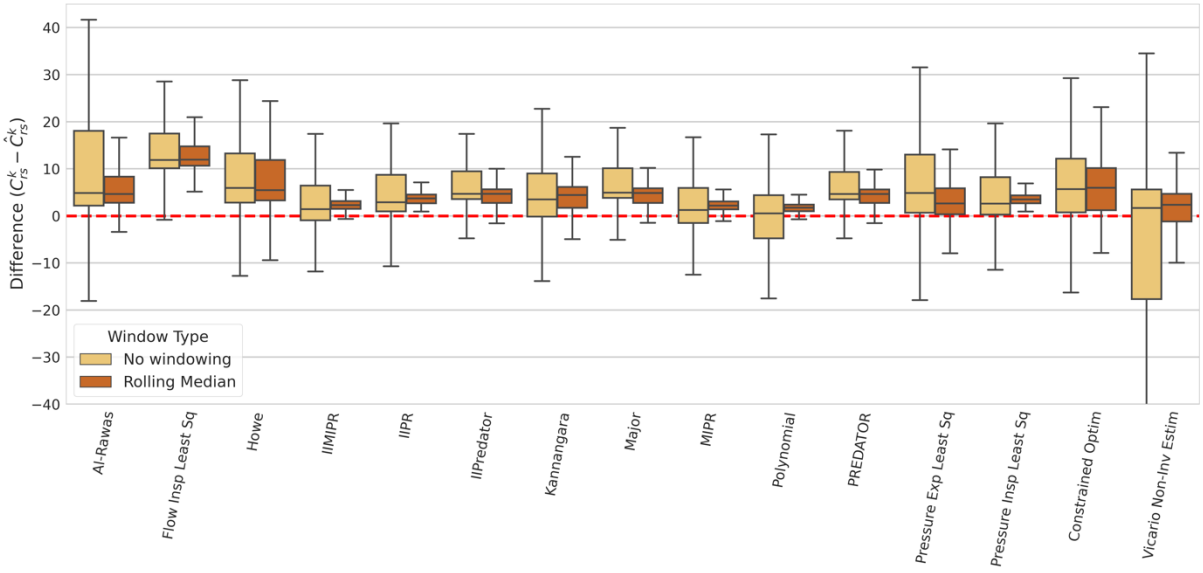


Figure 3.9: Shows the effect of no windowing (yellow) and rolling median windowing (orange) when examined at breath level.

We also hypothesized there may be certain characteristics of mode-specific algorithms (Table 3.1) that translate to modes outside those they were developed in. So, we investigate the utility of analyzing VM-specific algorithms in modes they were not tested for. In our first investigation, we compute the windowed, per-patient results for all PC/PRVC breaths with VC mode-specific algorithms and compare them to baseline results derived our 2 PC/PRVC-specific methods (Table 3.1, Figure 3.10). Our findings show us that the polynomial method, MIPR, and IIMIPR outperform both PC/PRVC-specific algorithms in per-patient median AD, and the flow-targeted inspiratory least squares method dramatically underperforms pressure-targeted least squares. Our results imply that even though VC algorithms are clearly tailored for specific aspects of VC, that they possess some generic analytic properties that translate to PC/PRVC. Furthermore, the fact that flow-targeted inspiratory least squares dramatically underperforms pressure-targeted least squares is surprising, given that flow-targeted inspiratory least squares was mathematically formulated for PC/PRVC(89). This shows that rearranging the single-

chamber model to account for a specific VM can yield worse results than using the standard formulation.

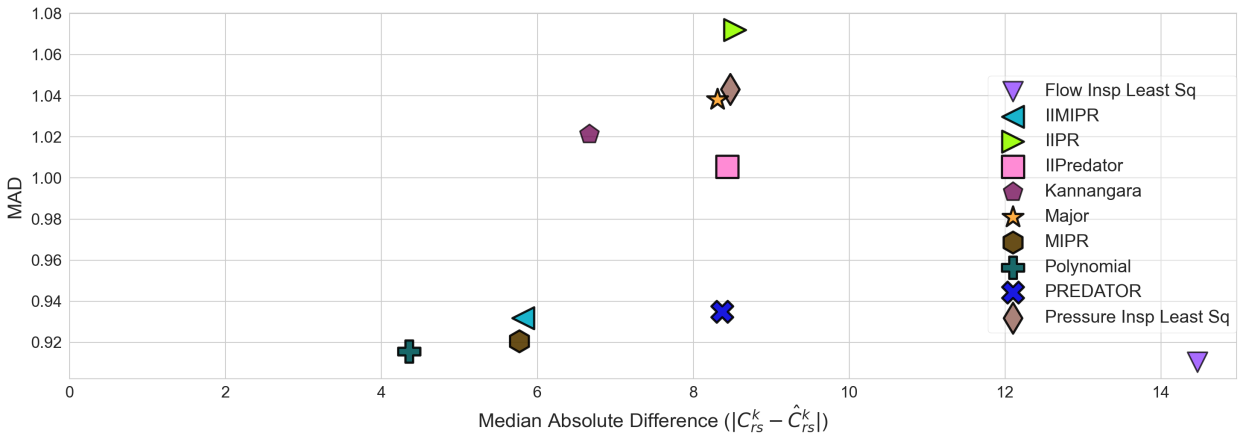


Figure 3.10: Shows the per-patient, windowed results of the VC specific algorithms applied to PC/PRVC. For comparison, we provide baseline calculations with PC/PRVC-specific algorithms.

We perform similar analysis on VC breaths. We analyze all VC breaths with our 2 PC/PRVC specific methods and provide baseline comparison using VC-specific algorithms. Unlike previously, none of the PC/PRVC specific methods outperform the best VC methods, but Kannangara's method can achieve relatively close performance to the best-performing VC algorithms. This experiment gives further evidence that there is generic utility to Kannangara's method even though its correction of flow will not function in VC-related asynchronies.

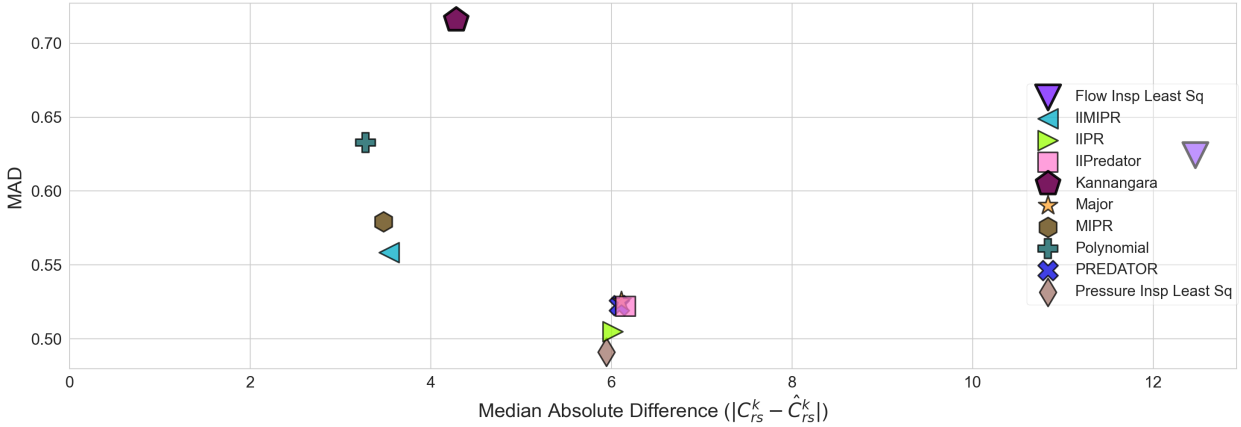


Figure 3.11: We find windowed, per-patient results for PC/PRVC algorithms when applied to VC. For comparison, we provide baseline calculations with VC-specific algorithms.

3.3 DISCUSSION

In this chapter, we have described our efforts to benchmark a comprehensive list of existing algorithms used to estimate C_{rs} using the single-chamber model of a human lung. Our work showcases that the Polynomial method (150) works best in our patient population across all surveyed VMs (Figure 3.2, Figure 3.10). We also show that utilizing a median windowing strategy can improve overall consistency of algorithm results and will function best when clinicians desire real time calculations for C_{rs}. Finally, we showed that all algorithms surveyed maintained good performance when applied to VMs outside those they were derived for. This result stands in contrast to existing assumptions that 1) mode-specific algorithms will only work in the VM they were developed for; and 2) the single chamber model needs to be rearranged to accommodate varying VMs (89). In addition to these findings, we have generated the largest available database of open ventilation data that can be used for benchmarking single-chamber algorithms. We hope that this contribution will speed up the development of future C_{rs} estimators.

Our study provides the most comprehensive validation of available single-chamber algorithms along varying clinical scenarios, with the largest dataset of patients and breaths to date. Our study design differs from prior validation efforts by Redmond et al. where the authors analyze 6 algorithms with a dataset of 4 patients, where each patient receives sedation during monitoring(92). Estimated C_{rs} is then measured for 30 breaths before sedation, and 30 breaths after sedation, for a total of 240 breaths (92). Our dataset was gathered over long periods of regular care during which time providers performed routine monitoring of C_{rs} for a total of

39,660 breaths (Appendix Table B.1). These differences in dataset size, study design, and the greater percentage of synchronous breathing in our dataset may explain some difference in results between ours and Redmond's study. For instance, when measuring C_{rs}, Redmond found that IIPREDATOR performed best when correcting for VC PVA. We note that IIPREDATOR was among the best performing algorithms in our study as well (Figure 3.5). However Redmond et al. did not evaluate expiratory pressure methods(47,104) in their work. Our findings show using expiratory pressure methods(47,104) function better than inspiratory pressure-targeted methods like IIPREDATOR when estimating C_{rs} during moderate-severe FA. We also provide more granular discrimination into different types of PVA than Redmond(92). Our work shows that all algorithms perform surprisingly well in periods of mild PVA such as during mild FA and DCA (Appendix Figure B.4, Appendix Figure B.5). It is only during more severe types of PVA that algorithms return variable C_{rs} estimates. Redmond et al. did not perform this level of discrimination and treated all breaths in the period after sedation wore off as asynchronous(92). Given the results of Redmond et al. and this study (Figure 3.5), it is possible that all subjects in Redmond et al. experienced moderate-severe FA after their sedation wore off(92).

Our results using VC algorithms in PC/PRVC, and vice-versa, requires elaboration. Algorithms that correct for PVA in VC like IIPR, MIPR, and the polynomial method will not function to correct PVA in PC/PRVC(102,150), nor can Kannangara's method correct PVA in VC(89). Our findings, however, show there exist generic computational methods in these algorithms that allow them to perform well under VMs different than what they were designed for. For instance, Kannangara's implementation of their algorithm only utilizes a small portion of inspiratory data between two of the inspiratory "shoulders" to calculate C_{rs}(Appendix Figure B.8)(89). Empirical evidence shows this implementation is significantly more effective than

using the entire inspiratory portion of the breath that is done in flow-targeted inspiratory least squares(89). Likewise, the way the polynomial method is calculating C_{rs} may be yielding overall benefits that are mode-generic as well, however it is beyond the scope of this work to investigate the specific mechanism at play(150). Furthermore, by comparing between results of pressure-targeted inspiratory least squares, and flow-targeted inspiratory least squares we find that rearranging the single-chamber model by VM does not yield improved C_{rs} estimates (Figure 3.10). First, it is unclear if there is actual mathematic rationale for the rearrangement because VM does not change how lungs operate, and by proxy the single-chamber model, VM just changes how the ventilator delivers a breath. Second, rearranging the single-chamber model doesn't allow isolation of R_{aw} and C_{rs} components in the single-chamber model(89,104). This may contribute to the larger estimation errors seen in the flow-targeted inspiratory least squares method.

Although we have been able to provide further levels of clinical validation for the single-chamber model in different clinical scenarios our study also has limitations. First, our work is an observational, single-center study that did not include any study-specific alterations in patient care. For a vast majority of patients, we were also unable to capture valid plateau pressures above RASS of -3 when patients were truly spontaneously breathing. So, our results should only be viewed as applicable for patients with RASS≤ -3. We also experienced large confidence intervals in our estimation of windowing effects due to having a relatively small patient cohort of 18 subjects. Furthermore we are unable to make any claims about which C_{rs} estimation algorithms work best under varying pathophysiologic states(47,95). Future work can add new open-source patient data to our published dataset so that the community can further improve

confidence intervals for algorithm results and understand performance of these algorithms under varying lung pathophysiology at different clinical centers.

Chapter 4: ARDS Detection with Classical Machine Learning Models

We hypothesized that a ML model, using only physiologic information extracted from raw VWD, would be able to discriminate between patients with and without ARDS at early time points in MV without the need for CXR, ABG, or other electronic health record (EHR)-derived data. In this chapter, we explore this hypothesis along with several other surrounding experimental questions.

4.1 Methods

All patient data were obtained as part of a prospective, Institutional Review Board approved study collecting raw VWD from mechanically ventilated adults admitted to the medical ICU at UC Davis Medical Center. Three clinicians (JYA, ICP, BTK) performed retrospective chart review to identify the cause of respiratory failure in subjects from the VWD study cohort enrolled between 2015–2019. Subjects were split into two cohorts: 1) patients with confirmed moderate or severe ARDS diagnosis using Berlin consensus criteria within 7-days of intubation(151); and 2) patients with no suspicion of ARDS during their course of MV to avoid phenotypic ambiguity. Patients with chronic obstructive pulmonary disease and/or asthma were excluded from the ARDS patient cohort. Causes of ARDS and indications for mechanical ventilation in the non-ARDS cohort are shown in Table 4.1. Both cohorts required at least 1h of VWD collected in the first 24h after ARDS criteria were first met, or after the start of MV (Appendix Table C.1 and Appendix Figure C.1). All cases meeting study inclusion criteria were reviewed by two clinicians and disagreements were settled by consensus. No sample size

calculation was conducted for this study however, sample size was guided by the range of cohort sizes in previous studies of VWD analysis (51–54) and to achieve a balanced dataset for ML model development as standard ML algorithms are biased towards the majority class, resulting in a higher mis-classification rate for the minority class.(152)

	ARDS (n=50)	**Non-ARDS (n=50)**
Age (median, IQR)	57 (38-65)	58 (49-67)
Female (%)	13 (26)	23 (46)
BMI (median, IQR)	26.4 (22.3-33.8)	25.9 (22.1-28.7)
Obstructive lung disease (n, %)		
COPD	0 (0)	12 (24)
Asthma	0 (0)	5 (10)
Reason for ICU admission (n, %)		
Acute hypoxemic respiratory failure	24 (48)	-
COPD/asthma exacerbation	-	17 (34)
Sepsis	11 (22)	-
Metabolic encephalopathy/drug overdose	2 (4)	15 (30)
Airway edema/anaphylaxis	-	5 (10)
Stroke	-	4 (8)
Cardiac arrest	9 (18)	3 (6)
Heart Failure	-	2 (4)
Upper gastrointestinal bleeding	-	2 (4)
Trauma/Surgery	3 (6)	2 (4)
Pancreatitis	1 (2)	-
SOFA score (median, IQR)	13 (10-16)	7.5 (5-10)
Days from intubation to Berlin criteria (median, IQR)	0.1 (0.0-0.2)	NA
Median PaO2/FiO2 first 24h (median, IQR)	176 (134-210)	318 (267-423)
Worst PaO2/FiO2 24h (median, IQR)	108 (66-137)	278 (147-385)
ARDS insult type (n,%)		
Pneumonia	18 (36)	-
Aspiration	14 (28)	-

Non-Pulmonary Sepsis	10 (20)	-
Trauma	2 (4)	-
Diffuse Alveolar Hemorrhage	2 (4)	-
Pancreatitis	1 (2)	-
Other	3 (6)	-
Hospital Length of Stay (median, IQR)	13.3 (6.6-25.4)	7.0 (4.2-13-4)
Hospital Mortality (n, %)	24 (48%)	10 (20%)
Ventilator- free days in 28 days (median, IQR)	6.6 (0-23.0)	25.3 (10.6-26.9)

Table 4.1: Clinical characteristics of study subjects.

Feature Name	Units	Description
I-time	seconds	Total inspiratory time
E-time	seconds	Total expiratory time
I:E ratio	N/A	Ratio of I-time divided by E-time
Respiratory rate	breaths/min	Instantaneous respiratory rate, defined as: 60/(I-time+E-time)
PEF to 0	NA	An expiratory time constant surrogate we defined by taking the slope of the expiratory flow from peak expiratory flow to where flow reaches close to 0.
PEF+0.16 to 0	NA	An expiratory time constant surrogate we defined by taking the slope of the expiratory flow from 0.16 seconds after peak expiratory flow to where flow reaches close to 0.
Mean expiratory flow	ml/min	The mean flow observation from the point in time peak expiratory flow (PEF) occurred to the point where the breath terminated and a new one began
Dynamic compliance (C_{dyn})	N/A	This measure is derived via: $C_{dyn} = \frac{TVi}{PIP-PEEP}$ where TVi is the inspiratory tidal volume. PIP is peak inspiratory pressure, and PEEP is positive end expiratory pressure.
Tidal volume ratio	N/A	Ratio of inspiratory tidal volume divided by expiratory tidal volume.

Table 4.2: Features calculated for each breath in the analysis. Extraction code is publicly accessible at GitHub.

We used our ventMAP software platform(54) to extract nine physiological features from raw VWD, representing pressure and flow, sampled at 50 Hz, obtained from Puritan-Bennett™ model 840 ventilators.(122) Features were extracted and aimed to capture relevant respiratory

pathophysiology while avoiding features that might strongly correlate with ARDS management such as tidal volume (TV) or positive end expiratory pressure (PEEP) (Table 4.2). Observations for each subject were derived by taking the median value of each feature across windows of 100 consecutive VWD breaths (approximately 5 minutes), with the 100-breath window size based on empirical sensitivity analysis. We processed all available VWD in the 24h after Berlin criteria were first met and 24 hours after the start of MV for patients with and without ARDS, respectively (Figure 4.2A), but not all patients had 24 hours of data based on variable start times of data acquisition. Each observation feature vector was tagged with a subject identifier and clinical class label of ARDS versus non-ARDS. Observations were excluded if a feature in the observation window met any of following: 1) not a number or an infinite value; 2) more than 50% of expected breaths in a window were missing; or 3) window start time was prior to charted MV start and end times in the EHR.

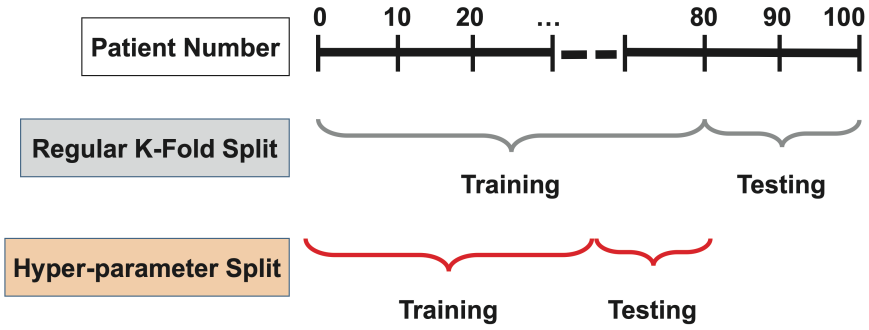


Figure 4.1: General methodology for hyperparameter selection. This methodology follows the k-fold cross validation technique, where we derive our hyperparameters based on a training set, and then split the training set into multiple k-folds itself. Hyperparameters are then chosen based on the optimal parameters in this new hyperparameter split.

Split-Type	Model	N Trees	Max Tree Depth	Split Criterion
K-Fold	Train 24	33	2	Gini
Holdout	Train 24	5	6	Information Gain
Bootstrap	Train 24	33	2	Gini

Table 4.3: Hyperparameters used for our Random Forest algorithm in accordance with our hyperparameter search methodology outlined above. We used the same hyperparameters for our bootstrap experiments as we did in k-fold experiments because the random nature of bootstrapping would have led to an un-fixed series of hyperparameters that changed each time we attempted to evaluate model hyperparameters. Other than the hyperparameters mentioned here, all other arguments were based on Scikit-learn Random Forest default arguments.

We evaluated 7-algorithms using the Python scikit-learn software library (Table 4.7).(139) Despite comparable performance across algorithms, we chose the random forest (RF) algorithm for further model development and testing based on its resistance to overfitting and tolerance to outliers.(143) Given our small sample size, we evaluated model performance using k-fold cross validation (k=5), a 70/30 holdout split, and bootstrapping. For the 5-fold cross validation, 80 subjects were used for training in each of the five k-folds, and 20 were used for validation, with no overlapping data between the training and validation sets in each fold (Figure 4.1). For the 70/30 holdout split, we randomly selected 70 subjects for model training, and withheld 30 for final model validation. For bootstrapping, 100 bootstrapping runs were performed. In each run, 80 patients were randomly selected with replacement for training and the remaining patients were used for testing, with performance averaged over 100 bootstraps. We performed feature selection for each model using chi^2 and Gini selection methods. Model hyperparameters were selected using a Python grid search (Table 4.3).(139) Because feature importance was comparable across both methods, we used sequential feature selection to maximize the area under the receiver operating curve (AUC) with chi^2 for all models (Table 4.4-Table 4.6, Figure 4.3).

Feature	Average P-value	Average Rank
Mean flow from peak expiratory flow (PEF)	1.82e-45	1.0
Respiratory rate	2.25e-33	2.0
PEF to 0	9.88e-28	3.0
PEF+0.16 to 0	3.22e-22	4.0
I-time	2.21e-5	5.0
E-time	3.15e-4	6.0
I:E ratio	8.69e-3	7.0
Dynamic compliance (C_{dyn})	0.382	8.0
Tidal volume ratio	0.626	9.0

Table 4.4: Chi^2-based feature importance based on train 24/test 24-hour (24/24) model dataset. Ranks are based on the average of all k-folds. Each feature achieved a whole number for a rank because there was no variation in feature ranking from fold to fold.

Feature	Average Score	Average Rank
Mean flow from peak expiratory flow (PEF)	0.355	1.0
Respiratory rate	0.2556	2.0
PEF to 0	0.1846	3.0
PEF+0.16 to 0	0.1038	4.0
I-time	0.0508	5.0
E-time	0.0298	6.0
I:E ratio	0.0127	7.0
Dynamic compliance (C_{dyn})	0.0061	8.0
Tidal volume ratio	0.0016	9.0

Table 4.5: Feature importance using Gini importance values based on train 24/test 24-hour (24/24) model dataset. All scores were rounded to 4 significant digits. Ranks are based on the average of all k-folds. Each feature achieved a whole number for a rank because there was no variation in feature ranking from fold to fold. Note that feature rankings here were the same as they were in the chi^2 feature rankings.

Split-Type	Model	Features Used
K-Fold	Train 24 / Test 24 Hours	8
K-Fold	Train 24 / Test 6 Hours	3
Holdout	Train 24 / Test 24 Hours	5
Bootstrap	Train 24 / Test 24 Hours	7

Table 4.6: Number of features selected for each model using the Random Forest algorithm. Feature selection was based on the AUC and accuracy selection method detailed in Figure 4.3.

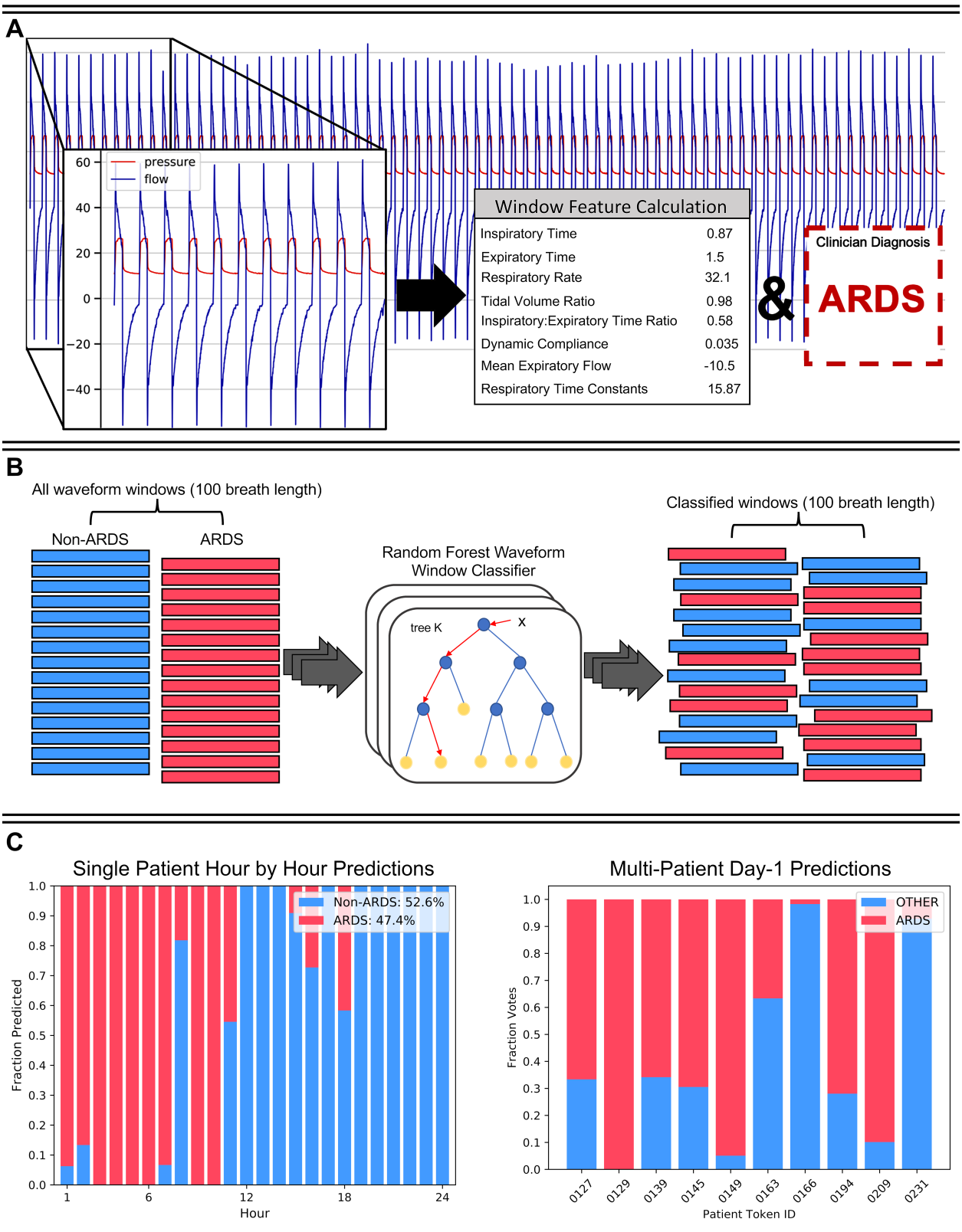

A
pressure
flow
Window Feature Calculation
Inspiratory Time 0.87
Expiratory Time 1.5
Respiratory Rate 32.1
Tidal Volume Ratio 0.98
Inspiratory:Expiratory Time Ratio 0.58
Dynamic Compliance 0.035
Mean Expiratory Flow -10.5
Respiratory Time Constants 15.87
&
Clinician Diagnosis
ARDS
B
All waveform windows (100 breath length)
Non-ARDS
ARDS
Random Forest Waveform Window Classifier
tree K
X
Classified windows (100 breath length)
C
Single Patient Hour by Hour Predictions
Non-ARDS: 52.6%
ARDS: 47.4%
Fraction Predicted
Hour
Multi-Patient Day-1 Predictions
OTHER
ARDS
Fraction Votes
Patient Token ID
0127
0129
0139
0145
0149
0163
0166
0194
0209
0231

Figure 4.2: Visual overview of data processing and classifier model development. A. Ventilator waveform data (VWD) from each subject are divided into consecutive 100-breath observation

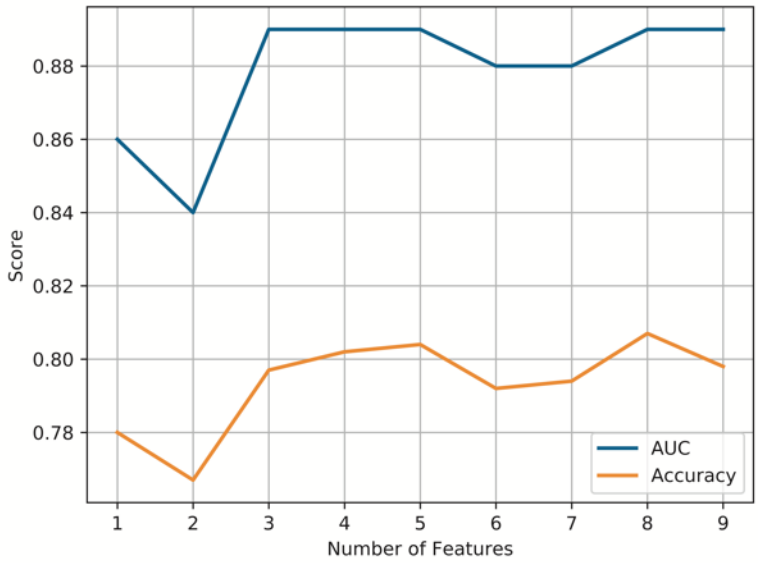


Figure 4.3: Example of how the number of features was selected for each model. The figure shows the change in AUC and patient-level prediction accuracy of the train 24/test 24-hour (24/24) model as a function of the number of features included in the model after chi^2 feature selection. If two sets of features had similar maximal AUC, then the tie was broken by selecting the set with the highest accuracy. *AUC*, area under the curve.

windows. Physiologic features are calculated for each breath in a window, and median values are used to represent the entire window. Each window is labeled as ARDS or non-ARDS and tagged with a subject identifier. B. Feature vectors for each labeled window are fed to a supervised machine learning algorithm for training and evaluation. C. Classified windows are aggregated at the patient level and threshold-based, patient-level predictions can be made based on the percentage of ARDS and non-ARDS windows within any given period (e.g., 24 hours).

ARDS screening ML models were developed using a two-step process. First, we trained the model to classify all individual 100-breath windows from the training set as either ARDS or non-ARDS (Figure 4.2B). We then determined patient-level model performance by attributing all breath window predictions from the validation sets to each subject and assigning the subject class as ARDS or non-ARDS using a specific threshold for the percentage of individual windows classified as ARDS in any given time bin (Figure 4.2C). We examined ML model performance to screen for early ARDS using either 24h or 6h of VWD. We trained the 24/24 model using the

first 24h of available VWD in the training set, and then tested using the first 24h of available VWD from the validation set (24/24 model; n = 100). Our second model, the 24/6 model, was trained using the first 24h of available data, but was validated using data available in the first 6h after Berlin criteria were first met or after the start of MV for ARDS and non-ARDS subjects, respectively (24/6 model: n = 70).

Model performance was assessed using AUC, sensitivity, specificity, positive predictive value (PPV), and negative predictive value (NPV). Performance was compared using a simple majority voting threshold (e.g., more than 50% of 100-breath windows were classified as ARDS in each period), and across a range of voting threshold deciles between 0-100%.

4.2 Results

A total of 100 adult mechanically ventilated patients were included in the study, including 50 with ARDS and 50 without evidence of ARDS during MV. Table 4.1 provides demographic, clinical, and physiologic characteristics of subjects. We analyzed a median of 21.2 hours of VWD per subject from ARDS patients and 13.3 hours of VWD from non-ARDS patients, representing 19,777 100-breath window observations. The dataset contained a total of 2,020,556 breaths, with 1,331,285 breaths from patients with ARDS and 689,271 from patients without ARDS.

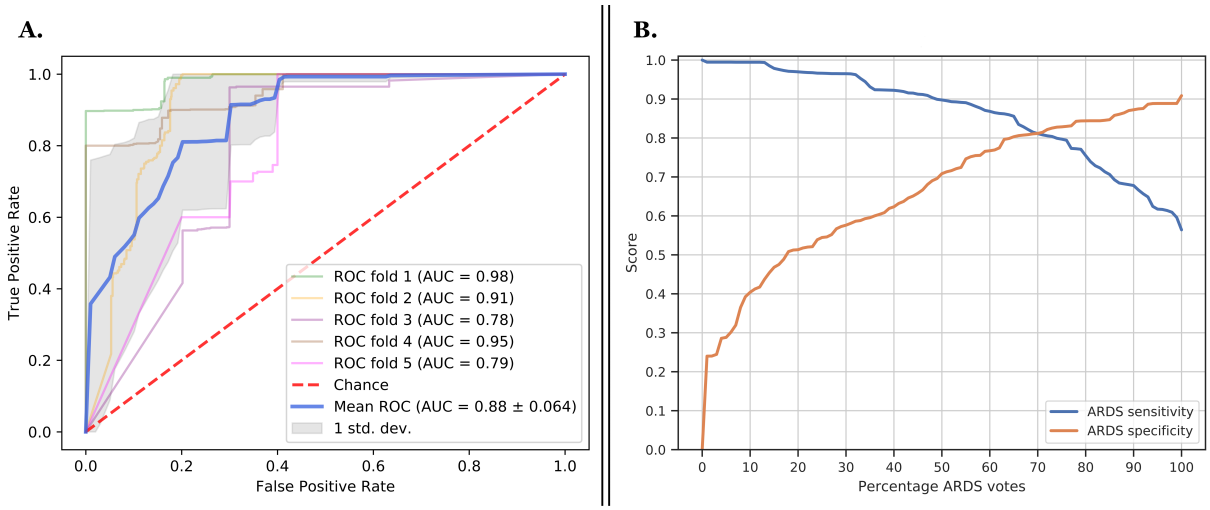


Figure 4.4: **A.** Receiver operating curve (ROC) curves for individual k-folds in the train 24/test 24 hour 5-fold cross validation model. Mean area under the ROC across all k-folds is shown in blue (95% confidence interval displayed in figure legend). **B.** Sensitivity and specificity of ARDS detection change as the voting threshold required to classify ARDS in the first 24 hours increases.

Performance of our primary ML model discriminating between ARDS and non-ARDS cases using the first 24h of VWD (24/24 model) are shown in Figure 4.4A and Table 4.8. For our main analyses, we used a simple majority voting scheme to determine patient-level predictions. Thus, if 51% or more observations from a patient were classified as ARDS, the patient was classified as ARDS. Using this voting threshold, the 24/24 VWD model was able to discriminate between ARDS and non-ARDS subjects with a mean AUC across all five k-folds of 0.88 (95% CI 0.816-0.944). Discriminative performance was similar in the 70/30 holdout and bootstrapping experiments (Table 4.10). Figure 4.4B and Table 4.9 show how model sensitivity and specificity varied by changing the threshold used to classify ARDS across the range of prediction thresholds from 0 to 100% prediction votes, and at specific threshold deciles from 10-100%, respectively.

Algorithm	Sensitivity	Specificity	PPV	NPV	AUC
Random Forest	0.90±0.059	0.71±0.089	0.77±0.082	0.90±0.059	0.88±0.064
Neural Network	0.95±0.043	0.73±0.087	0.80±0.078	0.95±0.043	0.90±0.059
Adaboost	0.92±0.053	0.74±0.086	0.80±0.078	0.92±0.053	0.90±0.059
Logistic Regression	0.96±0.038	0.74±0.086	0.81±0.077	0.96±0.038	0.90±0.059
Naïve Bayes	0.82±0.075	0.80±0.078	0.82±0.075	0.85±0.070	0.89±0.061
SVM	0.98±0.027	0.68±0.091	0.77±0.082	0.98±0.270	0.89±0.061

Table 4.7: Performance of different machine learning algorithms for ARDS classification. All numbers are rounded to 2 significant digits and reported along with 95% confidence intervals. *PPV*, positive predictive value; *NPV*, negative predictive value; *AUC*, area under the curve.

Model	Train/Test Split (n)	K-Fold Number	Sensitivity	Specificity	PPV	NPV	AUC
Train 24 / Test 24	80/20	1	1.0	0.73	0.79	1.0	0.98
-	-	2	1.0	0.79	0.83	1.0	0.92
-	-	3	0.70	0.70	0.69	0.70	0.78
-	-	4	0.80	0.91	0.90	0.82	0.94
-	-	5	1.0	0.44	0.64	1.0	0.79
-	NA	Mean of 5 k-folds	0.90 ± 0.059	0.71 ± 0.089	0.77 ± 0.082	0.90 ± 0.059	0.88 ± 0.064
Train 24 / Test 6	80/14	Mean of 5 k-folds	0.90 ± 0.07	0.75 ± 0.101	0.83 ± 0.088	0.83 ± 0.088	0.89 ± 0.073

Table 4.8. Model performance statistics for both train 24/test 24-hour and train 24/test 6-hour models. Mean (with 95% confidence intervals) performance across all 5 k-folds is shown for both models, and results of individual k-folds are displayed for the train 24/test 24-hour model to illustrate the spectrum of performance variability. Note that only 70 subjects had VWD available in the 1st 6 hours resulting in a smaller sample size for the test cohort in the train 24/test 6-hour model. *PPV*, positive predictive value; *NPV*, negative predictive value; *AUC*, area under the curve.

% ARDS votes in 1st 24 hours	Sensitivity	Specificity	PPV	NPV
10	0.99±0.02	0.4±0.096	0.63±0.095	0.99±0.02
20	0.97±0.033	0.51±0.098	0.68±0.091	0.96±0.038
30	0.96±0.038	0.58±0.097	0.71±0.089	0.96±0.038
40	0.92±0.053	0.63±0.095	0.74±0.086	0.92±0.053
50	0.9±0.059	0.71±0.089	0.77±0.082	0.91±0.056
60	0.87±0.066	0.77±0.082	0.81±0.077	0.87±0.066
70	0.81±0.077	081±0.077	0.83±0.074	0.83±0.074
80	0.75±0.085	0.85±0.07	0.85±0.07	0.79±0.08
90	0.68±0.091	0.87±0.066	0.86±0.068	0.74±0.086
100	0.57±0.097	0.91±0.056	0.86±0.068	0.69±0.091

Table 4.9: Performance characteristics of the train 24/test 24-hour model for detection of ARDS across deciles of voting thresholds, illustrating the tunable nature of our two-step ARDS classification methodology. *PPV*, positive predictive value; *NPV*, negative predictive value.

Split-Type	Sensitivity	Specificity	PPV	NPV	AUC
K-Fold	0.90±0.059	0.71±0.089	0.77±0.082	0.90±0.059	0.88±0.064
Holdout	0.90±0.107	0.75±0.155	0.79±0.146	0.89±0.112	0.94±0.085
Bootstrap	0.91±0.056	0.74±0.086	0.78±0.081	0.90±0.059	0.88±0.064

Table 4.10: Comparative performance of k-fold, 70/30 holdout, and bootstrapping methods for our train 24/test 24-hour (24/24) model with the Random Forest algorithm. Results are displayed along with 95% confidence intervals. Note that confidence intervals for the 70/30 holdout split are wider than for k-fold and bootstrapping methods because only 30 subjects were used in the testing set, whereas bootstrapping and k-fold methods used all 100 subjects. *PPV*, positive predictive value; *NPV*, negative predictive value; *AUC*, area under the curve.

Our second ML model explored the ability of an ML algorithm trained on the first 24h of VWD to differentiate between ARDS and non-ARDS in a validation set using only the first 6h of VWD after meeting Berlin criteria or starting MV for the ARDS and the non-ARDS cohorts, respectively (24/6 model). Discriminative performance in this 24/6 model was comparable to the

24/24 model with AUCs of 0.89 (95% CI 0.817-0.963) and 0.88 (95% CI 0.816-0.944), respectively, using 5-fold cross validation (Table 4.11).

Model	Train/Test Split (n)	K-Fold Number	Sensitivity	Specificity	PPV	NPV	AUC
Train 24 / Test 6	80/14	1	1.0	0.59	0.74	1.0	0.89
-	-	2	0.96	0.85	0.88	0.95	0.97
-	-	3	0.71	0.67	0.74	0.65	0.7
-	-	4	0.82	1.0	1.0	0.83	0.99
-	-	5	1.0	0.66	0.77	1.0	0.88
-	NA	Mean of 5 k-folds	0.90 ± 0.07	0.75 ± 0.101	0.83 ± 0.088	0.89 ± 0.073	0.89 ± 0.073

Table 4.11: Performance statistics for the train 24/test 6-hour (24/6) model. Mean (with 95% confidence intervals) performance across all 5 k-folds is shown, and results of individual k-folds are displayed to illustrate the spectrum of performance variability. Note that only 70 subjects had VWD available in the 1st 6 hours resulting in a smaller sample size for the test cohort in the train 24/test 6-hour (24/6) model. *PPV*, positive predictive value; *NPV*, negative predictive value; *AUC*, area under the curve.

4.3 Discussion

We developed an automated ARDS screening algorithm that can detect potential cases of moderate-severe ARDS early during MV without need for CXR, ABG, or other EHR-derived data. Using ML techniques and physiologic features derived from raw VWD, our model demonstrated robust discriminative performance for detecting ARDS in the first 24h after meeting Berlin criteria that was reproducible across a variety of experimental conditions. We further showed that our ARDS detection model could identify potential ARDS cases as early as 6h after Berlin criteria were first documented, and that our model architecture enabled adjustment of model performance according to desired levels of sensitivity and specificity.

To address limitations of prior studies into automated ARDS detection methods, our methods differed from previous research in several notable ways. Our use of ML with VWD-

derived features may overcome some limitations of previous feature extraction methods by capturing the physiologic signatures present in waveforms instead of relying on EHR or imaging data alone. Because VWD are generated from the start of MV, our methods enable continuous patient monitoring and may allow for more timely identification of potential ARDS cases, independent of ordering or documentation, which was suggested by our finding that ARDS could be detected as early as 6 hours after Berlin criteria were first met. As access to physiologic waveform data becomes more common, our exclusive use of VWD may also extend automated ARDS screening to resource-constrained care environments such as community and rural hospitals lacking well-developed EHRs, and in developing nations, battlefields, and disaster relief zones where tests required to fulfill Berlin criteria may be in short supply or unavailable.(48) Use of ventilator waveform analysis may thus improve both the timeliness and the ability to apply automated ARDS screening in diverse settings, which are particularly important since delayed or missed diagnosis is thought to be a major contributor to suboptimal implementation of evidence-based therapies for ARDS.(13,20,23,153)

In addition to extending previous research into developing automated ARDS screening systems, our work further demonstrates the potential value of ML in the analysis of large volumes of untapped streaming physiologic waveform data generated from patient monitoring devices in the ICU. The use of patient-derived physiologic data has gained increasing attention in recent years as the availability of both high-volume, high-sampling rate data types, and advanced computing power have become more commonplace. In this regard, automated processing of VWD has been demonstrated by multiple investigators in the study of patient-ventilator asynchrony (52–54) and several groups have shown the potential to computationally extract physiologic features from VWD including data derived from animal models of ARDS pertaining

to airway resistance and respiratory system compliance.(45,47,108) Sottile et al. (55) and Rehm et al. (56) have further investigated the ability to use ML to detect common types of patient-ventilator asynchrony without the need to explicitly code rule-based, expert systems, illustrating the ability of ML algorithms to learn relevant knowledge from the physiologic information embedded in raw VWD.

Our work also fits into a broader context of recent research using ML and sensor-derived physiologic data to develop so called digital biomarkers to screen for and monitor diseases, improve disease phenotyping, and predict clinical trajectories.(154) Recent studies in critical care have demonstrated the potential of digital biomarker signatures include the use of convolutional deep neural networks to process ECG waveforms to screen for hyperkalemia (155) and detect arrhythmias,(156) and the use of continuous electroencephalography waveforms and deep learning to predict neurologic outcome after cardiac arrest.(157) Within this framework, our results suggest the potential of ML and VWD to generate digital biomarker signatures of ARDS, either alone or in combination with conventional biomarkers,(158) to aid clinicians in early detection, monitoring ARDS progression, and prognostication of patient outcomes.

This study has several limitations that should be addressed in future studies. First, our study was limited to a single academic medical center, which could affect model generalizability despite our exclusive use of quantitative physiologic data.(33) Second, our limited sample size may have resulted in model overfitting. We attempted to address this issue by using the RF algorithm, which may be inherently more resistant to overfitting,(143) and several different frameworks for model validation. While most prior studies using VWD have been similarly limited in size,(52–55) research on larger cohorts will be necessary to understand the full limitations of waveform-based ARDS screening. Similarly, our subject selection was

intentionally biased to ensure phenotypic separation between ARDS and non-ARDS subjects to test the hypothesis that VWD and ML could be used to discriminate between clear phenotypes. Our cohort was comprised mostly of moderate-severe persistent ARDS that was present on intubation, and it is unclear how our model would perform in late-onset ARDS, mild ARDS patients, rapid resolvers,(159) those with an uncertain diagnosis when Berlin criteria are first met, or in those with pre-existing chronic lung disease.(36,43,160) Third, we focused on clinician-driven feature extraction and one ML algorithm. It is possible that the use of other input features, including non-waveform EHR-derived features, or algorithms capable of end-to-end model development and automated featurization such as deep learning (161) may have improved performance. Fourth, while VWD are ubiquitous at the bedside, widespread access to these data for research purposes remains a challenge at present. Finally, studies aimed at developing ARDS classifiers, including ours, are limited by the inherent imprecision of the Berlin criteria.(36,162) Recognizing this fundamental limitation, our study focused on developing a tunable screening algorithm rather than one aimed at diagnosis. Development of ARDS classifiers that generalize well and are trusted by clinicians will require additional study with larger, more heterogenous populations, may require improved methods of class assignment such as advanced imaging and physiologic or digital biomarkers,(24,31,158,163) and will ultimately require external validation followed by thoughtful integration into decision support workflows to realize intended benefits to patients.

Chapter 5: Deep Learning Based ARDS Detection

5.1 Methods

All data for this study were collected as part of an IRB-approved, observational study in the ICUs at UC Davis Health, an academic tertiary care medical center in Northern California. All patients or surrogates provided informed consent for data collection. Three clinicians performed dual-adjudicated, retrospective chart review to identify the cause of respiratory failure in subjects enrolled between 2015–2020. Subjects were split into two cohorts: 1) 50 patients with confirmed moderate or severe ARDS diagnosed using Berlin consensus criteria within 7-days of intubation,(164) and 2) 50 patients with no evidence of ARDS during their course of MV (referred to here as non-ARDS).(110) Our cohort was split into clear phenotypes as a proof of concept study to test the feasibility of using VWD and ML to detect ARDS. Patients were excluded from the cohort if: 1) they were transferred to ICU from an outside facility; 2) had a chronic tracheostomy prior to admission; 3) had a persistent bronchopleural fistula; 4) diffuse chronic fibrotic lung disease; or 5) Berlin criteria were met for < 24 hours. Included patients had at least 1hr of VWD within either the first 24 hours of ventilation if they were non-ARDS, or 1 hour of VWD in the 24 hours after first meeting Berlin diagnostic criteria for ARDS. For specific cohort descriptive statistics see Table 4.1.

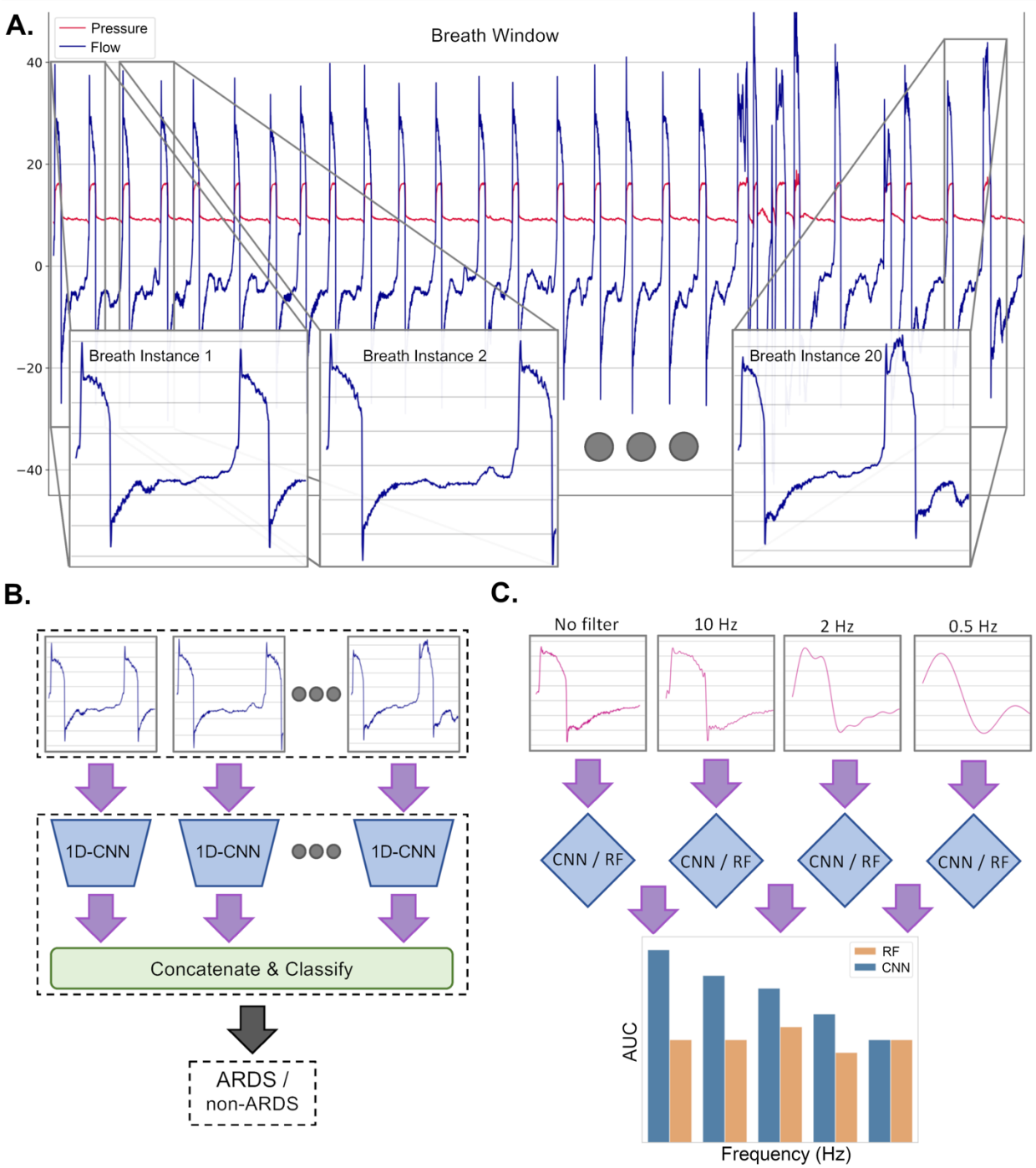


Figure 5.1: **A.** Ventilator data is extracted as a continuous series of raw flow data from the mechanical ventilator sampled at 50 Hz. **B.** Breath instances that were extracted from the above windows are fed into a 1-D convolutional neural network (CNN) to make an ARDS or non-ARDS classification on the instance. **C.** For interpretability, lowpass filtering was performed on the waveforms and then a CNN or random forest (RF) model was trained using the filtered data. CNN and RF model results were compared to determine how waveform data was utilized by the CNN.

VWD representing air flow sampled at 50 Hz over time was extracted from the Puritan Bennet-840 mechanical ventilator using an established data collection architecture.(122) Our VWD sample contained 2,020,556 breaths, including 1,331,285 breaths from patients with ARDS and 689,271 breaths from patients without ARDS. For generation of model inputs, flow time series data were split into successive breath "instances", which we define as a sequence of flow samples that are 224 observations in length (Figure 5.1A). Breath instances were formatted so that each instance begins at the start of inhalation. We then arranged the data as a series of 20 breath instances, which we refer to as a "breath window", for our ARDS classification experiments (Figure 5.1A). All breath windows were labeled according to their subject's corresponding pathophysiology (ARDS or non-ARDS).

Once we processed our data, we trained a one-dimensional (1-D), 18-layer, DenseNet CNN model to perform binary classification on which breath windows were classified as ARDS or non-ARDS.(165) Since there are > 1 breath windows per-patient, we can aggregate breath-window classifications to the patient by finding the majority classification of all breath windows for a patient. Thus, if $> 50\%$ of breath windows for a patient are classified as ARDS, then the patient was predicted as ARDS. Our primary reporting metric is model AUC based on the fraction of correct patient-level predictions made for ARDS. We also report patient-level accuracy, sensitivity, and specificity.

We evaluated our model performance by first using the 5-fold K-fold splitting technique using data from 80 patients out of the 100-patient cohort, evenly split between ARDS and non-ARDS patients. Each fold is split evenly in terms of the ratio of non-ARDS patients to ARDS patients to avoid class imbalance at the patient level. We also evaluate a 70:30 holdout split, random K-fold splits, and bootstrapping with replacement at 80:20 split (see online supplement).

Because the number of ARDS breath instances outnumbers non-ARDS by a 2:1 ratio, we correct this imbalance by performing random oversampling of non-ARDS breath instances.(166–168) Because of data noise, we used standard stochastic gradient descent instead of adaptive learning rate algorithms like ADAM.(169) We performed a grid search for optimal hyperparameters. Learning rate was found to be best at 0.001. To determine a consistent approximation of our model's performance we performed 10 trials of experiments for 10 epochs and then averaged the results to determine a final estimate of performance.(111)

We used multiple approaches for model interpretation. To understand the relative contributions of different components of the frequency domain to model performance, we transformed our VWD using the fast Fourier transform (FFT) from the SciPy library.(170) Since data was sampled at 50 Hz, FFT decomposes the signal into the frequency bandwidth between -25 and 25 Hz. FFT-transformed data were otherwise utilized in our model identically to unprocessed VWD in that breath instances were transformed into the frequency domain and then inputted into the model and classified. For model interpretability, we utilized Grad-Cam(171) to highlight a saliency map for VWD. We chose Grad-Cam in our application because it passed tests (i.e. data and model randomization tests) necessary for consideration as a model explanation method.(172) Grad-Cam was then used to visualize specific frequencies the model appeared to be using in classification.

Finally, we performed signal filtering experiments using FFT and threshold filtering to investigate the contributions of specific frequency components of the VWD signal on model performance. For these experiments we set a lowpass filter at a bandpass between 0.5-20Hz. VWD is then decomposed with FFT, and selected frequency ranges are removed. VWD is then reconstructed into the time-domain using an inverse FFT. Our model is then retrained on lowpass

filtered data and standard model performance metrics are then reported. All code was developed using Python's scientific computing software suite and PyTorch.(139,173) All code is available (*https://github.com/hahnicity/deepards*) and anonymized datasets are available upon request.

Fold N/length of sequence	**2048**	**1024**	**512**	**224**	**128**
Fold 1	0.8	0.8473	0.8210	0.8210	0.8315
Fold 2	0.8052	0.8157	0.8	0.8157	0.7578
Fold 3	0.8157	0.8052	0.7947	0.7894	0.8421
Fold 4	0.8789	0.8789	0.8526	0.9315	0.9210
Fold 5	0.7777	0.7666	0.7722	0.8222	0.8
Mean Accuracy	***0.82***	***0.82***	***0.81***	***0.84***	***0.83***

Table 5.1: Accuracy of model based on 5-fold cross validation.

Fold N/length of sequence	**2048**	**1024**	**512**	**224**	**128**
Fold 1	0.8844	0.9522	0.9411	0.9522	0.9044
Fold 2	0.9011	0.9277	0.9211	0.9377	0.8855
Fold 3	0.8411	0.8566	0.8889	0.8805	0.9244
Fold 4	0.9722	0.9577	0.94	0.9877	0.9266
Fold 5	0.8156	0.79	0.8175	0.865	0.8937
Mean AUC	***0.88***	***0.90***	***0.90***	***0.92***	***0.91***

Table 5.2: AUC of model based on 5-fold cross validation

5.2 Results

We utilized 1-D flow time-series data to train our DL model. Because our deep neural network architecture requires a fixed number of input samples, we first explored the optimal input sequence length. Based on experimentation (Table 5.1, Table 5.2), we found that breath instances of 224 inputs were optimal for input into our network. We then compared the performance of several DL architectures (CNN, LSTM, or hybrid CNN+LSTM) and found optimal performance with the CNN model (Table 5.3). We also found model performance improves when we ensure that each breath instance begins at the start of inhalation. We report

results for our final model in Table 5.4 across all 5 K-folds. Results from K-folds cross-validation were similar to results from experiments using a random 70:30 split of the dataset where 70% of patients' data were used for model development and 30% of patients were reserved for holdout testing, and to experiments using bootstrapping to split the dataset (Table 5.5)

Network	AUC	Accuracy	Sensitivity	Specificity
CNN only	**0.93**	**0.84**	0.83	**0.84**
LSTM only	0.90	0.81	0.81	0.81
CNN+LSTM	0.92	0.79	**0.95**	0.62

Table 5.3: Performance of CNN, LSTM, or combined CNN+LSTM networks for classification of the acute respiratory distress syndrome (ARDS) using ventilatory waveform data as the sole input. *AUC*, area under the receiver operating characteristic curve; *CNN*, convolutional neural network; *LSTM*, long short-term memory network.

K-Fold Number	AUC	Accuracy	Sensitivity	Specificity	PPV	NPV
1	0.99±0.01	0.85±0.056	1.0±0.0	0.7±0.113	0.78±0.07	1.0±0.0
2	0.88±0.025	0.79±0.069	0.65±0.15	0.92±0.037	0.9±0.06	0.74±0.094
3	0.94±0.018	0.85±0.037	0.9±0.075	0.8±0.113	0.83±0.068	0.9±0.067
4	0.98±0.005	0.89±0.019	0.85±0.05	0.92±0.037	0.92±0.042	0.86±0.037
5	0.97±0.013	0.85±0.031	1.0±0.0	0.7±0.075	0.77±0.045	1.0±0.0
Mean of 5 k-folds	0.95±0.019	0.84±0.026	0.88±0.068	0.81±0.06	0.84±0.038	0.9±0.05

Table 5.4: Displays results from individual K-Folds for our convolutional neural network classifier. 95% CI are shown after the mean result of each metric from individual trials. AUC - area under the curve. PPV - positive predictive value (also known as precision). NPV - negative predictive value

Once we finalized our deep learning network, we compared the performance of our DL model to our previously published traditional ML model (Figure 5.2).(110)

Split Type	N Trials	Accuracy	AUC	Sensitivity	Specificity
Holdout (110)	10	0.83±0.017	0.91±0.008	0.91±0.02	0.75±0.04
Random K-fold	50	0.84±0.01	0.93±0.007	0.86±0.014	0.83±0.016
Bootstrap	150	0.82±0.013	0.91±0.011	0.85±0.02	0.78±0.026

Table 5.5: Holdout, Random K-fold, Bootstrapping (80/20 split, with replacement) split results

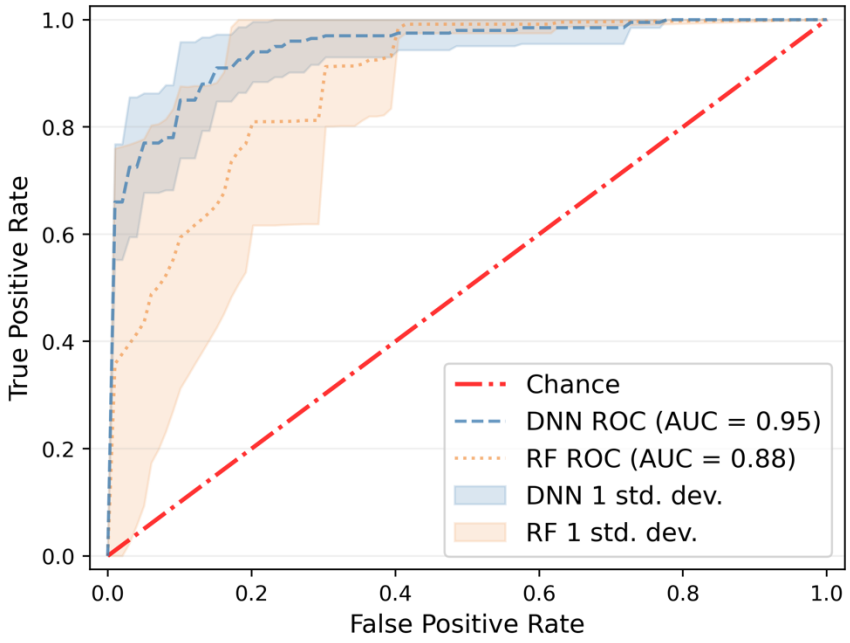


Figure 5.2: Comparison of ROC curves for both our initial random forest (RF) model, and our 1D-CNN model. Shaded parts of the plot are 95% CI interval.

After finding that our deep learning model outperformed the random forest model, we sought to investigate factors explaining the improved performance of deep learning. We began by exploring VWD components in the frequency domain that might be contributing to differential model performance by using 1-D FFT transformations of our data and then retraining the model with these transformed data (Table 5.6). When our network performed well with this experiment, we investigated the frequencies our network was using for classification. To do this, we utilized Grad-CAM to create a saliency map that highlights where in the frequency domain the network was focusing to make classifications, using all inputs into our neural network, and

averaging all Grad-CAM outputs. The Grad-CAM saliency map shows that the model's cam intensities for ARDS were focused primarily on higher frequencies while non-ARDS classification was focused at both high and low-frequency bands.

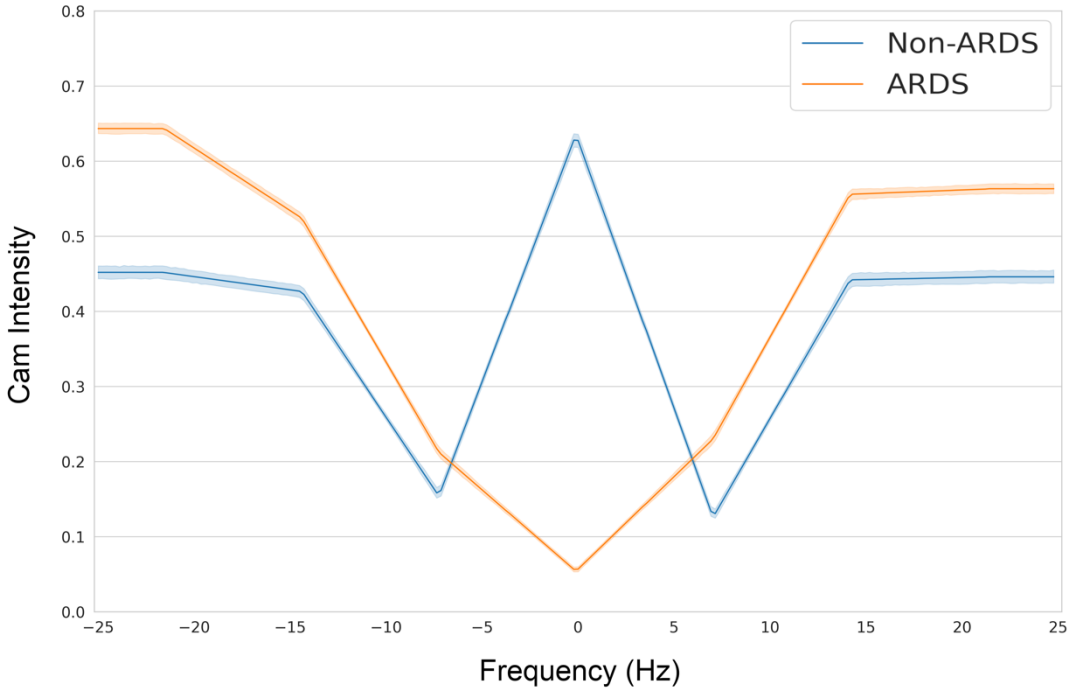


Figure 5.3: Shows an average Grad-CAM intensity (range 0-1.0) of our CNN that were trained only with FFT data. 95% CI is also shown in the slight boundary zone around each line. Non-ARDS focus is shown in blue, and ARDS in orange. Higher cam intensity values on the y-axis means that the network focused on this frequency more heavily for prediction of a specific class.

Dataset	***Deep NN?***	***AUC***	***Accuracy***	***Sensitivity***	***Specificity***
VWD only (Random Forest)	✗	0.88	0.80	0.90	0.71
VWD only (best CNN)	✓	0.95	0.84	0.88	0.82
FFT only	✓	0.87	0.76	0.79	0.72
VWD & FFT	✓	0.89	0.80	0.84	0.75

Table 5.6: Neural network performance as a function of different input data types. Results from a random forest classifier reported in our previous work are noted as a reference. All other rows

show results of experiments with different network inputs including VWD alone, FFT transformed VWD alone, and combined input of both data types.

Based on the above observations, we sought to better understand the relative contributions of both high and low frequency components of the raw VWD signal to the improved performance observed in our DL model (Table 5.7). We thus performed targeted frequency ablation experiments by applying FFT to the raw VWD, with progressive lowpass filtering cutoff frequencies ranging from 0.5-20 Hz. The ablated VWD were then used for model development and K-folds cross validation using methods identical to those with the non-ablated raw VWD for both DL and traditional RF model development and cross-validation.(110) These frequency ablation experiments revealed a decrease in DL model performance after ablation of high frequency waveform components beginning at a 20 Hz lowpass cutoff down to a cutoff of 0.5 Hz, with performance comparable to our traditional ML approach at the cutoff of 0.5 Hz (Table 5.7B). Our RF model, which uses expert-derived features computed from raw VWD, did not see comparable decrements in performance at higher lowpass frequency cutoffs (Table 5.7C), which implies traditional ML features are well-represented in low frequency ranges. We were unable to ablate frequencies below 0.5Hz because we were unable to extract manual features for the traditional ML model because waveform ablation prevented the extraction of manual features (Figure 5.5)

A. Baselines	*Network*	*AUC*	*Accuracy*	*Sensitivity*	*Specificity*
	RF(110)	0.88±0.064	0.80±0.078	0.90±0.059	0.71±0.089
	CNN	0.95±0.019	0.84±0.026	0.88±0.068	0.81±0.06

B. FFT Filtering on CNN	*Cutoff Frequency (Hz)*	*AUC*	*Accuracy*	*Sensitivity*	*Specificity*
	20	0.94±0.015	0.83±0.017	0.85±0.038	0.81±0.035
	15	0.93±0.014	0.83±0.018	0.89±0.034	0.76±0.042
	10	0.92±0.021	0.81±0.026	0.84±0.039	0.77±0.046
	8	0.90±0.02	0.80±0.023	0.85±0.04	0.74±0.045
	6	0.91±0.021	0.81±0.026	0.85±0.037	0.78±0.042
	4	0.91±0.018	0.83±0.025	0.83±0.037	0.82±0.042
	2	0.91±0.016	0.82±0.025	0.81±0.037	0.83±0.04
	1	0.90±0.021	0.82±0.029	0.78±0.051	0.85±0.035
	0.5	0.89±0.015	0.81±0.18	0.77±0.03	0.86±0.031

C. FFT Filtering on RF	*Cutoff Frequency (Hz)*	*AUC*	*Accuracy*	*Sensitivity*	*Specificity*
	20	0.88±0.063	0.83±0.074	0.86±0.068	0.79±0.08
	15	0.88±0.063	0.81±0.077	0.79±0.08	0.84±0.072
	10	0.88±0.063	0.80±0.079	0.87±0.066	0.73±0.087
	8	0.88±0.064	0.81±0.078	0.89±0.061	0.72±0.088
	6	0.87±0.065	0.81±0.078	0.88±0.063	0.73±0.087
	4	0.89±0.062	0.83±0.073	0.91±0.057	0.76±0.084
	2	0.87±0.065	0.81±0.076	0.87±0.065	0.75±0.085
	1	0.86±0.068	0.82±0.076	0.89±0.06	0.74±0.086
	0.5	0.88±0.065	0.84±0.072	0.87±0.067	0.81±0.078

Table 5.7: **A.** shows the baseline performances of the traditional machine learning based random forest (RF) model, and the convolutional neural network (CNN) model. **B.** shows results of our network trained with only VWD after a lowpass FFT filter is applied. **C.** - Results of the traditional ML model after FFT lowpass filtering is applied. Note that AUC maintains similar performance through all ablations and matches performance of the FFT filtered model on CNN at very low frequencies.

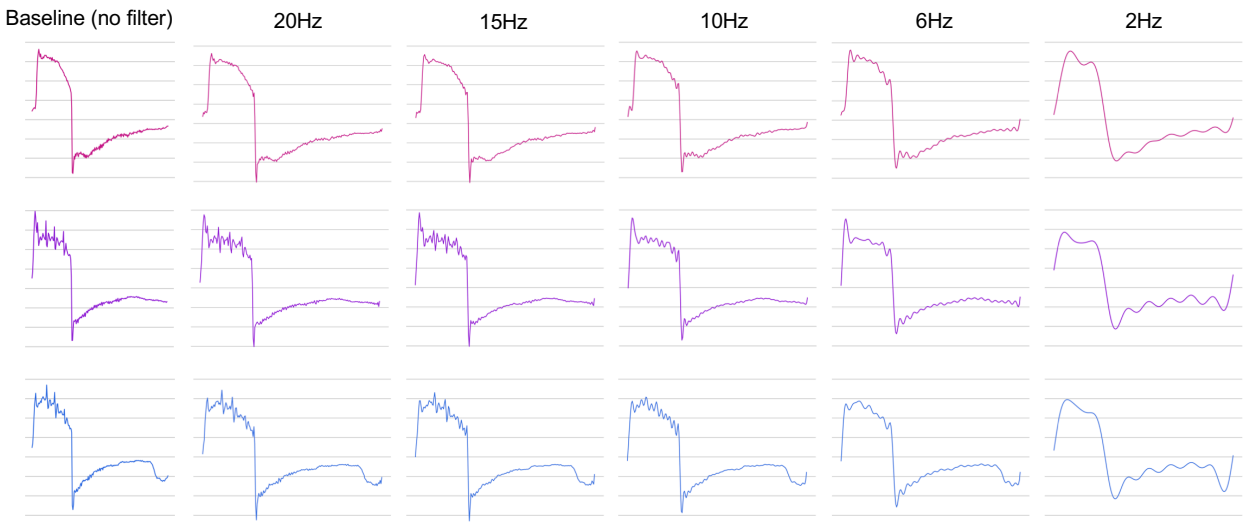


Figure 5.4: Examples of lowpass fast Fourier transform (FFT) filter applied to 3 different breaths at different passband thresholds ranging from 20Hz to 2Hz.

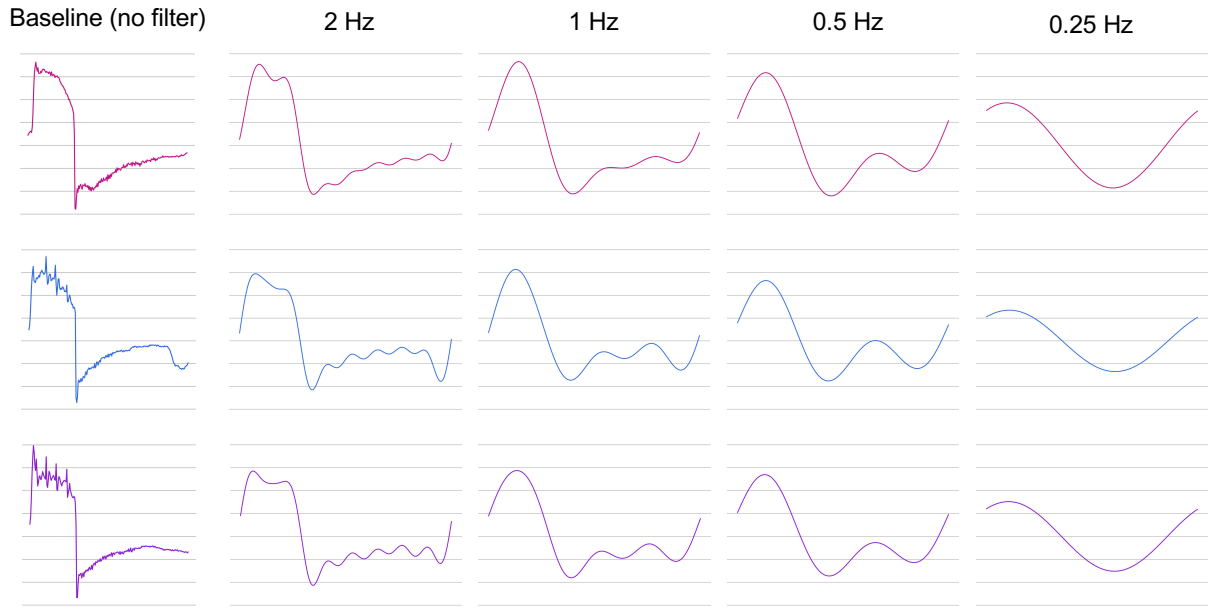


Figure 5.5: Examples lowpass fast Fourier transform (FFT) filter applied to 3 different breaths at different passband thresholds ranging from 2Hz to 0.25Hz.

5.3 Discussion

In this chapter, we detail our work developing deep learning (DL) models for detecting ARDS using raw VWD as the only data source. We found that a 1-D deep CNN improved discrimination between ARDS and non-ARDS subjects compared to a random forest classical ML model, without need for labor-intensive and potentially biased expert-informed feature engineering. Furthermore, using a mix of descriptive and experimental methods to explain what our DL model was learning, we were surprised to find that relatively high-frequency regions of the 1-D VWD time series contained the implicit information that our DL model had learned to improve performance over our classical ML model.

Our work expands on previous research related to the development of algorithm-based screening for ARDS. Nearly every study to date has focused on processing data derived from the EHR including laboratory data, flowsheet data such as vital signs, text documents including, and radiographic images.(30) While early efforts used rule-based systems,(28,29,44,60,174) more recent studies have employed classical machine learning methods.(34,42,43) Many of these approaches have demonstrated good-excellent ability to discriminate between patients with and without ARDS at their originating centers, but several aspects of design potentially limit widespread applicability. In particular, reliance on infrequently sampled, non-continuous model inputs derived from clinical diagnostic testing may limit the timeliness of ARDS screening and/or model generalizability(33) as a function of clinician or institutional ordering and documentation practices. To address these potential limitations, recent work by our group used physiologic features extracted from VWD as inputs into a classical ML model to screen for ARDS.(110) Our results showed discrimination comparable to prior screening systems, even in the first six hours after Berlin diagnostic criteria for ARDS were first evident, without the need

for additional model inputs from the EHR or other clinical systems. Despite the potential advantages of using VWD as the sole model input, our previous methods were limited by the need for expert-informed featurization of the raw VWD, which may have introduced biases into the model that compromised performance. Results of our current study suggest that deep learning may overcome limitations of our prior approach by learning from implicit information present in raw data whose value may not have been recognized by clinicians in the process of manual feature engineering.(50,110) While we are unaware of other studies directly comparing the performance of traditional ML and DL models operating on physiologic waveform data, multiple studies have demonstrated the potential utility of either classical ML or DL for classification and prediction using high sampling rate physiologic data.(155,157,175) Experimental comparisons of alternative ML approaches will be critical to developing a comparative effectiveness framework to guide future research while balancing competing considerations in terms of computational requirements and model development time, availability of technical and subject matter expertise, interpretability, regulatory complexity, and ease of deployment and maintenance.

One of the disadvantages of DL compared to some commonly used classical ML algorithms is a lack of interpretability, which can limit hypothesis-driven experimentation and compromise trust in model outputs.(50,176,177) Many works that have performed interpretability evaluations have utilized saliency methods.(178–181) This body of prior work motivated us to adopt Grad-CAM for our interpretability experiments. We also explored a number of different methods of ablating different parts of our VWD, or by searching for specific parts of the input that were salient, such as asynchronous breathing.(77) However, all these preliminary experiments failed to substantially impair the network's ability to learn. The failure of these analyses led to our investigation of using a lowpass FFT filter to remove high-frequency

components from VWD. Based on the results of these experiments, we saw that we were able to heavily filter VWD and still have it perform well in the task of ARDS detection. This suggests that there are key features that exist in the low-frequency representation of the waveform that are discriminative of ARDS.

To validate this question, we then performed learning on the lowpass filtered waveforms using featurization methods originally developed for traditional ML algorithms. This experiment builds upon prior work examining the importance of signal frequencies for predictive performance of machine learning models. In particular, there have been prior studies examining: the importance of spectroscopic wave spectrum for traditional machine learning;(182) the use of Grad-CAM to highlight important frequencies in sound recordings for bearing fault;(178) and Fourier transforms to improve interpretability of DNA models and atrial fibrillation detection.(183,184) We build upon this work by ablating model inputs over the frequency spectrum and then retrain our models with filtered data to obtain better understanding of how the models would perform. To the best of our knowledge however, no other surveyed studies have taken additional steps to retrain their models using filtered waveforms. We found that after retraining our traditional ML model on filtered waveforms model performance did not change. This shows us that the expert features that were originally derived exist on the low-frequency spectrum and are available to the DL model to learn. From observation of filtered waveforms (Figure 5.4,Figure 5.5), we note that low frequency data is associated with waveform periodicity, i.e., the rate at which a patient is breathing. Given that DL model's performance is equivalent to performance of traditional ML when using heavily filtered waveforms, it is possible that our DL model is learning the same expert-derived features that were originally used for detecting ARDS. The fact that DL models are superior to traditional ML when higher frequency waveform data is

included shows that DL models are likely able to learn discriminative features in higher-frequency data that would be difficult for experts to manually featurize. This observation fits broadly with the perceived benefit of using CNN,(111) and we hope our experiments will cause researchers to consider frequency ablation as a tool that may be used to better understand features that may or may not be present in PWD. Additional research into ventilator and other physiologic signals could also help to highlight useful features that are found in certain frequency bands.

Our study has several limitations. First, our dataset was limited to a single academic medical center. Second, our input data was limited to a single physiologic data stream (flow), and it is unclear if our findings regarding the importance of high frequency signal components to model performance will generalize to other non-ventilator physiologic waveforms. Third, Grad-CAM provides low fidelity mappings for FFT inputs. This hinders our ability to reason about which specific frequencies are being heavily used in our network. SHAP or LIME may be better alternatives to Grad-CAM for this purpose, however currently, these tools are only available for use with 2-D image data for DL models and not 1-D time series. Our limited sample size and more restrictive cohort enrollment parameters may have resulted in model overfitting. We attempted to address this issue by using a 5 K-folds approach to model validation where we found relative stability of performance across K-folds, and with separate experiments using randomized K-folds. However, future research on larger, more diverse, patient cohorts will be required to determine the best uses of deep learning and VWD for ARDS screening. Finally, our subject selection criteria were chosen to ensure phenotypic separation between ARDS and non-ARDS subjects to test our hypothesis that VWD and ML could be used to discriminate between these clearly delineated phenotypes.(110) It is unclear how the performance of deep learning

classifiers will be affected by the use of a more heterogeneous patient cohort, such as patients with unclear timing of ARDS onset, mild ARDS, or additional non-ARDS causes of respiratory failure.

Chapter 6: Conclusion and Future Work

6.1 An End-to-End System for ARDS Detection in Smart Hospitals

Once we developed our ARDS detection model, we desired to integrate it into a data collection architecture capable of supporting large, multi-center, clinical studies of patient-ventilator interactions, and IoT based multi-sensor, multi-patient monitoring. Our system requirements include: 1) continuous and automated data collection from multiple concurrently operating mechanical ventilators; 2) unobtrusive, non-disruptive operation so as not to influence patient care; 3) ability to maintain temporally accurate data and preserve correct data linkage between patient and collected ventilator waveform data (VWD); 4) ease of use of the data acquisition hardware by non-technical users. 5) database archival storage; and 6) ability to generate alerts to and receive feedback from doctors to improve mechanical ventilator management.

To accomplish these goals, we used a small, unobtrusive IoT device that acts as an information aggregator by collecting data from mechanical ventilators and other sensors or medical devices. For our prototype architecture, we chose to use the Raspberry Pi™ (RPi) microcomputer, a small Linux-based computer that, with customized software, can be attached to a ventilator to collect and stream VWD to a server through a wireless access point (Figure 6.1 (1)). Once collected, VWD is attributed to a specific patient by having physicians link VWD files to a specific patient via mobile application (Figure 6.1 (2)). The linkage process is performed without use of private patient information by referencing the patient via an anonymized token. Linkage of tokens to protected health information extracted from the electronic health record (EHR) is ensured

with use of a secure encrypted file. To ensure temporally accurate linkage of collected VWD to EHR data we required the RPi's to connect to the hospital's Network Time Protocol (NTP) servers before commencing data collection, followed by time stamping of VWD files.

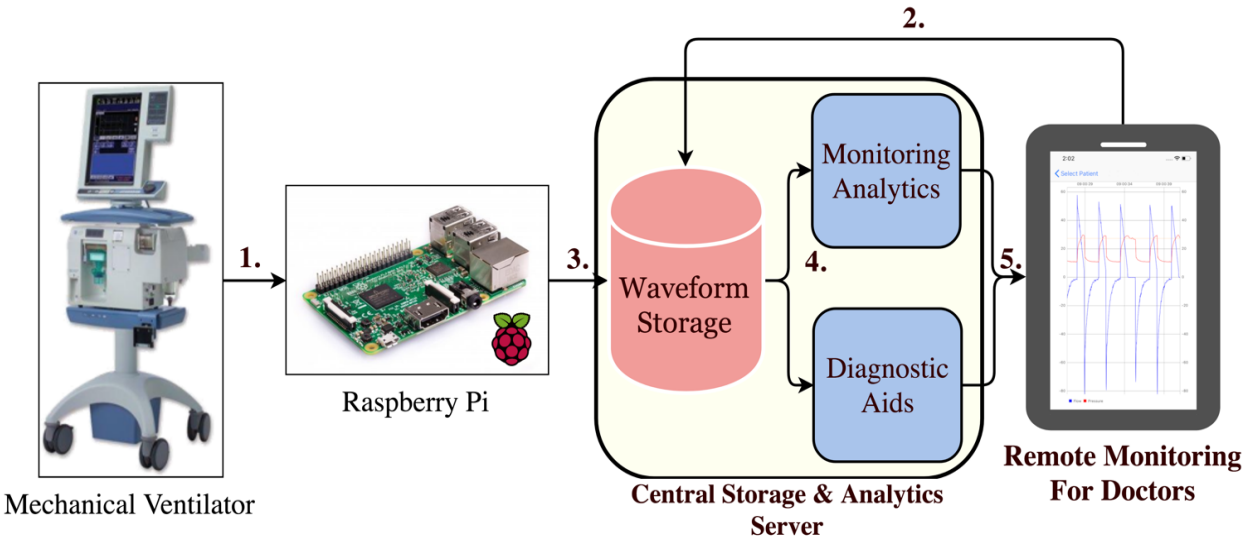


Figure 6.1: **1.)** Raspberry Pi microcomputers collect data from the mechanical ventilator. **2.)** A doctor performs linkage of a patient to a Raspberry Pi. **3.)** Ventilator waveform data (VWD) is stored in a database with proper patient attribution. **4.)** VWD is processed by analytic modules aimed at diagnostic aid and detection of abnormalities. **5.)** Alerts are sent to clinicians to review and take appropriate actions to improve patient care.

Our data attribution and time alignment protocol can be extended to collect other types of medical device data. In a pilot study, we have extended our RPi-based architecture to acquire patient blood oxygenation (SpO2) data from wireless pulse oximeters, allowing synchronous acquisition and aggregation of both VWD and SpO2 data. Other device data can be incorporated for aggregation as well, provided they can communicate with the RPi over Bluetooth, Wi-Fi, or wired cable.

Once device data are collected, it is forwarded to a database for storage (Figure 6.1 (3)). Analytic algorithms can then be applied to the data for anomaly detection and diagnostic purposes, with analytic outputs subsequently accessed retrospectively for research or in near real-time for

decision support (Figure 6.1 (4)). Providers can then monitor results of anomaly detection diagnostic algorithms, while providing real-time feedback to learning algorithms to improve their performance (Figure 6.1 (5)). As a result of our work, we have been able to collect one of the largest collections of breath-level VWD reported to date having collected 467 patients, and 47,990,952 recorded breaths for use in developing clinically validated analytic algorithms to support CDS system development.(185)

6.2 Mobile Applications for Ventilator Waveform Data

There are two major limitations of existing mechanical ventilators that present barriers to effective patient monitoring and limit the adoption of ventilation focused CDSS. First, the best available ventilator alarms use simple, threshold-based rules (e.g. – alarm for any breath with volume over 'x') that lack flexibility in terms of customization, and sophistication regarding analytics. Second, alarm settings cannot be configured remotely, and in most hospitals, alerts cannot be viewed using mobile devices. Clinicians must therefore be in a patient's room to directly observe how a patient is breathing, and are forced to abandon monitoring when called away.(186) Even when physicians are bedside, limited alarm sophistication and configurability can cause frequent false alerts, resulting in overly wide alarm thresholds that can cause long periods of asynchronous breathing and deterioration in a patient's physiologic state to go unnoticed. These problems highlight the need for mobile device-based CDSS to improve the monitoring and management of patients requiring mechanical ventilation.

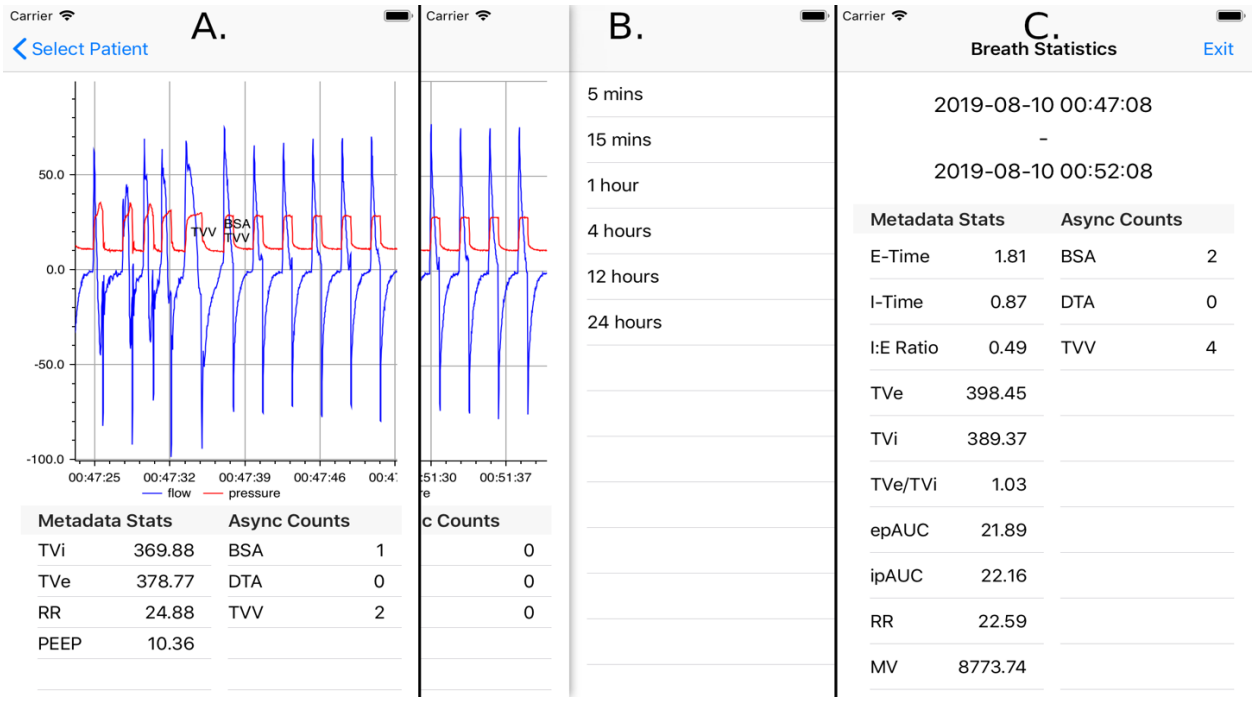


Figure 6.2: **A.** Mobile application displaying waveform data of one patient, with breaths labeled with detected asynchronies and excessive tidal volumes. The area below the chart contains statistics of breaths currently being displayed. Pinch-zoom functionality allows custom selection of time frames for waveform display, summary statistics, and event labeling. **B.** Discrete look back time frames over which breath statistics can be calculated. These options are selected via left swipe from the screen displaying patient waveform information. **C.** Example result of breath statistics calculated for a 5-minute time frame. Both clinically relevant metadata and PVA statistics are shown.

To address these problems, we have developed an iOS application and associated architecture to enable research and development of real-time monitoring and CDSS for VWD. Several core application features were designed to address existing deficiencies in ventilation monitoring, to enable innovations in decision support algorithm development, and to integrate into real-world clinical practice workflows. First, we allow clinicians to remotely view a patient's waveform data in near real time, to provide on-demand snapshots of overall clinical trends in ventilation (Figure 6.2A). Second, real time processing of VWD by our computing architecture and ventMAP software package enables remote alerting of clinicians to the presence of ventilator asynchrony and other forms of off-target ventilation.(54,121) Breaths that are determined to be

asynchronous are labeled on the screen, enabling clinicians to get an overview of asynchrony trends and their duration. The application also includes the ability to compute breathing statistics over variable, clinician-configurable periods of time (Figure 6.2B, C). The application's flexibility in this regard both enables clinicians to validate that prescribed treatment protocols are being implemented properly and allows greater sophistication in alarm logic including use of event class, severity, frequency, and proportion over configurable periods of time.

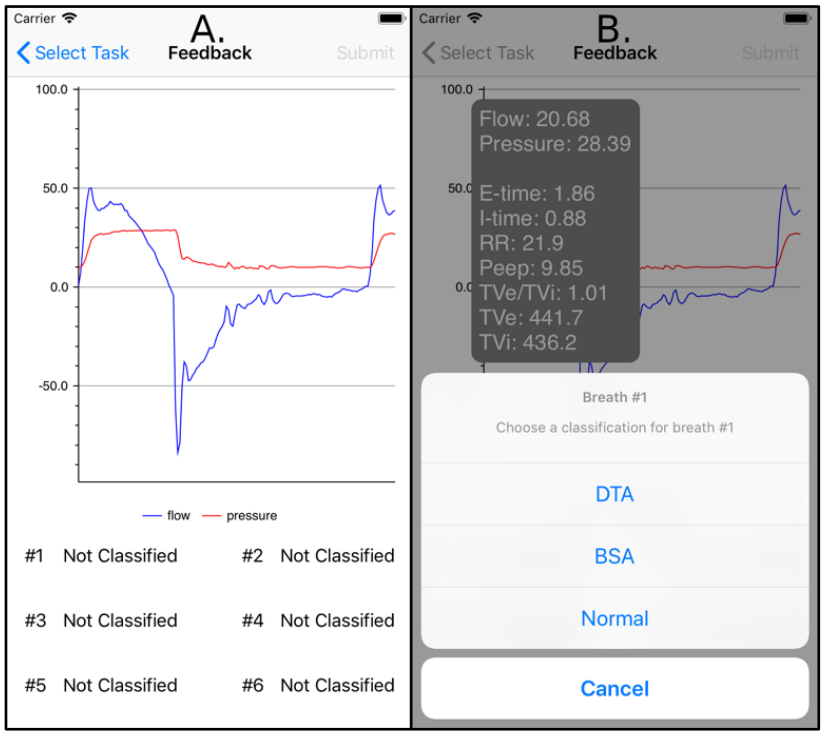


Figure 6.3: **A.** Breaths that were classified ambiguously by the machine learning classifier are displayed to clinicians for clarification. **B.** After selecting a breath, clinicians are presented with relevant breath-level statistics to assist with classification, and a configurable list of breath classes to select.

We utilized Apple™ push notifications to directly alert clinicians to ventilation problems. Alert settings are configurable on the mobile application, allowing clinicians to set separate alert configuration for each patient. This allows each clinician to receive alerts that are relevant to their practice and each patient's physiology. In addition to more traditional alert parameters such as

respiratory rate and tidal volume, we enable alerts for the occurrence of asynchronies such as DTA and BSA that are derived from our ML models,(56) and we employ artifact recognition algorithms to reduce false positive event detection.(54) All these alerts have provider-configurable boundaries and adjustable rolling time windows that can be modified on the device rather than the ventilator and turned on and off as a patient's condition evolves. This ability may prove useful to individualize alert logic and to reduce the occurrence of clinically irrelevant and false alarms.

To address the limited availability of ground truth data sets for ML algorithm development, the application was also designed to include a real time breath annotation mechanism. In the case of an uncertain prediction probability, the application can query the clinician to classify the ambiguous breaths (Figure 6.3). In doing so, the application enables the accumulation of labeled data to improve the alert system's accuracy and usability.

Finally, we built a prototype of the ML-driven ventilation management and alert system using a client-server model with modest cloud computing resources (2 CPUs, 4G memory, Ubuntu 14.04) on Amazon Web Services. Our goal was to investigate the performance of our system when implemented using real-time data processing in a cloud computing framework. We benchmarked the server-side processing delays to complete the following three key operations while simulating 1, 10, and 20 simultaneous patients (20 represents a typical full ICU patent load):

- Micro-batch processing: Time taken to process and store new incoming data (20-breath batch), perform feature extraction, and PVA detection.
- Data Retrieval: Time taken to retrieve 5 minutes of data (ventilator data, breath meta data, and PVAs) from an iPhone application (5 minutes was the default polling interval)
- Alert Processing: Time taken to digest classification results for all patients and generate alerts

For each task, we repeated the experiments 20 times to ensure statistical validity. We found our system was able to perform PVA detection in 1.047 seconds and perform data retrieval and all

alert processing in 0.125, and 0.107 seconds on average for 10 patients (Table 6.1). In general, data retrieval and alert processing time were negligible (sub-seconds) over different loads. Even at full ICU load (20 patients), the average micro-batch processing time was less than 2 seconds and less than 4 second 90% of the time. Given that most breaths on a ventilator last 2-3 seconds, we conclude that our system is capable of real time data processing.

N	Task	Mean (s)	Std (s)	90% (s)	Max (s)
1	Micro-Batch Processing	0.329	0.059	0.329	0.973
	Data Retrieval	0.098	0.031	0.118	0.122
	Alert Processing	0.002	0.000	0.002	0.003
10	Micro-Batch Processing	1.047	0.641	1.789	4.075
	Data Retrieval	0.125	0.145	0.311	0.559
	Alert Processing	0.107	0.079	0.209	0.274
20	Micro-Batch Processing	1.942	1.347	3.457	9.512
	Data Retrieval	0.272	0.220	0.346	2.045
	Alert Processing	0.357	0.283	0.681	0.914

Table 6.1: A summary of the server-side processing delays for three tasks: *Micro-Batch Processing, Data Retrieval*, and *Alert Processing* under different patient loads. The mean, standard deviation, 90th percentile, and maximum delays are reported in seconds, rounding to 3 decimal places.

Nevertheless, our prototype cannot guarantee better than worst case performance (9.512 seconds for 20 patients) due to the lack of dedicated resources. Variations in the processing delay were due to competing background workloads on the same server. This demonstrates the potential implications of using cloud platform for real-time data analytics in an intelligent CDSS system. Future research is needed to further explore the advantages and disadvantages of dedicated edge

computing platforms on premise versus cloud platforms, especially for future application scenarios where the data-driven analytics may be part of a closed loop systems controlling fluids and medication administration, ventilators, or other medical devices where low computation time variance and sub-second latency will be critical.

6.3 Future Work

6.3.1 Background

Our diagnostic modeling for ARDS could also serve as the foundation for improving prognostication of the ARDS disease in patients. Indeed, predictive models for disease can have clinical utility: Diseases such as cancer, sepsis, and ARDS all have increased mortality rates as disease severity increases.(5,13) There is also reason to believe that ARDS treatment protocols may be more effective when used early, especially in patients with moderate-severe disease, and some therapies may not work or even cause harm in patients with less severe ARDS.(24,26,187) In this regard, there are a variety of options available for the management of patients with ARDS. (25,26,188) Attention has thus focused on limiting ARDS progression through early recognition and implementation of ARDS specific treatments. (174,189) However, identifying which patients may benefit from more aggressive treatments is still an open challenge.

To help address this issue, we plan to develop models to predict clinical trajectories amongst patients with ARDS. Currently, we do not know of any best modeling architecture to use for prognostication, but we hypothesize hidden Markov models (HMMs) and advanced ML models such as deep learning will be well suited for the application of time series prognosis with VWD and EHR information.(5). As a result, we propose to explore a variety of HMM and DNN

architectures and modeling techniques to determine which avenue is best for prognosticating ARDS disease state.

Ultimately, our models and approach will not just apply to prognosis of ARDS. Our techniques will likely be broadly generalizable to prognostication of any disease where physicians utilize hospital waveform data to assist their practice. These modeling techniques will also satisfy a need for automated prognostication in critical care medicine. With the increasing digitization of medicine, we have entered an age when healthcare providers are continually presented with diverse types of patient data without the right tools to translate these data into the information necessary to create effective diagnoses and treatment plans.(1,4,190) Automating the prognostication of complex diseases stands to reduce cognitive burden on providers, prevent medical errors, and save patient lives.

6.3.2 Methods

We will definitively identify which patients had ARDS in our cohort of over 500 patients,(122) will then determine the exact time at which ARDS is capable of being diagnosed. Then we will determine the state of ARDS at the time of diagnosis via P/F measurement following the Berlin Criteria P/F severity ranges. Next, we will determine whether the patient deteriorated, improved, or stayed the same in their ARDS state in the next 24 hours post-ARDS onset. To do this, we will utilize oxygen saturation index (OSI) as a surrogate, and more available, measure of P/F ratio.(191,192) We will determine the patient as improving if the mean OSI for the next 24 hours was higher than the mean OSI for the prior 24 hour interval.

Once a ground truth is established, we will use the retrospectively collected VWD first 24 hours of data for ARDS patients, and VWD collected for the subsequent 24-hour period to

evaluate patient respiratory function. We will also utilize EHR derived features identified to be predictive of ARDS diagnosis in *Chapter 2* and *Chapter 4* (static features) and add a temporal dimension to them to by utilizing Calvert's and Kam's methodology of determining when a feature value is increasing, decreasing, or not changing (dynamic features).(193–195) In addition, we will also investigate whether magnitude and rate of change in features such as respiratory compliance will be predictive of worsening ARDS severity.(196,197)

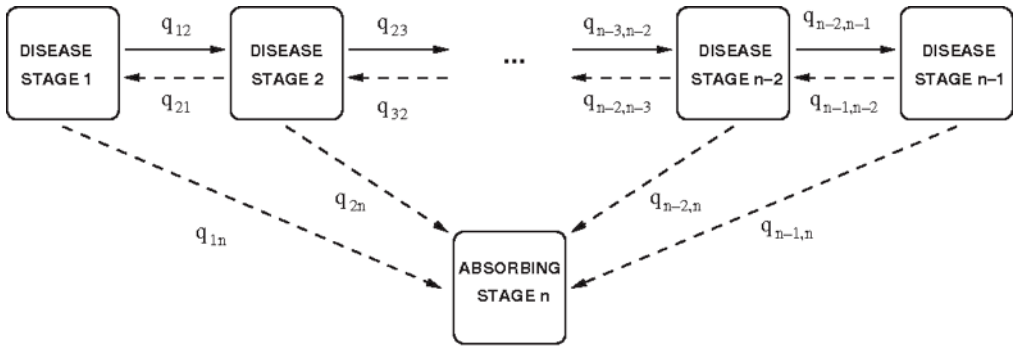


Figure 6.4: A multi-stage hidden Markov model (HMM) typically used to model disease progression in patients. Each disease stage corresponds with a model state, and each q_{ij} is a transition probability that the patient will move from one disease stage to the next. The final "absorbing stage" can be considered as patient death.

Using our dataset and parameterization approach, we will develop an HMM to predict if a subject's ARDS severity class will decrease, increase, or stay static in the next 24 hours. HMM's work by establishing multiple system states that are connected to each other by state transition probabilities.(198–202) These links dictate at what probability a system will move from one state to another.(203) Using a HMM, we will establish a probabilistic model that determines whether a subject will change ARDS severity state in the next 24 hours, and if so to which ARDS disease state, while validating the accuracy of model predictions. We can perform this disease progression modeling in multiple ways, and it is not clear which methodology would be best to use in our current situation. Multi-state disease models are attractive because of their

conceptual simplicity (Figure 6.4), but other models such as an unsupervised learning model proposed by Wang should also be examined.(198,199) We will also pursue an exploratory approach to modeling ARDS progression with deep learning.

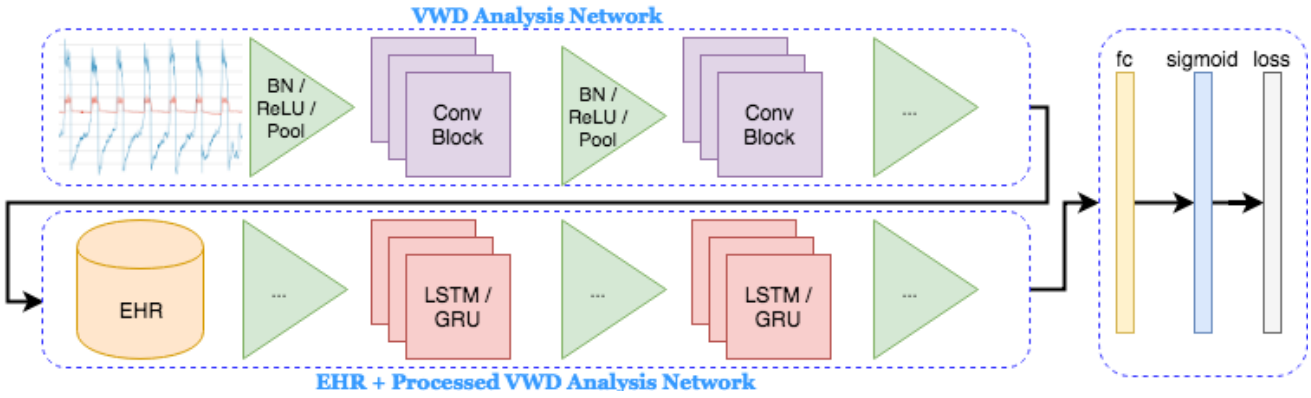


Figure 6.5: Approximation of what our neural network will look like. VWD will first be fed into a preprocessing step involving either batch normalization (BN), pooling, or a rectified linear unit layer. This output will be passed to CNN layers and the process will repeat to derive featurized information from VWD. Then featurized information will be passed to our second network which integrates the featurized information with EHR derived data. These two data sources will then be passed through a series of long short-term memory (LSTM) or gated recurrent unit (GRU) gates. Finally, the output of this network will be classified with a single fully connected and sigmoid layer. Loss will be calculated at the end of the classification process.

Next, we will create a model for predicting change in ARDS severity class using a type of ML known as deep learning. Deep learning models use specialized neural networks that are used to generate novel features from the available "raw" input feature set, thus precluding the necessity of feature extraction via specialized expert knowledge.(111,204–206) For model development we will use multiple different deep learning models: LSTM, GRU, and CNN, that have shown success in extraction of temporal patterns in EHR, and in the early detection of sepsis.(5,194,207) Given that important physiologic information such as respiratory compliance is embedded in VWD, we hypothesize that CNN networks will be able to identify pertinent features including temporal patterns diagnostic of worsening severity of ARDS after the patient has met Berlin criteria (Figure 6.5). As described above, we will use OSI data to group ARDS

patients in our cohort according to whether they worsened, remained stable, or improved in ARDS severity class in subsequent 24 after ARDS onset. A “raw” feature set of static and temporally dynamic features derived from VWD and EHR data will be used for model development. Once completed, we will test the hypothesis that our deep learning model will have better sensitivity and specificity than the HMM for prognosticating ARDS severity class.

6.3.3 Discussion

We hypothesize that a HMM, using the best features identified in *Chapter 2* and *Chapter 4* and additional temporal information, will be able to accurately predict changes in ARDS severity. We hypothesize that the addition of deep learning will allow us to computationally derive additional predictive information that will result in superior predictive performance of our model compared to an HMM.

It is possible that neither HMM nor deep learning models will accurately predict changes in ARDS severity. Despite this possibility, the promising performance of our ML models in distinguishing between patients with and without ARDS strongly suggests that use of physiologic features will be useful in predicting changes in clinical trajectories in ARDS patients. Our use of static and dynamic features creates a novel temporal framework for predicting changes in ARDS, and the use of advanced ML methods that can discover useful temporal patterns in a raw dataset to develop novel predictive features may further mitigate risk of model failure. Finally, the use of a 24-hour window in which to assess for change in ARDS severity during model development may be overly restrictive and compromise model training. It possible that a longer time frame (e.g., 72 or 96 hours from ARDS onset) could better separate patients into groups with distinct trajectories, leading to better model training and thus prognostic

performance, which may require further model development and validation experiments with different cutoffs.

Beyond our specific use modeling, our proposed work in predicting clinical trajectories in ARDS could lead to more appropriate treatment of patients with ARDS, and it could also contribute to the science of using new predictive models for other diseases such as chronic obstructive pulmonary disease (COPD), sepsis, or cancer.(163,194,208) These types of predictive models will become increasingly important as medicine becomes more personalized by utilizing high frequency data collection and machine learning analytics to generate highly accurate and actionable prognostic information from a wealth of data.

6.4 Summary

ARDS is a prevalent and deadly disease that affected approximately 10% of ICU admissions pre-COVID, and with mortality rates of 29.7%-42.9%. Despite prevalence and severity, ARDS remains widely underrecognized and only 60% of those with ARDS are ever recognized as having the syndrome during their hospitalization. It is suspected that the high mortality rates of ARDS patients are partially because of under recognition, yet despite years of improvements in ARDS clinical definitions, unaided clinicians remain deficient at timely ARDS diagnosis. Automated diagnostic solutions have been proposed to ameliorate the problem of ARDS diagnosis; however, their performance varies highly based on inter-center radiologist practices, availability of necessary EHR data, timeliness of data reporting, and subjectivity of the Berlin criteria.

We propose to improve this situation by utilizing VWD as a primary or surrogate mechanism for ARDS detection. VWD is unbiased, pervasive, and ubiquitous for patients with

respiratory failure. Our results in *Chapter 4* and *Chapter 5* show that an automated, ML-based ARDS screening algorithm using VWD can detect ARDS with strong discrimination performance within the first 24 hours after Berlin criteria. Our model functioned without need for a CXR, ABG, or other EHR-derived data, and displays VWD shows promise for use as a digital biomarker of ARDS pathophysiology. While our results represent a first proof-of-concept that digital biomarkers derived from physiologic monitoring data can be used for early ARDS detection, additional research is needed to determine how broadly such methods can be applied, how best to incorporate them into traditional Berlin criteria-based diagnostic workflows, and how they might complement EHR and biochemical approaches to clinical phenotyping and prognosis.

This dissertation is broadly aimed at showing the research challenges we solved and software that we built to create our ML-based ARDS detection model in *Chapter 4*. ML models require large amounts of labeled data to function well. So, we developed APL to solve this issue in the use-case of VWD. In *Chapter 2.2* we showed how APL represents a new, effective tool to improve the consistency and speed of generating multi-reviewer gold standard VWD datasets that will help drive the development of novel CDS systems. With the use of APL, we created 2 datasets comprising over 300,000 annotated breaths for classifying PVA and VM. Our classifiers for PVA in *Chapter 2.3* and VM in *Chapter 2.4* will facilitate development of more advanced automated MV CDSS to improve the management of patients experiencing respiratory failure.

Patients with respiratory failure need extensive physiologic monitoring of vital signals. One such signal that can be measured non-invasively is respiratory compliance, which can also help warn and monitor for onset and progression of ARDS. In *Chapter 3* we provided further clinical validation of a set of 15 algorithms used to monitor respiratory compliance based on the

single-chamber model of the lung. Our exploration is intended to give scientists and clinicians more faith in performance of the single-chamber model under varying clinical scenarios for both clinical purposes and so we can improve the featurization of our ARDS detection model explored in *Chapter 4*.

Besides better feature engineering, performance of ML algorithms can also be improved by applying more modern ML techniques such as deep learning. Classical ML algorithms operate largely using potentially biased, manually crafted features based on clinical knowledge. Deep learning in contrast, promises to potentially remove this bias by extracting features directly from raw data such as VWD. So, in *Chapter 5* we explored the hypothesis of whether our ARDS ML classifier would benefit from a more unbiased feature set. Our work gives evidence that deep learning may have advantages to expert-derived featurization in extracting physiologically useful information from raw ventilator waveform data. Furthermore, our use of frequency ablation to highlight salient waveform information may have potential for future applications of physiologic waveform data analysis so that scientists may be able to better understand the performance impact of different waveform morphologies during model development.

Next, we incorporated our improved ARDS detection model into an automated platform for collecting, monitoring, and performing diagnosis on physiologic data collected in the ICU. Our work in *Chapters 6.1 and 6.2* showcases a proof-of-concept system for future improvements in healthcare delivery where data will be continuously streamed from the bedside, analyzed in the cloud, and returned to clinicians at the point of care in the form of actionable diagnostic and predictive alerts. In this regard, we envision a future where CDSS are designed specifically around IoT sensors, cloud computing and EHR integration, and mobile device-based access to CDSS feedback in a "provider-in-the-loop" implementation framework where inaccurate decisions made

by ML algorithms can be corrected by clinicians to continuously refine algorithm performance over time.

The development of effective healthcare models will in turn depend on clinical studies testing CDSS effectiveness. Such studies will evaluate potential improvements in care gained from rapidly alerting physicians to events such as PVA or diagnoses like ARDS. These trials will be a key part of future learning healthcare systems that will design, test, and implement automated CDSS, where data will be continuously streamed from the bedside, analyzed in the cloud, and returned to clinicians at the point of care in the form of actionable diagnostic and predictive alerts. In this regard, we envision a future where CDSS are designed specifically around IoT sensors, cloud computing and EHR integration, and mobile device-based access to CDSS feedback in a "provider-in-the-loop" implementation framework where inaccurate decisions made by ML algorithms can be corrected by clinicians to continuously refine algorithm performance over time.

In conclusion, we have developed an end-to-end system for developing and exploring the use of VWD in the application of ARDS detection. Our system highlights methodologies to explore the use of future data technologies in the ICU for purpose of syndrome detection using non-invasive VWD. These methodologies will help create advanced CDSS that will reduce the cognitive burden on care providers, improve quality of care, reduce patient suffering, and reduce overall hospital mortality from deadly diseases such as ARDS.

References

1. Drew BJ, Harris P, Zègre-Hemsey JK, Mammone T, Schindler D, Salas-Boni R, et al. Insights into the problem of alarm fatigue with physiologic monitor devices: A comprehensive observational study of consecutive intensive care unit patients. PLoS One. 2014;9(10).
2. Phillips JA, Barnsteiner JH. Clinical alarms: Improving efficiency and effectiveness. Vol. 28, Critical Care Nursing Quarterly. 2005.
3. Blum JM, Kruger GH, Sanders KL, Gutierrez J, Rosenberg AL. Specificity improvement for network distributed physiologic alarms based on a simple deterministic reactive intelligent agent in the critical care environment. J Clin Monit Comput. 2009;23(1).
4. Graham KC, Cvach M. Monitor alarm fatigue: Standardizing use of physiological monitors anddecreasing nuisance alarms. Am J Crit Care. 2010;19(1).
5. Ravi D, Wong C, Deligianni F, Berthelot M, Andreu-Perez J, Lo B, et al. Deep Learning for Health Informatics. IEEE J Biomed Heal Informatics. 2017;21(1).
6. Murdoch TB, Detsky AS. The inevitable application of big data to health care. Vol. 309, JAMA - Journal of the American Medical Association. 2013.
7. Berner ES, Lande TJ La. Overview of Clinical Decision Support Systems. Clin Decis Support Syst. 2007;3–22.
8. Raghupathi W, Raghupathi V. Big data analytics in healthcare: promise and potential. Heal Inf Sci Syst. 2014;2(1):1–10.
9. Raghupathi W. Health Care Information Systems. Commun ACM. 1997;40(8).
10. Chawla N V., Davis DA. Bringing big data to personalized healthcare: A patient-centered

framework. Vol. 28, Journal of General Internal Medicine. 2013.

11. Beitler JR, Goligher EC, Schmidt M, Spieth PM, Zanella A, Martin-Loeches I, et al. Personalized medicine for ARDS: the 2035 research agenda. Intensive Care Med. 2016;42(5):756–67.

12. Mathay MA, Zemans RL, Zimmerman GA, Arabi YM, Beitler JR, Mercat A, et al. Acute Respiratory Distress Syndrome. Nat Rev Dis Prim. 2019;5(1):18.

13. Bellani G, Laffey JG, Pham T, Fan E, Brochard L, Esteban A, et al. Epidemiology, patterns of care, and mortality for patients with acute respiratory distress syndrome in intensive care units in 50 countries. JAMA - J Am Med Assoc. 2016;315(8):788–800.

14. Kor DJ, Carter RE, Park PK, Festic E, Banner-Goodspeed VM, Hinds R, et al. Effect of aspirin on development of ARDS in at-risk patients presenting to the emergency department the LIPS-a randomized clinical trial. JAMA - J Am Med Assoc. 2016;315(22):2406–14.

15. Pierrakos. Acute Respiratory Distress Syndrome: Pathophysiology and Therapeutic Options. J Clin Med Res. 2012;

16. Thompson BT, Chambers RC, Liu KD. Acute respiratory distress syndrome. N Engl J Med. 2017;377(6):562–72.

17. Yadav H, Thompson BT, Gajic O. Is acute respiratory distress syndrome a preventable disease? Am J Respir Crit Care Med. 2017;195(6):725–36.

18. Stapleton RD, Wang BM, Hudson LD, Rubenfeld GD, Caldwell ES, Steinberg KP. Causes and timing of death in patients with ARDS. Chest. 2005;128(2).

19. Donahoe M. Acute respiratory distress syndrome: A clinical review. Vol. 1, Pulmonary Circulation. 2011.

20. Weiss CH, Baker DW, Weiner S, Bechel M, Ragland M, Rademaker A, et al. Low tidal volume ventilation use in acute respiratory distress syndrome. Crit Care Med. 2016;44(8).

21. ADTF. Acute respiratory distress syndrome: The Berlin definition. JAMA - J Am Med Assoc. 2012;307(23):2526–33.

22. Lanspa MJ, Gong MN, Schoenfeld DA, Lee KT, Grissom CK, Hou PC, et al. Prospective assessment of the feasibility of a trial of low–tidal volume ventilation for patients with acute respiratory failure. Ann Am Thorac Soc. 2019;16(3).

23. Sjoding MW, Hyzy RC. Recognition and appropriate treatment of the acute respiratory distress syndrome remains unacceptably low. Vol. 44, Critical Care Medicine. 2016.

24. Bellani G, Pham T, Laffey JG. Missed or delayed diagnosis of ARDS: a common and serious problem. Vol. 46, Intensive Care Medicine. 2020.

25. Brower RG, Matthay MA, Morris A, Schoenfeld D, Thompson BT, Wheeler A. Ventilation with lower tidal volumes as compared with traditional tidal volumes for acute lung injury and the acute respiratory distress syndrome. N Engl J Med. 2000;342(18):1301–8.

26. Papazian L, Forel J-M, Gacouin A, Penot-Ragon C, Perrin G, Loundou A, et al. Neuromuscular Blockers in Early Acute Respiratory Distress Syndrome. N Engl J Med. 2010;363(12).

27. Sinha P, Delucchi KL, Thompson BT, McAuley DF, Matthay MA, Calfee CS. Latent class analysis of ARDS subphenotypes: a secondary analysis of the statins for acutely injured lungs from sepsis (SAILS) study. Intensive Care Med [Internet]. 2018;44(11):1859–69. Available from: https://doi.org/10.1007/s00134-018-5378-3

28. Azzam HC, Khalsa SS, Urbani R, Shah C V., Christie JD, Lanken PN, et al. Validation

Study of an Automated Electronic Acute Lung Injury Screening Tool. J Am Med Informatics Assoc. 2009;16(4):503–8.

29. Herasevich V, Yilmaz M, Khan H, Hubmayr RD, Gajic O. Validation of an electronic surveillance system for acute lung injury. Intensive Care Med. 2009;35(6):1018–23.

30. Wayne MT, Valley TS, Cooke CR, Sjoding MW. Electronic sniffer systems to identify the acute respiratory distress syndrome. Ann Am Thorac Soc. 2019;16(4):488–95.

31. Zeiberg D, Prahlad T, Nallamothu BK, Iwashyna TJ, Wiens J, Sjoding MW. Machine learning for patient risk stratification for acute respiratory distress syndrome. PLoS One. 2019;14(3):1–14.

32. Le S, Pellegrini E, Green-Saxena A, Summers C, Hoffman J, Calvert J, et al. Supervised machine learning for the early prediction of acute respiratory distress syndrome (ARDS). J Crit Care [Internet]. 2020;60:96–102. Available from: https://doi.org/10.1016/j.jcrc.2020.07.019

33. McKown AC, Brown RM, Ware LB, Wanderer JP. External Validity of Electronic Sniffers for Automated Recognition of Acute Respiratory Distress Syndrome. J Intensive Care Med. 2017;1–9.

34. Yetisgen-Yildiz M, Gunn ML, Xia F, Payne TH. A text processing pipeline to extract recommendations from radiology reports. J Biomed Inform. 2013;46(2).

35. Yetisgen-yildiz M, Bejan CA, Wurfel MM. Identification of Patients with Acute Lung Injury from Free-Text Chest X-Ray Reports. Work BioNLP. 2013;(2009).

36. Sjoding MW, Hofer TP, Co I, Courey A, Cooke CR, Iwashyna TJ. Interobserver Reliability of the Berlin ARDS Definition and Strategies to Improve the Reliability of ARDS Diagnosis. Chest [Internet]. 2018;153(2):361–7. Available from:

https://doi.org/10.1016/j.chest.2017.11.037

37. Scholten EL, Beitler JR, Prisk GK, Malhotra A. Treatment of ARDS With Prone Positioning. Vol. 151, Chest. 2017.

38. Neto AS, Cardoso SO, Manetta JA, Pereira VGM, Espósito DC, Pasqualucci MDOP, et al. Association between use of lung-protective ventilation with lower tidal volumes and clinical outcomes among patients without acute respiratory distress syndrome: A meta-analysis. JAMA - J Am Med Assoc. 2012;308(16).

39. Choi E, Biswal S, Malin B, Duke J, Stewart WF, Sun J. Generating Multi-label Discrete Patient Records using Generative Adversarial Networks. 2017;68:1–20. Available from: http://arxiv.org/abs/1703.06490

40. Fan-Minogue H, Maslove D, Lamb P, Levitt J, Paik D, Rubin D. Extracting Computational and Semantic Features from Portable Chest X-rays for Diagnosis of Acute Respiratory Distress Syndrome. AMIA Jt Summits Transl Sci proceedings AMIA Jt Summits Transl Sci. 2013;2013.

41. Afshar M, Joyce C, Oakey A, Formanek P, Yang P, Churpek MM, et al. A Computable Phenotype for Acute Respiratory Distress Syndrome Using Natural Language Processing and Machine Learning. AMIA . Annu Symp proceedings AMIA Symp. 2018;2018:157–65.

42. Chbat NW, Chu W, Ghosh M, Li G, Li M, Chiofolo CM, et al. Clinical knowledge-based inference model for early detection of acute lung injury. Ann Biomed Eng. 2012;40(5):1131–41.

43. Reamaroon N, Sjoding MW, Lin K, Iwashyna TJ, Najarian K. Accounting for label uncertainty in machine learning for detection of acute respiratory distress syndrome. IEEE

J Biomed Heal Informatics. 2019;23(1):407–15.

44. Solti I, Cooke CR, Xia F, Wurfel MM. Automated classification of radiology reports for acute lung injury: Comparison of keyword and machine learning based natural language processing approaches. In: Proceedings - 2009 IEEE International Conference on Bioinformatics and Biomedicine Workshops, BIBMW 2009. 2009.

45. Lucangelo U, Bernabé F, Blanch L. Respiratory mechanics derived from signals in the ventilator circuit. Respir Care [Internet]. 2005;50(1):55--67. Available from: http://www.ncbi.nlm.nih.gov/pubmed/15636645

46. Epstein SK. How often does patient-ventilator asynchrony occur and what are the consequences? Respir Care. 2011;56(1).

47. van Drunen EJ, Chiew YS, Chase JG, Shaw GM, Lambermont B, Janssen N, et al. Expiratory model-based method to monitor ARDS disease state. Biomed Eng Online. 2013;12(1):1–15.

48. Riviello ED, Kiviri W, Twagirumugabe T, Mueller A, Banner-Goodspeed VM, Officer L, et al. Hospital incidence and outcomes of the acute respiratory distress syndrome using the Kigali modification of the Berlin definition. Am J Respir Crit Care Med. 2016;193(1):52–9.

49. Colombo D, Cammarota G, Alemani M, Carenzo L, Barra FL, Vaschetto R, et al. Efficacy of ventilator waveforms observation in detecting patient-ventilator asynchrony. Crit Care Med. 2011;39(11).

50. Hinton G. Deep Learning—A Technology With the Potential to Transform Health Care. JAMA - J Am Med Assoc. 2018;320(11):1101–2.

51. Gutierrez G, Ballarino GJ, Turkan H, Abril J, De La Cruz L, Edsall C, et al. Automatic

detection of patient-ventilator asynchrony by spectral analysis of airway flow. Crit Care. 2011;15(4).

52. Blanch L, Villagra A, Sales B, Montanya J, Lucangelo U, Luján M, et al. Asynchronies during mechanical ventilation are associated with mortality. Intensive Care Med. 2015;41(4):633–41.

53. Beitler JR, Sands SA, Loring SH, Robert L, Malhotra A, Spragg RG, et al. Breath Stacking Dyssynchrony in ARDS : the BREATHE Criteria. 2017;42(9):1427–36.

54. Adams JY, Lieng MK, Kuhn BT, Rehm GB, Guo EC, Taylor SL, et al. Development and Validation of a Multi-Algorithm Analytic Platform to Detect Off-Target Mechanical Ventilation. Sci Rep. 2017;7(1):1–11.

55. Sottile PD, Albers D, Higgins C, Mckeehan J, Moss MM. The Association between Ventilator Dyssynchrony, Delivered Tidal Volume, and Sedation using a Novel Automated Ventilator Dyssynchrony Detection Algorithm. Crit Care Med. 2018;46(2):e151–7.

56. Rehm G, Han J, Kuhn B, Delplanque J-P, Anderson N, Adams J, et al. Creation of a Robust and Generalizable Machine Learning Classifier for Patient Ventilator Asynchrony. Methods Inf Med. 2018;57(04):208–19.

57. Chanques G, Kress JP, Pohlman A, Patel S, Poston J, Jaber S, et al. Impact of ventilator adjustment and sedation-analgesia practices on severe asynchrony in patients ventilated in assist-control mode. Crit Care Med. 2013;41(9).

58. Beitler JR, Sands SA, Loring SH, Owens RL, Malhotra A, Spragg RG, et al. Quantifying unintended exposure to high tidal volumes from breath stacking dyssynchrony in ARDS: the BREATHE criteria. Intensive Care Med. 2016;42(9).

59. Mulqueeny Q, Ceriana P, Carlucci A, Fanfulla F, Delmastro M, Nava S. Automatic detection of ineffective triggering and double triggering during mechanical ventilation. Intensive Care Med. 2007;33(11).

60. Koenig HC, Finkel BB, Khalsa SS, Lanken PN, Prasad M, Urbani R, et al. Performance of an automated electronic acute lung injury screening system in intensive care unit patients. Crit Care Med. 2011;39(1).

61. Sinderby C, Liu S, Colombo D, Camarotta G, Slutsky AS, Navalesi P, et al. An automated and standardized neural index to quantify patient-ventilator interaction. Crit Care. 2013;17(5).

62. Piquilloud L, Jolliet P, Revelly JP. Automated detection of patient-ventilator asynchrony: New tool or new toy? Vol. 17, Critical Care. 2013.

63. Kim B, Wattenberg M, Gilmer J, Cai C, Wexler J, Viegas F, et al. Interpretability beyond feature attribution: Quantitative Testing with Concept Activation Vectors (TCAV). 35th Int Conf Mach Learn ICML 2018. 2018;6:4186–95.

64. Gramfort A, Luessi M, Larson E, Engemann DA, Strohmeier D, Brodbeck C, et al. MNE software for processing MEG and EEG data. Neuroimage. 2014;86.

65. Letunic I, Bork P. Interactive Tree of Life v2: Online annotation and display of phylogenetic trees made easy. Nucleic Acids Res. 2011;39(SUPPL. 2).

66. Ernst J, Kellis M. Large-scale imputation of epigenomic datasets for systematic annotation of diverse human tissues. Nat Biotechnol. 2015;33(4).

67. Fouse AS, Weibel N, Hutchins E, Hollan JD. ChronoViz: A system for supporting navigation of time-coded data. In: Conference on Human Factors in Computing Systems - Proceedings. 2011.

68. Sousa E, Malheiro T, Bicho E, Erlhagen W, Santos J, Pereira A. MUVTIME: A Multivariate Time Series Visualizer for Behavioral Science. In 2016.

69. Stockbridge N, Brown BD. Annotated ECG waveform data at FDA. In: Journal of Electrocardiology. 2004.

70. Brugman H, Russel A. Annotating multi-media/multi-modal resources with ELAN. In: Proceedings of the 4th International Conference on Language Resources and Evaluation, LREC 2004. 2004.

71. Müller C, Strube M. Multi-level annotation of linguistic data with MMAX2. Corpus Technol Lang Pedagog New Resour new tools, new methods 3. 2006;

72. Winslow RL, Granite S, Jurado C. WaveformECG: A platform for visualizing, annotating, and analyzing ECG data. Comput Sci Eng. 2016;18(5).

73. Russell BC, Torralba A, Murphy KP, Freeman WT. LabelMe: A database and web-based tool for image annotation. Int J Comput Vis. 2008;77(1–3).

74. Götz S, García-Gómez JM, Terol J, Williams TD, Nagaraj SH, Nueda MJ, et al. High-throughput functional annotation and data mining with the Blast2GO suite. Nucleic Acids Res. 2008;36(10).

75. Kilicoglu H, Demner-Fushman D. Bio-SCoRes: A smorgasbord architecture for coreference resolution in biomedical text. PLoS One. 2016;11(3).

76. Murias G, Montanyà J, Chacón E, Estruga A, Subirà C, Fernández R, et al. Automatic detection of ventilatory modes during invasive mechanical ventilation. Crit Care. 2016;20(1).

77. Thille AW, Rodriguez P, Cabello B, Lellouche F, Brochard L. Patient-ventilator asynchrony during assisted mechanical ventilation. Intensive Care Med.

2006;32(10):1515–22.

78. Slutsky AS, Ranieri VM. Ventilator-Induced Lung Injury. N Engl J Med [Internet]. 2013;369(22):2126–36. Available from: http://www.nejm.org/doi/10.1056/NEJMra1208707

79. Rittayamai N, Katsios CM, Beloncle F, Friedrich JO, Mancebo J, Brochard L. Pressure-Controlled vs Volume-Controlled Ventilation in Acute Respiratory Failure. Chest. 2015;148(2).

80. Murias G, Lucangelo U, Blanch L. Patient-ventilator asynchrony. Vol. 22, Current Opinion in Critical Care. 2016.

81. Mellott KG, Grap MJ, Munro CL, Sessler CN, Wetzel PA, Nilsestuen JO, et al. Patient ventilator asynchrony in critically ill adults: Frequency and types. Hear Lung J Acute Crit Care. 2014;43(3).

82. Grieco DL, Bitondo MM, Aguirre-Bermeo H, Italiano S, Idone FA, Moccaldo A, et al. Patient-ventilator interaction with conventional and automated management of pressure support during difficult weaning from mechanical ventilation. J Crit Care. 2018;48.

83. Pham T, Brochard LJ, Slutsky AS. Mechanical Ventilation: State of the Art. Vol. 92, Mayo Clinic Proceedings. 2017.

84. Gattinoni L, Marini JJ, Pesenti A, Quintel M, Mancebo J, Brochard L. The “baby lung” became an adult. Vol. 42, Intensive Care Medicine. 2016.

85. Desai JP, Moustarah F. Pulmonary compliance. StatPearls [Internet]. 2020;

86. West JB. Pulmonary pathophysiology: the essentials. Lippincott Williams & Wilkins; 2011.

87. Chiew YS, Chase JG, Shaw GM, Sundaresan A, Desaive T. Model-based PEEP

optimisation in mechanical ventilation. Biomed Eng Online. 2011;10:1–16.

88. Hess DR. Respiratory mechanics in mechanically ventilated patients. Respir Care. 2014;59(11):1773–94.

89. Kannangara DO, Newberry F, Howe S, Major V, Redmond D, Szlavecs A, et al. Estimating the true respiratory mechanics during asynchronous pressure controlled ventilation. Biomed Signal Process Control [Internet]. 2016;30:70–8. Available from: http://dx.doi.org/10.1016/j.bspc.2016.06.014

90. Lourens MS, Van Den Berg B, Hoogsteden HC, Bogaard JM. Flow-volume curves as measurement of respiratory mechanics during ventilatory support: The effect of the exhalation valve. Intensive Care Med. 1999;25(8):799–804.

91. Vialet R, Monnier A, Lagier P, Jammes Y, Toméi C, Martin C. Passive expiratory flow-volume curve is not an accurate method to measure the respiratory time constant in rabbits. Acta Anaesthesiol Scand. 1999;43(10):1017–20.

92. Redmond DP, Chiew YS, Major V, Chase JG. Evaluation of model-based methods in estimating respiratory mechanics in the presence of variable patient effort. Comput Methods Programs Biomed. 2019;171:67–79.

93. Chiew YS, Pretty C, Docherty PD, Lambermont B, Shaw GM, Desaive T, et al. Time-varying respiratory system elastance: A physiological model for patients who are spontaneously breathing. PLoS One. 2015;10(1):1–13.

94. Larraza S, Dey N, Karbing DS, Jensen JB, Nygaard M, Winding R, et al. A mathematical model approach quantifying patients' response to changes in mechanical ventilation: Evaluation in volume support. Med Eng Phys [Internet]. 2015;37(4):341–9. Available from: http://dx.doi.org/10.1016/j.medengphy.2014.12.006

95. Damanhuri NS, Docherty PD, Chiew YS, Van Drunen EJ, Othman NA, Desaive T, et al. Relation of respiratory system elastance and expiratory time constant: Are they from the same lung? IFAC Proc Vol. 2014;19(August):4784–9.

96. Vicario F, Buizza R, Truschel WA, Chbat NW. Noninvasive estimation of alveolar pressure. Proc Annu Int Conf IEEE Eng Med Biol Soc EMBS. 2016;2016-Octob:2721–4.

97. Vicario F, Albanese A, Karamolegkos N, Wang D, Seiver A, Chbat NW. Noninvasive estimation of respiratory mechanics in spontaneously breathing ventilated patients: A constrained optimization approach. IEEE Trans Biomed Eng. 2016;63(4):775–87.

98. Al-Rawas N, Banner MJ, Euliano NR, Tams CG, Brown J, Martin AD, et al. Expiratory time constant for determinations of plateau pressure, respiratory system compliance, and total resistance. Crit Care [Internet]. 2013;17(1):R23. Available from: http://ccforum.com/content/17/1/R23

99. Brunner JX, Laubscher TP, Banner MJ, Iotti G, Braschi A. Simple method to measure total expiratory time constant based on the passive expiratory flow-volume curve. Crit Care Med. 1995;23(6):1117–22.

100. Lourens M, Van den Berg B, Aerts J, Verbraak A, Hoogsteden H, Bogaard J. Expiratory time constants in mechanically ventilated patients with and without COPD. Intensive Care Med. 2000;26(11):1612–8.

101. Babuška R, Alic L, Lourens MS, Verbraak AFM, Bogaard J. Estimation of respiratory parameters via fuzzy clustering. Artif Intell Med. 2001;21(1–3):91–105.

102. Newberry F, Kannangara O, Howe S, Major V, Redmond D, Szlavecz A, et al. Iterative interpolative pressure reconstruction for improved respiratory mechanics estimation during asynchronous volume controlled ventilation. IFMBE Proc. 2016;56:133–9.

103. Redmond DP, Major V, Corbett S, Glassenbury D, Beatson A, Szlavecz A, et al. Pressure reconstruction by eliminating the demand effect of spontaneous respiration (PREDATOR) method for assessing respiratory mechanics of reverse-triggered breathing cycles. IECBES 2014, Conf Proc - 2014 IEEE Conf Biomed Eng Sci “Miri, Where Eng Med Biol Humanit Meet.” 2014;(April):332–7.

104. Howe SL, Chase JG, Redmond DP, Morton SE, Kim KT, Pretty C, et al. Inspiratory respiratory mechanics estimation by using expiratory data for reverse-triggered breathing cycles. Comput Methods Programs Biomed [Internet]. 2020;186:105184. Available from: https://doi.org/10.1016/j.cmpb.2019.105184

105. Major V, Corbett S, Redmond D, Beatson A, Glassenbury D, Chiew YS, et al. Respiratory mechanics assessment for reverse-triggered breathing cycles using pressure reconstruction. Biomed Signal Process Control [Internet]. 2016;23:1–9. Available from: http://dx.doi.org/10.1016/j.bspc.2015.07.007

106. Vicario F, Albanese A, Wang D, Karamolegkos N, Chbat NW. Simultaneous Parameter and Input Estimation of a Respiratory Mechanics Model. Model Simul Optim Complex Process HPSC 2015. 2017;235–47.

107. Briel M, Meade M, Mercat A, Brower RG, Talmor D, Walter SD, et al. Higher vs lower positive end-expiratory pressure in patients with acute lung injury and acute respiratory distress syndrome: Systematic review and meta-analysis. Vol. 303, JAMA - Journal of the American Medical Association. 2010.

108. Sundaresan A, Chase JG, Shaw GM, Chiew YS, Desaive T. Model-based optimal PEEP in mechanically ventilated ARDS patients in the Intensive Care Unit. Biomed Eng Online. 2011;10.

109. Jiang X, Coffee M, Bari A, Wang J, Jiang X. Towards an Artificial Intelligence Framework for Data-Driven Prediction of Coronavirus Clinical Severity. Comput Mater Contin. 2020;63(1):537–51.

110. Rehm GR, Cortés-Puch I, Kuhn BT, Nguyen J, Fazio SA, Johnson MA, et al. Use of Machine Learning to Screen for Acute Respiratory Distress Syndrome Using Raw Ventilator Waveform Data. Crit Care Explor. 2021;3(1):1–12.

111. Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. Adv Neural Inf Process Syst. 2012;25:1097–105.

112. Acharya UR, Fujita H, Lih OS, Hagiwara Y, Tan JH, Adam M. Automated detection of arrhythmias using different intervals of tachycardia ECG segments with convolutional neural network. Inf Sci (Ny). 2017;405(April):81–90.

113. Spampinato C, Palazzo S, Kavasidis I, Giordano D, Souly N, Shah M. Deep learning human mind for automated visual classification. Proc - 30th IEEE Conf Comput Vis Pattern Recognition, CVPR 2017. 2017;2017-Janua:4503–11.

114. Kiranyaz S, Ince T, Hamila R, Gabbouj M. Convolutional Neural Networks for patient-specific ECG classification. Proc Annu Int Conf IEEE Eng Med Biol Soc EMBS. 2015;2015-Novem:2608–11.

115. Zebin T, Chaussalet TJ. Design and implementation of a deep recurrent model for prediction of readmission in urgent care using electronic health records. 2019 IEEE Conf Comput Intell Bioinforma Comput Biol CIBCB 2019. 2019;

116. Lee W, Na J, Kim G. Multi-task self-supervised object detection via recycling of bounding box annotations. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. 2019;2019-June:4979–88.

117. Papandreou G, Chen LC, Murphy KP, Yuille AL. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. Proc IEEE Int Conf Comput Vis. 2015;2015 Inter:1742–50.

118. Tomlinson DR, Bashir Y, Betts TR, Rajappan K. Accuracy of manual QRS duration assessment: Its importance in patient selection for cardiac resynchronization and implantable cardioverter defibrillator therapy. Europace. 2009;11(5).

119. Baumgartner WA, Cohen KB, Fox LM, Acquaah-Mensah G, Hunter L. Manual curation is not sufficient for annotation of genomic databases. In: Bioinformatics. 2007.

120. Bird C, Bachmann A, Rahman F, Bernstein A. LINKSTER: Enabling efficient manual inspection and annotation of mined data. In: Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering. 2010.

121. Ramirez II, Arellano DH, Adasme RS, Landeros JM, Salinas FA, Vargas AG, et al. Ability of ICU health-care professionals to identify patient-ventilator asynchrony using waveform analysis. Respir Care. 2017;62(2):144–9.

122. Rehm GB, Kuhn BT, Delplanque JP, Guo EC, Lieng MK, Nguyen J, et al. Development of a research-oriented system for collecting mechanical ventilator waveform data. J Am Med Informatics Assoc. 2018;25(3):295–9.

123. Merkel D. Docker: lightweight Linux containers for consistent development and deployment. Vol. 2014, Linux Journal. 2014.

124. Pohlman MC, McCallister KE, Schweickert WD, Pohlman AS, Nigos CP, Krishnan JA, et al. Excessive tidal volume from breath stacking during lung-protective ventilation for acute lung injury. Crit Care Med. 2008;36(11).

125. Blanch L, Sales B, Montanya J, Lucangelo U, Oscar GE, Villagra A, et al. Validation of

the Better Care® system to detect ineffective efforts during expiration in mechanically ventilated patients: A pilot study. Intensive Care Med. 2012;38(5).

126. Gholami B, Phan TS, Haddad WM, Cason A, Mullis J, Price L, et al. Replicating human expertise of mechanical ventilation waveform analysis in detecting patient-ventilator cycling asynchrony using machine learning. Comput Biol Med [Internet]. 2018;97(April):137–44. Available from: https://doi.org/10.1016/j.compbiomed.2018.04.016

127. Rehm GB, Kuhn BT, Nguyen J, Anderson NR, Chuah CN, Adams JY. Improving mechanical ventilator clinical decision support systems with a machine learning classifier for determining ventilator mode. In: Studies in Health Technology and Informatics. 2019.

128. Blanch A, Balada F, Aluja A. Presentation and AcqKnowledge: An application of software to study human emotions and individual differences. Comput Methods Programs Biomed. 2013;110(1).

129. Kim JG, Snodgrass M, Pietrowicz M, Karahalios K. Visual analysis of relationships between behavioral and physiological sensor data. In: Proceedings - 2015 IEEE International Conference on Healthcare Informatics, ICHI 2015. 2015.

130. Hienert D, Wegener D, Schomisch S. Making sense of open data statistics with information from Wikipedia. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2013.

131. Benedek M, Kaernbach C. Decomposition of skin conductance data by means of nonnegative deconvolution. Psychophysiology. 2010;47(4).

132. Yu C, Zhong Y, Smith T, Park I, Huang W. Visual data mining of multimedia data for social and behavioral studies. Inf Vis. 2009;8(1).

133. Silva I, Moody GB. An Open-source Toolbox for Analysing and Processing PhysioNet Databases in MATLAB and Octave. J Open Res Softw. 2014;2.

134. Moody GB, Mark RG, Goldberger AL. PhysioNet: physiologic signals, time series and related open source software for basic, clinical, and applied research. Conf Proc IEEE Eng Med Biol Soc. 2011;2011.

135. Jing J, Dauwels J, Rakthanmanon T, Keogh E, Cash SS, Westover MB. Rapid annotation of interictal epileptiform discharges via template matching under Dynamic Time Warping. J Neurosci Methods. 2016;274.

136. Sauermann S, David V, Schlögl A, Egelkraut R, Frohner M, Pohn B, et al. Biosignals, standards and FHIR -The way to go? In: Studies in Health Technology and Informatics. 2017.

137. Mason JW, Hancock EW, Gettes LS. Recommendations for the Standardization and Interpretation of the Electrocardiogram. Part II: Electrocardiography Diagnostic Statement List A Scientific Statement From the American Heart Association Electrocardiography and Arrhythmias Committee, Council on Clinical Cardiology; the American College of Cardiology Foundation; and the Heart Rhythm Society Endorsed by... Vol. 49, Journal of the American College of Cardiology. 2007.

138. Kligfield P, Gettes LS, Bailey JJ, Childers R, Deal BJ, Hancock EW, et al. Recommendations for the Standardization and Interpretation of the Electrocardiogram. Part I: The Electrocardiogram and Its Technology A Scientific Statement From the American Heart Association Electrocardiography and Arrhythmias Committee, Council on Clinical Cardiology; the American College of Cardiology Foundation; and the Heart Rhythm Society Endorsed by... Vol. 49, Journal of the American College of Cardiology.

2007.

139. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. J Mach Learn Res [Internet]. 2011;12:2825–30. Available from: http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

140. Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, et al. Automatic differentiation in PyTorch. In 2017.

141. Guenther N, Schonlau M. Support vector machines. Stata J. 2016;16(4).

142. Gers FA, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM. Neural Comput. 2000;12(10).

143. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.

144. Oliveira S, Portela F, Santos MF, Machado J, Abelha A, Silva Á, et al. Predicting plateau pressure in intensive medicine for ventilated patients. Adv Intell Syst Comput. 2015;354(Dm):179–88.

145. Rehm GB, Kuhn BT, Lieng MK, Cortes-Puch I, Nguyen J, Guo EC, et al. An Analytic Platform for the Rapid and Reproducible Annotation of Ventilator Waveform Data. bioRxiv. 2019;

146. Peslin R, Da Silva JF, Chabot F, Duvivier C. Respiratory mechanics studied by multiple linear regression in unsedated ventilated patients. Eur Respir J. 1992;5(7):871–8.

147. Lauzon AM, Bates JHT. Estimation of time-varying respiratory mechanical parameters by recursive least squares. J Appl Physiol. 1991;71(3):1159–65.

148. Becher T, Schädler D, Frerichs I, Weiler N. Non-invasive determination of respiratory system mechanics in pressure support ventilation using the expiratory time constant? Crit Care. 2013;17(2).

149. Tan CP, Chiew YS, Geoffrey Chase J, Chiew YW, Pretty C, Desaive T, et al. Model iterative airway pressure reconstruction during mechanical ventilation asynchrony: Shapes and sizes of reconstruction. In: IFMBE Proceedings. 2018.

150. Redmond DP, Docherty PD, Chiew YS, Chase JG. A polynomial model of patient-specific breathing effort during controlled mechanical ventilation. Proc Annu Int Conf IEEE Eng Med Biol Soc EMBS. 2015;2015-Novem(L):4532–5.

151. Force ADT, Ranieri V, Rubenfeld G, Others. Acute Respiratory Distress Syndrome. JAMA - J Am Med Assoc. 2012;307(23):2526--2533.

152. López V, Fernández A, García S, Palade V, Herrera F. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Inf Sci (Ny). 2013;250.

153. Needham DM, Yang T, Dinglas VD, Mendez-Tellez PA, Shanholtz C, Sevransky JE, et al. Timing of low tidal volume ventilation and intensive care unit mortality in acute respiratory distress syndrome:A Prospective Cohort Study. Am J Respir Crit Care Med. 2015;191(2):177–85.

154. Coravos A, Khozin S, Mandl KD. Developing and adopting safe and effective digital biomarkers to improve patient outcomes. Vol. 2, npj Digital Medicine. 2019.

155. Galloway CD, Valys A V., Shreibati JB, Treiman DL, Petterson FL, Gundotra VP, et al. Development and Validation of a Deep-Learning Model to Screen for Hyperkalemia from the Electrocardiogram. JAMA Cardiol. 2019;4(5).

156. Hannun AY, Rajpurkar P, Haghpanahi M, Tison GH, Bourn C, Turakhia MP, et al. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. Nat Med. 2019;25(1).

157. Tjepkema-Cloostermans MC, da Silva Lourenço C, Ruijter BJ, Tromp SC, Drost G, Kornips FHM, et al. Outcome Prediction in Postanoxic Coma With Deep Learning. Crit Care Med. 2019;47(10).

158. Calfee CS, Janz DR, Bernard GR, May AK, Kangelaris KN, Matthay MA, et al. Distinct molecular phenotypes of direct vs indirect ARDS in single-center and multicenter studies. Chest. 2015;147(6).

159. Madotto F, Pham T, Bellani G, Bos LD, Simonis FD, Fan E, et al. Resolved versus confirmed ARDS after 24 h: insights from the LUNG SAFE study. Intensive Care Med. 2018;44(5).

160. Rubenfeld GD, Caldwell E, Granton J, Hudson LD, Matthay MA. Interobserver variability in applying a radiographic definition for ARDS. Chest. 1999;116(5).

161. Wainberg M, Merico D, Delong A, Frey BJ. Deep learning in biomedicine. Nat Biotechnol. 2018;36(9).

162. Thille AW, Esteban A, Fernández-Segoviano P, Rodriguez JM, Aramburu JA, Peñuelas O, et al. Comparison of the berlin definition for acute respiratory distress syndrome with autopsy. Am J Respir Crit Care Med. 2013;187(7).

163. Calfee CS, Delucchi K, Parsons PE, Thompson BT, Ware LB, Matthay MA. Subphenotypes in acute respiratory distress syndrome: Latent class analysis of data from two randomised controlled trials. Lancet Respir Med. 2014;2(8).

164. Ranieri VM, Rubenfeld GD, Thompson BT, Ferguson ND, Caldwell E, Fan E, et al. Acute respiratory distress syndrome: The Berlin definition. JAMA - J Am Med Assoc. 2012;307(23):2526–33.

165. Huang G, Liu Z, van der Maaten L, Weinberger KQ. Densely Connected Convolutional

Networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. p. 4700–8.

166. Iwana BK, Uchida S. An Empirical Survey of Data Augmentation for Time Series Classification with Neural Networks. 2020;1–17.

167. Wen Q, Sun L, Song X, Gao J, Wang X, Xu H. Time Series Data Augmentation for Deep Learning: A Survey. arXiv. 2020;

168. Johnson JM, Khoshgoftaar TM. Survey on deep learning with class imbalance. J Big Data. 2019;6(1).

169. Kingma DP, Ba JL. Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. 2015.

170. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods. 2020;17(3).

171. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In: Proceedings of the IEEE International Conference on Computer Vision. 2017. p. 618–26.

172. Adebayo J, Gilmer J, Muelly M, Goodfellow I, Hardt M, Kim B. Sanity checks for saliency maps. Adv Neural Inf Process Syst. 2018;2018-Decem(Nips):9505–15.

173. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Advances in Neural Information Processing Systems [Internet]. 2019. p. 8024--8035. Available from: http://arxiv.org/abs/1912.01703

174. Herasevich V, Tsapenko M, Kojicic M, Ahmed A, Kashyap R, Venkata C, et al. Limiting

ventilator-induced lung injury through individual electronic medical record surveillance. Crit Care Med. 2011;39(1).

175. Pirracchio R, Cohen MJ, Malenica I, Cohen J, Chambaz A, Cannesson M, et al. Big data and targeted machine learning in action to assist medical decision in the ICU. Vol. 38, Anaesthesia Critical Care and Pain Medicine. 2019.

176. Ghassemi M, Naumann T, Schulam P, Beam AL, Chen IY, Ranganath R. A Review of Challenges and Opportunities in Machine Learning for Health. AMIA Jt Summits Transl Sci proceedings AMIA Jt Summits Transl Sci [Internet]. 2020;2020:191–200. Available from: http://www.ncbi.nlm.nih.gov/pubmed/32477638%0Ahttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC7233077

177. Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat Mach Intell. 2019;1(5):206–15.

178. Kim JY, Kim JM. Bearing fault diagnosis using grad-CAM and acoustic emission signals. Appl Sci. 2020;10(6).

179. Burkart N, Huber MF. A survey on the explainability of supervised machine learning. arXiv. 2020;1–74.

180. Porumb M, Stranges S, Pescapè A, Pecchia L. Precision Medicine and Artificial Intelligence: A Pilot Study on Deep Learning for Hypoglycemic Events Detection based on ECG. Sci Rep. 2020;10(1):1–16.

181. Fukui H, Hirakawa T, Yamashita T, Fujiyoshi H. Attention Branch Network: Learning of Attention Mechanism for Visual Explanation. arXiv. 2018;0–9.

182. Song CL, Vardaki MZ, Goldin RD, Kazarian SG. Fourier transform infrared spectroscopic

imaging of colon tissues: evaluating the significance of amide I and C–H stretching bands in diagnostic applications with machine learning. Anal Bioanal Chem. 2019;411(26):6969–81.

183. Tseng AM, Shrikumar A, Kundaje A. Fourier-transform-based attribution priors improve the interpretability and stability of deep learning models for genomics. bioRxiv. 2020;

184. Zolotarev AM, Hansen BJ, Ivanova EA, Helfrich KM, Li N, Janssen PML, et al. Optical mapping-validated machine learning improves atrial fibrillation driver detection by multi-electrode mapping. Circ Arrhythmia Electrophysiol. 2020;(October):1199–212.

185. Rajkomar A, Oren E, Chen K, Dai AM, Hajaj N, Hardt M, et al. Scalable and accurate deep learning with electronic health records. NPJ Digit Med [Internet]. 2018;1(1):1–10. Available from: http://dx.doi.org/10.1038/s41746-018-0029-1

186. Halpern NA, Pastores SM, Oropello JM, Kvetan V. Critical Care Medicine in the United States Addressing the Intensivist Shortage and Image of the Specialty. Crit Care Med. 2013;41(12):2754–61.

187. Mosier JM, Kelsey M, Raz Y, Gunnerson KJ, Meyer R, Hypes CD, et al. Extracorporeal membrane oxygenation (ECMO) for critically ill adults in the emergency department: History, current applications, and future directions. Vol. 19, Critical Care. 2015.

188. Taccone P, Pesenti A, Latini R, Polli F, Vagginelli F, Mietto C, et al. Prone positioning in patients with moderate and severe acute respiratory distress syndrome: A randomized controlled trial. JAMA - J Am Med Assoc. 2009;302(18).

189. Huang DT, Angus DC, Moss M, Thompson BT, Ferguson ND, Ginde A, et al. Design and rationale of the reevaluation of systemic early neuromuscular blockade trial for acute respiratory distress syndrome. Ann Am Thorac Soc. 2017;14(1).

190. Halpern NA. Innovative designs for the smart ICU : Part 1: From initial thoughts to occupancy. Chest. 2014;145(2).

191. Rawat M, Chandrasekharan PK, Williams A, Gugino S, Koenigsknecht C, Swartz D, et al. Oxygen saturation index and severity of hypoxic respiratory failure. Neonatology. 2015;107(3).

192. Vadi S. Correlation of oxygen index, oxygen saturation index, and pao2/fio2 ratio in invasive mechanically ventilated adults. Indian J Crit Care Med. 2021;25(1).

193. Calvert JS, Price DA, Chettipally UK, Barton CW, Feldman MD, Hoffman JL, et al. A computational approach to early sepsis detection. Comput Biol Med. 2016;74.

194. Kam HJ, Kim HY. Learning representations for the early detection of sepsis with deep neural networks. Comput Biol Med. 2017;89.

195. Kim J, Blum JM, Scott CD. Temporal Features and Kernel Methods for Predicting Sepsis in Postoperative Patients. Manuscript. 2010;

196. Baedorf Kassis E, Loring SH, Talmor D. Mortality and pulmonary mechanics in relation to respiratory system and transpulmonary driving pressures in ARDS. Intensive Care Med. 2016;42(8).

197. Brower RG, Morris A, MacIntyre N, Matthay MA, Hayden D, Thompson BT, et al. Effects of recruitment maneuvers in patients with acute lung injury and acute respiratory distress syndrome ventilated with high positive end-expiratory pressure. Crit Care Med. 2003;31(11).

198. Wang X, Sontag D, Wang F. Unsupervised learning of disease progression models. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2014.

199. Jackson CH, Sharples LD, Thompson SG, Duffy SW, Couto E. Multistate Markov models for disease progression with classification error. J R Stat Soc Ser D Stat. 2003;52(2).

200. Guihenneuc-Jouyaux C, Richardson S, Longini IM. Modeling markers of disease progression by a hidden Markov process: Application to characterizing CD4 cell decline. Biometrics. 2000;56(3).

201. Andreão R V., Dorizzi B, Boudy J. ECG signal analysis through hidden Markov models. IEEE Trans Biomed Eng. 2006;53(8).

202. Marco E, Meuleman W, Huang J, Glass K, Pinello L, Wang J, et al. Multi-scale chromatin state annotation using a hierarchical hidden Markov model. Nat Commun. 2017;8.

203. Rabiner LR. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proc IEEE. 1989;77(2).

204. Schmidhuber J. Deep Learning in neural networks: An overview. Vol. 61, Neural Networks. 2015.

205. Bengio Y. Learning deep architectures for AI. Found Trends Mach Learn. 2009;2(1).

206. Arel I, Rose D, Karnowski T. Deep machine learning-A new frontier in artificial intelligence research. IEEE Comput Intell Mag. 2010;5(4).

207. Lipton ZC, Kale DC, Elkan C, Wetzel R. Learning to diagnose with LSTM recurrent neural networks. In: 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings. 2016.

208. Mannino DM, Diaz-Guzman E, Pospisil J. A new approach to classification of disease severity and progression of COPD. Chest. 2013;144(4).

209. Stevenson D, Revie J, Chase JG, Hann CE, Shaw GM, Lambermont B, et al. Algorithmic processing of pressure waveforms to facilitate estimation of cardiac elastance. Biomed

Eng Online. 2012;11.

210. Ikeda T, Yamauchi Y, Uchida K, Oba K, Nagase T, Yamada Y. Reference value for expiratory time constant calculated from the maximal expiratory flow-volume curve. BMC Pulm Med. 2019;19(1):1–9.

211. Wiriyaporn D, Wang L, Aboussouan LS. Expiratory time constant and sleep apnea severity in the overlap syndrome. J Clin Sleep Med. 2016;12(3):327–32.

Appendix A: APL

A.1: Installation

APL was developed with Python2.7. If you do not have a Python2.7 environment, you can set one up using anaconda or virtualenv. Running a development server for APL is not difficult, but you will need Redis installed on your machine before you attempt to run these instructions:

```
conda create -n apl python=2.7
conda activate apl
cd source
pip install -r requirements.txt
redis-server &
python development.py
```

If you want to run APL on Docker then follow these steps: First, you'll need to configure the app properly for a production environment.

```
touch prod_config.py
```

Then enter the prod_config.py file with a text editor and add the following information:

```
REDIS = {
   “host”: “apl-redis”,
   “port”: 6379,
   “db”: 0,
}
SECRET_KEY = “XXX”
```

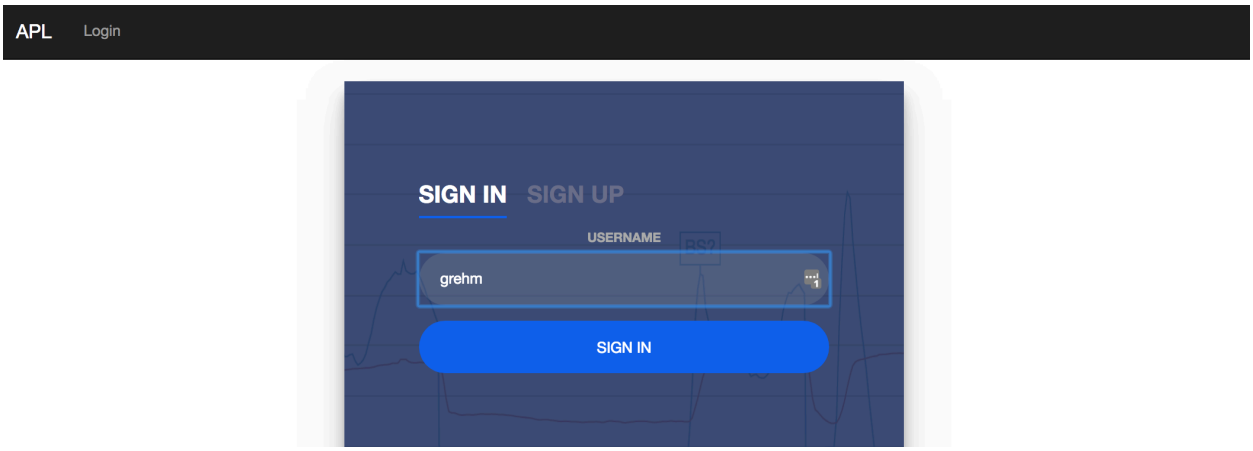
`SECRET_KEY` should be changed to whatever you deem appropriate.

Running the annotation pipeline will require Docker to be installed on your computer. Docker is a flexible piece of software that can quickly deploy complex applications in a reproducible manner. You will need to install Docker and Docker-Compose. Instructions for this can be readily found on Google for your operating system. Once Docker and Docker-Compose are installed you can execute the following command on the CLI

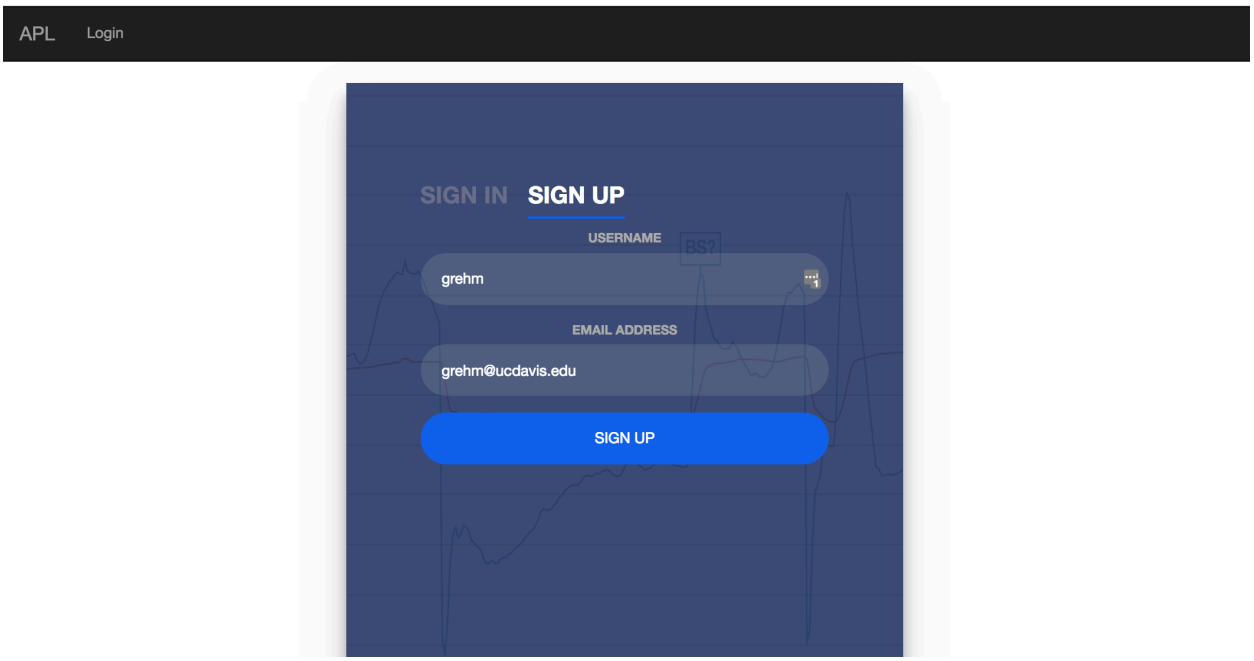
```
docker-compose up
```

A.2: Usage

Using APL is designed to be as simple as possible to untrained users. Sign-In is accomplished by entering a username, while sign-up is done by entering a username and an email address. We currently have not implemented password protection mechanisms because we haven't found a need for it yet, but if multi-center users desire to use the same APL instance, then password protection will undoubtedly be necessary.

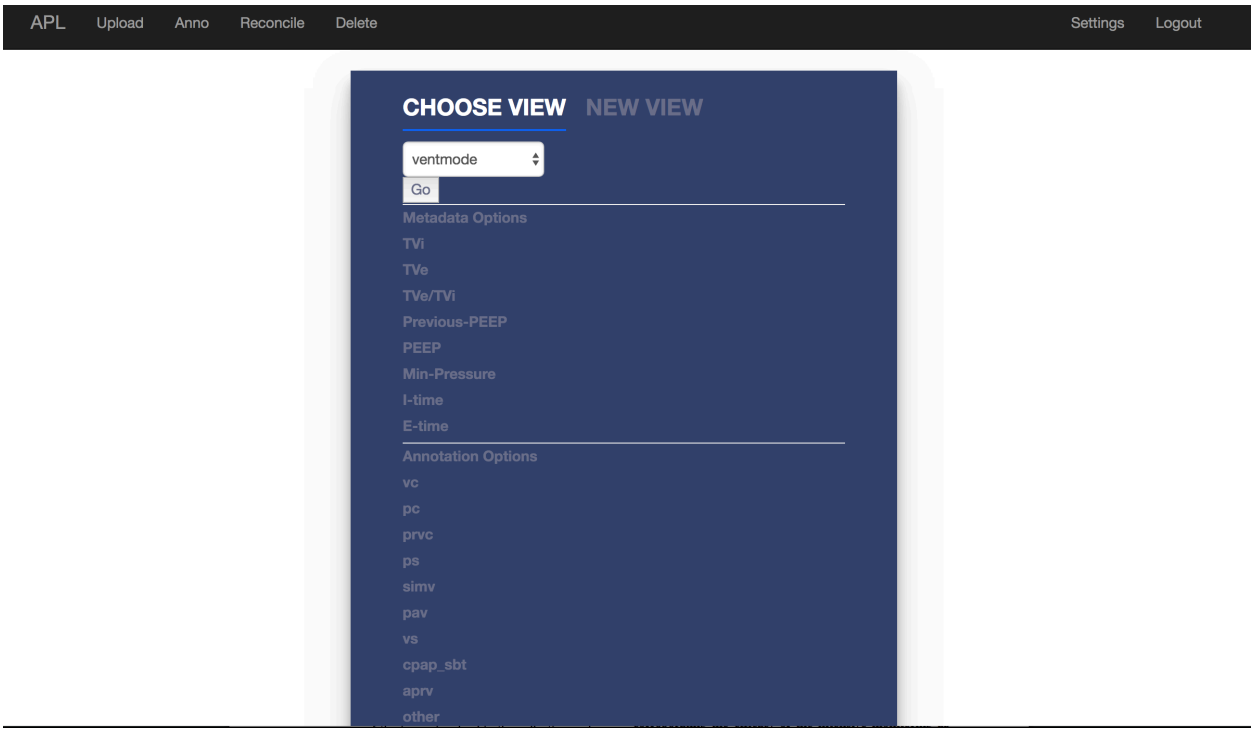


Appendix Figure A.1: Displays sign in screen for APL

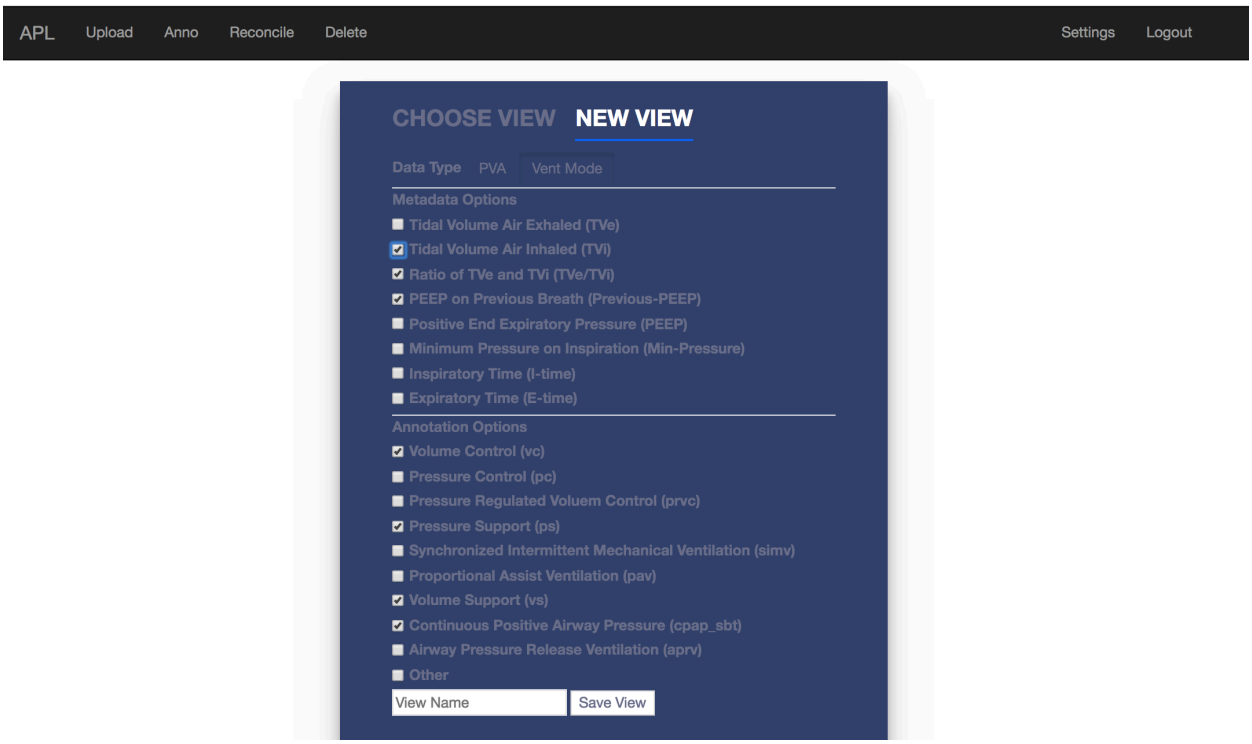


Appendix Figure A.2: Displays sign up screen for APL

Normally users can just choose an annotation view. After choosing a view a list of metadata and annotation options will be displayed to ensure the correct view has been chosen.

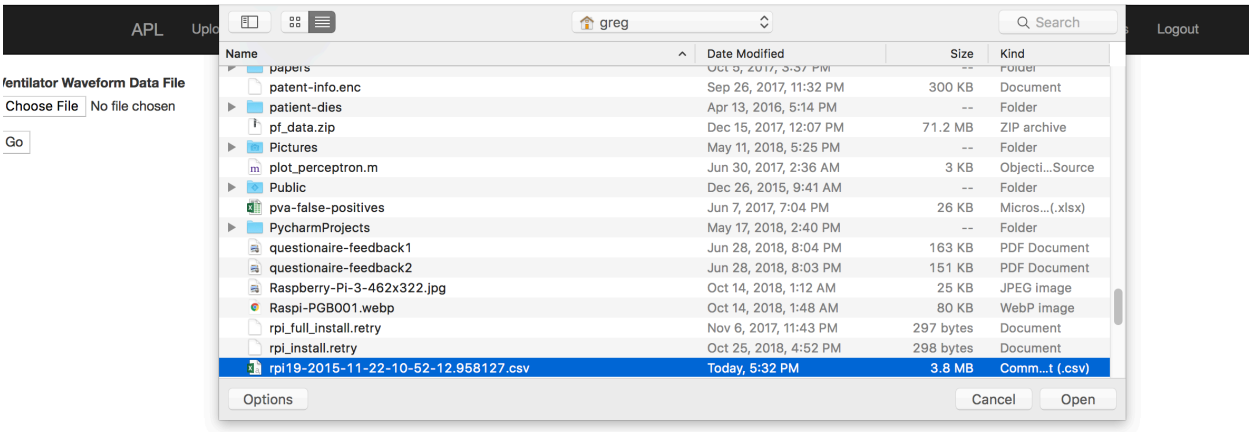


Appendix Figure A.3: Annotation views can be chosen based on preset views that are default to APL. Users can also create a new annotation view if they desire as well. New annotation views will be marked with a username to ensure proper attribution is made.

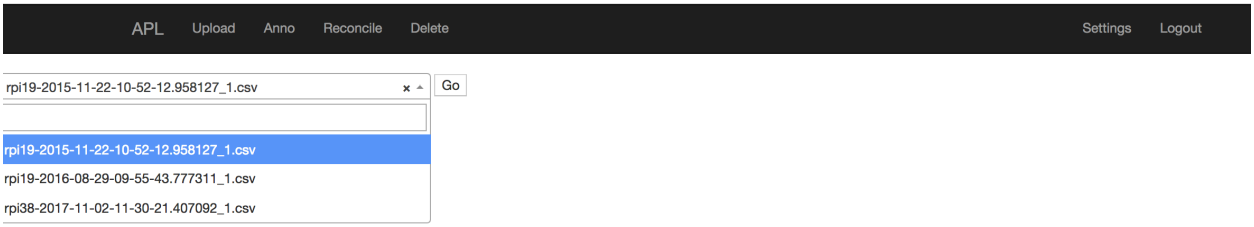


Appendix Figure A.4: Shows how to create a new annotation view with APL.

Upload occurs using files collected from the Puritan Bennet 840 ventilator. The files are then analyzed with the ventMAP software and processed into a form that can be rendered by APL.



Appendix Figure A.5: Files can then be chosen via drop-down menu.



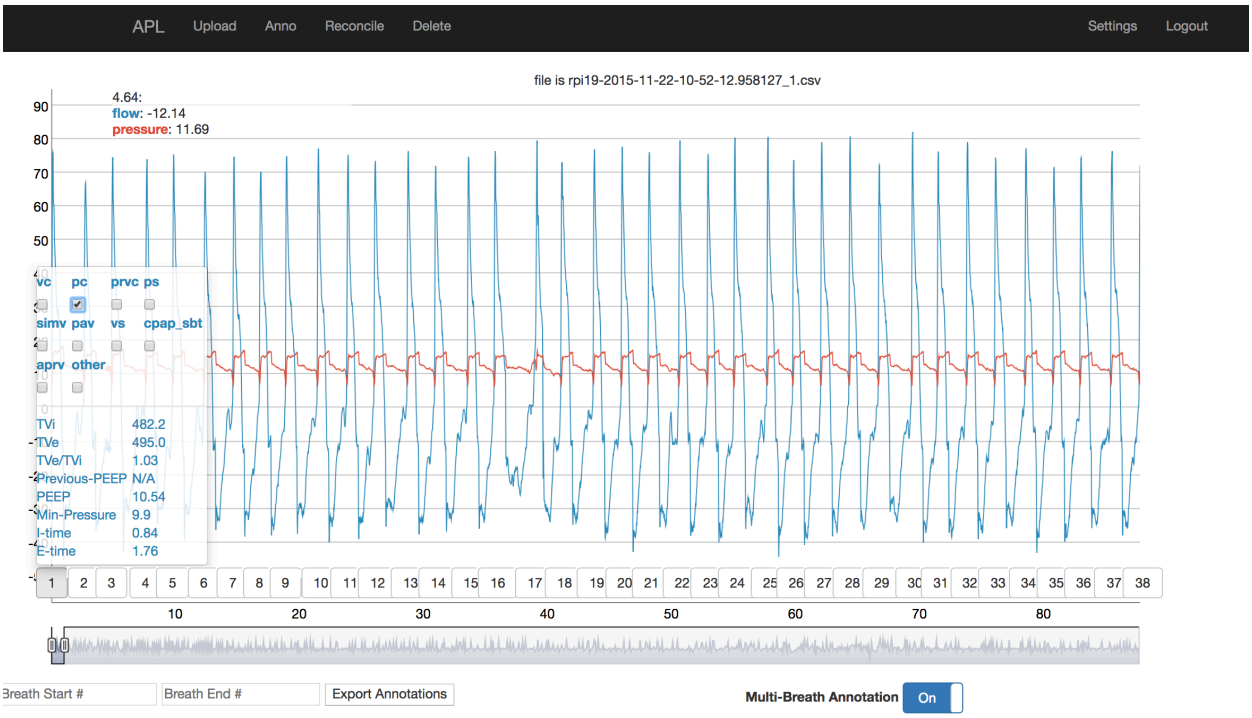
Appendix Figure A.6: After processing the file will be available to view.

Annotating a file can begin after a view and a file are chosen to annotate.

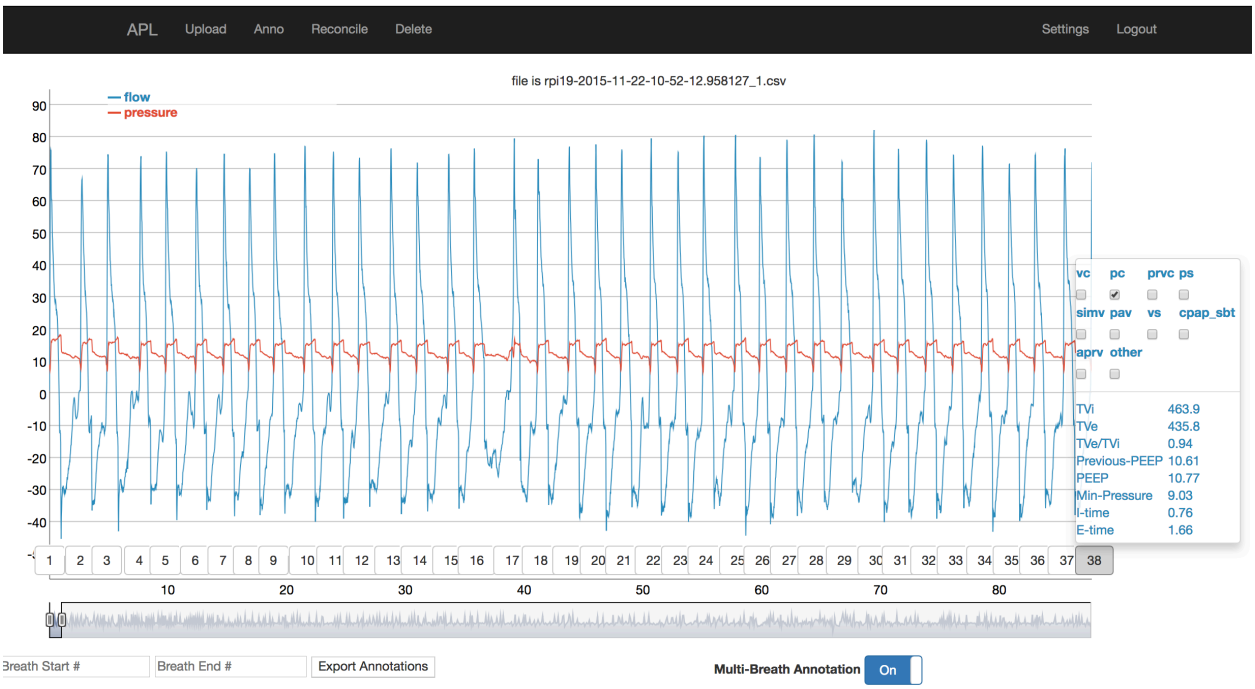
Appendix Figure A.7: Upon choosing a file, users will see breath data. Flow data is marked in blue, and pressure is in red.

Appendix Figure A.8: Clicking on a breath number shows a drop-up window with annotations and metadata

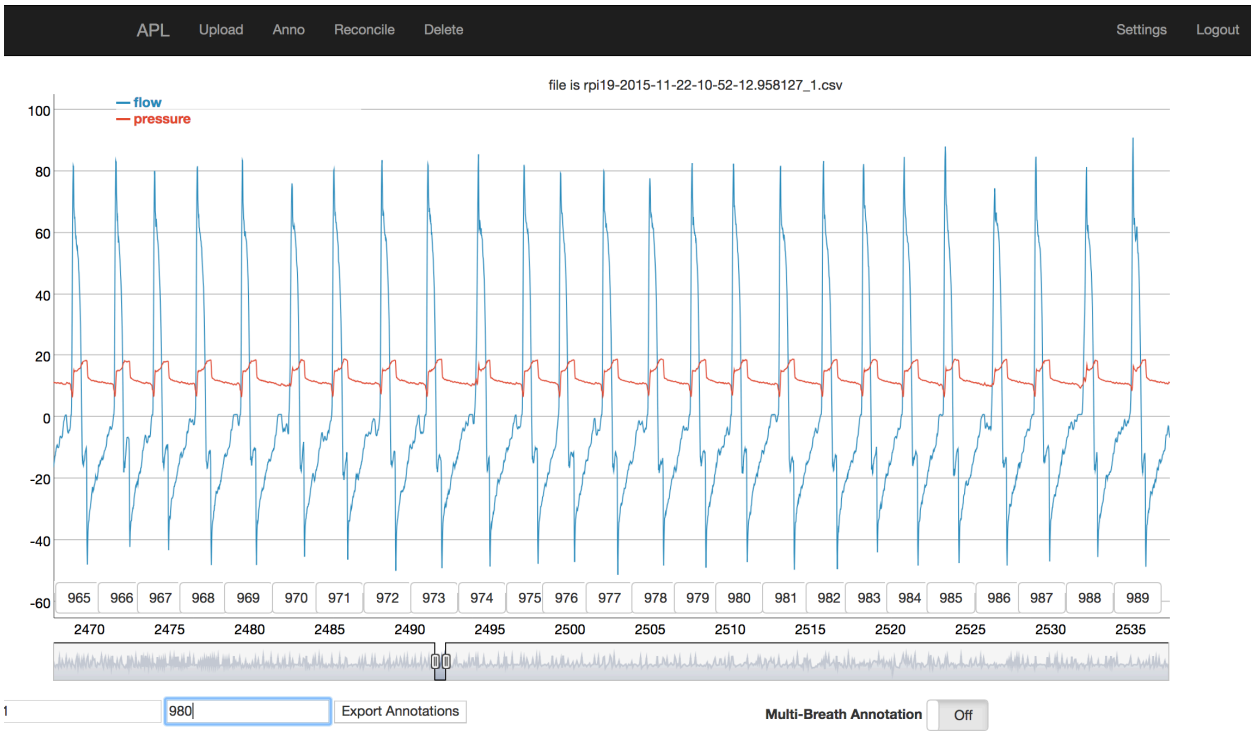
Annotation occurs simply by clicking on a breath number and by clicking a checkbox associated with a breath annotation. Breaths that are not annotated are assumed to be normal. If multiple breaths at once are desired to be annotated then a user simply picks a breath to start annotations at, and then selects a breath to end annotations, and all intermediate breaths will be automatically annotated. Annotations can be exported to CSV format by clicking a button at the bottom left of the annotation screen. There is also option for output of a specific breath range of annotations as well.



Appendix Figure A.9: Selecting the first breath of a multi-breath annotation

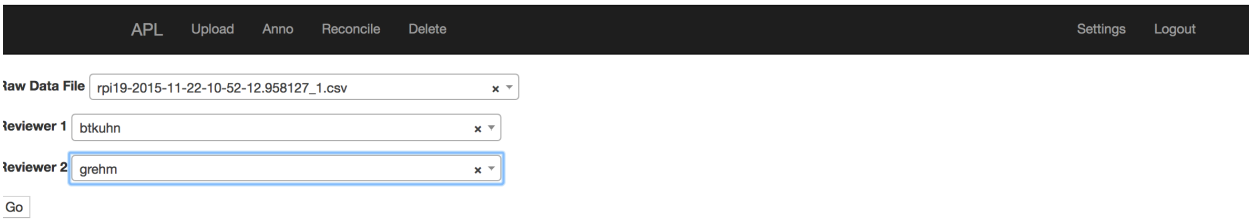


Appendix Figure A.10: Selecting the final breath a multi-breath annotation. All breaths in between will be marked with the chosen annotation.



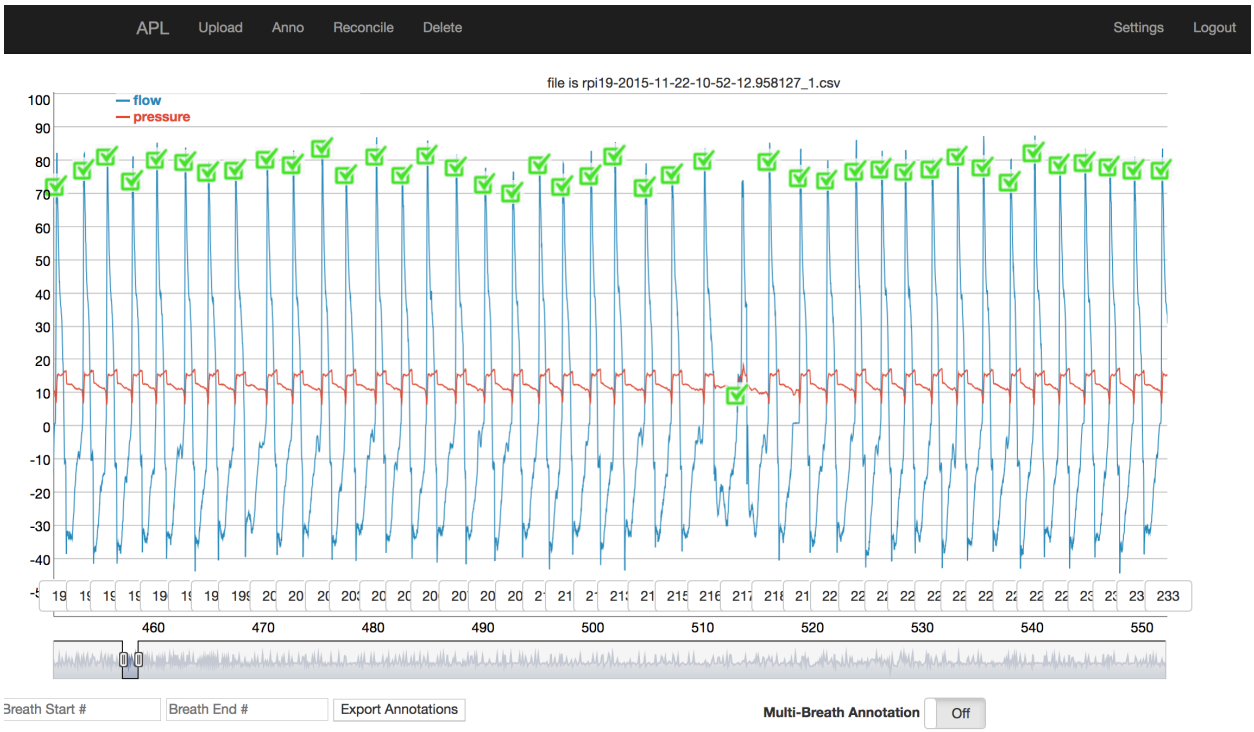
Appendix Figure A.11: Shows choosing breath numbers to export to CSV format at the bottom left corner of the screen.

If another reviewer has annotated the same file you are working on, in the same view then you can select the file you want to reconcile and the name of the reviewer you want to reconcile against.



Appendix Figure A.12: To begin the reconciliation process choose a file and then two different reviewers who annotated the file.

If there are no breaths that need to be reconciled, then all breaths will have a green checkmark above them.



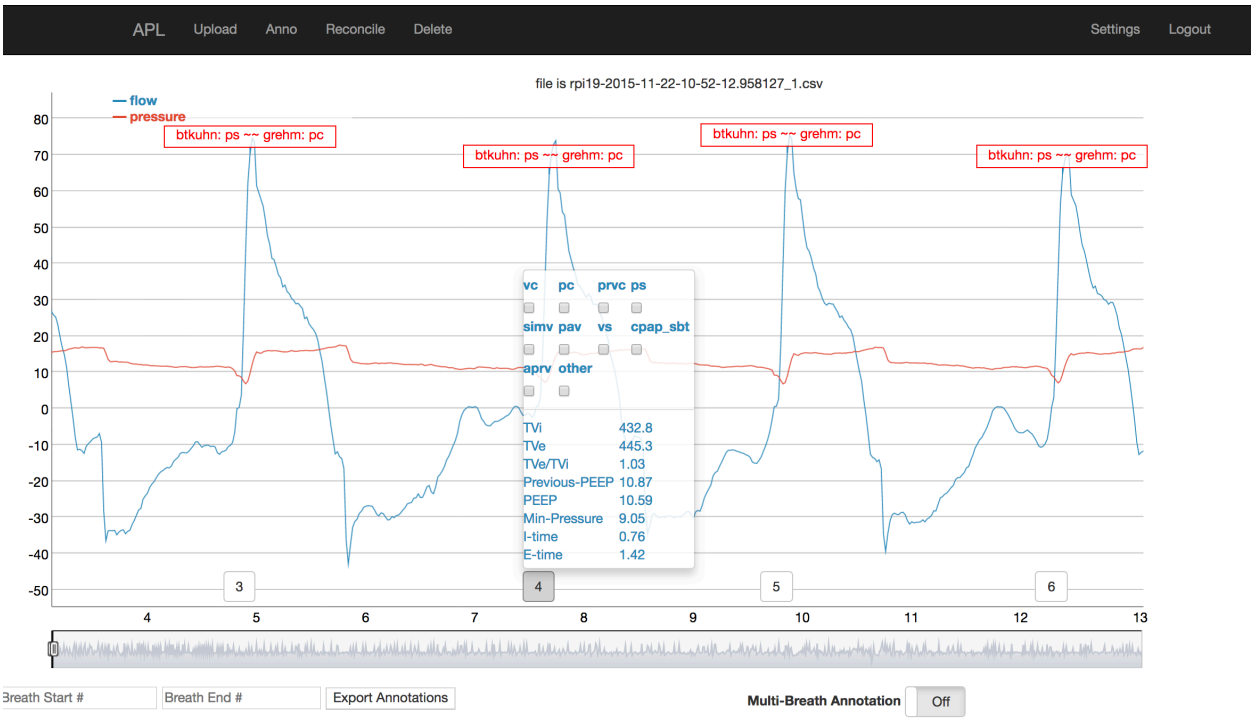
Appendix Figure A.13: If everyone agreed on all annotations then all breaths will be displayed with checkmarks

If breaths need to be reconciled, then breaths will appear with a red label above them, marking the differing choices each reviewer made when annotating them.

Appendix Figure A.14: If there are annotation disagreements then specific issues will be shown for the breath, along with reviewers and the annotation that each reviewer chose.

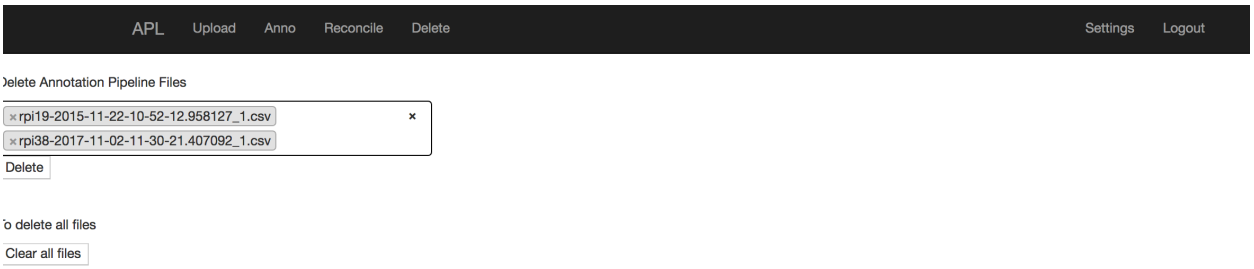
Importantly, breaths needing reconciliation will have differing annotations be unchecked,

allowing for tabula rasa of choice.



Appendix Figure A.15: Annotation disagreements can be reconciled by choosing a consensus annotation and then exporting the file again.

If files need to be deleted from APL to clear disk space, then that is possible via the `Delete` Tab. Here users have option of deleting single files, or all files if space needs to be quickly found.



Appendix Figure A.16: Breath data can be deleted if necessary to make space for new files.

Appendix B: Algorithms for Estimating Static Compliance

B.1: Cohort Breath Descriptive Statistics

	VC (n=15)	PC (n=10)	PRVC (n=3)
Number breaths	24,938	10,111	4,610
RASS			
-1	22	1,641	0
-2	0	0	0
-3	1,908	4,849	0
-4	7,389	2,717	2,095
-5	12,061	904	2,515
N/A	3,558	0	0
Asynchronous Breaths			
Double Trigger Asynchrony	49	134	2
Breath Stacking Asynchrony	664	690	41
Flow Asynchrony (FA)			
Mild	6,729	-	-
Moderate	35	-	-
Severe	34	-	-
Delayed Cycling (DCA)	-	1426	55

Appendix Table B.1: Details per-breath counts under different ventilation modes, Richmond agitation and sedation score (RASS), and patient ventilator asynchrony conditions.

B.2: Flow Asynchrony Algorithm

Here we detail in Python-style pseudo-code the algorithm used to perform flow asynchrony calculations. Our inputs into the function are the minimum inspiratory pressure for a breath, and the set PEEP. Minimum inspiratory pressure is found by taking the minimum pressure from the inspiratory limb of a breath, excluding the first 5 pressure observations of

inspiration. PEEP is found by mean averaging of the last 5 pressure observations of a breath. PEEP calculations for multiple breaths can be mean/median weighted to find a rolling windowed PEEP. This method is more resistant to outlier pressure observations at the end of a breath such as failed trigger asynchronies and other breathing artifact such as suction or cough.

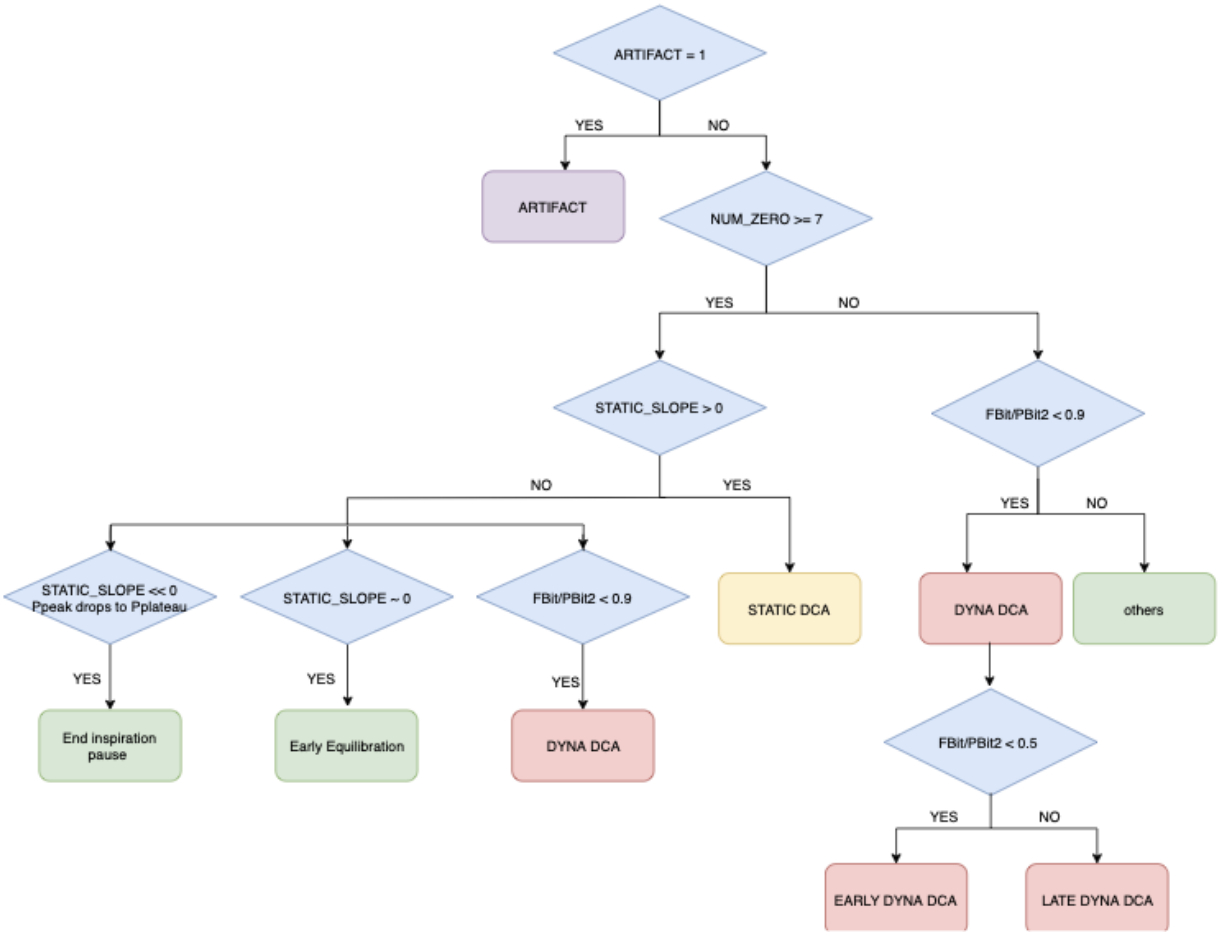
```
def is_breath_flow_async(insp_min_p, peep):
    """
    FA severity is graded from 0-3, with
    0: no flow async
    1: mild flow async
    2: moderate flow async
    3: severe flow async

    :param insp_min_p: the minimum pressure during inspiration
    :param peep: the set positive end expiratory pressure (PEEP)
    """
    diff = insp_min_p - peep
    if diff <= 8: # it's an FA, now find how severe.
        if peep < insp_min_p: # mild FA
            fa = 1
        elif 0 < insp_min_p <= peep: # moderate FA
            fa = 2
        elif insp_min_p <= 0: # severe FA
            fa = 3
    else:
        fa = 0

    return fa
```

B.3: Delayed Cycling Asynchrony Algorithm

The delayed cycling asynchrony (DCA) algorithm is designed to tell us if a patient is attempting to prematurely terminate (exhale) their breath before a pressure-based ventilation mode is set to allow them to exhale. There are two types of DCA that we test for 1) static DCA; 2) dynamic DCA. Static DCA can be defined as when a patient performs a DCA but does not breathe and passively resists any additional air inhalation. Dynamic DCA is defined as when a patient actively exhales when the ventilator pressure is still elevated.



Appendix Figure B.1: Flow chart for making delayed cycling asynchrony (DCA) classifications. Artifact - is the breath an artifact (e.g., cough/suction) or not. Num zero - The number of consecutive flow observations close to 0 in the breath. FBit - the flow-defined inspiratory time in a breath. Is found by finding the amount of time in a breath that flow is above 0, PBit - the pressure-based inspiratory time of a breath. Static Slope - the slope of a period of at least 7 consecutive flow observations that are near 0.

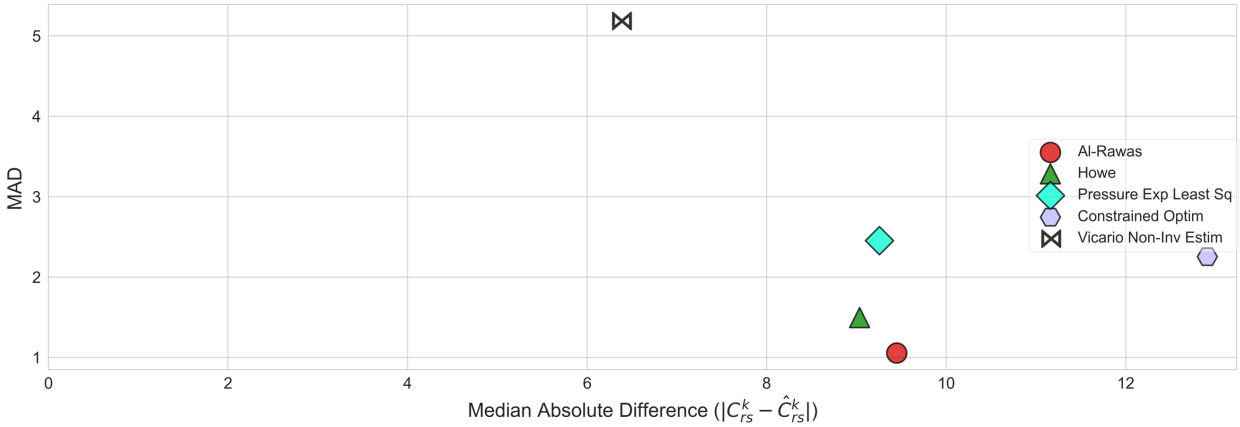
Static DCA can be found by determining if the flow is close to 0 for an extended period. If the slope (i.e., static slope) of this period of breathing that is close to 0 is positive, then we determine the breath is a static DCA. We find dynamic DCA by analyzing the flow-based inspiratory time (FBit) and the pressure-based inspiratory time (PBit). FBit can be found by determining the amount of time from breath start until the time that flow crosses below 0 and the patient is attempting to exhale. PBit can be defined as the amount of time during a pressure-mode that pressure is elevated. PBit is more difficult to accurately measure compared to FBit

because of additional logic and rules that is necessary to implement it. As a relatively general rule, if FBit/PBit < 0.9 then the breath is classified as a delayed cycling asynchrony if the following corner cases are not found in the breath:

1. The breath must not be indicative of an artifact such as a suction, cough, or ventilator disconnect
2. The breath must not be an inspiratory pause.
3. The breath must not have a static slope of 0 (if static slope is 0 then the ventilator is just equilibrating early)

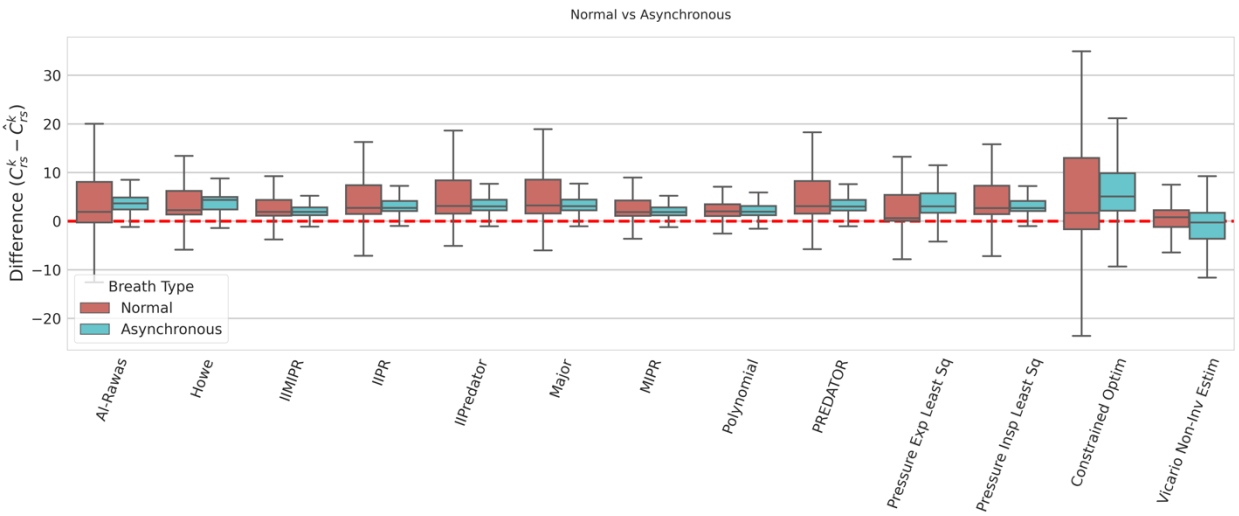
For additional details see Appendix Figure B.1. Code for FBit and PBit can be found in ventMAP.

B.4: Additional Per-Patient Results

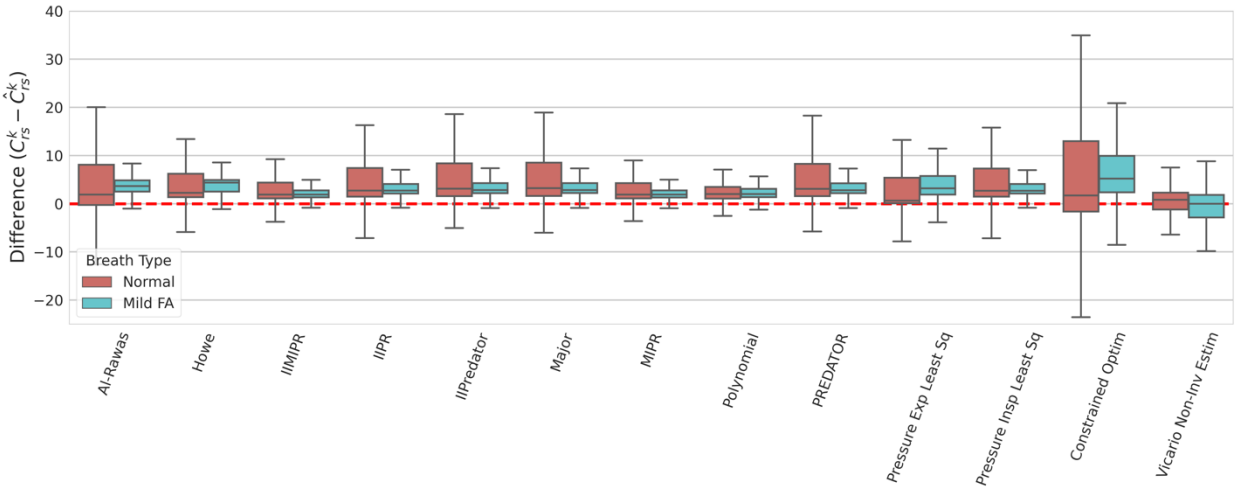


Appendix Figure B.2: Mode-agnostic algorithm performance on a per-patient basis across all ventilation modes.

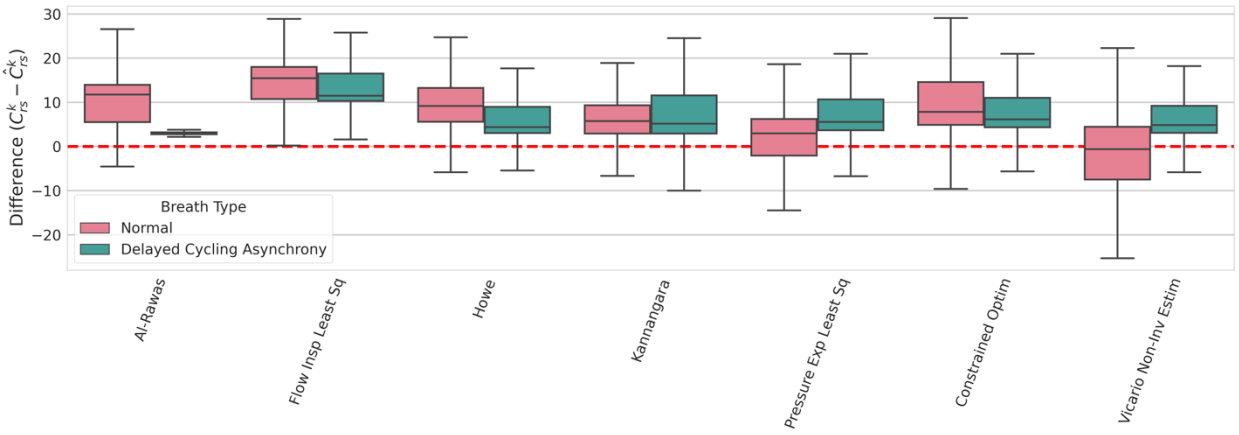
B.5: Algorithm Performance During Asynchrony



Appendix Figure B.3: Comparison between non-asynchronous and asynchronous breathing in volume control mode.



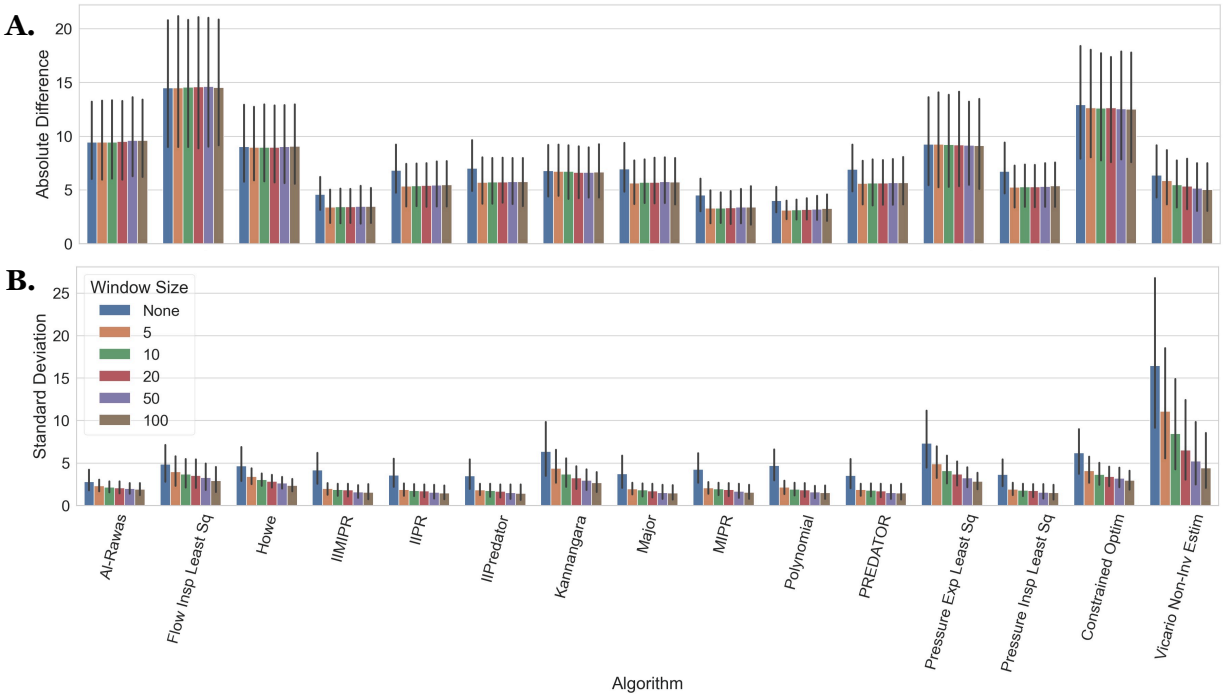
Appendix Figure B.4: Comparison between non-asynchronous and mild flow asynchrony breathing in volume control (VC) mode.



Appendix Figure B.5: Comparison between normal and delayed cycling asynchrony breathing in pressure modes (pressure control / pressure-regulated volume control).

B.6: Window Sizes

We also investigate the effect of window size on both mean per-patient AD and MAD. Our findings show that simply implementing windowing may have effect of decreasing mean AD for some algorithms (Appendix Figure B.6A). A more significant result can be seen when examining MAD, where larger window sizes tend to decrease overall MAD for many algorithms (Appendix Figure B.6B).

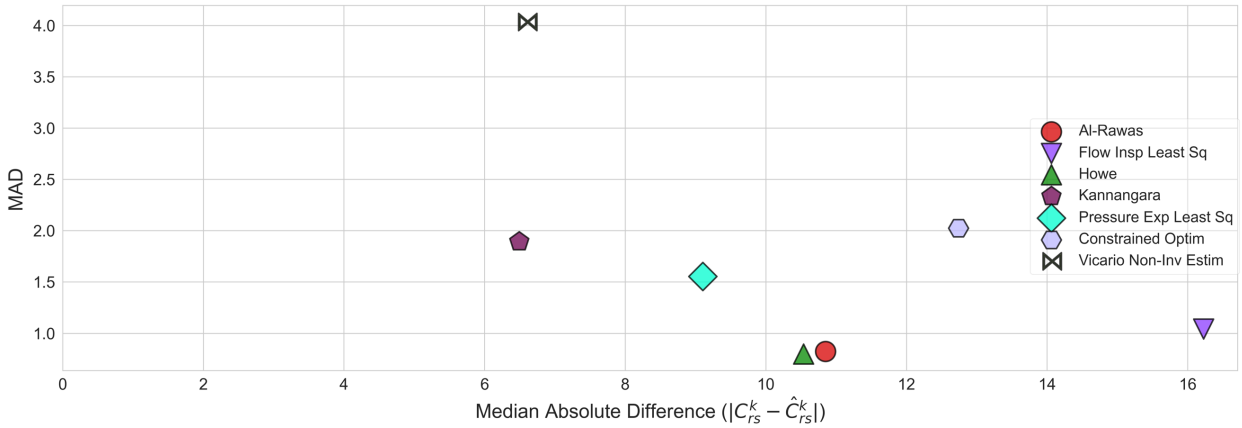


Appendix Figure B.6: **A**. Effect of windowing on absolute difference. Again, we are unable to find statistical significance in per-patient AD. **B**. Effect of algorithm standard deviation with different window sizes. Note: Window size of "None" is the recording of a baseline result where no windowing was used. 95% confidence is shown in bar plot whiskers

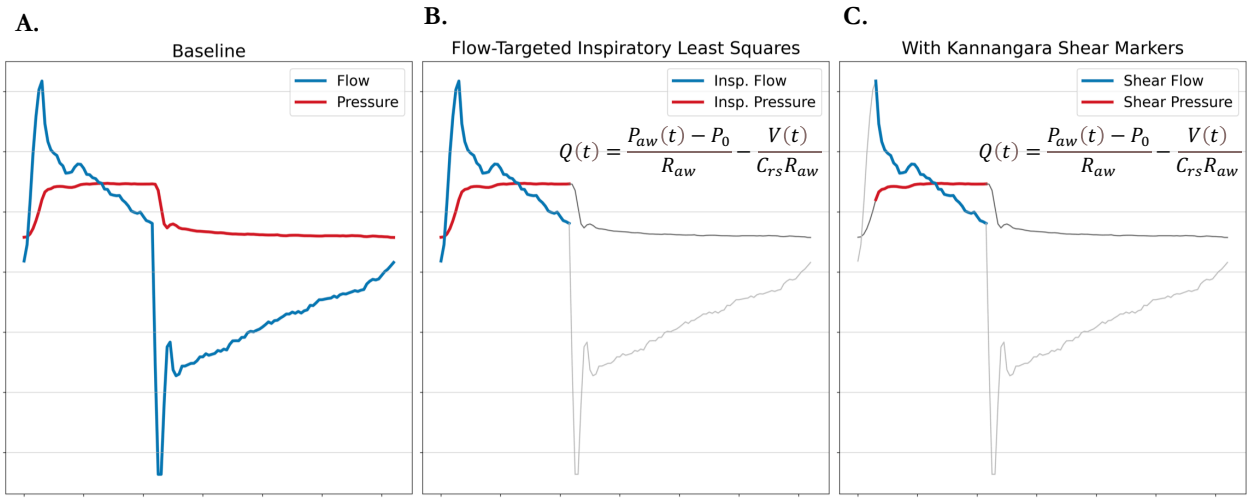
B.7: Kannangara Explanation

From Chapter 3.2, we already have shown that the standard flow-targeted least squares significantly underperforms Kannangara's method. Here, we show why this occurs. The two methods both use the same formulation of the single-chamber model (Appendix Figure B.8 B,

C). Kannangara's method does correct for asynchrony, but this cannot account for the complete difference in performance between the two methods. We show in Appendix Figure B.7 that when we examine per-patient calculations between the two methods, Kannangara's method has lower median absolute difference compared to flow-targeted inspiratory least squares. So, the difference in performance between the two algorithms must be derived from differences in which both methods compute compliance. Kannangara's method calculates compliance using a shear transform to identify two "shoulders" of the inspiratory section of the breath where breath flow begins decelerating.(209) The pressure and flow between the two shoulders are then provided to the flow-targeted inspiratory least squares formulation of the single chamber model (Appendix Figure B.8 C).



Appendix Figure B.7: Per-patient calculations for PC/PRVC breaths only.



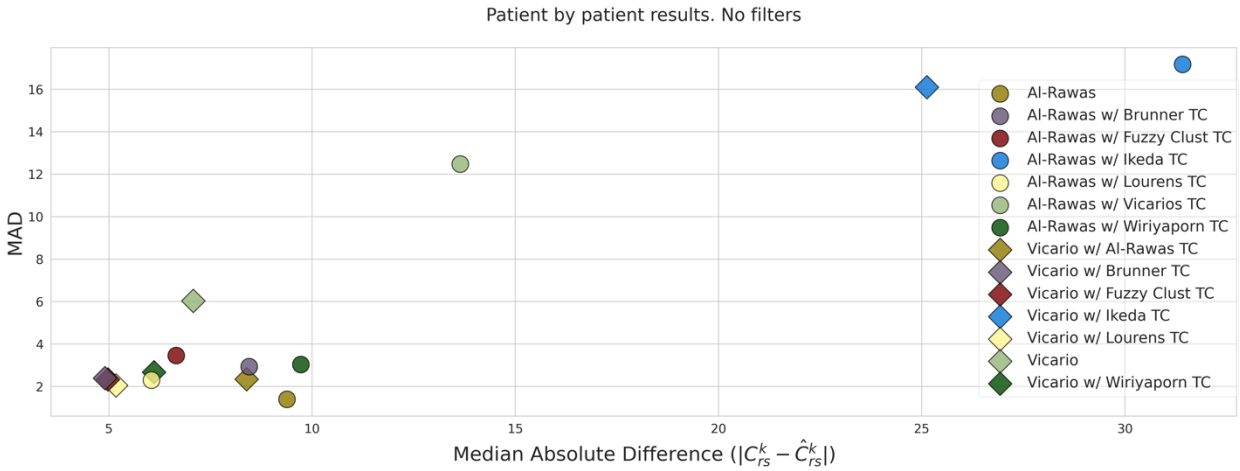
Appendix Figure B.8: The difference between flow-targeted inspiratory least squares calculations, and Kannangara method calculations**.** Also shows the formulation of the single-chamber model that is used. **A.** Shows baseline breath with flow/pressure observations. **B.** Shows how flow-targeted least squares is calculated. **C.** Shows how Kannangara's method is calculated.

B.8: Modifying Time Constants for Al-Rawas and Vicario's Methods

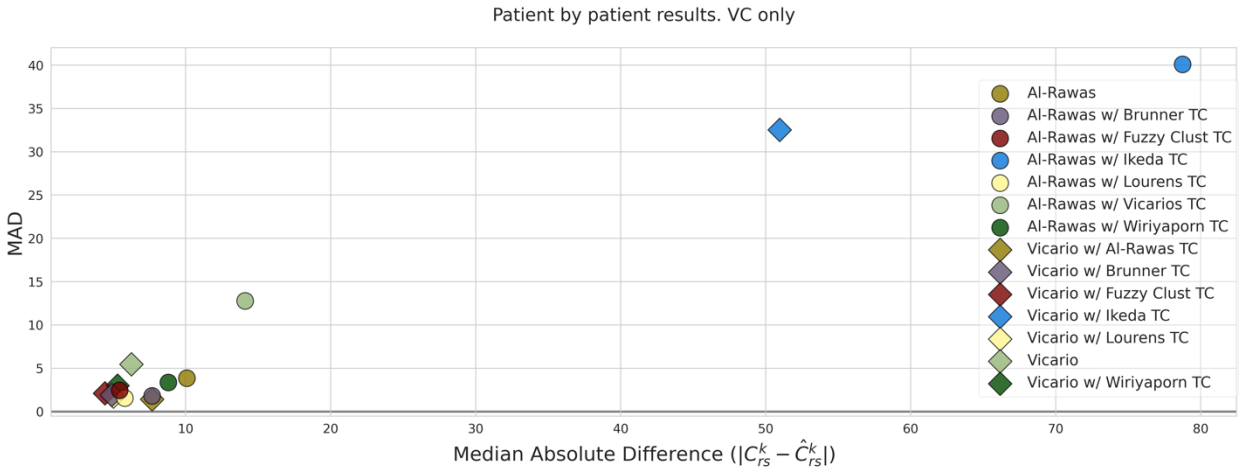
Two of the algorithms that we surveyed: Al-Rawas method, and Vicario's non-invasive estimation of alveolar pressure both utilize a respiratory time constant to calculate an analytic solution for a compliance estimate. The time constant is defined by being a product of both compliance (C_{rs}) and airway resistance (R_{aw}).(96)

$$\tau = R_{aw}C_{rs}$$

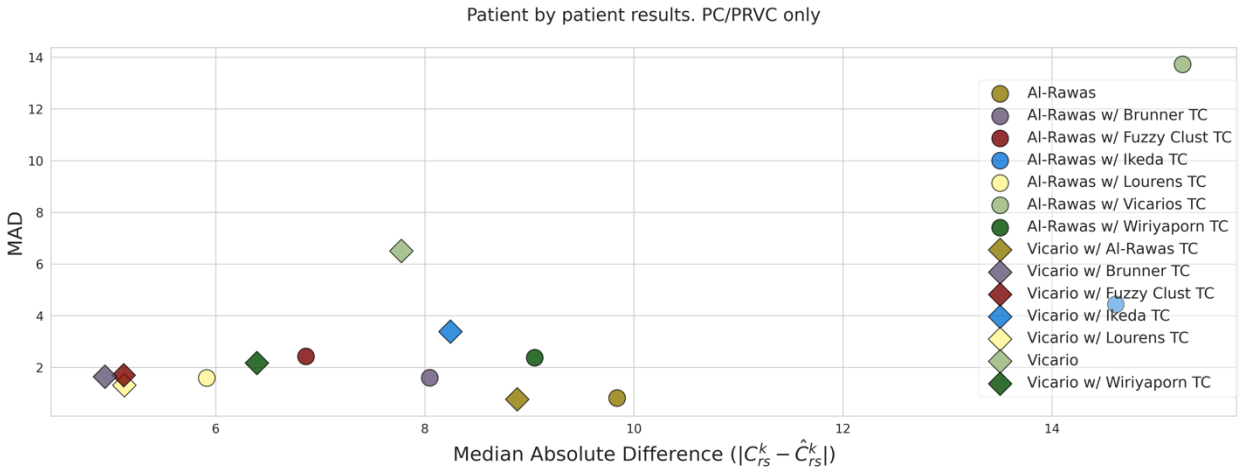
We hypothesized that by utilizing different time constants, other than the time constant that was originally published for the algorithm, that we could improve the estimation results for these algorithms. So we investigate the performance of 7 different time constant calculation methods: Al-Rawas,(98) Brunner,(99) Fuzzy Clustering,(101) Ikeda's,(210) Lourens,(100) Vicario's,(96) Wiriyaporn's.(211) We use these methods to benchmark performance of Al-Rawas and Vicario's method the same way that we did in Chapter 3.



Appendix Figure B.9: Overall per-patient results with different time constants (TCs).



Appendix Figure B.10: Different time constant (TC) results in volume control (VC) only.



Appendix Figure B.11: Different time constants (TC) in pressure control (PC) / pressure regulated volume control (PRVC) only.

Our results (Appendix Figure B.9-Appendix Figure B.11) show that we can improve performance of both Al-Rawas and Vicario's method through use of different time constants. This result implies it is possible that Al-Rawas and Vicario's time constant methods are not optimal for use in a clinical setting in the Al-Rawas and Vicario methods. We are, however, unable to make direct claim about which time constant is objectively superior because our experiments did not measure the airway resistance of the patient. Without this, we are unable to directly measure the time constant, and thus not able to determine which time constant has the lowest error.

Appendix C: ARDS Detection

C.1: General Study Information

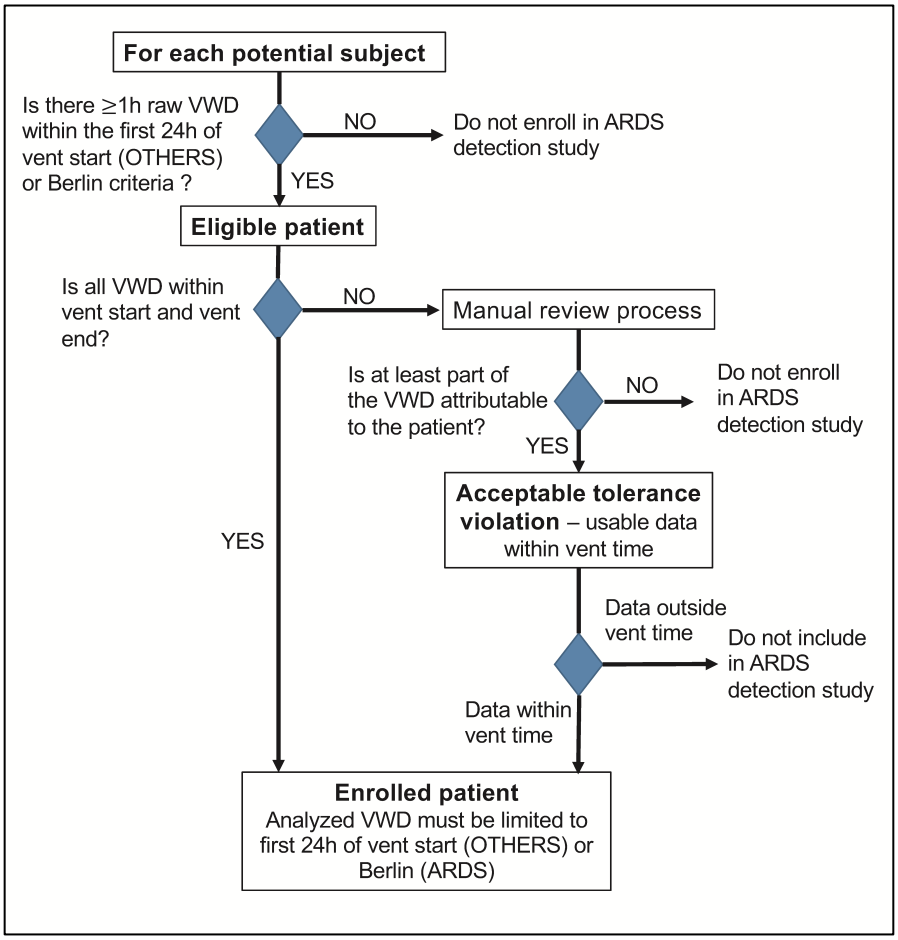
Criteria Type	Criterion
General Inclusion	Age ≥ 18 Mechanically ventilated in the ICU
General Exclusion	Dual clinician agreement on the absence of ARDS Transferred to ICU from outside facility Chronic tracheostomy prior to admission Known or suspected bronchopleural fistula Diffuse chronic fibrotic lung disease Pregnant women Prisoners
ARDS Patient Cohort	Dual clinician diagnosis of ARDS meeting Berlin criteria within 7 days of intubation Recorded lowest PaO2/FiO2 ratio ≤ 200 mm Hg Patients with chronic obstructive pulmonary disease and/or asthma were excluded from the ARDS patient cohort
Non-ARDS Patient Cohort	No suspicion of ARDS based on dual clinician adjudicated chart review

Appendix Table C.1: Inclusion/exclusion criteria for subject enrollment.

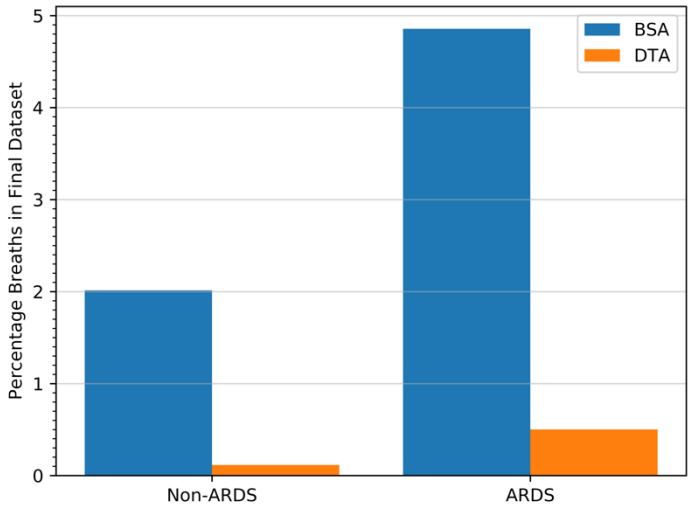
	ARDS	Non-ARDS
Primary Ventilator Mode (%)		
Assist Control-Pressure Control	60%	72%
Assist Control-Volume Control	26%	4%
Pressure-Regulated Volume Control	6%	2%
Pressure Support	6%	20%
Other	2%	2%
Mean RASS (range +4 to -5)	-3.1	-2.0
Cisatracurium Use (%)	26%	2%
Mean Duration of Cisatracurium (hr.)	12.5	14.9

Appendix Table C.2: Additional clinical characteristics of the study cohort in the first 24 hours of mechanical ventilation. *Other,* may include ventilator modes such as synchronized intermittent mandatory ventilation, volume support, and proportional assist ventilation; *RASS*,

Richmond Agitation-Sedation Scale, range from +4 indicating combative to -5 indicating unarousable, with 0 indicating alert and calm



Appendix Figure C.1: Ventilator waveform data (VWD) quality analysis workflow. We first determined if at least 1 hour of VWD was present in the first 24 hours after the start of mechanical ventilation (non-ARDS) or after Berlin criteria were first met (ARDS). If so, we then ensured that VWD time stamps overlapped with the timing of mechanical ventilation as charted in the electronic health record. We excluded VWD that occurred outside of the study window (i.e. - 1st 24 hours) to ensure that VWD from non-invasive ventilation used prior to intubation did not contaminate the data set.



Appendix Figure C.2: Proportion of common patient-ventilator asynchronies detected using previously validated software in ventilator waveform data collected in the first 24 hours in ARDS and non-ARDS subjects. *BSA*, breath stacking asynchrony; *DTA*, double trigger asynchrony

C.2: Machine Learning Features and Algorithms

Features were extracted and aimed to capture relevant respiratory pathophysiology while avoiding features that might strongly correlate with ARDS treatments such as tidal volume (TV) or positive end expiratory pressure (PEEP). Features were first extracted for each breath and then we calculated the median of each feature for consecutive 100 breath windows. These median values were used to construct individual training/testing observations.

To prevent erroneous feature values resulting from periodic episodes of poor-quality waveform data, breath data were dropped if there was a NaN (e.g., derived features with zero in the denominator) on infinite value (e.g., derived value with zero in the numerator) found for any of the features. Breath windows were dropped if more than 50% of a 100-breath frame consisted

of NaN or infinite values. Across all subjects, we found that 1.5% of breaths and 0.86% of windows were dropped.

Below we provide additional detail on the methods used to extract the features described in Table 4.2 in Python-style pseudocode to render them more human-readable. Not all methods are explicitly detailed, only code relevant to feature extraction. All extraction code is publicly accessible at GitHub.

```
def pef_to_zero(flow, time_offset, pef_index):
"""
      Find the slope from peak expiratory flow (pef) to approximately zero
      flow.
      To prevent calculation of non-representative slopes resulting from poor
      waveform quality or asynchrony, we calculate zero flow as the flow
      closest to zero +/- 2 liters per minute and closest to breath end.
    """
    # When flow is within range of -2 to 2.
    extra_offset = time_offset / 0.02
    flow_threshold = 2
    for obs in flow[pef_index+extra_offset:]:
        if abs(obs) < flow_threshold:
            flow_zero_index = flow.index(obs)

      # If expiratory flow never reached between -2 and 2 before breath end,
      then return a NaN
    if flow_zero_index is None:
        return NaN

    slope = (flow[flow_zero_index] - flow[pef_index]) / (flow_zero_index -
pef_index)
    # If somehow slope was negative then assume waveform quality problem and
return a NaN, otherwise return the slope
    if slope < 0:
        return NaN

    return slope

def feature_extraction(breath):
    """
    flow = extract_flow_data(breath)
```

```
    pressure = extract_pressure_data(breath)
    x0_index = find_point_where_inhalation_ends(flow, pressure)
    i_time = len(flow[:x0_index]) * 0.02
    e_time = len(flow[x0_index:]) * 0.02
    i_e_ratio = i_time / e_time
    resp_rate = 60 / (i_time + e_time)
    # pef stands for peak expiratory flow
    pef_index = flow.index(min(flow[x0_index:]))
    mean_flow_from_pef = mean(flow[pef_index:])
    pef_plus_0_to_zero = pef_to_zero(flow, 0, pef_index)
    pef_plus_16_to_zero = pef_to_zero(flow, 0.16, pef_index)

    # Extracts TVi for breath, but not used as a feature in model
    tvi = simpsons_method(flow[:x0_index])
    # Extracts TVe for breath, but not used as a feature in model
    tve = simpsons_method(flow[x0_index:])
    # Extract pip, but not used as a feature in model
    pip = max(pressure[:x0_index])
    # Extract peep but not used as a feature in model
    peep = mean(pressure[len(pressure)-5:])

    # Use TVi, pip and peep to extract dynamic compliance
    dyn_compliance = tvi / (pip - peep)
    tidal_volume_ratio = tvi / tve
```

BIBLIOGRAPHY[3]

Gregory Rehm

Candidate for the Degree of

Doctor of Philosophy

Dissertation: A Computational System for Detecting the Acute Respiratory Distress Syndrome Using Physiologic Waveform Data from Mechanical Ventilators

Major Field: Computer Science

ADVISER'S APPROVAL: Chen-Nee Chuah

---

[3] IF NECESSARY (should not exceed one page except for PhDs)