Transforming Critical Care: Machine Learning Prediction for Arterial Oxygen

Evaluating the Feasibility and Performance of a Transformer Model

Master's Thesis in MAS Data Science

Submitted:	January 27th, 2024
By:	Dominik Wolf born on September 05, 1988
Supervisor:	Dr. Jasmina Bogojeska

ZHAW
Zürcher Hochschule für Angewandte Wissenschaften
Gertrudstrasse 15
CH-8401 Winterthur
Homepage: https://www.zhaw.ch/

Abstract

Healthcare providers are currently facing various challenges, including a shortage of qualified personnel, increasing cost pressures, and workflow inefficiencies. If these issues are not addressed, they will ultimately lead to a decline in the quality of care. By leveraging the increasing digitalization in healthcare, clinical decision support systems can offer a promising solution to these problems by supporting healthcare professionals in their workflows and decision-making processes.

This thesis is a first step towards developing a software solution that utilizes machine learning to predict changes in blood gas levels of critically ill patients. If successful, this application will reduce the need for arterial blood gas tests, lower healthcare expenses, and enhance the quality of care and clinical outcome.

The research scope of this thesis is to investigate the feasibility of employing a transformer model for the prediction of a patient's partial oxygen pressure in arterial blood based on vital signs, lung mechanics, and ventilation data. This also involves examining the transformer's capacity to handle asynchronous time series data and comparing its performance to two simple baseline models.

The utilized transformer model comprises an encoder-decoder architecture that includes an altered attention mechanism that has near-linear computational complexity and, therefore, enables the model to process large input sequences. Supervised learning was performed based on 90,000 days’ worth of ventilation data from over 16,000 patients, extracted from the MIMIC-IV database.

The results show that the utilized architecture effectively handles the challenges of asynchronous time series in clinical datasets. However, the model's performance in predicting the partial pressure of oxygen was limited. Despite initial experiments showing marginal improvements over baseline models, the training loss exhibits high variability. Neither hyperparameter optimization, reducing model complexity, nor enhancing the training set yields significant improvement. This led to the conclusion that, in conjunction with the provided dataset, the chosen model has substantial limitations.

Among other potential issues, the primary constraint identified is the small data-to-model-complexity ratio. Future improvements could include transitioning to an encoder-only architecture to leverage transformer capabilities better and conducting an in-depth dataset exploration to enrich the dataset with more informative features.

In summary, for our specific use case, evidence suggests that transformers may not offer substantial benefits considering the required implementation efforts. Yet, under optimal conditions and with expertise for effective implementation, they can prove effective.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Clinical Background

1.1.1 Introduction to Mechanical Ventilation

The lungs' primary function is to add oxygen and remove carbon dioxide (CO2) from the blood. With every breath, air is drawn through the trachea and bronchi into small air sacs called alveoli at the ends of the bronchi. These sacs are surrounded by small blood vessels (capillaries). Oxygen moves through the thin membranes of the alveoli and enters the bloodstream. The red blood cells collect the oxygen and transport it to the body's organs and tissues. As the blood cells discharge the oxygen, they take up carbon dioxide, a metabolic waste product. The carbon dioxide is then transported back to the lungs and released into the alveoli. During each exhalation, carbon dioxide is expelled from the bronchi and out through the trachea.

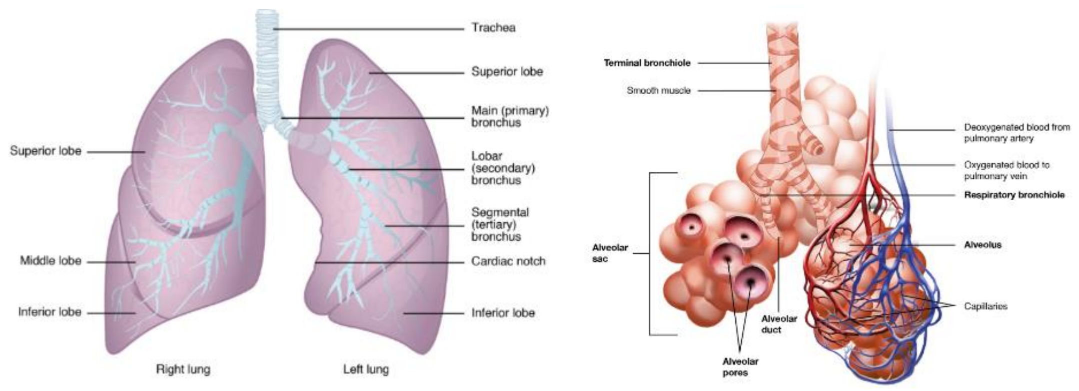


Figure 1 Anatomy of the lungs (left) and alveolar sacs (right), surrounded by small blood vessels (capillaries) that enable the exchange of oxygen and carbon dioxide between the lungs and the bloodstream (Betts et al., 2022).

If the respiratory system is unable to maintain either normal delivery of oxygen (hypoxemia) to tissues or the normal removal of carbon dioxide (hypercapnia) from the tissue, we talk about respiratory failure (which is further discussed in **1.1.2.**)

Mechanical ventilation is a common procedure used in the intensive care unit, with more than 50% of ICU patients requiring it within the first 24 hours of admission. It is a method of supporting intubated patients when their spontaneous breathing is inadequate to sustain life or to reach a therapeutic target. There are two types of mechanical ventilators: negative pressure ventilators and positive pressure ventilators. Positive pressure ventilators are the preferred choice in hospitals because they offer various ventilation modes and features that allow for more sophisticated delivery of conditioned gas and assistance in ventilation. These machines ensure adequate gas exchange by applying positive pressure inflation to the lung and terminate it according to a set target in volume, pressure, or time. Although mechanical ventilation can save the lives of critically ill patients, it is associated with multiple life-threatening complications, such as air leaks and pneumonia (Sandur & Stoller, 1999).

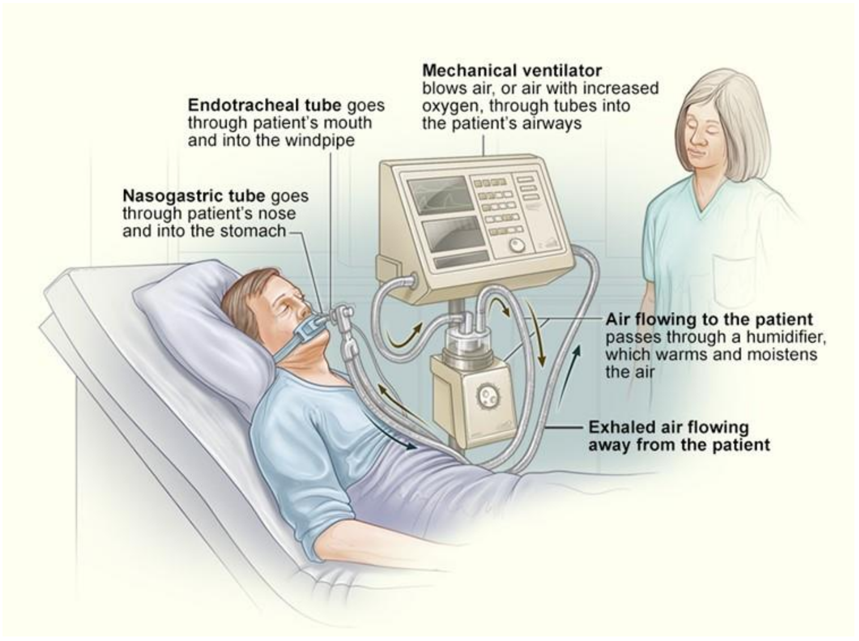

Mechanical ventilator blows air, or air with increased oxygen, through tubes into the patient's airways
Endotracheal tube goes through patient's mouth and into the windpipe
Nasogastric tube goes through patient's nose and into the stomach
Air flowing to the patient passes through a humidifier, which warms and moistens the air
Exhaled air flowing away from the patient

Figure 2 The illustration displays a typical arrangement of a mechanical ventilator used in hospitals. The ventilator supplies the patient with warm, humidified air or air containing additional oxygen through a breathing tube (known as an endotracheal tube) or a close-fitting mask (*What Is a Ventilator*, 2022).

When a patient no longer needs mechanical ventilation, they are ready to be weaned off the ventilator. Candidates for weaning must have adequate oxygenation, carbon dioxide elimination, respiratory muscle strength, and the ability to protect their airway. Non-invasive ventilation refers to ventilator assistance techniques that don't bypass the upper airway, they can serve as a bridge between extubation and spontaneous ventilation, where the patient breathes on their own.

To monitor patients on mechanical ventilation, health care professionals use arterial blood gases, pulse oximetry, and end-tidal CO2 measurements.

1.1.2 Acute Respiratory Failure

Acute respiratory failure (ARF) is defined as acute and progressive hypoxemia developing within hours, days, or up to a month caused by various respiratory, cardiovascular, or systemic disease in previously healthy patients. Among ARF, acute respiratory distress syndrome (ARDS) is a life-threatening condition which is characterized by the activation of white blood cells and pulmonary inflammation that requires immediate treatment (Fujishima, 2023). The causes of this condition include diseases such as COVID-19, inhaling smoke and toxic gases, as well as lung injuries.

Acute respiratory failure (ARF) encompasses a range of diseases that ultimately lead to the same physiological outcomes: arterial hypoxemia; partial pressure of oxygen (PaO2]) of < 60 mm Hg or hypercapnia; partial pressure of carbon dioxide (PaCO2) of > 45 to 50 mm Hg.

The overall efficiency of gas exchange can be assessed in terms of maintenance of normal PaO2 and PaCO2. The assessment can be performed by calculating the alveolar-arterial partial pressure oxygen difference (PAO2 – PaO2), which is also known as the A-a gradient. However, as most patients with ARF receive supplemental oxygen it is clinically more useful to use the ratio of PaO2 to FiO2[1] (normally > 400 mm HG) to assess gas exchange efficiency. This ratio also forms one of the basic criteria for the diagnosis of ARDS (Keyt & Peters, 2019).

[1] Fraction of Inspired Oxygen is 21% in room air and up to 100% during oxygen therapy or mechanical ventilation.

As seen above the partial pressure of oxygen (PaO2) in arterial blood is a vital clinical indicator for detecting acute respiratory failure. It is further used for assessing a patient's oxygenation status and therefore serves for optimizing mechanical ventilation settings and serves as a basis for important therapeutic decisions.

1.2 Relevance

1.2.1 Challenges in Healthcare

Healthcare providers are currently facing numerous challenges such as shortage of qualified personnel, increasing cost pressures, and workflow inefficiencies. If not addressed, these challenges will ultimately lead to a decrease in the quality of care provided to their patients. Therefore, innovative solutions using data-based systems are urgently needed to support clinical staff in their daily workflows and decision-making processes. These systems can provide valuable insights into patient care, help optimize treatment plans, and ensure that healthcare professionals have access to accurate and up-to-date information at all times. By leveraging these tools, healthcare providers can improve efficiency, reduce costs, and deliver better patient outcomes.

1.2.2 Economic Impact of Frequent ABG Tests

The monitoring of arterial oxygen levels in mechanically ventilated patients is common practice in intensive care units. Blood gas analyses (BGA) are used for this purpose, but they are resource-intensive and only provide snapshot information.

However, the process of drawing and analyzing ABG samples is invasive, time-consuming, and expensive. As one of the most commonly used diagnostic tests, it accounts for as much as 10-20% of all costs incurred during an ICU stay, while increasing the risk of infections and "anemia of chronic investigation" (DellaVolpe et al., 2014).

In a level 3 intensive care unit (ICU), an average of 4.8 to 8.5 ABG tests are conducted per patient per day (Walsh et al., 2020). Among these tests, 50% are considered inappropriate, often occurring during shift changes, after treatment discontinuation, or following the cessation of ventilatory support or oxygen delivery for stable patients. The combined expenses for materials and labor associated with each ABG test are estimated to be around $20. In a hypothetical scenario, considering a 12-bed ICU operating at 80% capacity, the number of inappropriate tests sums up to between $175,000 and $310,000 per year.

1.2.3 Clinical Decision Support Systems

The clinical decision support systems market is estimated around $1.7 billion in revenue in 2023. It is expected to grow steadily at a compound annual growth rate (CAGR) of 7.5%, reaching $2.5 billion by 2028 (*Clinical Decision Support Systems Market*, 2023). This growth demonstrates a clear trend toward adopting software solutions that help healthcare providers deliver high-quality care while managing financial and staffing constraints.

Intensive care units (ICUs) make up a significant portion of hospital budgets, despite treating a relatively small number of patients. This is due to the complex procedures involved, the use of expensive technology and medications, and the need for highly trained staff. Patients admitted to ICU with acute respiratory distress syndrome (ARDS), which is an inflammatory lung condition, are especially expensive to treat. They have a high mortality rate and often experience a lower quality of life compared to other critically ill patients. Patients with ARDS often require mechanical ventilation (MV), a particularly costly life-sustaining therapy (Marti et al., 2016).

According to the LUNG SAFE study, 34.9% of the patients enrolled suffered from acute hypoxemic respiratory failure (Pham et al., 2021). Among these patients, 69% met the criteria for acute respiratory distress syndrome (ARDS) according to the Berlin definition (“Acute Respiratory Distress Syndrome,” 2012). The mortality rate for severe ARDS was found to be as high as 46.1%. However, only 34.0% of clinicians were able to recognize ARDS at the time of fulfillment of the criteria. This highlights the need for decision-support tools to enable healthcare professionals the timely detection of respiratory failure.

1.3 Research Aim

1.3.1 Clinical Use Case

Our research aim is to develop a software solution that can assist in managing critically ill patients in intensive care units who require frequent Arterial Blood Gas (ABG) tests to monitor their oxygenation and guide treatment decisions.

To achieve this goal, we want to leverage state-of-the-art machine learning techniques for time series forecasting. We aim to continuously analyze a set of patient parameters, including vital signs, lung mechanics, and ventilator settings, to intelligently predict fluctuations in blood gas levels.

We strive to have a positive clinical impact by reducing the number of unnecessary ABG tests and thus decrease healthcare costs. Additionally, we hope to provide clinicians with timely and informed decision-making capabilities that improve the quality of care and patient outcomes.

This thesis represents an initial step towards the clinical use case mentioned above and aims to achieve the following research objectives.

1.3.2 Research Objectives

Within this thesis, the primary objective is to investigate the feasibility of employing a transformer model for a prediction task on asynchronous time series data. The thesis aims to address the following key research questions.

1. Investigate, whether the transformers model is capable of effectively handle asynchronous time series data.
2. Evaluate the performance of the transformer model to predict the partial pressure of arterial oxygen based on ventilation parameters and the patient’s lung mechanics.

2 Theoretical Foundation

2.1 Introduction to Time Series Analysis

Time series are everywhere and arise naturally in many contexts. Data is often collected at fixed intervals, such as the daily maximum temperature or the total number of passengers each month. Time series are prevalent in various fields, including transportation (e.g., traffic data), business (e.g., sales figures), economics (e.g., stock prices), official statistics (e.g., population demographics), natural sciences (e.g., population sizes, solar activity), environmental sciences (e.g., precipitation, temperature, air pollutant measurements), and more.

Statistical analysis of such data is referred to as time series analysis. Its main goals are understanding past trends and predicting future values. While simple (graphical) methods from descriptive statistics can significantly improve data understanding, a model-based analysis often provides more insights into relationships.

In cases where a good model for the data can be found, it becomes possible to generate predictions for future values. These predictions serve as a basis for budgeting, procurement decisions, etc. Furthermore, time series models can also be used to assess fluctuations in the data, determining whether observed deviations are within the realm of random variability or if substantial changes have occurred (Hofer, 2021).

2.1.1 Characteristics of Health Care Data

In the landscape of health care, the utilization of patient data has become increasingly pivotal for understanding and managing various aspects of patient well-being. In critical care medicine, one of the most data intensive medical specialty, time series data is routinely utilized for understanding the patient trajectory and managing treatment strategies. Digitalized electronic health records (EHR) have become the single source of truth for health care professionals and the go to tool for analyzing monitoring and treatment data. In their study, Manor-Shulman and colleagues reported a median of 1,348 clinical data points for each 24-hour period per patient are being collected in EHR (Manor-Shulman et al., 2008).

Those data points come in various forms, including clinical observations, lab reports, hospitalizations and discharges, demographics, medications, and billing information. Therefore, EHRs can be considered heterogeneous (Denny, 2012). The data format can be structured (e.g. procedure codes, administrative data, laboratory test results), unstructured (e.g. clinical notes for admission, procedure notes, medical history) or semi-structured (e.g. ultrasound data, MRI data, ECG data) (Sarwar et al., 2023). Additionally to the data format, temporality, sparsity, irregularity and imbalanced data pose a significant challenge for the analysis of health care data (Sarwar et al., 2023).

Temporality: Observations such as temperature or blood pressure can change over time, making time-series analysis a suitable tool. However, patients seek medical care based on their unique health conditions and needs, which means that the number of visits and duration can vary from patient to patient. Therefore, it is challenging to compare different patients directly due to the variations in observations. Additionally, patients who require more medical attention may have more frequent visits, which can impact the accuracy of the analysis outcomes.

Sparsity: Electronic Health Record (EHR) data often has missing values due to variations in medical requirements and data collection processes, which can adversely affect its quality. Missing data can complicate the analysis, leading to biased outcomes. Therefore, it's crucial to handle missing data appropriately to prevent biased conclusions.

Irregularity: EHR data is irregularly recorded, with two levels of irregularities observed: visit-level and feature-level. Visit-level irregularity refers to the irregularity observed in the patient’s visits, while feature-level irregularity refers to the appearance of the same feature irregularly in the EHR dataset.

Imbalanced data: EHR data often has class imbalance, with the class of interest being underrepresented. This can result in poor discrimination and calibration of data mining models, leading to biased and poor performance.

2.1.2 Traditional Time Series Models

Traditional time series models have been fundamental in analyzing sequential data and establishing a foundation for forecasting techniques. Two prominent examples of traditional time series models are Autoregressive Integrated Moving Average (ARIMA) and Seasonal ARIMA (SARIMA).

ARIMA stands out as a classic method for analyzing non-stationary time series data. Unlike regression models, ARIMA explains a time series by looking at its past or lagged values. The name "ARIMA" comes from its combination of autoregressive (AR), integration (I), and moving average (MA) operations. The autoregressive component captures the relationship between an observation and several lagged observations, while the moving average component models the dependency between an observation and a residual from previous observations. Additionally, differencing is employed to stabilize the mean of the time series (Box et al., 2016).

Building upon ARIMA, SARIMA incorporates seasonality components to handle periodic patterns in time series data. This model is particularly effective when dealing with data that exhibits repeating patterns over time, such as sales figures affected by yearly trends (Hyndman & Athanasopoulos, 2018). The ARIMA and SARIMA models are popular for their statistical properties and effective modeling process. They are easy to implement, but they can only detect linear relationships in stationary time series data without any missing values. Artificial neural networks, on the other hand, are useful for time series forecasting because they can handle nonlinear relationships and more complex data. (X. Zhang et al., 2013).

2.1.3 Neural Network Approaches

In 1998, Zhang and his colleagues summarized numerous studies on Artificial Neural Networks (ANNs) for forecasting and highlighted several reasons why ANNs are considered suitable for forecasting compared to traditional statistical methods. Firstly, ANNs can model unknown relationships in data without requiring many assumptions. Secondly, they can apply what they have learned to new, unseen data. Thirdly, ANNs, particularly non-linear relationships, can be modeled well as they are universal approximators, capable of representing a wider range of functions than traditional time series models (Hewamalage et al., 2019; G. Zhang et al., 1998).

RNNs are a class of neural networks that are commonly used for sequence prediction problems. In contrary to Feed Forward Neural Network (FFNN), RNNs have connections that form cycles, allowing them to maintain a hidden state and capture temporal dependencies (Hewamalage et al., 2019). Every Recurrent Neural Network (RNN) is composed of multiple RNN units. Among the most widely used RNN units for sequence modeling tasks are Long Short-Term Memory (LSTM) cell, and Gated Recurrent Unit (GRU) (Cho et al., 2014; Hochreiter & Schmidhuber, 1997).

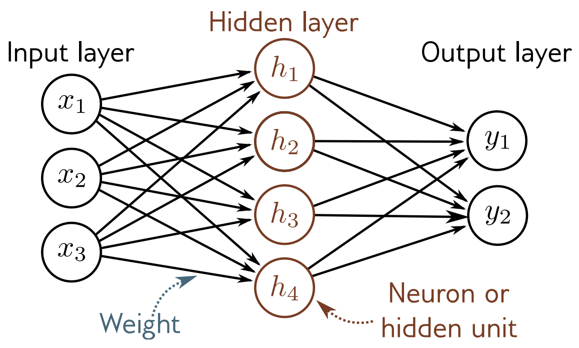


Figure 3 A shallow neural network includes an input layer, a hidden layer, and an output layer. These layers are connected through forward connections, forming a feed-forward network, with each connection corresponding to a weight (Prince, 2023).

Traditional RNNs face the vanishing gradient problem as the network depth increases. In gradient-based learning methods like backpropagation, the network weights are updated based on the gradient value after each training iteration. Due to certain activation functions and network architectures, the gradient value may become too small during backpropagation, hindering weight updates and, in extreme cases, causing the network to halt training (Basodi et al., 2020).

Long Short-Term Memory (LSTM) Networks are a specific type of RNN designed to address the vanishing gradient problem. LSTMs include memory cells and gating mechanisms that enable them to capture long-term dependencies more effectively. They are well-suited for time series forecasting tasks where understanding and remembering patterns over extended periods are crucial (Gers, 1999).

Gated Recurrent Units (GRUs) are similar to Long Short-Term Memory (LSTM) networks, with a simpler structure. As a result, GRUs use fewer training parameters and may be less computationally expensive than LSTMs. However, LSTMs are better suited for tasks that require a deep understanding of context. Other than that, both techniques perform similarly, and it can be challenging to determine which one is better in a given situation (Bianchi et al., 2017).

2.2 Transformer Models

Transformer models have brought a significant change in the field of natural language processing (NLP) and beyond. They offer an architecture that excels at capturing complex patterns in sequential data. This model was introduced by Vaswani et al. in 2017, and it differs from traditional recurrent and convolutional architectures. The Transformer model relies on self-attention mechanisms to process input sequences in parallel, making it more efficient. With its ability to capture long-range dependencies and scalability to handle diverse tasks, Transformer models have become the backbone of state-of-the-art models in machine translation, text generation, and other domains (Devlin et al., 2019; Radford et al., 2018; Vaswani et al., 2017).

2.2.1 Transformer Model Architecture

Traditional neural network architectures, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs), have shown great success in sequential and spatial data processing. However, these models face challenges in capturing long-range dependencies efficiently. Recurrent models process input and output sequences symbol by symbol in a sequential manner. This means that the model generates a sequence of hidden states based on the previous hidden state and the input at that position. However, this sequential approach makes it difficult to parallelize training examples, especially when dealing with longer sequences. To put it simply, memory limitations restrict the ability to batch across examples which is crucial for efficient training (Vaswani et al., 2017).

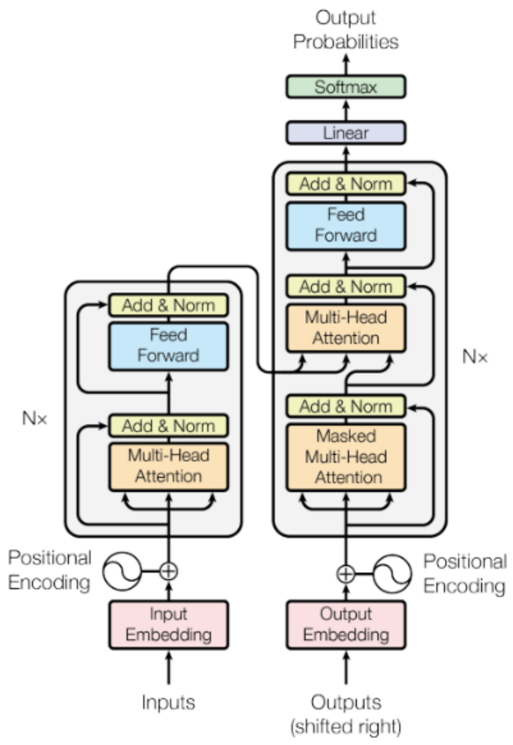


Figure 4 The transformer model architecture presented by Vaswani et al., 2017 showing the encoder on the left and the decorder on the right side.

High-Level Architecture

The Transformer is a deep learning architecture and comprised of an encoder and a decoder (**Figure 4**). The encoder maps an input sequence of symbol representations to a sequence of continuous representations. Given those representations, the decoder then generates an output sequence of symbols one element at a time (Vaswani et al., 2017).

Both, encoder and decoder contain multiple layers. Each layer consists of two sub-layers: a multi-head self-attention mechanism and a feed-forward neural network (**Figure 5**). The multi-head self-attention mechanism allows the model to focus on different parts of the input sequence, while the feed-forward network applies a point-wise fully connected layer to each position separately and identically (Vaswani et al., 2017).

To facilitate training and prevent over-fitting, the Transformer model uses residual connections and layer normalization. Additionally, Vaswani et al.

introduce a positional encoding scheme that encodes the position of each token in the input sequence. This enables the model to capture the order of the sequence without the need for recurrent or convolutional operations.

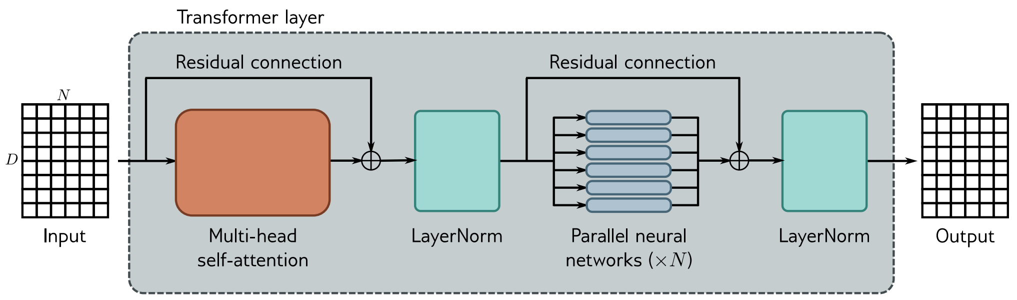


Figure 5 Transformer layer with multi-head self-attention and feed-forward neural network (Prince, 2023)

Embeddings

To utilize transformers or in general neural networks for text processing, the initial step involves converting the text into a numerical format. This process, known as embedding, aims to create a numerical representation that can capture important characteristics of the text, such as the relationships between words or the sentiment of the text. The ideal embedded representation should accurately reproduce these characteristics (Jurafsky & Martin, 2023).

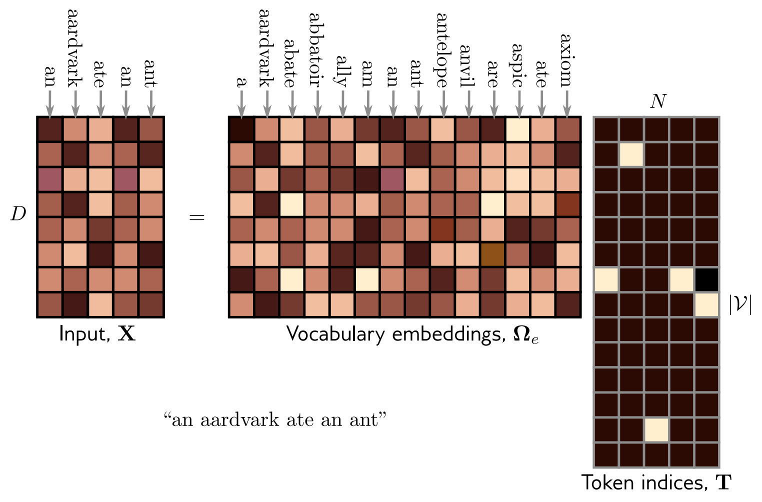


Figure 6 The input embedding matrix X is created by multiplying the vocabulary matrix with a matrix containing one-hot vectors in its columns, corresponding to word indices. This results in N embeddings of length D (Prince, 2023).

Each word (or token) in the vocabulary is associated with a unique word embedding. These embeddings are combined to form a matrix that represents the embeddings of the entire vocabulary. In this Matrix building a multidimensional vector space of Dimension D, words that are similar in meaning are located close to each other, allowing the model to establish semantic relationships. For transformers, the embedding matrix is a network parameter that needs to be learned like any other (Prince, 2023).

Self-Attention

The Transformer architecture was developed to overcome the limitations of traditional sequence processing solutions by providing a scalable and parallelizable alternative.

The self-attention mechanism lies at the core of the Transformer, allowing the model to assign importance to different positions in the input sequence while making predictions. Unlike RNNs, which process sequences sequentially, the self-attention mechanism enables the Transformer to consider all positions simultaneously. This, in turn, enables the Transformer to capture contextual information, regardless of distance, which is particularly useful when dealing with long sequences (Bahdanau et al., 2016; Vaswani et al., 2017).

The matrix computation in the self-attention layer of the model is visualized in **Figure 7**. The input matrix X contains N word embeddings with a dimension of D. Query matrix Q, key matrix K, and value matrix V are then separately operated on the input X. The dot products of matrix Q and K are then calculated using matrix multiplication, and each column of the resulting matrix is independently applied with a softmax operation to calculate the attentions. Finally, the attentions are multiplied by the values to create an output that has the same size as the input (Prince, 2023).

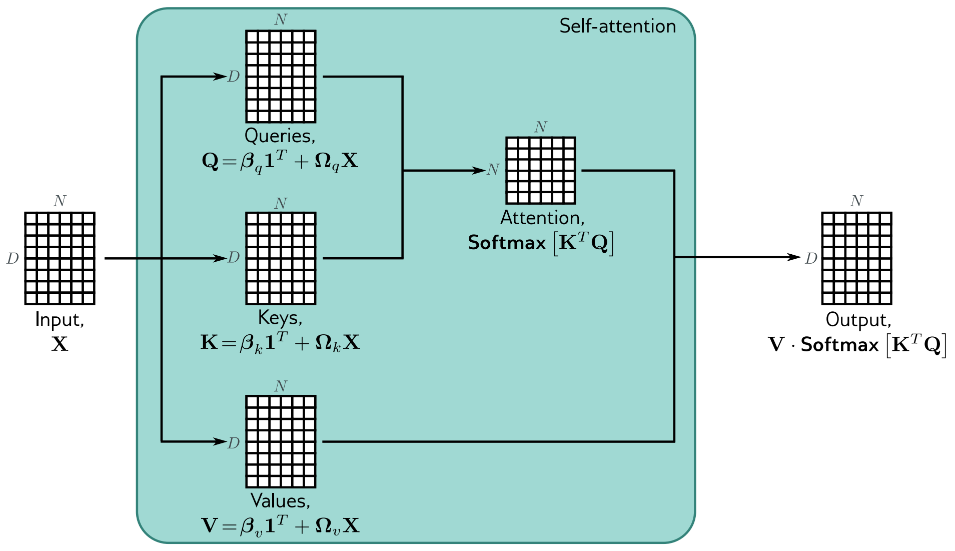

Self-attention
N
D
Queries,
$\mathbf{Q} = \boldsymbol{\beta}_q \mathbf{1}^T + \boldsymbol{\Omega}_q \mathbf{X}$
Keys,
$\mathbf{K} = \boldsymbol{\beta}_k \mathbf{1}^T + \boldsymbol{\Omega}_k \mathbf{X}$
Values,
$\mathbf{V} = \boldsymbol{\beta}_v \mathbf{1}^T + \boldsymbol{\Omega}_v \mathbf{X}$
Attention,
Softmax $[\mathbf{K}^T\mathbf{Q}]$
Input,
$\mathbf{X}$
Output,
$\mathbf{V} \cdot$ Softmax $[\mathbf{K}^T\mathbf{Q}]$

Figure 7: Self-attention computation in matrix form (Prince, 2023)

In their paper, Vaswani et al. introduced a model that employs multi-head self-attention layers to process the input matrix X. This means that several attention layers are working simultaneously. To achieve this, the input matrix is split along D, and the outputs are then joined together vertically. Another linear transformation is applied to recombine them (Vaswani et al., 2017). The reason for using multiple heads is to make the self-attention network more resilient to poor initializations. This technique is believed to enhance the robustness of the self-attention network (Prince, 2023).

The Transformer model by Vaswani displays three distinct ways to employ multi-head attention:

1. The encoder includes **“self-attention”** layers as described in **Figure 7**. Where all the keys, queries, and values come from the same source - in this case, the output of the previous layer in the encoder. This enables each position in the encoder to attend to all positions in the previous encoder layer.
2. In **"encoder-decoder attention"** also known as **“cross-attention”** layers, the queries are derived from the previous decoder layer, while the memory keys and values are taken from the encoder's output. This allows every position in the decoder to attend to all positions in the input sequence.

3. Similarly, to self-attention layers in the encoder, the decoder employs a **“masked attention layer”** enabling each decoder position to attend to all positions in the decoder, up to and including that position, which means that each word in the sequence can only attend to its own embedding and those of words earlier in the sequence.

Types of Transformer Models

The process of passing an embedding matrix X through a series of K transformer layers is called the transformer model. There are three types of transformer models: encoder, decoder, and encoder-decoder. The encoder transforms text embeddings into a representation that can support a range of tasks. The decoder can predict the next token to continue the input text. Encoder-decoders are used in sequence-to-sequence tasks like machine translation, where one text string is converted into another. The next chapter will discuss some applications of the three transformer models.

2.2.2 Applications of Transformer Models

Text and Word Classification: Google's BERT (Bidirectional Encoder Representations from Transformers) is an encoder model that was trained in two stages. In the pre-training stage, BERT was trained under self-supervision to predict missing words from sentences from a large internet corpus. The fine-tuning process was done by appending an extra layer to the transformer to convert the output vector to the desired format, which specialized the network to the corresponding task. The final network was successfully applied for sentiment analysis (where the passage is labeled as having a positive or negative emotional tone) or for named entity recognition to classify each word as an entity type (e.g. person, place, organization) (Devlin et al., 2019; Prince, 2023).

Text Generation: OpenAI's GPT (Generative Pre-trained Transformer) is a decoder model that has demonstrated remarkable proficiency in text generation. It was trained unsupervised on vast datasets using language modeling as a training signal and then fine-tuned on smaller supervised datasets for specific tasks. The approach is task-agnostic and achieves state-of-the-art performance across various tasks. Unsupervised learning is chosen as it overcomes the limitations of supervised learning that require large, carefully curated, and expensive datasets (Radford et al., 2018).

Machine Translation: Translation from one language to another requires a sequence-to-sequence model consisting of an encoder and a decoder. The encoder processes the source sentence and produces a representation for each word. The decoder generates each output word by attending to the previous output words and the source sentence using a self-attention layer that allows the decoder embeddings to attend to the encoder embeddings

The model introduced by Vaswani et al. has outperformed all the models reported before in the English-to-German translation task, resulting in a new state-of-the-art. It has also surpassed all the previously published models in English-to-French translation while costing less than one-fourth of the training cost of the previous state-of-the-art model.

Further Applications: Initially started in the domain of natural language processing Transformers now progress and find applications in many other fields. These applications include Vision Transformer (ViT) for image classification, conformer for speech recognition, detection transformers (DETR), Text-to-Image generative models such as DALL-E from OpenAI, and many more (Carion et al., 2020; Dosovitskiy et al., 2021; Gulati et al., 2020; Shi et al., 2020).

3 Methodology

3.1 Data Set

In order to test the model on a dataset with irregularly sampled and non-synchronized measurement, the MIMIC-IV database was chosen (Johnson et al., 2023). The MIMIC-IV is the fourth iteration of the MIMIC database and is widely used in the healthcare and research communities, particularly in the field of critical care. It is a publicly available database that includes de-identified health-related data from over 430,000 patients admitted hospitals and over 73,000 patients admitted to intensive care units. This database contains a wealth of information, including clinical notes, demographic details, vital sign measurements, laboratory test results, medications, and more.

3.1.1 Parameter Selection

To examine the dataset and select our parameters of interest, various ventilation procedures were plotted and visually analyzed. **Figure 8** illustrates nicely that the dataset is highly imbalanced, with differing and fluctuating sampling frequencies. The example nicely confirms the challenges for the analysis of health care data as describe chapter **2.1.1**. The five parameters selected variable and their characteristics are further described below.

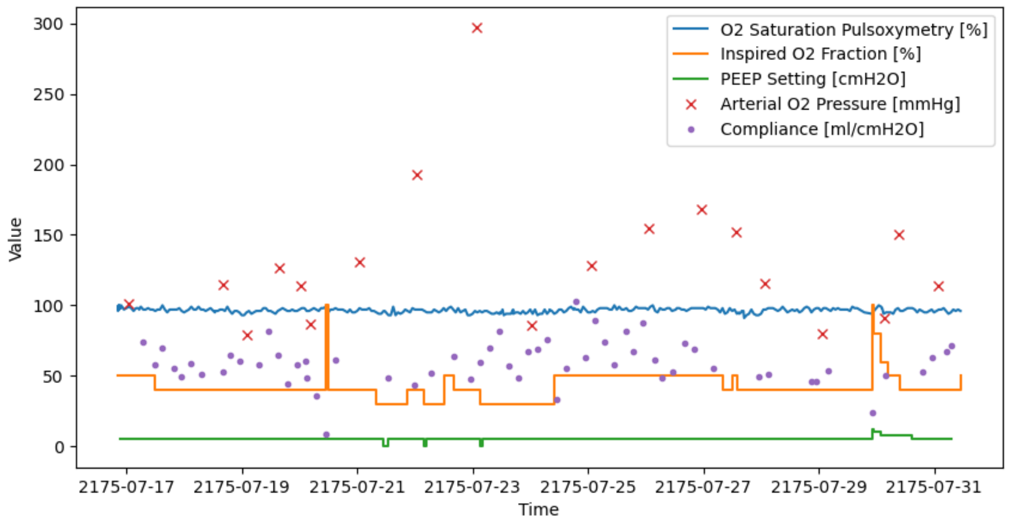


Figure 8 Data sample for a patient's ventilation procedure over several days

Arterial O2 Pressure: As described before the partial oxygen pressure in arterial blood is the most important parameter for monitoring a patient oxygenation status and guide treatment decisions. The parameter is derived by drawing a sample of the patient’s arterial blood with subsequent laboratory test. The sampling interval of 4.37 hours matches roughly with the regular bedside visits of the respiratory therapists who is responsible for a patient’s respiratory care. However, it also displays a high standard deviation, indicating that it’s availability is influence by a variety of factors such as the patient’s condition, or r external factors such as staffing constraints, bedside expertise, or other clinical priorities.

PEEP and FiO2: Positive end-expiratory pressure (PEEP) and fraction of inspired Oxygen (FiO2) are two important settings of the mechanical ventilator that greatly affect the patient's oxygenation during mechanical ventilation. FiO2 refers to the percentage of oxygen that the patient receives per breath, which can range from 21% (room air) to 100%. PEEP, on the other hand, refers to the baseline pressure provided to the patient by the device. With increasing PEEP,

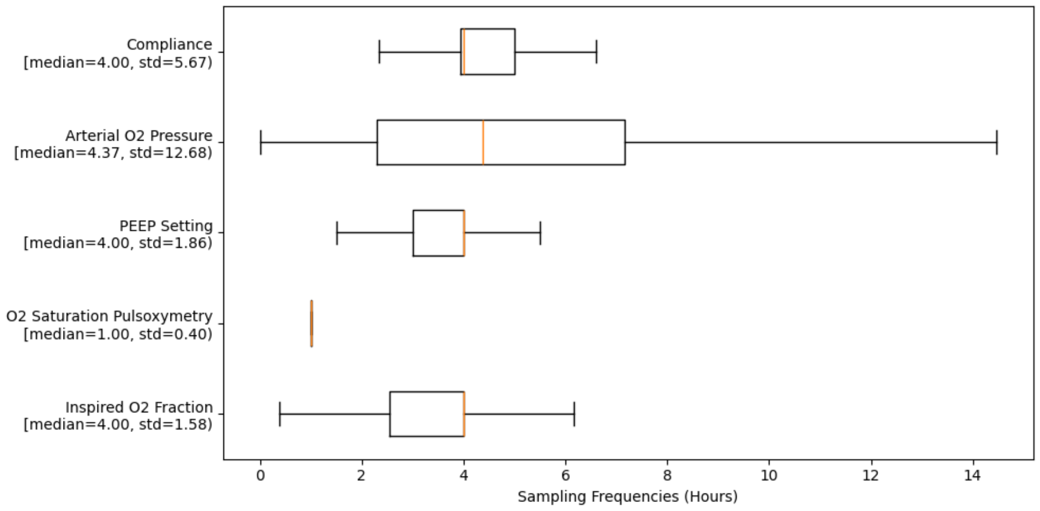


Figure 9 Sampling frequencies of the parameters. Outliers are not shown for increased readability.

the lungs remain more inflated after expiration, leading to more surface area for gas exchange. As these two values are typically automatically transmitted by the device, their availability should be constant. However, the data indicates they are requested every 4 hour or on change by the Electronic Medical Record (EMR) system.

Compliance: Pulmonary compliance describes the patient’s lung mechanics and is measuring the extent to which the lung will expand for each unit increase of applied pressure. Health care professionals rely on that concept to understand some pathologies (e.g. ARDS) and help to guide therapy. An decreased compliance value can indicate pulmonary inflammations that greatly reduces a lungs gas exchange efficiency. Compliance can be measured manually or automatically be the mechanical ventilator. The data displays a similar availability like FiO2 and PEEP, therefore it is assumed that it is also transmitted automatically.

Pulse oximetry: Pulse oximetry is a widely used method for monitoring patients in critical care. It measures the oxygen saturation level by shining light through the skin and measuring the changes in light absorption of oxygenated and deoxygenated blood. This is usually done through a finger probe. It serves as real time monitoring for every patient in the ICU, and therefore it is not surprising that it shows a high occurrence in the datasets. Although there are some inaccuracies associated with pulse oximetry, it is still the most used tool for monitoring a patient’s oxygenation.

3.1.2 Data Preprocessing

The process of collecting and preparing the data involved four main steps: identification, extraction, de-noising, and normalization.

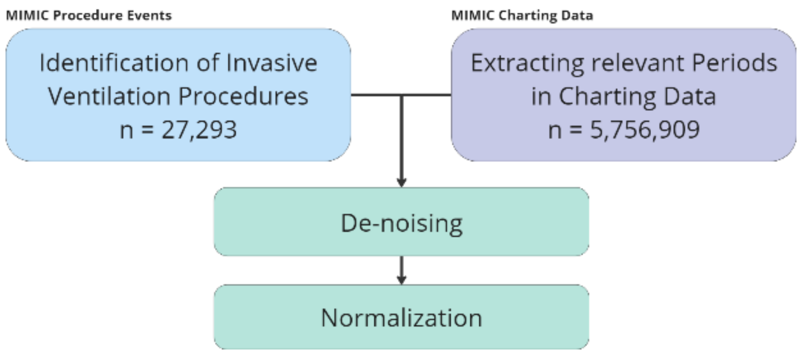

MIMIC Procedure Events
Identification of Invasive Ventilation Procedures n = 27,293
MIMIC Charting Data
Extracting relevant Periods in Charting Data n = 5,756,909
De-noising
Normalization

Figure 10 Data collection and preprocessing workflow

Identification: To efficiently extract the relevant data from the database, we first identified the periods of interest in the MIMIC procedure events table. This table contains the start and end times of all medical procedures and therapies that were applied. In order to effectively train the model, we needed to have our target variable, which is the partial pressure of arterial oxygen, available for analysis. To increase its availability, we focused on events of invasive mechanical ventilation, as ABG tests are frequently performed for patients undergoing this type of ventilation to monitor and adjust therapy. Additionally, as the average number of daily ABG tests performed is between 4.8 to 8.5, we further increased its availability by only considering ventilation events that lasted longer than 4 hours. As seen in **Table 1**, the identified periods resulted in a total of 60,778 days' worth of ventilation data from 16,428 patients.

Descriptive Statistics[2]	
Total Patients [n]	16,428
Patient Gender (female) [%]	40.1
Patient Age [years]	65.0 (54.0-75.0)
Total Hospital Admissions [n]	17,733
Total Ventilation Events [n]	20,207
Total Time on Mechanical Ventilation [days]	60,778
Time on MV per Procedure [hours]	26.8 (10.0-86.8)

Table 1 presents the demographics and statistics of the data that has been extracted.

Extraction: With the periods of interest identified the relevant charting data was accessed and retrieved. The five previously mentioned parameters were extracted, including two ventilation settings (PEEP Setting and Fraction of Inspired Oxygen), two patient monitoring parameters (Lung Compliance and Pulse Oximetry), and the target variable, which is the partial pressure of arterial oxygen.

De-noising: Upon exploring the gathered dataset, numerous errors were detected in the observations. For instance, some observations showed a fraction of inspired oxygen below 21%, which is an unrealistic scenario. To minimize data distortion and enhance dataset quality, extreme values were identified and removed. To achieve this, the 0.1% and 99.9% percentiles were computed, and

[2] Quantitative data are expressed as median (interquartile range), percentage or counts

any observation outside this range was removed. The resulting parameter distribution can be viewed in **Figure 11**.

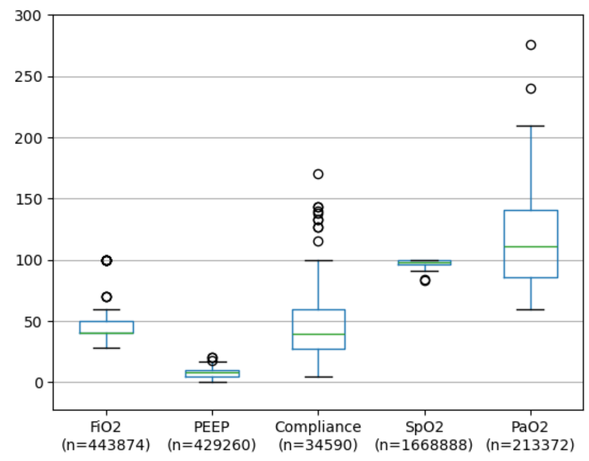


Figure 11 Parameter distribution after the de-noising step

Normalization: To prevent the model from being sensitive to the scale of input features and to ensure that each feature contributes equally to the learning process, we have implemented normalization measures. For each parameter, we calculated the mean and standard deviation of all observations. Then, for each observation, we subtracted the mean and divided the residual by the standard deviation.

3.2 Transformer Model Architecture

As the starting point for development served the Tripletformer, an encoder-decoder Transformer architecture presented by Yalavarthi et al., 2022. If not stated otherwise the architecture remained unchanged. The Tripletformer model was chosen for its ability to tackle several challenges related to working with healthcare data, which were discussed in chapter 2.1.1. This model possesses an architecture that enables it to manage multivariate time series with:

- Non-periodic observations in a channel
- Single examples with very few observations in a channel
- Completely unobserved channels

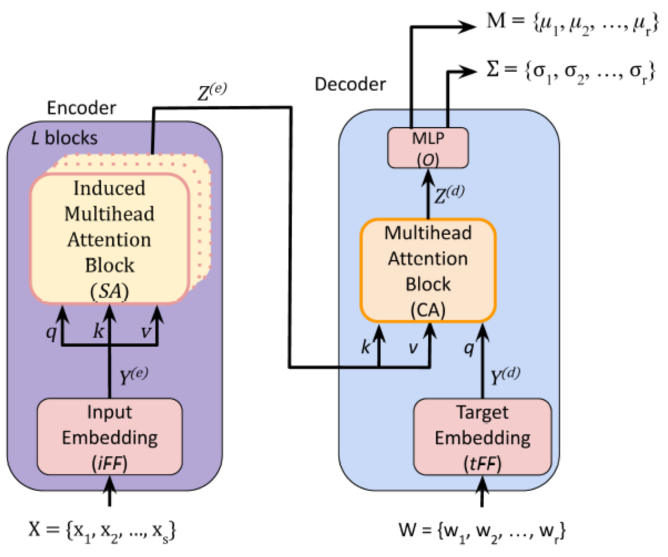

Encoder
L blocks
Induced Multihead Attention Block (SA)
q k v
$Y^{(e)}$
Input Embedding (iFF)
$X = \{x_1, x_2, ..., x_s\}$
$Z^{(e)}$
Decoder
MLP (O)
$Z^{(d)}$
Multihead Attention Block (CA)
k v q
$Y^{(d)}$
Target Embedding (tFF)
$W = \{w_1, w_2, ..., w_r\}$
$M = \{\mu_1, \mu_2, ..., \mu_r\}$
$\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_r\}$

Figure 12 Tripletformer architecture as presented by Yalavarthi et al., 2022

As seen in **Figure 12**, to the encoder (left) maps the input series X which is a set of observations, to a set of representations $Z^{(e)}$. The decoder (right) outputs the probability distribution parameters (mean and, standard distribution) based on the encoder output (Z) and the target queries (W).

The Tripletformer was originally designed for probabilistic interpolation in asynchronous time series (Yalavarthi et al., 2022). However, some minor modifications have been made to repurpose it for our research objective, the prediction of partial oxygen pressure in arterial blood of invasively ventilated patients.

3.2.1 Encoder

As seen in **Figure 12** the input series $X = \{x_1, x_2, ..., x_s\}$ consists of a set of observations (sequence) of length s. Each observation $x_i = (t_i, c_i, u_i)$ consists of a triple of time, (parameter) channel, and observation measurement. Therefore, the model does not require a fixed length of the input sets as they can vary for each example. Additionally, unlike to the traditional transformer model, it does not need positional encodings as the position of the observation is naturally given by the observation timestamp within the triplet.

Embedding Layer: In the encoder, the input embedding layer (iFF) generates the learned embeddings for each observation in the set. It processes the information pointwise, meaning it operates on the individual observations independently. However, as the iFF layer only works on vectors, channel, time, and value is concatenated into one vector for each observation. Additionally, as the channel indicator uses one-hot encoding, the vector is of space $\mathbb{R}^{C+2}$ with C the number of channels. In our case, we have five parameters. An example is given below.

$$x_1 = (t_1, 4, 0.1) \rightarrow y_1 = (t_1, 0, 0, 0, 1, 0, 0.1)$$

The output of the iFF layer therefore is a set of embeddings $\mathrm{Y}^{(e)} = \{y_1^{(e)}, \dots, y_s^{(e)}\}$ of length s.

Attention Block: The initial transformer architecture employs a multihead attention layer, and therefore consists of multiple stacked self-attention layers (Vaswani et al., 2017). As seen in **Figure 7**, the attention matrix is calculated by the dot product of matrix Q and K with a subsequent softmax operation. In our case the attention matrix would be of dimension $s \times s$, and therefore have a quadratic computational complexity. With large input sequences, computing the attention matrix A would become a bottleneck.

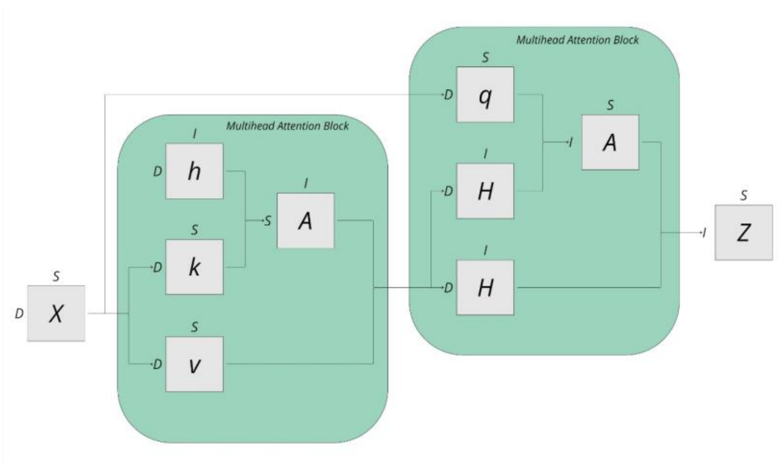


Figure 13 Architecture of the Induced Multihead Attention Block

In order to tackle the issue of quadratic complexity, Yalawarti et. Al utilized an Induced Multihead Attention Block (IMAB). This block is composed of two consecutive multihead attention layers and a trainable parameter matrix, h. As depicted in **Figure 13**, the IMAB resolves the computational bottleneck by employing two attention matrices, A, which exhibit only a linear dependency on the input sequence length. The output Z, is a set of continuous embeddings $Z^{(e)} = \{z_1^{(e)}, \dots, z_s^{(e)}\}$.

3.2.2 Decoder

As shown in **Figure 12**, the decoder takes the encoder output $Z^{(e)}$ and the set of target queries as inputs and returns the mean (M) and standard deviations (Σ) of the target values. The decoder consists of three main components, a target embedding layer (tFF), the cross attention block (CA), and the output layer (O).

Embedding Layer: The target embedding layer (tFF) works analogous to the iFF. It is a point wise feedforward layer that provides the presentation of the learned embedding of the target query. As the value will be estimated by the model, the target queries $W = \{w_1, \dots, w_r\}$ are the concatenated information of time and channel (again with one-hot encoding). In our case w_r is a vector of space $\mathbb{R}^6$. The target queries are then passed through to obtain a set of their latent representation $Y^{(d)} = \{y_1^{(d)}, \dots, y_r^{(d)}\}$

Cross Attention Layer: The cross attention layer takes the latent embeddings $Z^{(e)}$ from the encoder and feeds it as key and values in the multihead attention block. Through the multiplication with representation of the target embeddings $Y^{(d)}$ the cross attention layer outputs the set of learned embeddings $Z^{(d)} = \{z_1^{(d)}, \dots, z_r^{(d)}\}$, which are then passed on to the output layer.

Output Layer: The output layer (O) is a feedforward layer with two output heads. Assuming a Gaussian distribution the output layer produces the distribution parameter $M = \{\mu_1, \dots, \mu_r\}$ and $\Sigma = \{\sigma_1, \dots, \sigma_r\}$ for each target query in W.

3.2.3 Supervised Learning

In the framework created by Yalavarthi and collueges, the model outputs a probability distribution for each query w of the set W. This was considered a significant advantage of the model for a possible future application. By predicting both the mean and standard deviation, the model improves its transparency, enabling the user to see not just the predicted value but also the level of confidence in that prediction.

$\widehat{Pr}(u'|X, w)$ characterizes the probability distribution of the target observation u' and is represented as follows:

$$\widehat{Pr}(u'|X, w) = \mathcal{N}(u'; \mu, \sigma)$$

With the assumption, that the data adheres to a Gaussian distribution, the model utilizes Negative Log-Likelihood (NLL) as the primary loss for model training. To prevent the model from getting trapped in local optima, a mean square error was added to the loss (L).

$$\mathcal{L} = NLL + \lambda\, Mean\, Square\, Error$$

The weight (λ) for mean square error is one of the hyperparameters of the model.

4 Experiments

4.1 Training and Validation Splits

The complete dataset contains data from 16,428 patients who underwent 17,733 hospital admissions, totaling 20,207 ventilation procedures. It's possible for one patient to have multiple admissions and ventilation events, so to improve the model's generalization and prevent it from validating on already encountered patients, the data was sorted by patient and admission ID before splitting into sets. Finally, the data was then split in training (70%), validation (15%) and test-set (15%).

4.2 Sampling

In a clinical setting, ABG tests are commonly initiated by physicians or respiratory therapists. These tests may be performed to monitor the impact of therapy changes or in accordance with hospital guidelines and workflows. Regardless of the reason, the main objective is to obtain the patient's current state of oxygenation. To train the model for this specific scenario, a sampling algorithm ensured that the latest PaO2 value for each ventilation period was selected as the target observation and served as input to the decoder, while all previous data points (including previous PaO2 measurements) were used as input for the encoder. To prevent the model from learning from future observations, data trailing the target observation was excluded from the sample.

4.3 Training Procedure

The training of a transformer model involves several key steps to optimize its parameters and enable it to make accurate predictions. Below some of the key steps of the training procedure will be discussed.

Model inputs: the model takes a variety of inputs to specify the transformers architecture including encoder and decoder components (e.g. number of layer, attention heads). Some of them were modified during hyperparameter tuning to optimize the model’s performance.

Optimizer: The model was optimized using the built-in Adam optimizer from PyTorch, which is a commonly used optimization algorithm for training deep neural networks. Adam stands for Adaptive Moment Estimation and it adjusts the learning rates for each parameter based on the historical gradients and squared gradients. By combining the benefits of both momentum and RMSprop algorithms, Adam aims to provide a more adaptive and efficient optimization process. This makes it effective in training deep neural networks with various architectures and data characteristics.

Learning rate scheduler: The model employed the PyTorch 'ReduceLROnPlateau' scheduler. The purpose of this scheduler is to dynamically adjust the learning rate during training. When the validation loss stops improving (after patience epochs), it reduces the learning rate to help the optimization process converge more effectively. This can be particularly useful in preventing the model from getting stuck in a local minimum or oscillating around the optimal solution.

Training loop: During each training iteration, the model is fed with batches of the training dataset. For every batch, a sampling algorithm is used to define the target observations and their corresponding context information (input series X). From the target observations, the channel and time information are utilized to construct the target queries W. During the forward pass, the input series X and target queries W are passed through the model to produce the predictions. The value of the target observation is considered as the ground truth and is compared to the prediction to calculate the training loss. The gradients of the loss with respect to the model parameters are calculated during the subsequent backpropagation. To avoid any problems with exploding gradients, the gradients are clipped before being fed to the optimizer to update the model parameters. The training dataset is iterated through the model until either a defined number of iterations is reached or the condition for early stopping is met.

Early stopping: During each iteration, the training and validation loss are calculated to evaluate training performance. To prevent overfitting on the training set, training is stopped if there is no improvement in performance on the validation set for 30 iterations.

4.4 Hyperparameter Tuning

To optimize model performance, a hyperparameter search was conducted. Various parameters such as the number of layers, number of hidden units, and batch size were adjusted to optimize for performance on the validation set.

The following Hyperparameter were searched:

- Learning Rate: [0.01, 0.001, 0.0001, 0.00001]
- Batch Size: [32, 64, 128, 256, 512]
- Attention layer dimension [16, 32, 64, 256]
- Hidden units in feedforward layers [16, 32, 64, 256]
- IMAB layers [1, 2, 4]

4.5 Hardware and Software

The experiments were conducted using PyTorch as the primary deep learning framework. The neural network models were trained using many of PyTorch's build in functionalities (optimizer, scheduler, etc.). The coding and experimentation processes were facilitated by the Visual Studio Code (VSCode) integrated development environment. And to enhance computational performance, an NVIDIA GeForce RTX 3070 graphics processing unit (GPU) was utilized.

5 Results and Discussion

5.1 Training Performance

During the training process, the models exhibited a high level of fluctuation in their training loss. The fluctuations were so intense that it was difficult to detect any clear trend in the model's performance, and its apparent improvement seemed to be occurring purely by chance. Several reasons were considered to be causing this behavior.

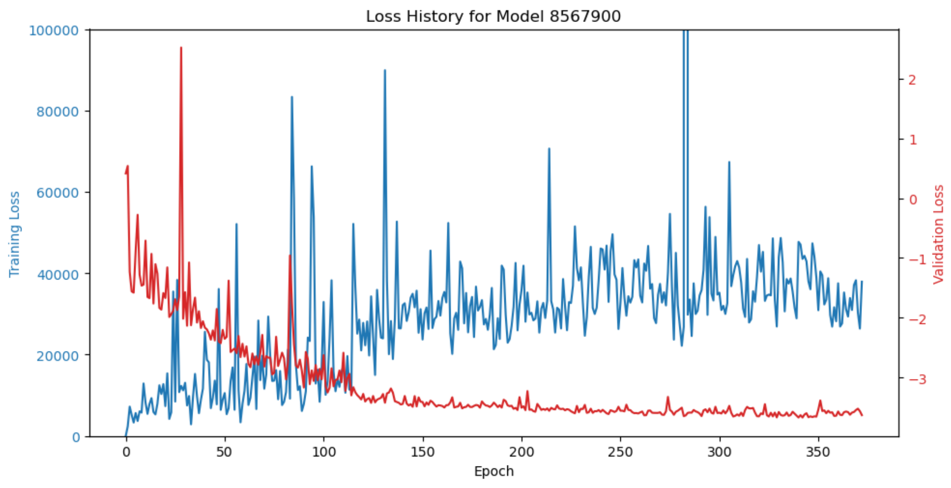


Figure 14 Example training performance with negative likely hood (NLL) on training and validation set

Inappropriate learning rate: The learning rate determines the magnitude of the adjustments made to the network's parameters by the optimizer. If the learning rate is set too high, the network might overshoot the optimal solution and diverge, leading to poor results. To avoid overshooting and improve the efficiency of the model in fitting the data, a lower learning rate was assumed to be a potential solutions and addressed during the hyperparameter search.

High model complexity: A complex model may not converge due to various reasons. One of the most common reasons is overfitting the training set, where the model fits too closely to the training data, making it difficult to generalize to new unseen data. However, as the training loss is highly fluctuating it is hard to see any convergence. In the example above it even seems that the loss is increasing over time.

Another important factor to consider is that models with high complexity relative to the amount of training instances can lead to high variance in the training process. Each subset of data may significantly impact the model parameters, resulting in instability and failure to converge. To address both issues during the hyperparameter search, parameters were changed that affect the transformer's architecture to monitor the effect of model complexity. Additionally, there was a focus on larger batch sizes to reduce the variance effect.

Insufficient training data: If the dataset used to train a model is insufficient in size or lacks diversity, the model may not receive enough information to accurately identify patterns in the data. This can lead to incorrect predictions for certain inputs, causing fluctuations in the training accuracy. To examine the impact of the amount of training data, some experiments involved breaking down ventilation periods into smaller pieces, thereby significant increasing the target observations in the training dataset.

5.2 Hyperparameter Tuning

The aim of the hyperparameter tuning was to enhance the accuracy of the model and resolve the issues of high fluctuations and low convergence in the training loss that were initially encountered. In order to achieve this, a search space was defined within the set of hyperparameters, and the model's performance on the validation dataset was evaluated. A comprehensive performance overview of all models can be found in Appendix A.

5.2.1 Initial Hyperparameter Search

In their study, Yalavarthi et al., evaluated their models on a dataset retrieved from an older version of the MIMIC database. Therefore, their best-performing set of hyperparameters was taken as a promising starting point for the search.

Following hyperparameters have been explored: Learning rate {0.01, 0.001}, batch size {32, 64, 128}, IMAB layers {2, 4}. The performance of the model was assessed on the validation set using negative log likelihood (NLL), mean squared error (MSE), and mean absolute error (MAE).

Model	Val. Loss (NLL)	Val. Loss (MSE)	Val. Loss (MAE)	Model Param. (n)	Learn. Rate	Batch Size	Enc Heads	Dec Heads	IMAB Dim	CAB Dim	Dec Dim	N Layer
8298643	**-3.9704**	0.0472	**0.0315**	482818	0.001	32	2	2	64	256	256	2
7746487	-3.9663	0.0471	0.0320	551426	0.001	32	2	2	64	256	256	4
762072	-3.7976	**0.0462**	0.0330	482818	0.001	64	2	2	64	256	256	2
1047520	-3.8996	0.0467	0.0319	551426	0.001	64	2	2	64	256	256	4
3933066	-3.7129	0.0469	0.0331	482818	0.001	128	2	2	64	256	256	2
6093749	-3.5318	0.0469	0.0341	551426	0.001	128	2	2	64	256	256	4
7297106	-3.8821	0.0469	0.0341	551426	0.01	32	2	2	64	256	256	4
1491929	-3.8576	0.0472	0.0323	551426	0.01	64	2	2	64	256	256	4
2216878	-3.6472	0.0489	0.0355	551426	0.01	128	2	2	64	256	256	4

Table 2 Results of the initial hyperparameter search. Evaluation metrics are NLL, MSE, and MAE on the validation dataset, lower the best.

In **Table 2**, it is observed that the two models with the best performance have both used a lower learning rate of 0.001 along with two IMAB layers. As we have discussed earlier, a potential way to reduce high training fluctuation is to reduce the model complexity. Since the initial search models were quite complex in relation to the amount of available training data, with over 450,000 trainable parameters, and the two best-performing models have slightly lower complexity compared to the others, it was decided to conduct an additional search with models that have significantly lower complexity.

5.2.2 Search with Lower Complexity Models

To decrease the number of trainable parameters, the dimensions of the attention layer (IMAB and CAB) were set to 64, along with the layers of the decoder's feed-forward layer (Dec Dim). Additionally, the layers in IMAB were reduced to one. These modifications reduced the number of trainable parameters by approximately one order of magnitude, which is equivalent to roughly 65,000 parameters.

The search space for the lower complexity models (also referred to as “small models”) was as follows: Learning Rate {0.01, 0.001, 0.0001, 0.00001}, and batch size {32, 64, 128}.

Model	Val. Loss (NLL)	Val. Loss (MSE)	Val. Loss (MAE)	Model Param. (n)	Learn. Rate	Batch Size	Enc Heads	Dec Heads	IMAB Dim	CAB Dim	Dec Dim	N Layer
848517	106.0063	0.0470	0.0356	64514	1e-05	64	2	2	64	64	64	1
9728787	0.5709	0.0468	0.0358	64514	1e-05	64	2	2	64	64	64	1
8567900	-2.4196	0.0468	0.0337	64514	0.0001	128	2	2	64	64	64	1
9851768	**-3.9636**	0.0476	**0.0321**	64514	0.001	32	2	2	64	64	64	1
9892	-3.6831	0.0471	0.0327	64514	0.001	64	2	2	64	64	64	1
6007937	-3.2194	**0.0465**	0.0381	64514	0.001	128	2	2	64	64	64	1
910911	-3.8053	0.0466	0.0344	64514	0.01	128	2	2	64	64	64	1

Table 3 Results of hyperparameter search with reduced complexity models. Evaluation metrics are NLL, MSE, and MAE on the validation dataset, lower the best.

Although the highest performing model accomplished similar results as the more complex models, the observed variability in the NLL's was still extremely high. The experiments with learning rates below 0.001 were conducted in order to stabilize the training loss. However, they were stopped almost immediately by the early stopping condition due to their complete failure to improve performance on the validation set. The best performance in this group was achieved by a model with a learning rate of 0.001 and a relatively low batch size of 32.

5.2.3 Search with Enhanced Training Set

As the reduction of model parameters did not yield any performance gains, it was decided to approach the issue from the dataset side. To improve the ratio between data and model complexity the number of target observations, to be predicted by the model, needed to be increased. However, the model's architecture does not allow for predicting an observation in the middle of the ventilation sequence without considering future observations. Therefore, an alternative solution was found by dividing the ventilation sequences into smaller segments (referred to as "ventilation snippets"), each with subsequent PaO2 observation. This measure led to a sixfold increase in the number of target observations. This change only affected the training dataset. The validation and test datasets remained unchanged to enable comparison with results from the previous groups.

The following hyperparameter search was conducted as follows. Batch Size {32, 64, 128}, attention layer dimension {16, 32, 64, 256}, hidden units in feed forward layer {16, 32 256}, and IMAB layers {2, 4}.

Model	Val. Loss (NLL)	Val. Loss (MSE)	Val. Loss (MAE)	Model Param. (n)	Learn. Rate	Batch Size	Enc Heads	Dec Heads	IMAB Dim	CAB Dim	Dec Dim	N Layer
897223	-3.8978	**0.0470**	**0.0331**	482818	0.001	32	2	2	64	256	256	2
8069965	-3.8456	0.0495	0.0366	551426	0.001	32	2	2	64	256	256	4
774821	-3.5553	0.0516	0.0377	482818	0.001	64	2	2	64	256	256	2
3815465	**-3.9106**	0.0533	0.0383	551426	0.001	64	2	2	64	256	256	4
6879220	-3.8297	0.0498	0.0366	482818	0.001	128	2	2	64	256	256	2
1842991	9.6677	0.0475	0.0347	7042	0.001	256	2	2	16	16	16	2
9881996	-3.8741	0.0517	0.0375	43778	0.001	256	2	2	32	32	32	4
5502529	-3.8435	0.0492	0.0352	7042	0.001	512	2	2	16	16	16	2

Table 4 Results of hyperparameter search with reduced complexity models. Evaluation metrics are NLL, MSE, and MAE on the validation dataset, lower the best.

Unfortunately, the search did not provide the expected results. The best performing model within the “ventilation snippets” group was not able to outperform the previous results. Even in the experiment with drastic reduction of learnable parameters no increased performance could be observed.

5.3 Comparative Analysis

5.3.1 Efficiency Comparison with Baseline Models

To provide a better understanding of the models' performances, they were compared to a basic mean imputation algorithm. This algorithm follows a fixed imputation strategy and always predicts the target observation with the mean value of PaO2 of the dataset. Additionally, a forward imputation algorithm was implemented, that follows the strategy to always predict the last observed PaO2 value. If there is no previous observation in the sample, the algorithm will predict the mean PaO2 value of the dataset. Although the implementation of those two algorithms is straightforward, it cannot capture any patterns in the data.

The best models from each group was compared based on their performance on the test set, measured in terms of mean squared error (MSE) and mean absolute error (MAE).

Model		Test Loss (MSE)	Test Loss (MAE)
Baseline	Mean Imputation	3.5983	0.0767
Baseline	Forward Imputation	3.5983	0.0736
Large Model	8298643	**3.5948**	**0.0663**
Small Model	9851768	3.5960	0.0667
Snippets	3815465	3.6031	0.0736

Table 5 Results of comparison to baseline models. Evaluation measure is MSE and MAE on the test dataset, lower the best.

Overall, the model from the initial group (model 8298643) showed the best performance in terms of MSE and MAE. The best model coming from the "small model" group was also able to outperform the two baseline algorithms. The models trained on the "ventilation snippets" however, failed to surpass the baseline.

To summarize, none of the models showed a significant improvement compared to the baseline. Therefore, no further effort was made to compare them to more sophisticated models. The performance of the trained models was similar to that of the imputation algorithms, indicating that the models failed to learn any relationships or patterns in the data. Additionally, this raises concerns about the current setup, as it has serious limitations that hinder the models' ability to learn effectively.

5.3.2 Visual Inspection of Results

To visually inspect the results and understand the real-world implications of the performance, we generated predictions using the best-performing model. We denormalized the data, except for the time channel, and plotted it as shown in **Figure 15**. The last PaO2 value, which served as the ground truth, is represented by the blue circle, while the corresponding prediction made by the model is shown in red.

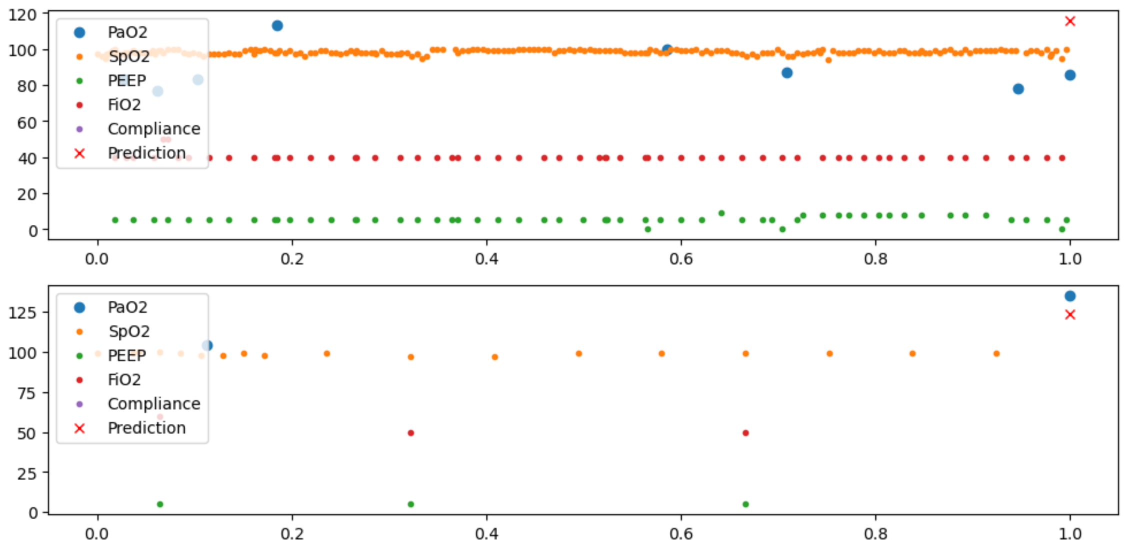


Figure 15 Visualization of prediction results. The model predicts (red cross) the value for the last PaO2 value (blue circles) based on the previous data of the ventilation sequence.

The accuracy of the predictions shows a high variance, with some examples exhibiting a difference of more than 30 mm Hg when compared to the ground truth. In a real-world scenario, this variance is significant and can mean the difference between a healthy patient and one suffering from respiratory failure. Thus, these variances can have a major influence on the clinical decisions that need to be made. In Appendix B, there are additional examples that demonstrate that most channels are quite stable, while PaO2 shows a high degree of variability with seemingly little correlation to other parameters. This leads to the question of whether the data contains the necessary information to predict arterial oxygen levels accurately.

5.4 Limitations and Challenges

Transformers have shown impressive capabilities in multiple natural language processing and computer vision tasks, which has generated a considerable interest in the time series community. However, adapting transformers for time series data presents significant challenges. The subchapters below will discuss the general challenges of adapting transformers to time series data, as well as the limitations regarding our specific use case.

5.4.1 Challenges in Adapting Transformers to Time Series Data

Computation Complexity: When the input sequences are short, the bottleneck of the transformer lies in the point-wise feed-forward network. However, as the input sequences grow longer, the sequence length gradually drives the complexity of the model. As previously discussed, the quadratic complexity of the original self-attention blocks is one of the major limitations for their application of transformer models to time series data. To overcome this limitation, most models use specialized self-attention blocks to reduce the complexity to a more linear relationship with the drawback of creating more complex model architectures.

Incapability of capturing temporal relationships: Another drawback of self-attention is its inability to capture the order of observations in time series data. While this is not a significant problem for NLP applications where rearranging words does not significantly affect the overall meaning of the sentence. However, it is a major issue for time series data, which have strong temporal relationships. Using various positional encoding techniques can preserve some ordering information, but applying self-attention on top of them still results in inevitable temporal information loss (Zeng et al., 2022). Therefore, to address this, many models use specialized architectures to preserve some of the temporal information.

Unclear state of the art: In their work, Zeng et al. introduced a set of very simple one-layer linear models that outperformed most established sophisticated transformer models. This calls into question the model evaluation for those models and the current state-of-the-art on various datasets (Zeng et al., 2022).

5.4.2 Limitations Regarding the Specific Use Case

The self-attention layer at the core of the transformer model assigns importance to different positions in the input sequence. Applying a masked attention layer would additionally prevent the model from attending to future information. Initially, this seemed like a reasonable approach to leverage the transformer's capability for parallelizing the calculations and considering the whole sequence simultaneously. However, in the utilized encoder-decoder architecture, the context information is fed into the encoder while the target queries serve as input for the decoder. This information then merges in the decoder's cross-attention block. This fact prevented any reasonable use of masked attention layers and therefore the effective use of the attention mechanism. To ensure that the model does not access future information, the input sequences are modified so that the target queries always represent the latest timepoint of the sample.

One of the most appealing advantages of Transformers is their capability to capture long-range dependencies and interactions, making them particularly suitable for time series modeling. However, as one of the most common reasons for measuring arterial oxygen is to observe the effect of the latest therapy adjustment, long-range dependencies might not be as relevant for an accurate prediction of the current PaO2 value. Moreover, a patient's oxygenation status can be considered as a steady state, which is frequently perturbed by disease progression and therapeutic measures. Unlike other time series data, it shows no global trend or seasonal patterns.

5.4.3 Future Directions for Improvement

When applying the proposed model to predict arterial oxygen in the described use case, several challenges were encountered. These challenges included high variability in training loss, limited convergence of the models, and marginal performance gains compared to the baseline models. To overcome these issues, directions for improvements have been identified.

Architecture: The Transformer architecture provides significant advantages when working with asynchronous multivariate time series. It can handle non-periodic observations, channels with few or no observations, and input sequences of any length. This makes it particularly useful for medical data applications. Furthermore, its parallelized processing of the whole sequence makes it computationally more efficient than other neural network approaches. To take advantage of these benefits, an encoder-only architecture is proposed, combined with a masked attention layer. This approach allows multiple observations to be predicted in one sample, ensuring that the model can only attend to the past information for predicting a specific data point. The key challenge here is to create an efficient embedding mechanism that can mask the value but provide information about channel and time.

Feature Exploration: The selected variables were primarily chosen from a application perspective of mechanical ventilation. However, there are many other factors that can affect a patient's oxygen saturation, such as disease progression, cardiac output, or blood pH. The variability in the training loss suggests that there may be issues with the selected variables, so it would be beneficial to explore the dataset more thoroughly to identify informative features. A subsequent principal component analysis could then be used to reduce the dimensionality and provide a set of variables that contribute to improved model performance.

6 Conclusion

6.1 Summary of Findings

The following research questions were addressed in this thesis: Can transformers handle asynchronous time series data efficiently? How well does a transformer model perform in predicting the partial pressure of oxygen using ventilation parameters?

The results show that transformers have an architecture that allows them to overcome the challenges of working with asynchronous time series data. They can handle variable sequence lengths and samples with non-periodic observations, and channels with little or no observations. The attention mechanism allows the model to process all observations in parallel and detect long-term relationships.

However, the model's performance on the described use case is limited. The initial experiments showed only marginal improvements to the baseline models, and the training loss had a high variability. In the subsequent experiments, neither the hyperparameter search, nor the reduced model complexity, nor the enhanced training set was able to bring significant improvement. These findings led to the conclusion that the used model, in combination with the provided dataset, has significant limitations.

The most important limitation was assumed to be the rather small data to model complexity ratio. Additionally, the attendance to all datapoint of the sequence and the unclear effect of the loss of temporal information in transformers could also have contributed to the reduced performance.

For future improvement, two opportunities were identified. One is to change the model towards an encoder-only architecture leveraging the transformers capabilities more effectively. The other is to thoroughly explore the dataset to include more informative features and to ensure that the required information is provided by the feature set.

From a personal point of view, transformers seem to be very powerful, especially with their impressive achievements in the field of generative AI and NLP. However, adapting them to other domains such as time series data requires much effort, often resulting in new building blocks and increasingly complex model architecture. With increased complexity comes an increased requirement for training data. Additionally, the architecture has a major influence on the model's explainability. Complex models with sophisticated building blocks and many layers make it hard to understand how the input features are transformed into predictions, making it a black box for the users. Especially in our use case where healthcare professionals rely on the model to help them making better treatment decisions, explainability and trust becomes a crucial factor for product success.

In summary, with regards to our specific use case, there is still not enough evidence to suggest that Transformers offer significant benefits considering their implementation costs. However, under the right circumstances, and especially given the appropriate expertise to effectively implement and maintain them, they can be effective.

As the journey through this research concludes, it leaves behind not only a comprehensive understanding of transformer models in the context of asynchronous time series but also ideas for future exploration, improvement, and collaboration. The pursuit of innovation is ongoing, and the quest for data driven solution that improve patients’ life will continue.

Reference List

Acute Respiratory Distress Syndrome: The Berlin Definition. (2012). *JAMA*, *307*(23). https://doi.org/10.1001/jama.2012.5669

Bahdanau, D., Cho, K., & Bengio, Y. (2016). *Neural Machine Translation by Jointly Learning to Align and Translate* (arXiv:1409.0473). arXiv. http://arxiv.org/abs/1409.0473

Basodi, S., Ji, C., Zhang, H., & Pan, Y. (2020). Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*, *3*(3), 196–207. https://doi.org/10.26599/BDMA.2020.9020004

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., & Jenssen, R. (2017). *An overview and comparative analysis of Recurrent Neural Networks for Short Term Load Forecasting*. https://doi.org/10.1007/978-3-319-70338-1

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2016). *Time series analysis: Forecasting and control* (Fifth edition). John Wiley & Sons, Inc.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). *End-to-End Object Detection with Transformers* (arXiv:2005.12872). arXiv. http://arxiv.org/abs/2005.12872

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation* (arXiv:1406.1078). arXiv. http://arxiv.org/abs/1406.1078

Clinical Decision Support Systems Market. (2023, September). https://www.marketsandmarkets.com/Market-Reports/clinical-decision-support-systems-market-18085342.html

DellaVolpe, J. D., Chakraborti, C., Cerreta, K., Romero, C. J., Firestein, C. E., Myers, L., & Nielsen, N. D. (2014). Effects of implementing a protocol for arterial blood gas use on ordering practices and diagnostic yield. *Healthcare*, *2*(2), 130–135. https://doi.org/10.1016/j.hjdsi.2013.09.006

Denny, J. C. (2012). Chapter 13: Mining Electronic Health Records in the Genomics Era. *PLoS Computational Biology*, *8*(12), e1002823. https://doi.org/10.1371/journal.pcbi.1002823

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (arXiv:1810.04805). arXiv. http://arxiv.org/abs/1810.04805

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale* (arXiv:2010.11929). arXiv. http://arxiv.org/abs/2010.11929

Fujishima, S. (2023). Guideline-based management of acute respiratory failure and acute respiratory distress syndrome. *Journal of Intensive Care*, *11*(1), 10. https://doi.org/10.1186/s40560-023-00658-3

Gers, F. A. (1999). Learning to forget: Continual prediction with LSTM. *9th International Conference on Artificial Neural Networks: ICANN '99*, *1999*, 850–855. https://doi.org/10.1049/cp:19991218

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., & Pang, R. (2020). *Conformer: Convolution-augmented Transformer for Speech Recognition* (arXiv:2005.08100). arXiv. http://arxiv.org/abs/2005.08100

Hewamalage, H., Bergmeir, C., & Bandara, K. (2019). *Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions*. https://doi.org/10.48550/ARXIV.1909.00590

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hofer, C. (2021). *Zeitreihen und Prognosen*.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice, 2nd edition, OTexts: Melbourne, Australia*. Forecasting: Principles and Practice. https://otexts.com/fpp2/

Johnson, A. E. W., Bulgarelli, L., Shen, L., Gayles, A., Shammout, A., Horng, S., Pollard, T. J., Hao, S., Moody, B., Gow, B., Lehman, L. H., Celi, L. A., & Mark, R. G. (2023). MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, *10*(1), 1. https://doi.org/10.1038/s41597-022-01899-x

Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing* (3rd ed. draft). https://web.stanford.edu/~jurafsky/slp3/

Keyt, H., & Peters, J. I. (2019). Acute Respiratory Failure. In *Cardiac Intensive Care* (pp. 308-317.e1). Elsevier. https://doi.org/10.1016/B978-0-323-52993-8.00030-8

Manor-Shulman, O., Beyene, J., Frndova, H., & Parshuram, C. S. (2008). Quantifying the volume of documented clinical information in critical illness. *Journal of Critical Care*, *23*(2), 245–250. https://doi.org/10.1016/j.jcrc.2007.06.003

Marti, J., Hall, P., Hamilton, P., Lamb, S., McCabe, C., Lall, R., Darbyshire, J., Young, D., & Hulme, C. (2016). One-year resource utilisation, costs and quality of life in patients with acute respiratory distress syndrome (ARDS): Secondary analysis of a randomised controlled trial. *Journal of Intensive Care*, *4*(1), 56. https://doi.org/10.1186/s40560-016-0178-8

Pham, T., Pesenti, A., Bellani, G., Rubenfeld, G., Fan, E., Bugedo, G., Lorente, J. A., Fernandes, A. D. V., Van Haren, F., Bruhn, A., Rios, F., Esteban, A., Gattinoni, L., Larsson, A., McAuley, D. F., Ranieri, M., Thompson, B. T., Wrigge, H., Brochard, L. J., & Laffey, J. G. (2021). Outcome of acute hypoxaemic respiratory failure: Insights from the LUNG SAFE Study. *European Respiratory Journal*, *57*(6), 2003317. https://doi.org/10.1183/13993003.03317-2020

Prince, S. J. D. (2023). *Understanding Deep Learning*. MIT Press. http://udlbook.com

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. *Preprint*. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

Sandur, S., & Stoller, J. K. (1999). PULMONARY COMPLICATIONS OF MECHANICAL VENTILATION. *Clinics in Chest Medicine*, *20*(2), 223–247. https://doi.org/10.1016/S0272-5231(05)70139-8

Sarwar, T., Seifollahi, S., Chan, J., Zhang, X., Aksakalli, V., Hudson, I., Verspoor, K., & Cavedon, L. (2023). The Secondary Use of Electronic Health Records for Data Mining: Data Characteristics and Challenges. *ACM Computing Surveys*, *55*(2), 1–40. https://doi.org/10.1145/3490234

Shi, Z., Zhou, X., Qiu, X., & Zhu, X. (2020). *Improving Image Captioning with Better Use of Captions* (arXiv:2006.11807). arXiv. http://arxiv.org/abs/2006.11807

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. https://doi.org/10.48550/ARXIV.1706.03762

Walsh, O. M., Davis, K., & Gatward, J. (2020). Reducing inappropriate arterial blood gas testing in a level III intensive care unit: A before-and-after observational study. *Critical Care and Resuscitation*, *22*(4), 370–377. https://doi.org/10.51893/2020.4.OA10

Yalavarthi, V. K., Burchert, J., & Schmidt-thieme, L. (2022). *Tripletformer for Probabilistic Interpolation of Asynchronous Time Series* (arXiv:2210.02091). arXiv. http://arxiv.org/abs/2210.02091

Zeng, A., Chen, M., Zhang, L., & Xu, Q. (2022). *Are Transformers Effective for Time Series Forecasting?* (arXiv:2205.13504). arXiv. http://arxiv.org/abs/2205.13504

Zhang, G., Eddy Patuwo, B., & Y. Hu, M. (1998). Forecasting with artificial neural networks: *International Journal of Forecasting*, *14*(1), 35–62. https://doi.org/10.1016/S0169-2070(97)00044-7

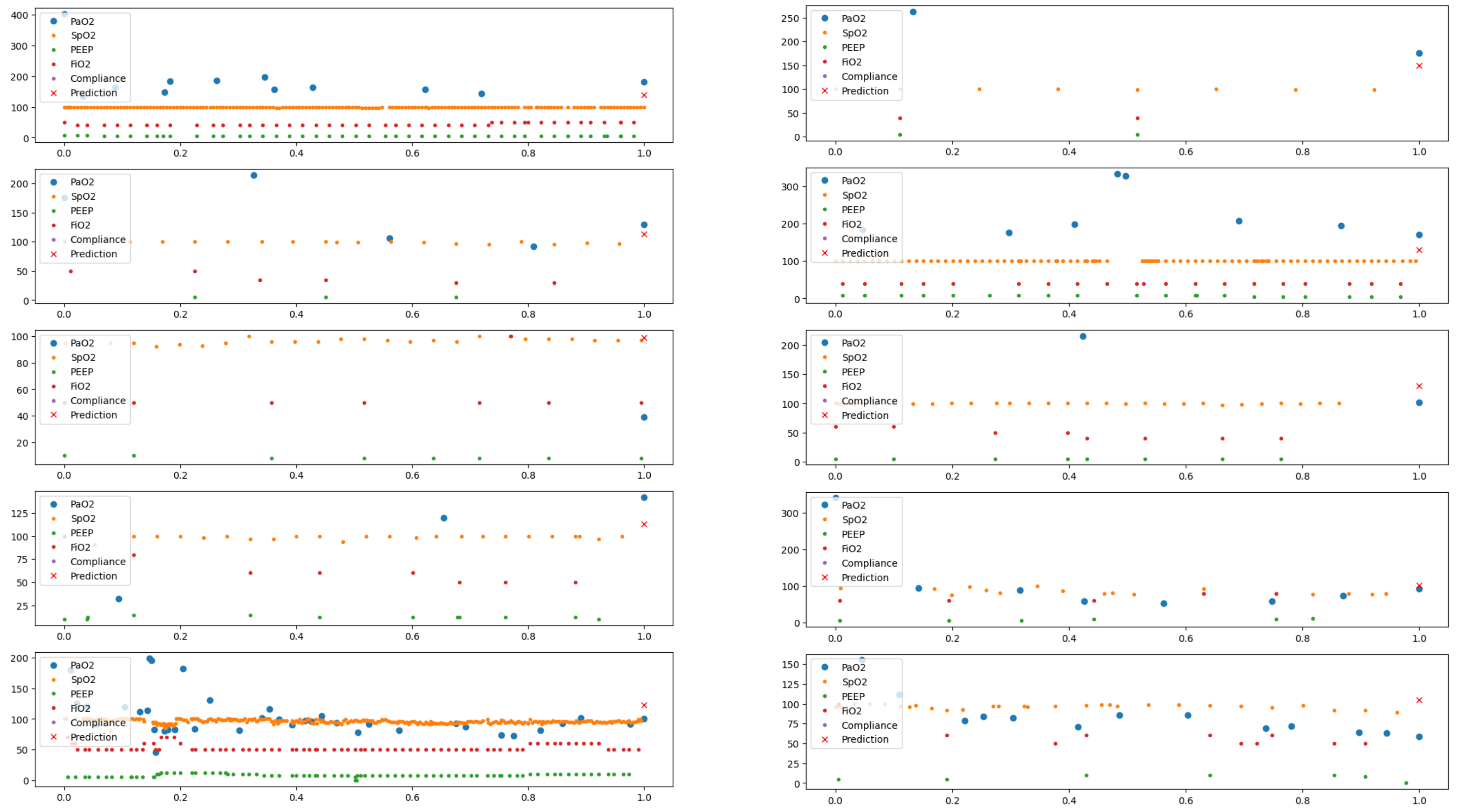
Zhang, X., Liu, Y., Yang, M., Zhang, T., Young, A. A., & Li, X. (2013). Comparative Study of Four Time Series Methods in Forecasting Typhoid Fever Incidence in China. *PLoS ONE*, *8*(5), e63116. https://doi.org/10.1371/journal.pone.0063116

.

Appendix A

	Model	Val. Loss (NLL)	Val. Loss (MSE)	Val. Loss (MAE)	Model Param. (n)	Learn. Rate	Batch Size	Enc Heads	Dec Heads	IMAB Dim	CAB Dim	Dec Dim	Layers
Large Models	8298643	-3.9704	0.0472	0.0315	482818	0.001	32	2	2	64	256	256	2
	7746487	-3.9663	0.0471	0.0320	551426	0.001	32	2	2	64	256	256	4
	762072	-3.7976	0.0462	0.0330	482818	0.001	64	2	2	64	256	256	2
	1047520	-3.8996	0.0467	0.0319	551426	0.001	64	2	2	64	256	256	4
	3933066	-3.7129	0.0469	0.0331	482818	0.001	128	2	2	64	256	256	2
	6093749	-3.5318	0.0469	0.0341	551426	0.001	128	2	2	64	256	256	4
	7297106	-3.8821	0.0469	0.0341	551426	0.01	32	2	2	64	256	256	4
	1491929	-3.8576	0.0472	0.0323	551426	0.01	64	2	2	64	256	256	4
	2216878	-3.6472	0.0489	0.0355	551426	0.01	128	2	2	64	256	256	4
Small Models	848517	106.0063	0.0470	0.0356	64514	1e-05	64	2	2	64	64	64	1
	9728787	0.5709	0.0468	0.0358	64514	1e-05	64	2	2	64	64	64	1
	8567900	-2.4196	0.0468	0.0337	64514	0.0001	128	2	2	64	64	64	1
	9851768	-3.9636	0.0476	0.0321	64514	0.001	32	2	2	64	64	64	1
	9892	-3.6831	0.0471	0.0327	64514	0.001	64	2	2	64	64	64	1
	6007937	-3.2194	0.0465	0.0381	64514	0.001	128	2	2	64	64	64	1
	910911	-3.8053	0.0466	0.0344	64514	0.01	128	2	2	64	64	64	1
Snippets	897223	-3.8978	0.0470	0.0331	482818	0.001	32	2	2	64	256	256	2
	8069965	-3.8456	0.0495	0.0366	551426	0.001	32	2	2	64	256	256	4
	774821	-3.5553	0.0516	0.0377	482818	0.001	64	2	2	64	256	256	2
	3815465	-3.9106	0.0533	0.0383	551426	0.001	64	2	2	64	256	256	4
	6879220	-3.8297	0.0498	0.0366	482818	0.001	128	2	2	64	256	256	2
	1842991	9.6677	0.0475	0.0347	7042	0.001	256	2	2	16	16	16	2
	9881996	-3.8741	0.0517	0.0375	43778	0.001	256	2	2	32	32	32	4
	5502529	-3.8435	0.0492	0.0352	7042	0.001	512	2	2	16	16	16	2

Appendix B


PaO2
SpO2
PEEP
FiO2
Compliance
Prediction