Neonatal Intensive Care Unit Text Summarization: Project Proposal

SYSC4907

Jesse Levine (101185127)
Gurshan Riarh (101182603)

Supervisor: Dr. James Green
Department: Systems & Computer Engineering
Date: October 28^{th} 2024

Contents

1 Introduction

This proposal outlines the problem statement for our Systems & Computer Engineering Capstone project, detailing the motivation behind the problem and our proposed solution. We will cover the implementation plan, including how we intend to test and validate our solution. Additionally, we will review the background context, previous work completed in this domain, and highlight the objectives we aim to achieve throughout this project.

1.1 Motivation

Neonatal Intensive Care Unit (NICU) patients require continuous health monitoring. In particular, NICU patients' vital signs are monitored and tracked around-the-clock and a summary of procedures performed throughout the day is recorded in the patient's medical records; this is known as charting [2].

Medical charting is time consuming for the medical staff and therefore has the potential to further encumber an already understaffed hospital [3] [4].

Charting is not the only source for potential time bottlenecks; a patient handover is the transfer of essential information and responsibility for the care of the patient from one health care provider to another. Ineffective and inefficient handovers not only have the potential to slow down the flow of the hospital's operations but also poses severe risks for the patient. The handover, by necessity, contains much critical information about the patient and the patients status throughout the previous health-care providers shift [1].

Another motivator in this project, are parents. Having a child in the NICU is extremely stressful and parents cannot always be in the unit with their child. Other than discussing directly with the health care provider, parents have limited methods for getting real-time updates on the status of their child [5] [6].

Charting, patient handovers, and parent updates would benefit from semi-automated charting and text-generated summarizing systems that could alleviate the workload of the

hospital staff, streamline patient care and lighten some stress for parents.

1.2 Past Work

Previous work done in the CUBIC lab focused on developing Machine Learning (ML) based models for analyzing patient video to estimate vital signs, recognize clinical and routine interventions and analyze environmental conditions. However, currently, there lacks an existing system that can summarize all the produced data for differing audiences: parents, clinical colleagues and the Electronic Health Record (EHR) [2].

Our project will build upon, and complement, work done in the CUBIC lab towards automated text summarization of medical charting data. Previous work, like Last year's capstone project, led by Ms Toyin Adams, demonstrated a proof-of-concept system capable of generating text summaries from JSON datafiles. However, these JSON files contained NICU vital data at a singular point in time and therefore their pipeline did not consider a temporal approach: summarizing the status of the NICU patient over the course of a few hours, a nurses entire shift, or even a day.

It should also be noted that last year's rendition of the text-summarizer produced hallucinations and occasionally editorialized the data, providing opinionated analysis of patient care quality despite this content not being requested. It is imperative that a future summarizer system does not make-up its own data nor begin making diagnoses or comments on the care of the patient.

The previous group also worked on a patient simulator to support the validation of the semi-automatic text-summarizer. The patient simulator generates JSON data structures that include fields representing vital signs and the presence of ongoing clinical or routine care interventions. The simulator also replicates in-the-moment health monitoring of neonates by "capturing" vital signs including, but not limited to, body temperature, respiration rate, heart rate, and $Sp0_2$ data. The simulator's output is compiled into a JSON file, containing all the simulated data for use in the text-summarizer.

The patient simulator developed last year was meant to provide an approximation to real-life scenarios however, much like with the text-summarizer, the patient simulator developed last year was incapable of producing a temporal or continuous data stream.

1.2.1 Large Language Models

Recent advances in large language models (LLMs) have shown significant promise in generating natural language summaries across a variety of domains, including healthcare. LLMs like GPT-4o have demonstrated an ability to generate human-like text based on input data, making them suitable for complex tasks like summarizing patient data. These advancements create an opportunity to leverage state-of-the-art (SOTA) LLMs to assist with medical charting, patient handovers, and parent updates, offering a more cohesive and streamlined approach to handling NICU patient information [15] [16].

2 Objectives

The primary goal of the project is to build upon and enhance current work done in the domain of non-contact patient care in the NICU. As mentioned above, past work has shown proof-of-concept systems for semi-automated text-summaries of patient data and patient simulator systems for the validation of the semi-automated summarizers.

As such, our project is composed of two main goals: expanding the scope and functionality of the patient simulator and the text-summarizer. Within these goals lie sub-goals discussed below.

2.1 Automated Text Summarizer

Clinicians spend nearly half their day on filling out EHRs and desk work [3]. Current medical electronic record systems are inefficient and are primarily utilized for administrative and legal purposes, and not clinical efficiency [4].

As such, health-care providers waste a tremendous amount of time filling in paper work and updating other health-care providers on the status of the patient at handover, rather than actually caring for the patient.

There exists a need to create a system, given real-time patient data, capable of semi-automatically generating a health record, automatically generating a summary of a patient's status over a period of time and also developing a chat-bot to deliver real-time updates to parents when prompted.

2.1.1 Semi-Automated Charting & Patient Handover Summarizer

Over the course of a nurse's 8 -12 hour shift, a patient's vital signs and any treatment done on the patient is recorded in the patients chart. This is done, in theory, to keep a streamlined medical record of the patient but also to get the following nurse up to speed at patient handover [3].

As such, there exist two main sub-goals in continuing the work on the automated text-summarizer.

Given the patient data, over the length of the nurse's shift, the fist goal is to create a system capable of automatically filling in part of an EHR with the data captured. Past work has looked at single time points, but realistic medical scenarios are temporal; they appear over time and can't be seen in a single moment in time. Manually filling in an EHR can be prone to error, especially in the case when clinicians suffer from burn out as they are over-worked with administrative tasks. Automatically filling in an EHR, directly from extracted patient data, would tremendously alleviate the adminitrative work seen by the clinicians and reduce their burn-out.

Similarly, at patient handovers, the exiting clinician details to the incumbent the status of the patient(s), any treatments done and if there are any markers to look out for; not to mention the incumbent clinician also needing to look over the EHR of the patient(s) prior to actually caring for them.

The second goal then, in this domain, is to create an automatic text-summarizer based directly off the patient(s)'s data. A system capable of automatically summarizing the status of a patient over an 8, 10, or 12 hour shift would help streamline the patient handover but also offer the incumbent clinician a quick summary in which they could refer back to over the course of their own shift.

The automatic text-summarizer should briefly explain how the patient is doing purely based on the vital signs monitored over the chosen time period, and also note when certain treatments were administered. The summarizer, should in no way make a diagnosis, or comment on the quality of care.

The only interpretations the summarizer should perform is to flag any irregularities to the health care provider. For example, if the patient's heart rate were to begin beating in an irregular pattern, the system should be able to flag this and include it in the summary.

2.1.2 Automated Real-Time Parent Update Summarizer

Having a child in intensive care is among the most stressful and distressing periods for new parents. Currently, there is a lacking area in offering parents the capability of quick, concise, real-time updates on their children. Exacerbating the distress is the sudden vast amount of information they receive coupled with the likelihood of missing weeks to months on end of work [6].

Parents who must return to work, while their child is in intensive care, lack the technology to receive real-time updates on the status of their child.

Part of our objective, is to create a real-time automated text summarizer capable of updating the parent when prompted. The summarizer should emulate a chatbot, where the parent can easily prompt it and receive an update. The tonality of the chatbot should approximate how a nurse would update a parent, and much like the handover simulator, it should in no way make any diagnoses or comments about the level of care. The chatbot summarizer should simply offer a concise update on the status of the baby and any interventions

done up to that point in time.

To ensure the quality and accuracy of the updates provided by the chatbot, a human-in-the-loop approach may be necessary for oversight and quality assurance before deploying such a system. While this project aims to explore the potential of SOTA LLMs for generating these updates, it is likely that human supervision would be required to validate the system's outputs, ensuring that the information provided to parents is reliable and appropriate.

2.2 Testing SOTA LLMs

As part of the validation, we will examine various state-of-the-art LLMs and explore methods to fine-tune their behaviour for our specific use cases. This includes evaluating different LLM models and determining the optimal way to adapt them for NICU-specific tasks, such as medical charting, patient handovers, and parent communication.

2.3 Patient Simulator

The patient simulator aims to generate comprehensive NICU nurse care data over a variable shift length (e.g., 8-12 hours) to validate the text summarizer. The simulator includes parameters such as vital signs (heart rate, respiratory rate, blood oxygen levels, etc.), and interventions (e.g., diaper changes, feedings, x-ray imaging, and repositioning), as well as natural neonate movements. The data generated will be timestamped and collected continuously over the shift.

LLMs require validation to be dependable and effective; without validation, they can provide inaccurate or an incomplete output, which can lead to misinterpretations or errors when dealing with patient care. The summarizers will need to output reliable and true text regarding the patient as false reporting of vital sign fluctuations or other variables being monitored can lead to the wrong action plan or medical decisions. To avoid this, the simulator will be used as a way for the team to validate the output generated by the text summarizer.

The goal of the simulator is to emulate real-world NICU data to feed into the text summarizer. To create an accurate emulation, the simulator must include all the vital signs, interventions, and other monitoring criteria that are recorded in the NICU. As such the simulator will contain vital signs such as heart rate (bpm), Respiratory rate (breaths per minute), blood pressure (mmHg - systolic and Diastolic), body temperature (°C), Oxygen saturation (SpO_2) and skin tone (normal, blueish, yellow-jaundice, pink, red).

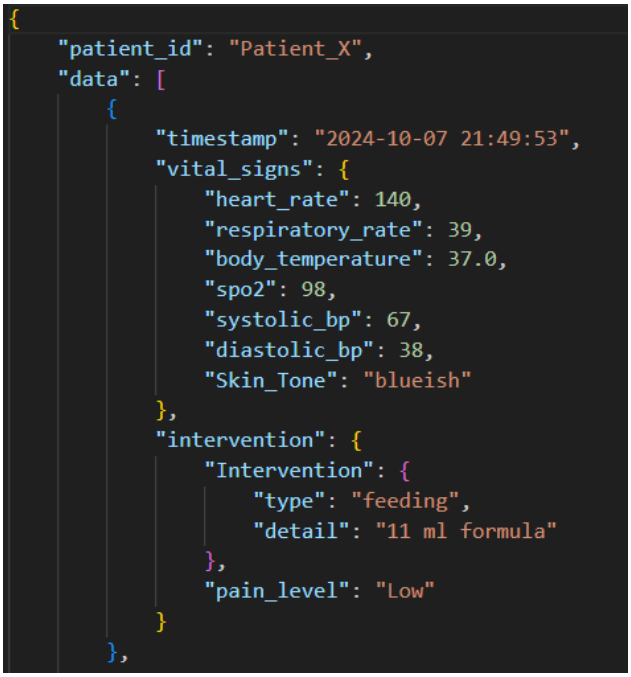

Figure 1: Example Output of JSON data from the simulator

Generating visuals in regards to the vital signs can greatly aid in quick understanding of trends and making sure that the vital sign data the simulator is outputting is accurate without having to scroll through thousands of lines in the generated data.

Another parameter to monitor is interventions. These parameters will be split into two categories, instantaneous and ongoing interventions. The interventions include but are not limited to, sensory control (light, noise and other environmental factors), baby positioning (supine, left lateral, right lateral), feeding (amount of formula (ml)), diaper change, and administration of medication and oxygen. The pain levels during these interventions will

also be added to allow for understanding of patient needs and preferences.

One of the main priorities of the simulator will be to implement continuous time data. This entails providing outputs for all aforementioned parameters over the course of an entire NICU shift. The improved version of the simulator will not only generate data for any amount of specified time but will allow for different user selected data generation intervals to more accurately represent the NICU environment.

The JSON output from the simulator will differ depending on what option is selected, where the options will vary from regular and irregular NICU observations. This means that one version will have regular-varying variables, and the other version will be simulating irregular patterns within the NICU allowing the team to validate and fine-tune the text-summarizer.

By creating the patient simulator, it will allow for replication of real-world NICU care; Generating vital signs and nurse interventions over variable shift lengths, which will be utilized in validating the LLM's ability to generate accurate responses in correspondence with the simulated data. By implementing all of these innovations, the team will build a fully functional NICU patient simulator to be used in tandem with the text summarizer.

3 Background

3.1 Continuous Health Monitoring

Vital sign monitoring has a long and storied history; in the 17th century Galileo developed an experiment on how to measure body temperature using ethanol and a mercury thermometer. Sir John Floyer wrote a paper in 1707 on measuring heart rate using a pendulum. In 1852, Taube included respiratory rate as a variable to track while examining a fever and in 1896, the blood pressure cuff was invented [7]. With greater technological advancements, tracking temperature, pulse rate, respiratory rates and blood pressure became common practice.

However it wasn't until the 1950s, when electronic vital sign monitors arose, could vital

signs be continuously tracked. Prior to electronic monitors, vital signs were periodically and manually tracked, and underlying health conditions weren't discovered [7] [8].

The development of continuous health monitoring has transformed healthcare practice, allowing for real-time tracking of vital signs, which are key indicators of a patient's overall health. The significance of these vital signs is paramount in the early detection of medical conditions, as continuous monitoring enables healthcare providers to promptly identify abnormalities. For instance, a sudden drop in blood pressure or an irregular heart rate may indicate a cardiac event, allowing for immediate intervention and potentially saving lives [9].

Continuous monitoring also plays a crucial role in evaluating the efficacy of treatments. By observing changes in vital signs, healthcare professionals can assess whether a particular medication, therapy or intervention is achieving the desired results, leading to informed decisions about the patients's care. Over time, this data allows for the identification of trends, such as deterioration or improvement in a patient's condition. Early detection of such trends can prevent complications and reduce the need for more aggressive interventions [9].

The emergence of electronic vital sign monitors in the 1950s marked a pivotal moment, as it enabled vital signs to be continuously tracked, replacing the periodic manual measurements that had previously been common practice. With electronic monitors, vital signs can be recorded as frequently as needed, providing a clearer picture of a patient's overall health. Additionally, remote monitoring systems further enhanced this capability by allowing healthcare providers to track multiple patients simultaneously and to receive early warnings when vital signs deviate from normal thresholds. These advancements have significantly improved the efficiency of patient monitoring, enabling timely and precise interventions, even from a distance [7].

The continuous collection of vital sign data through remote systems also offers long-term insights, aiding research and quality improvement efforts while contributing to evidence-based care.

3.1.1 Vital Sign Fluctuations

The vital signs in neonates give important information in regards to overall health. Neonates usually have abnormal vital signs when observing their basic pathophysiology. Continuous monitoring of heart rate, respiratory rate, oxygen saturation (SpO2), temperature, and blood pressure may outline slight deviation that may indicate sepsis, Necrotizing enterocolitis (NEC), or respiratory deterioration. For instance, Heart Rate Variability (HRV), being an index of autonomic nervous activity, has been found to be lower among neonates who are either stressed or sick. Low HRV and transient episodes of bradycardia or tachycardia have been warning signals regarding infection in its early course, with the latter often related to the anti-inflammatory response of the vagus during a septic condition. Such early warnings are invaluable, considering the lessened mortality rates possible with timely intervention [22] [23].

On the other hand, respiratory rate variability such as periods of tachypnea and apnea are another indicator of neonatal health. Apnea, common in preterm infants, is due to poor respiratory control and when combined with infection, the frequency and intensity of incidents increase. These respiratory irregularities in combination create a complete picture about the neonate's health status when decreased oxygen saturation levels are involved. Oxygen saturation monitoring allows for early detection of hypoxemia or hyperoxemia to prevent adverse outcomes such as brain injury or Retinopathy of Prematurity (ROP) [22]. By understanding these oscillations and interrelationships, the clinicians may predict and prevent critical diseases before severe symptoms manifest [23].

Continuous monitoring of these vital signs brings a lot of insight, but without advanced analytics, important patterns might go unrecognized. Such analytics of continuous vital sign data have demonstrated the power of improving disease predictions such as those related to sepsis or NEC. Heart rate characteristics, for instance, are a metric that includes measures of HRV and transient deceleration's that early warning systems have employed in providing real-time alerts of health risks to a clinician. These improve outcomes through early detection

and intervention [22] [23].

The virtual environment simulations of fluctuating vital signs are important both in clinical training and in the development of predictive algorithms. A digital patient simulator modeling variability and interactions between vital signs can be used to exercise early detection and intervention strategies under controlled conditions. A typical example of a sophisticated simulator is the *Super TORY S2220 Advanced Newborn Patient Simulator* [24], which provides comprehensive tools for realistic simulation of neonatal scenarios. It will accurately capture a wide range of vital sign fluctuations-from cardiac monitoring with a richly detailed ECG library to the most advanced airway and respiratory management, including spontaneous breathing and end-tidal CO_2 measurement. These capabilities are of great value in refining the algorithms for application in real-time monitoring systems. These simulators test the performance of health abnormality early warning systems in continuous, real-world neonatal vital sign datasets to improve the accuracy of the detection model and reduce the workload of NICU staff, in turn, enhancing neonatal care. Simulation is an effective and safe way to improve neonatal outcomes by translating theory into practice [23] [24].

3.2 Transformer Architecture & Natural Language Processing

The Transformer architecture, introduced in the 2017 paper, *Attention is All You Need* [11] by Google, represents a significant breakthrough in sequence-to-sequence modelling, which had long been dominated by Hinton's work on Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) [13]. Unlike these earlier models, which rely heavily on sequential processing, Transformers use an attention mechanism to focus on different parts of the input data simultaneously, greatly enhancing the efficiency and scalability of the model [10] [12].

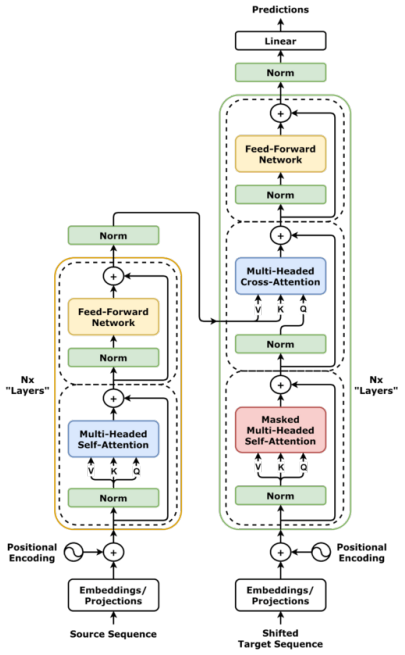

Figure 2: Transformer Model Architecture [11]

At the core of Transformer architecture is the concept of self-attention, which allows the model to weigh the relevance of different input tokens with respect to one another, regardless of their position in the sequence. This contrasts with RNNs, where information is passed step-by-step and can degrade over time. Transformers, on the other hand, can process all input tokens in parallel, thus improving both speed and performance, particularly for tasks involving long-range dependencies. A Transformer consists of two main components: an encoder and a decoder, both of which contain multiple layers of self-attention mechanisms and feed-forward neural networks. The encoder processes the input data and generates a series of context-aware embeddings, while the decoder uses these embeddings to produce an output sequence, such as a translated sentence or a text summary [10] [12].

Natural Language Processing (NLP) involves a wide range of tasks that require understanding and generating human language, from translation and summarization to sentiment analysis and question answering. Traditional models, such as RNNs, often struggled with capturing long-term dependencies between words, especially in lengthy sentences or docu-

ments. The advent of transformers has revolutionized NLP by enabling more efficient and effective handling of these dependencies. Their attention-based mechanism allows models to focus on key aspects of the input text, regardless of the distance between words, providing more accurate and contextually aware predictions [10] [15].

Transformers in NLP have become the foundation for SOTA LLMs, including BERT (Google's language model), ChatGPT, and Llama (Meta's language model), which are widely used for tasks like machine translation, summarizing, and even medical text analysis. In healthcare, for instance, transformer-based LLMs can help automate the generation of medical reports, extract critical information from patient records, or even analyze real-time patient data for abnormalities. By leveraging the power of self-attention and parallel processing, transformer-based LLMs significantly enhance the ability to process and generate complex text data, making them invaluable in a field like healthcare where timely, accurate information can greatly impact patient outcomes [15].

3.3 Patient Simulators

Historically, simulation in medical training started with simplistic physical models of anatomy even before the innovation of more advanced technologies such as computers. Medical simulation in the early days had been developed to merge the gap between theoretical knowledge and practical skills allowing the increase in confidence and performance in medical staff and procedures. Over the centuries, this concept evolved, particularly during the 1900s when medical training transitioned from simple apprenticeships to a more structured approach that demanded objective competence measures in knowledge, skills, and behaviour. The development and refinement of simulation based tools have been brought to the forefront of medical care based on the "Never the first time on the patient" principle. This principle entails the importance of rehearsed, risk-free application towards medical care procedures [30].

A major application of medical simulation is in neonatal care. The historical use of mannequins and standardized patients has further evolved into more complex simulations

that include technologies that can emulate real-life neonatal care routines. In this day and age it is important in scenarios such as neonatal resuscitation, where simulation training is regulated by guidelines and has been shown to effectively improve the procedure preparedness and response of health care professionals [31]. In regards to the technology available in present day, patient simulation has branched out from medical procedures to machine learning applications. With the emergence of Artificial Intelligence (AI), applications of medical simulation have become geared towards patient care efficiency [32].

Using LLMs in tandem with patient simulation reflects a shift towards digitized healthcare practices. LLMs can process and analyze extensive datasets resulting from simulations. Innovating these tools and using them in both training and real-time patient care routines can allow for more efficient care and execution of procedures [33] By feeding simulated neonatal data into LLMs, health care professionals can gain access to summarized insights that are important for quick decision making. Using this approach in advancing the usage of medical simulation allows for real-time clinical decision support, and in turn allowing for patients to gain the proper care required in a timely fashion [31] [32].

3.4 Generative AI & NLP In Healthcare

Recent advancements in AI, specifically generative AI and NLP, have significantly influenced healthcare, transforming both administrative and clinical practices. Generative AI models are AI systems, based on Transfomer architecture, capable of creating new content such as text, images, or music. LLMs are a subset of generative AI, trained on vast datasets of human language. LLMs, learn the structure and nuances of language, allowing them to generate coherent and contextually appropriate responses, making them suitable for complex applications like summarization, answering questions, and conversational tasks [15] [14].

One of the most impactful uses of generative AI in healthcare is the automation of clinical documentation. LLMs such as Microsoft's Copilot, which are integrated with EHRs and productivity software, can significantly reduce the administrative burden on healthcare

workers by automating repetitive tasks like form-filling and summarization. Mayo Clinic's early deployment of Copilot demonstrated that generative AI could effectively support clinical staff by freeing up time for more patient-focused care, thereby improving efficiency of overall healthcare delivery [16].

Additionally, NLP and generative AI are being used in clinical decision support, patient triage, and medical literature summarization. LLMs, such as GPT-4 and Med-PaLM, have demonstrated their capabilities by passing medical licensing exams and answering complex medical questions with near-human accuracy [15]. Med-PaLM, for instance, has been specifically trained on medical datasets to provide domain-specific insights, enhancing diagnostic support and enabling healthcare providers to make informed decisions [15].

Another noteworthy contribution to the field is Clinical Camel, an open-source medical LLM developed from LLaMA-2 (an LLM by Meta), explicitly tailored for clinical research [17]. Clinical Camel has been fine-tuned using Dialogue-Based Knowledge Encoding (DBKE), a novel method that converts dense clinical texts into synthetic conversations, enhancing the model's ability to handle medical dialogue tasks. Clinical Camel outperformed GPT-3.5 in multiple medical benchmarks, such as PubMedQA and the USMLE Sample Exam, showcasing the potential of open models to achieve expert-level performance in the healthcare domain [14] [17].

Generative AI also plays a role in augmenting patient interaction through chatbots and virtual assistants. These tools provide real-time health updates, answer patient queries, and enhance chronic disease management, contributing to a more patient-centric healthcare system [15]. However, challenges such as misinformation, hallucinations, and biases in training data still need to be addressed to ensure these models are safe and effective for clinical use. The use of generative AI at the Mayo Clinic and in the development of Clinical Camel, for instance, highlights the importance of privacy, ethics, and safety as critical considerations when deploying generative AI in healthcare [16] [17].

3.5 LangChain

LangChain is an open-source framework designed to simplify the process of building applications that make use of LLMs. It provides a set of tools and modular components that developers can use to create pipelines, combining LLMs with external data sources and other processing techniques. LangChain's goal is to streamline the development of LLM-driven solutions by managing the different elements of an AI application, such as document processing, model prompts, retrieval of relevant information, and integration of chat-like functionalities [18].

LangChain is particularly useful in applications that require managing and utilizing large volumes of text data effectively. For healthcare applications, LangChain allows for the integration of medical datasets with LLMs, making it possible to generate natural language summaries, answer patient-related queries, and facilitate communication between healthcare professionals and patients. In the context of NICU care, LangChain can streamline the creation of chatbot interactions and enable effective handover summaries, enhancing the quality and efficiency of healthcare services [19].

LangChain also supports retrieval-augmented generation (RAG), a process in which the language model is combined with an external knowledge base or dataset to provide more accurate, contextually relevant responses. This is especially valuable in healthcare, where precise information is critical. By allowing LLMs to access and retrieve relevant data points from external sources, LangChain ensures that the generated text is based on up-to-date and accurate information [18] [19].

3.6 Validation of LLMs

LLMs, such as those in the GPT series and BERT, hold promise for a wide array of applications, including healthcare, where these models are helpful for tasks ranging from patient management to information dissemination. The rate at which such technologies are being integrated into sensitive fields like medicine has given rise to significant concerns about their

reliability and the accuracy of their outputs. In view of the potential consequences of misinformation, there is an urgent need for mechanisms or modalities that ensure verification and validation regarding the factual accuracy and clinical relevance of information generated by LLMs [34].

The proliferation of LLMs in health is rivalled only by their capabilities to analyze massive datasets, which may even grow beyond human capabilities in terms of speed and efficiency. However, LLMs may perform with compromised accuracy in data provided, since quality variance in training datasets may or may not pass peer reviews or fact checks. This misinformation can have serious consequences, including self-diagnosis with something wrong or inappropriate medical intervention, which calls for a structured framework of validation before their deployment in practice [35].

In the aspect of LLM validation, MEDIC offers an external validation project developed for the performance evaluation of LLMs using a panoramic evaluation methodology. This project was not founded by our team and nor will it be implemented but it refers to comprehension regarding how well LLMs behave in several medical contexts. A lot of relevance has been accorded to the technological aspects and their applications in realistic situations. This framework further investigates the LLMs for their dimensions of medical reasoning, ethics and bias, and clinical safety to make certain that not only are these models technically sound, but they also align with medical standards and ethical considerations. The reason for such rigorous validation processes is to mitigate risks associated with automated decision-making and therefore to protect patient outcomes from the intrinsic limitations of LLMs, for instance, the output out information that sounds very convincing but is actually factually incorrect [34] [35].

Therefore, though LLMs can advance the field of healthcare, their deployment needs to be carefully managed. The establishment of protocols for its validation and constant monitoring of outputs emanating from them becomes a prerequisite to make sure the tools confer benefits without compromising safety and accuracy. These will go a long way in

leveraging LLMs while upholding trust and integrity in medical practices.

4 Implementation

4.1 Patient Simulator

To implement a NICU patient simulator, the first step will be to design a user interface (UI). This will allow healthcare professionals to easily input patient identification as required. The UI will be designed using Python's Tkinter library, a simple tool in creating graphical user interfaces. This will bring up the user interface to ask the user for the "Patient ID," which will uniquely identify this patient both in this system and in the exported data. It will also enable the user to select from different options types (regular/irregular vital sign patterns) for the generation of vital signs which will be discussed in a further section.

Following the user's input provided, the core of the implementation is the Simulator program. This is running for a pre-defined shift length. The simulator will deliver vital sign values for heart rate, respiratory rate, body temperature, SpO2, and blood pressure (both systolic and diastolic) at periodic intervals, usually every 30 minutes (can be changed). The simulator will also include clinical interventions-manipulation of lighting, feeding, medication administration, and repositioning of the patient-to simulate natural activities within a NICU environment.

Once the simulation is completed, this information is tabulated in a structured JSON format, where entries represent a timestamp, the vital signs generated at this point in time, and any interventions taking place. The JSON is hierarchical in nature, starting with a tree root being the patient ID, further containing a list of data points corresponding to the respective timestamps. This will ensure that this format is current and easily consumable by the text summarizer. The output filename will include the patient ID, making sure that the tracking and identification of patient data is done appropriately. This structured JSON output would then be fed into an text summarizer, which processes the information into

natural language summaries of various aspects that the medical staff can go through.

In addition to the generation of JSON data, the simulator also needs to be complemented with a data visualization component: one that generates plots of the vital signs over the x-hour shift (where x is a defined shift length). This module will use the *matplotlib* library to create time-series plots for each vital sign to demonstrate, for instance, a heart rate that is trending up/down, a temperature spiking/cooling, or oxygen saturation increasing/decreasing. These plots give a graphical presentation of the data, from which health professionals will find it easier to note deviations or patterns of concern. Each plot is saved as an image file, allowing data to be shared or included in patient reports with great ease. This feature adds an extra dimension to the simulator through the numerical and visual outputs needed for complete understanding of the patient's condition.

4.1.1 Discrete Time Variables in Patient Simulation

Apart from the continuous output of vital signs, it is important to consider discrete-time interventions that do not occur at every output interval such as changing light, feeding, or medicating a patient. These kinds of interventions must be modelled independently of the continuous physiological data since it is an external action of medical staff at a certain time. Separating the interventions from the continuous output allows the simulation to better emulate real clinical workflows, where interventions are infrequent and their effects can be delayed on vital signs. While essential, these interventions should not affect the output of constant monitoring in vital signs, though the impact of such could be modelled to appear after the event of the intervention is logged; for example, improved oxygen saturation following the administration of oxygen.

4.2 Vital Sign Modelling

For the analysis and modelling of vital signs, We consider a deep-learning approach to model neonate vital sign data effectively. Deep learning can be described as machine learning

concerned with algorithms inspired by the structure and function of the brain, called artificial neural networks. The deep learning model is characterized by powerful methods of processing complex and high-dimensional data, such as physiological signals regarding heart rate and respiration rate. This neural network structure, especially RNNs, have proved very effective in modelling temporal data. In temporal data analysis, tasks that normally involve time-series prediction are perfectly handled by the structure [27].

Our chosen deep learning approach is the Long Short-Term Memory Recurrent Neural Network (LSTM-RNNs). LSTM-RNNs are a sort of RNN especially designed to capture both short-term and long-term dependencies in sequential data. Unlike other RNNs, which are troubled by the vanishing gradient problem [29] in long sequences, LSTMs possess special memory units and gates that enable the network to capture information for long durations. Therefore, LSTM would be ideal in modeling vital signs, whereby changes in vital signs depend on both the recent and past physiological states. By incorporating LSTM-RNNs, we can predict not only the next few minutes of the vital signs but also potentially simulate the patterns over an entire nursing shift; thus, it is highly applicable to our neonate patient simulator [27] [28].

4.2.1 PhysioNet Database

PhysioNet is an online repository providing access to large collections of medical and physiological signals. Some examples include a number of research studies involving human health, whereby one can develop and also validate computational models [36] [37].

We will use the PhysioNet data to train a model that can simulate neonate vital signs. These will form the basis of our initial model, which will predict how those vital signs evolve over time.

4.2.2 Deep Learning Approach

First, we modeled the heart rate and the respiration rate with a random forest regressor. The Random Forest Regressor was trained on 1-hour subsets of data from the *Preterm Infant Cardio-Respiratory Signal Dataset* from PhysioNet. For feature preparation in this model, we used a sliding window approach and created lag features, with the input at each time being composed of the current and past values of heart rate and respiration over the previous five time steps. The lag features were, therefore, the input features, while the target was the heart rate and respiration rate for the next time step. We then split the dataset into training, validation, and test sets in the ratio 0.5:0.25:0.25, respectively.

While the Random Forest model was reasonably successful in yielding preliminary insights, it had some limitations. Random forest does not intrinsically model any temporal dependencies. In this case, for example, it uses only the lagged values of certain coded features to estimate trends or patterns in time series data. However, it is not capable of understanding or learning from the underlying natural fluctuations or changes that occur in physiological data over time. Essentially, it lacks the ability to grasp the deeper, dynamic processes affecting the data it analyzes.

The LSTM-RNN model overcomes the shortfalls of the Random Forest approach by providing a systematic approach for learning the dependencies across various time scales from the data. LSTMs are designed to retain relevant information across time steps, hence offering a more granular understanding of temporal sequences [27]. This becomes an important aspect in physiological modelling, where vital signs are modulated by a variety of physiological processes, some operating on short and others on long timescales.

4.2.3 Vital Sign Data Modelling Version 1: Heart Rate & Respiration Rate V_1

The implementation of the code in analyzing the datasets will involve several steps in building a Random Forest model to predict the heart rate and respiration rate of neonate babies.

First, pre-processing of the dataset involves the creation of lag features in order to capture temporal dependencies; past values of heart rate and respiration rate will be used as features to predict future values. The dataset will be split into training, validation, and test sets in the ratio of 0.5, 0.25, and 0.25, respectively. This work will train a Random Forest Regressor on those features and evaluate its performance by metrics like Mean Squared Error (MSE) and Mean Absolute Error (MAE). Initial input for the simulation will include the first row of test data, followed by a specific initial vector in order to simulate heart rate and respiration rate over an 8-hour period in 5-minute increments, i.e., 96 steps. That would involve an iterative prediction, updating the lag features to use as inputs for the next iteration. The simulated data derived will be written to a Comma Separated Values (CSV) file for further analysis.

Following the RF approach, an RNN using the Long Short Term Memory (LSTM) construction is presented that captures sequential dependencies that are inherent in this dataset in a much better fashion. A sliding window approach to prepare the dataset, where each window is a 5-minute segment of data, hence allowing the LSTM to learn temporal patterns effectively. It will again be divided into training, validation, and test sets with the same ratio: 0.5, 0.25, and 0.25. Define an LSTM-based architecture in light of the sequential nature of the time series data: comprising hidden layers, output layers, and dropout layers against overfitting. The model will be trained on the training dataset but with a touch of monitoring the validation loss in order to find overfitting and early stopping to stop the training when the validation loss is not improving. Once the training is complete, the group will go ahead and use the LSTM model in order to forecast simulated heart rate and respiration rate for an hour in 5-minute intervals. It first starts the simulation with a selected initial input window and iteratively continues the simulation by using the model's prediction of one step for subsequent steps.

The planned implementation will contrast these two approaches: a Random Forest model and an LSTM RNN. The Random Forest will serve as a more basic baseline, while it is

expected that the LSTM RNN will give a more detailed and temporally precise simulation due to its capability of learning elaborate temporal patterns. Both of the models will be checked on training, validation, and test sets while comparing the results of the simulation for predictive performance at some temporality. The experience already obtained from both approaches will come in handy in future efforts toward the creation of a larger framework for simulating neonatal vital signs comprehensively. Once research for the appropriate datasets is conducted, the group also intends to extend the approach so as to be able to model additional vital signs, such as oxygen saturation, to extend the scope of neonatal simulation to represent a more inclusive set of physiological processes.

4.3 Text Summarization

The text summarization component of our project addresses three core aspects of NICU patient care: nurse-to-nurse handover summarization, parent updater summarization via chatbot, and automated EHR data input from patient monitoring data. Each aspect plays a critical role in reducing workload, enhancing communication, and improving overall patient care.

4.3.1 Patient Handover Summarization

This aspect focuses on generating concise and context-aware summaries for nurse handovers, which typically occur at the end of each shift. The goal is to alleviate the administrative burden faced by healthcare workers and ensure the accurate and timely relay of patient status information. We achieve this by utilizing LangChain, a framework that allows seamless integration of various components for building LLM applications. By leveraging LangChain, we utilize an LLM that generates summaries based on data collected throughout a nurse's shift. The input data includes vital signs and clinical interventions represented in JSON format, which is then processed to extract key details for the summary. The system is designed to replicate a conversational handover, ensuring that the information conveyed

is easily understandable for the incoming nurse while avoiding diagnostic interpretation or clinical decision-making.

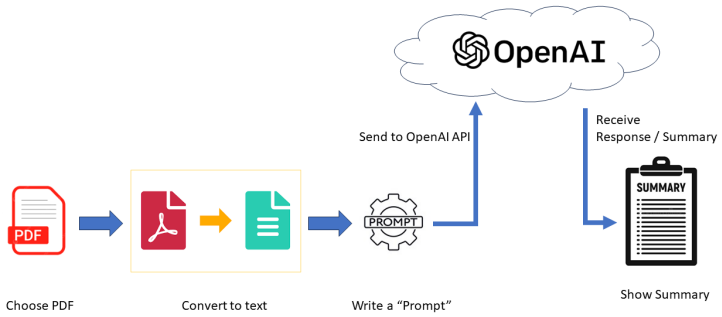

Figure 3: Example Block Diagram for Text Summarize. In this example PDFs are processed using the *PyPDFLoader* Function [20].

The first version of the Nurse-to-Nurse Summarizer utilizes a relatively straightforward approach with LangChain, leveraging JSON data from patient monitoring. The workflow involves using the *JSONLoader* function to load patient data into a LangChain document structure, which is then processed by a text splitter (*RecursiveCharacterTextSplitter*) to create manageable chunks. These chunks are embedded using *OpenAIEmbeddings* and stored in a *Chroma* vector store for retrieval. A custom prompt template is used with the *PromptTemplate* function to generate the summary based on the retrieved patient data. This version provides a comprehensive overview of patient status while handling a little context effectively.

The second version of the summarizer will incorporate Retrieval-Augmented Generation (RAG) to improve context relevance and ensure that key points are consistently highlighted during the handover process. In this setup, a pre-defined RAG script will be provided to guide the LLM's response, ensuring consistency across different shifts and providing a standardized format for handovers. The script will serve as a base, prompting the LLM to retrieve context-relevant data from the vector store, including information on 'normal' neonatal vital sign rates and typical fluctuations. This ensures that the LLM can contextualize the patient's

condition against expected norms, thereby generating a summary that highlights important events, changes in the patient's condition, and any anomalies that need attention. This structured approach helps ensure completeness and reduces the variability in the quality of summaries generated.

4.3.2 Parent Updater Summarization Via ChatBot

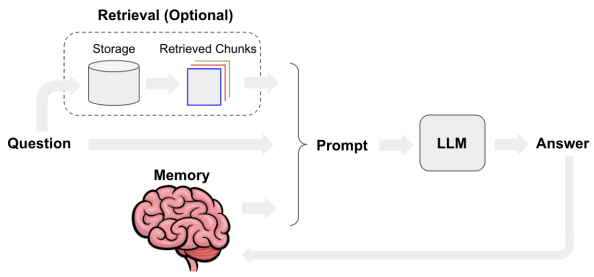

Figure 4: Simple ChatBot Implementation [21]

Parents of NICU patients often experience heightened anxiety due to the lack of real-time updates about their child's health. To address this, we implemented a chatbot-based summarizer that provides parents with quick, clear, and non-technical summaries of their child's condition. This chatbot allows parents to interact with the system by asking questions, and it generates responses based on the patient's real-time data.

Again, LangChain plays a central role in the implementation of the chatbot. The *JSONLoader* is used to load patient data into a format that can be easily processed by the LLM. The chatbot will also rely on a RAG approach to ensure that the responses are informed by the most up-to-date patient information. The RAG system retrieves relevant data from a vector store, where patient information is stored after being embedded using *OpenAIEmbeddings*. The vector store is queried to pull specific data points needed to answer parents' queries accurately.

The chatbot is configured with a predefined script that guides the LLM in generating empathetic and clear responses, similar to how a nurse would communicate with a concerned

parent. This script emphasizes providing reassurance, explaining vital sign trends, and informing parents about interventions without using overly technical language. LangChain enables chat history, which ensures that responses are contextually aware and consistent. This makes it easy to integrate the LLM with a user-friendly front-end, ensuring that parents receive informative and comforting updates in a timely manner.

4.3.3 Automated Charting

Given a PDF of the EHR template, we could directly map the data from the JSON files into the appropriate fields in the PDF. Using a library like *PyPDF2* or *pdfrw*, the JSON data can be parsed and inserted into the corresponding sections of the EHR form, thereby avoiding the need for complex natural language processing or LLMs. This approach ensures accurate data representation while simplifying the process and reducing the potential for errors.

5 Team Expertise in Relation to Project

5.1 Gurshan Riarh

For this project I am using skills from a background in Biomedical and Electrical Engineering that I have gained over the past 4 years, including the competence of mathematics-based problem-solving, programming language skill in various languages, and ethics-based decision-making in research.

This is further elaborated on the fact that my mathematics-based problem-solving skills are highly applicable for decomposing deep learning models and applying these to generate vital signs in patient simulation. These skills are also quite important in the analysis of trends in generated data of vital signs. Besides that, HTML, JavaScript, and Python allow me to create and optimize the patient simulator, in regard to the fact that it will be delivered by a thorough process. Regarding research, I will be aiming to use peer-reviewed sources and proper citation for academic integrity and compliance within the accepted standards.

In the second semester of this academic year, I will be enrolled in Intro to Machine Learning which will continue to deepen my knowledge on the subject and its subsections in relation to this project which in turn will allow for a higher level understanding of project goals.

5.2 Jesse Levine

I bring a strong foundation in both biomedical engineering and software development to this project. Throughout my Biomedical and Electrical Engineering degree at Carleton University, I have taken courses in Python programming and Functional Programming, which have equipped me with the coding skills necessary to handle tasks related to data processing and software development.

Additionally, my coursework in Bioelectrical Systems, Biomedical Systems Modeling, and Ethics in Research for Biomedical Engineering provides me with a deep understanding of the technical and ethical aspects crucial for biomedical applications like those involved in this project.

I also have direct research experience in integrating machine learning with biomedical applications. Specifically, I worked on a project involving the use of Raman spectroscopy and machine learning for the classification of bacterial species. This project not only sparked my interest in machine learning but also served as the motivation for my current involvement in this capstone project. My experience with machine learning in a biomedical context will be valuable for developing the predictive models required for patient simulators and ensuring the integration of machine learning for automated summarization is accurate and effective.

6 Timetable

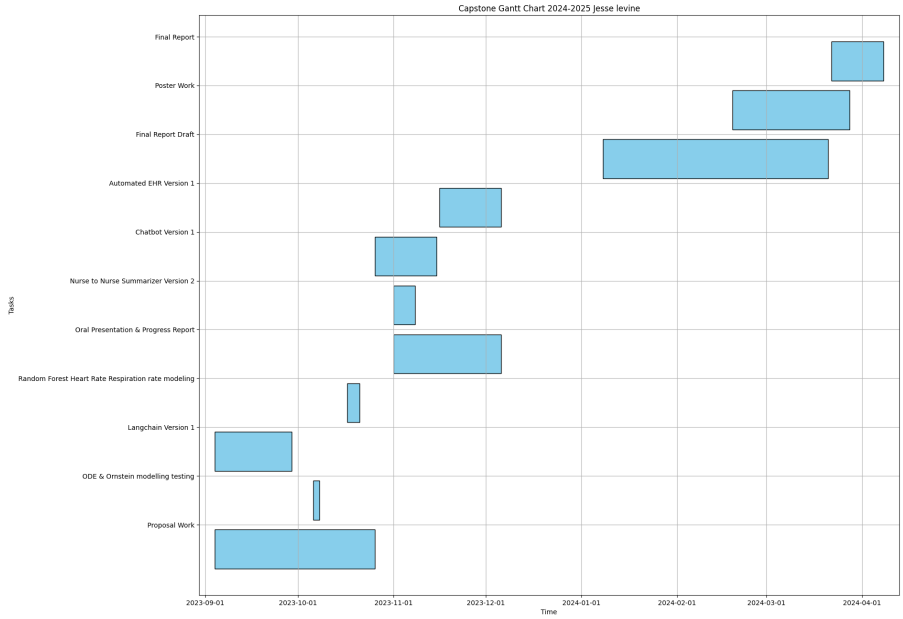

Figure 5: Gantt Chart for Fall 24 and Winter 25 Capstone Project - Jesse Levine

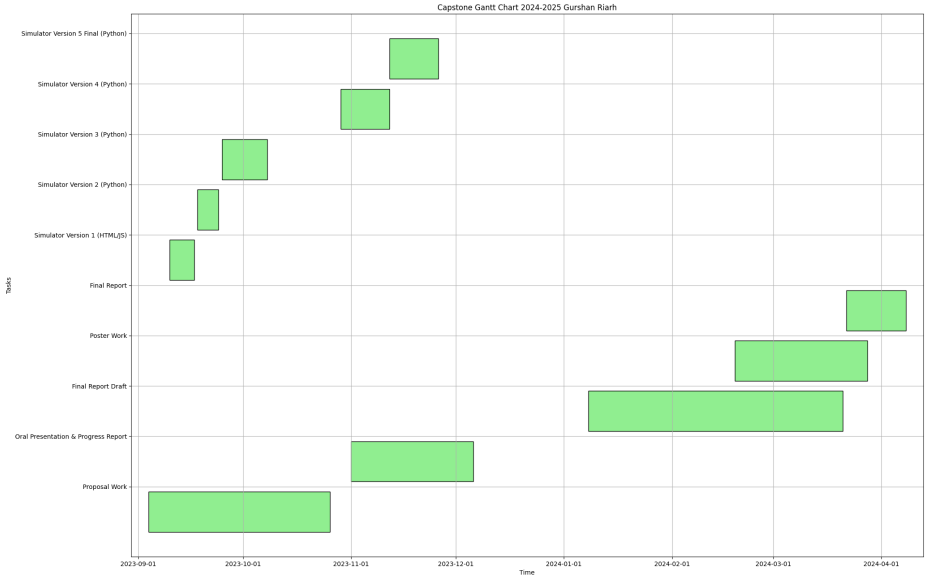

Figure 6: Gantt Chart for Fall 24 and Winter 25 Capstone Project - Gurshan Riarh

7 References

References

[1] M. A. Friesen, "Handoffs: Implications for nurses," Patient Safety and Quality: An Evidence-Based Handbook for Nurses., https://www.ncbi.nlm.nih.gov/books/NBK2649/

[2] T. Adams and I. Bogdanov, "Project Proposal as of October 2023," thesis, 2023

[3] M. Weiner, "Forced inefficiencies of the Electronic Health Record," Journal of general internal medicine, https://pmc.ncbi.nlm.nih.gov/articles/PMC6848393

[4] Dawes: Electronic medical records aren't helping patients or Doctors — Ottawa citizen, https://ottawacitizen.com/opinion/dawes-electronic-medical-records-arent-helping-patients-or-their-doctors

[5] S. E. Seaton et al., "Estimating neonatal length of stay for babies born very preterm," Archives of disease in childhood. Fetal and neonatal edition, https://pmc.ncbi.nlm.nih.gov/articles/PMC6580734/ (accessed Oct. 21, 2024).

[6] "App helps support parents of Nicu Babies no matter where they are," App Helps Support Parents of NICU Babies No Matter Where They Are, https://www.hcsc.com/newsroom/category/collaborative-care/app-helps-support-parent-nicu-babies

[7] A. Pola, "A history of alarms: Back to basics: CalmWave, inc..," CalmWave Inc, https://calmwave.ai/a-history-of-alarms-back-to-basics/

[8] Olive, "Vital signs - now and then," Vios Medical, https://www.viosmedical.com/in/resources/vital-signs-now-and-then/

[9] S. D. Health, "The importance of continuous vital sign monitoring in Hospitals," Patient Vital Signs Monitoring System, https://soteradigitalhealth.com/blog/the-importance-of-continuous-vital-sign-monitoring-in-hospitals

[10] R. Kulshrestha, "Transformers," Medium, https://towardsdatascience.com/transformers-89034557de14

[11] A. Vaswani et al., "Attention is all you need," arXiv.org, https://arxiv.org/abs/1706.03762

[12] "Transformer neural networks: A step-by-step breakdown," Built In, https://builtin.com/artificial-intelligence/transformer-neural-network

[13] PamC/FLYINGMUM, "Geoffrey Hinton: The godfather of deep learning and ai," Medium, https://flyingmum.medium.com/geoffrey-hinton-the-godfather-of-deep-learning-and-ai-1f5e10390b30

[14] J. Ma et al., "Segment Anything in Medical Images and Videos: Benchmark and Deployment," arvix.org,

[15] M. Cascella et al., "The breakthrough of large language models release for Medical Applications: 1-year timeline and perspectives," Journal of medical systems, https://pmc.ncbi.nlm.nih.gov/articles/PMC10873461/

[16] "Mayo Clinic to deploy and test Microsoft generative AI Tools - Mayo Clinic News Network," Mayo Clinic, https://newsnetwork.mayoclinic.org/discussion/mayo-clinic-to-deploy-and-test-microsoft-generative-ai-tools/

[17] A. Toma et al., "Clinical camel: An open expert-level medical language model with dialogue-based knowledge encoding," arXiv.org, https://arxiv.org/abs/2305.12031

[18] "Introduction,"LangChain, https://python.langchain.com/v0.2/docs/introduction/

[19] "Tutorials," LangChain, https://python.langchain.com/v0.2/docs/tutorials/

[20] P. Nair, "Document summarizer using open AI on Langchain," NinethSense oWnZ mE!, https://blog.ninethsense.com/general/document-summarizer-using-open-ai-on-langchain/

[21] "Chatbots," LangChain, https://python.langchain.com/v0.1/docs/use_cases/chatbots/

[22] Kumar, N., Akangire, G., Sullivan, B. et al. Continuous vital sign analysis for predicting and preventing neonatal diseases in the twenty-first century: big data to the forefront. Pediatr Res 87, 210–220 (2020). https://doi.org/10.1038/s41390-019-0527-0

[23] Sullivan, B.A., Fairchild, K.D. Vital signs as physiomarkers of neonatal sepsis. Pediatr Res 91, 273–282 (2022). https://doi.org/10.1038/s41390-021-01709-x

[24] Gaumard Scientific. (2024). Super TORY® S2220 - Advanced Newborn Patient Simulator. Gaumard. https://www.gaumard.com/supertory

[25] Sahai, N. (2023, September 6). Mastering Random Forest Regression: A Comprehensive Guide. AnalytixLabs. https://www.analytixlabs.co.in/blog/random-forest-regression/

[26] Poudel, S. (2023, August 28). Recurrent Neural Network (RNN) Architecture Explained. Medium. Retrieved from https://medium.com/@poudelsushmita878/recurrent-neural-network-rnn-architecture-explained-1d69560541ef

[27] Stryker, C. (2024, October 4). What is a Recurrent Neural Network (RNN)? IBM. https://www.ibm.com/topics/recurrent-neural-networks

[28] Tondak, A. (2023, June 23). Recurrent Neural Networks — RNN Complete Overview 2023. K21 Academy. Retrieved from https://k21academy.com/datascience-blog/machine-learning/recurrent-neural-networks/

[29] Amanatullah. (2023, June 12). Vanishing Gradient Problem in Deep Learning: Understanding, Intuition, and Solutions. Retrieved from https://medium.com/@amanatulla1606/vanishing-gradient-problem-in-deep-learning-understanding-intuition-and-solutions-da90ef4ecb54

[30] Rosen, K. R. (2008). The history of medical simulation. Journal of Critical Care, 23(2), 157-166. https://doi.org/10.1016/j.jcrc.2007.12.004

[31] Yousef N, Moreau R, Soghier L. Simulation in neonatal care: towards a change in traditional training? Eur J Pediatr. 2022 Apr;181(4):1429-1436. doi: 10.1007/s00431-022-04373-3. Epub 2022 Jan 12. PMID: 35020049; PMCID: PMC8753020.

[32] Ben Ahmed, H., & Dziri, C. (2020). History of Medical Simulation. Honoris Medical Simulation Center. Retrieved from https://honorismedicalsimulation.net/research/history-of-medical-simulation/

[33] Anibal J, Huth H, Gunkel J, Gregurick S, Wood B. Simulated Misuse of Large Language Models and Clinical Credit Systems. medRxiv [Preprint]. 2024 Sep 16:2024.04.10.24305470. doi: 10.1101/2024.04.10.24305470. PMID: 38645190; PMCID: PMC11030492.

[34] Kanithi, P. K., Christophe, C., Pimentel, M. A. F., Raha, T., Saadi, N., Javed, H., Maslenkova, S., Hayat, N., Rajan, R., & Khan, S. (2024). MEDIC: Towards a Comprehensive Framework for Evaluating LLMs in Clinical Applications. arXiv preprint arXiv:2409.07314. Retrieved from https://arxiv.org/abs/2409.07314

[35] Narula, S., Karkera, S., Challa, R., Virmani, S., Chilukuri, N., Elkas, M., Thammineni, N., Kamath, A., Jaiswal, P., & Krishnan, A. (2023). Testing the Accuracy of Modern LLMs in Answering General Medical Prompts. International Journal of Social Science and Economic Research, 8(9), 2793-2800.

[36] National Institutes of Health. (2023). PhysioNet: The research resource for complex physiologic signals. National Institutes of Health. https://commonfund.nih.gov/sites/default/files/SPARCMark-MIT-Sess2.pdf

[37] PhysioNet. (2023). PhysioNet Databases. Retrieved from https://physionet.org/about/database/

A Appendices

A.1 Nurse_to_Nurse_Summarizer_v1

```
from langchain_openai import ChatOpenAI
from langchain_community.document_loaders import JSONLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain_openai import OpenAIEmbeddings
from langchain_chroma import Chroma
from langchain_core.prompts import PromptTemplate
from langchain_core.runnables import RunnablePassthrough
from langchain_core.output_parsers import StrOutputParser

from dotenv import load_dotenv
load_dotenv()

def format_docs(docs):
    return "\n\n".join(doc.page_content for doc in docs)

def Nurse_2_Nurse_Summarization_v1(filepath: str):

    llm = ChatOpenAI(model = 'gpt-4o')

    loader = JSONLoader(file_path=filepath, jq_schema='.', text_content=False)
    docs  = loader.load()

    text_splitter = RecursiveCharacterTextSplitter(chunk_size = 10000, chunk_overlap = 1000, add_start_index = True)
    all_splits = text_splitter.split_documents(docs)

    embed = OpenAIEmbeddings(show_progress_bar=True, request_timeout= 10000)

    vectorstore = Chroma.from_documents(documents=all_splits, embedding=embed, persist_directory= './gurshan_3_db')
    retriever = vectorstore.as_retriever(search_type="similarity",search_kwargs={'k': 6})

    template = """Use the following pieces of context to summarize the data at the end.
    If you dont know, just say you dont know.
    Pretend you are a health care worker at the ned of your shift.
    Now is the time for patient handover. Explain and summarize to the next nurse the status of the patient described in the JSON file.
    In the case of irregular vital sign fluctuations, flag it and mention it

    Always say "The End!", in a new line at the end of the answer.
```

```
    {context}

    Question: {question}

    Helpful Answer:"""
    custom_rag_prompt = PromptTemplate.from_template(template)
    rag_chain = ({"context": retriever | format_docs, "question": RunnablePassthrough()} | custom_rag_prompt | llm | StrOutputParser())
    response = rag_chain.invoke("Summarize")

    return response

Nurse_2_Nurse_Summarization_v1('Gurshan_Data.json')
```

A.2 Output from Nurse_to_Nurse_Summarizer_v1

"The patient has experienced several fluctuations in vital signs throughout the shift, with notable irregularities in heart rate and respiratory rate. The heart rate has ranged from 130 to 158 bpm, and the respiratory rate has varied from 34 to 47 breaths per minute. The patient's skin tone has alternated between normal, red, blueish, pink, and yellow-jaundice, indicating potential underlying issues. There have been multiple interventions, including lighting adjustments, oxygen administration, feeding, diaper changes, medication, and position changes, to address varying pain levels ranging from low to extreme.irregularities include periods of elevated heart rate and respiratory rate, particularly when the skin tone was blueish or red, suggesting respiratory distress or inadequate oxygenation at times. Pain levels have also fluctuated significantly, affecting the patient's comfort and requiring frequent analgesic and antibiotic administration.patient requires close monitoring of vital signs and continuous assessment of interventions to ensure stability. There is a need for vigilant attention to pain management and respiratory support, given the frequent changes in skin tone and oxygen saturation levels. The End!"

A.3 Version 3 - Patient Simulator

A.3.1 NICU_Simulator_Params

```
    # Gurshan Riarh 101182603 - SYSC 4907 - SimulatorV3 (Vital Sign Params & JSON Formmating/Exporting)
import json
import random
import numpy as np
from datetime import datetime, timedelta

class NeonateSimulator:
    def __init__(self, shift_length=8, patient_id=None, model_choice='1'): # Change the shift length here but make sure to update mathematical models to accomodate time c
        """
        Initializes the simulator with shift length,
        output file name, patient ID, and the choice of model for vital sign simulation.
        """
        self.shift_length = shift_length # This is set to 8 hours by default

        self.start_time = datetime.now()
        self.data = []
        self.patient_id = patient_id
        self.output_file = (patient_id + "_Data.json")
        self.model_choice = model_choice
        if model_choice == '2':  # If option "2" is chosen then initialize the models
            self.heart_rate_model, self.respiratory_rate_model, self.body_temperature_model, self.spo2_model, self.systolic_bp_model, self.diastolic_bp_model = self.simul

    def simulate_vital_sign_models(self):
        """
        Mathematical models for vital signs -> Use sinusoidal functions along with noise to
        replicate realistic breathing patterns for neonates.
        """
        duration = self.shift_length * 60  # Total duration in minutes
        t = np.arange(0, duration, 1)  # Time vector for the simulation period

        # Heart rate model using sinusoidal variations for circadian rhythm and respiratory influences
        HR_base = 140  # Baseline heart rate
        HR = HR_base + 10 * np.sin(2 * np.pi * 1/1440 * t) + 5 * np.sin(2 * np.pi * 1/60 * t) + np.random.normal(0, 2, len(t))

        # Respiratory rate model with slower circadian variation and random noise
        RR_base = 40  # Baseline respiratory rate
        RR = RR_base + 5 * np.sin(2 * np.pi * 1/720 * t) + np.random.normal(0, 1, len(t))

        # Body temperature model that oscillates around a normal value with minimal variation
        BT_base = 37.0  # Baseline body temperature in Celsius
        BT = BT_base + 0.2 * np.sin(2 * np.pi * 1/1440 * t) + np.random.normal(0, 0.1, len(t))

        # Oxygen saturation model with minimal variation (normal for neonates)
        SpO2_base = 98  # Baseline oxygen saturation percentage
        SpO2 = SpO2_base + 1 * np.sin(2 * np.pi * 1/1440 * t) + np.random.normal(0, 0.5, len(t))

        # Systolic blood pressure model with small sinusoidal variation and random noise
        SBP_base = 70  # Baseline systolic blood pressure
        SBP = SBP_base + 5 * np.sin(2 * np.pi * 1/1440 * t) + np.random.normal(0, 2, len(t))

        # Diastolic blood pressure model with small sinusoidal variation and random noise
```

```
    DBP_base = 40  # Baseline diastolic blood pressure
    DBP = DBP_base + 3 * np.sin(2 * np.pi * 1/1440 * t) + np.random.normal(0, 1.5, len(t))

    return iter(HR), iter(RR), iter(BT), iter(SpO2), iter(SBP), iter(DBP)



def generate_vital_signs(self):
    """
    Generates vital signs for a single time step. Depending on the model choice,
    it either uses random values (default) or the predetermined mathematical models (def simulate_vital_sign_models).

    """
    if self.model_choice == '2':
        heart_rate = round(next(self.heart_rate_model))
        respiratory_rate = round(next(self.respiratory_rate_model))
        body_temperature = round(next(self.body_temperature_model), 1)
        spo2 = round(next(self.spo2_model))
        systolic_bp = round(next(self.systolic_bp_model))
        diastolic_bp = round(next(self.diastolic_bp_model))
    else:
        heart_rate = random.randint(120, 160)
        respiratory_rate = random.randint(30, 60)
        body_temperature = round(random.uniform(36.5, 37.5), 1)
        spo2 = random.randint(95, 100)
        systolic_bp = random.randint(60, 90)
        diastolic_bp = random.randint(30, 60)

    skin_tone =  random.choice(["normal", "blueish", "yellow-jaundice", "pink", "red"])

    return {
        "heart_rate": heart_rate,
        "respiratory_rate": respiratory_rate,
        "body_temperature": body_temperature,
        "spo2": spo2,
        "systolic_bp": systolic_bp,
        "diastolic_bp": diastolic_bp,
        "Skin_Tone": skin_tone
    }


def simulate(self):
    """
    Runs the simulation over the specified shift length, generating vital signs and interventions
    at defined time intervals and storing them for export.
    Time Stamping is accurate but uses a python library so it is not accurate real time.
    """
    current_time = self.start_time
    end_time = self.start_time + timedelta(hours=self.shift_length)
    while current_time < end_time:
        entry = {
            "timestamp": current_time.strftime("%Y-%m-%d %H:%M:%S"),
            "vital_signs": self.generate_vital_signs(),
            "intervention": self.generate_interventions()
        }
        self.data.append(entry)
```

```
            current_time += timedelta(minutes=30) # Change this if you want different output intervals (originally set to 1 minute)


    def export_data(self):
        """
        Exports all collected data into a JSON file, including the patient ID at the top of the file.
        """
        output_data = {
            "patient_id": self.patient_id,
            "data": self.data
        }
        with open(self.output_file, "w") as f:
            json.dump(output_data, f, indent=4)
        print(f"Data successfully exported to {self.output_file}")

    def generate_interventions(self):
        """
        Randomly selects an intervention from a predefined list and generates details for it.
        Ask Head Nurse what other interventions they look for.
        """
        interventions = [
            {"type": "lighting adjustment", "detail": random.choice(["dim", "bright"])},
            {"type": "position change", "detail": random.choice(["supine", "left lateral", "right lateral"])},
            {"type": "diaper change", "detail": None},
            {"type": "oxygen administration", "detail": random.choice(["increase", "decrease"]) + " flow"},
            {"type": "feeding", "detail": f"{random.randint(10, 30)} ml formula"},
            {"type": "medication", "detail": f"{random.choice(['antibiotic', 'analgesic'])} {random.randint(1, 5)} mg/kg"}
        ]
        #return random.choice(interventions)

        selected_interevention = random.choice(interventions) # Stores the choice made for intervention

        pain_level = random.choice(["Low", "High", "Moderate", " Extreme" ]) # Sets options for pain level during intervention

        return{
            "Intervention": selected_interevention,
            "pain_level": pain_level
        }
```

A.3.2 Vital_Sign_Plots

```
    # Gurshan Riarh 101182603 - SYSC 4907 - SimulatorV3 (Vital Sign Plotting)
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from datetime import datetime, timedelta

def plot_vital_signs(data, start_time, shift_length):
    """
    Generates and saves plots of vital signs over the duration of the simulation shift (Neo_Simulator_Parameters file on line 8).
    data: List of dictionaries containing timestamped entries of vital signs data.
    start_time: The datetime object representing when the simulation started.
    Shift_length: Integer representing the number of hours the simulation ran.
    """
    # dictionary to hold lists of average vital signs values for each hour.
    hourly_data = {sign: [] for sign in ["heart_rate", "respiratory_rate", "body_temperature", "spo2", "systolic_bp", "diastolic_bp"]}
```

```
    # Generate hourly intervals from the start time to the end of the simulation.
    hourly_intervals = [(start_time + timedelta(hours=i)) for i in range(shift_length + 1)]

    # Loop over each interval and collect vital sign data.
    for i in range(len(hourly_intervals) - 1):
        # Filter data to include only entries within the current hourly interval.
        interval_data = [entry for entry in data if hourly_intervals[i] <= datetime.strptime(entry["timestamp"], "%Y-%m-%d %H:%M:%S") < hourly_intervals[i + 1]]

        # For each vital sign, calculate the average value over the interval and store it.
        for sign in hourly_data:
            if interval_data:
                average = sum(entry["vital_signs"][sign] for entry in interval_data) / len(interval_data)
                hourly_data[sign].append(average)
            else:
                hourly_data[sign].append(None)  # Append None if there's no data for this interval.

    # Plot each vital sign over time.
    for sign, values in hourly_data.items():
        plt.figure()  # Create a new figure for each vital sign.
        plt.plot(hourly_intervals[:-1], values, label=sign, marker='o', linestyle='-')  # Plot the averages with markers and lines.
        plt.xlabel("Time (hours)")  # Label for the x-axis.
        plt.ylabel(sign)  # Label for the y-axis specific to the vital sign.
        plt.title(f"{sign} over Time")  # Title for the plot.
        plt.gca().xaxis.set_major_locator(mdates.HourLocator(interval=1))  # Set tick marks to every hour.
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))  # x-axis labels to show hours and minutes.
        plt.legend()  # Show a legend identifying the plotted line.
        plt.savefig(f"{sign}_hourly_plot.png")  # Save the plot to a file.
        plt.close()
```

A.3.3 Main

```
    # Gurshan Riarh 101182603 - SYSC 4907 - SimulatorV3 (Main - UI Inputs)
import tkinter as tk
from tkinter import simpledialog, messagebox
from NICU_Simulator_Parmas import NeonateSimulator
from Vital_Sign_Plots import plot_vital_signs
import sys  # Import sys to use sys.exit() for exiting the program

def main():
    """
    Main function to drive the neonate simulator application.
    This function initializes a GUI to accept a Patient ID, runs the simulation, exports data, and generates plots.
    """
    root = tk.Tk()  # Create the main window for the tkinter application.
    root.withdraw()  # Hide the main window since we only need the dialog prompt.

    # Function to ensure input is provided, exits if Cancel is pressed.
    def get_user_input(prompt, title):
        while True:
            response = simpledialog.askstring(title, prompt)
            if response is None:  # Exit if the user presses Cancel
                sys.exit(0)
            if response.strip():
```

```
                return response
            messagebox.showwarning("Input Required", "Please provide the necessary information to proceed.")

    # Function to make sure the model choice is either '1' or '2', exits if Cancel is pressed.
    def get_model_choice(prompt, title):
        while True:
            response = get_user_input(prompt, title)
            if response in ['1', '2']:
                return response
            messagebox.showwarning("Invalid Input", "Please enter '1' for default random signs or '2' for mathematical models.")

    # Prompt the user to enter a Patient ID using a simple dialog and ensure valid input.
    patient_id = get_user_input("Please enter the Patient ID:", "Patient ID")

    # Ask user to select which model to use for the simulation and ensure valid input.
    model_choice = get_model_choice("Enter '1' for default random signs, '2' for mathematical models:", "Model Selection")

    # Create an instance of the NeonateSimulator with the provided Patient ID and model choice.
    simulator = NeonateSimulator(patient_id=patient_id, model_choice=model_choice)
    simulator.simulate()  # Run the simulation to generate data.
    simulator.export_data()  # Export the simulated data to JSON file.
    #==========COMMENT BACK IN WHEN READY FOR PLOTTING====================#
    #plot_vital_signs(simulator.data, simulator.start_time, simulator.shift_length)  # Generate and save plots of the vital signs.

if __name__ == "__main__":
    main()
```

A.3.4 Random Forest Regression for Heart Rate and Respiration Rate

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.metrics import accuracy_score

data = pd.read_csv('all_infants_1h_subset_vital_signs.csv')

def create_lag_features(df, lag=5):
    for i in range(1, lag + 1):
        df[f'heart_rate_lag_{i}'] = df['heart_rate'].shift(i)
        df[f'respiration_rate_lag_{i}'] = df['respiration_rate'].shift(i)
    return df

data = create_lag_features(data, lag=5)
data = data.dropna()

features = [
    'heart_rate', 'respiration_rate',
    'heart_rate_lag_1', 'heart_rate_lag_2', 'heart_rate_lag_3', 'heart_rate_lag_4', 'heart_rate_lag_5',
    'respiration_rate_lag_1', 'respiration_rate_lag_2', 'respiration_rate_lag_3', 'respiration_rate_lag_4', 'respiration_rate_lag_5'
]
target = ['heart_rate', 'respiration_rate']

X = data[features]
y = data[target]
```

```
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.5, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predict on the test set
y_val_pred = rf_model.predict(X_val)
val_mse = mean_squared_error(y_val, y_val_pred)
print(f'Mean Squared Error on Validation Set: {val_mse:.2f}')
val_mae = mean_absolute_error(y_val, y_val_pred)
print(f'Mean Absolute Error on Validation Set: {val_mae:.2f}')

y_test_pred = rf_model.predict(X_test)
test_mse = mean_squared_error(y_test, y_test_pred)
print(f'Mean Squared Error on Test Set: {test_mse:.2f}')
test_mae = mean_absolute_error(y_test, y_test_pred)
print(f'Mean Absolute Error on Test Set: {test_mae:.2f}')

#initial_vector = [112.8, 60, 112.8, 75, 115.38, 20.41, 113.21, 61.22, 115.38, 48.39, 114.5, 120]
#initial_vector = [159.57, 41.1, 157.89, 27.027, 159.57, 54.54, 160.43, 58.82, 158.73, 23.26, 161.29, 63.83]
#current_input = np.array(initial_vector).reshape(1, -1)
current_input = X_test.sample(1).values.reshape(1, -1)



simulation_steps = 96
simulated_data = []

for step in range(simulation_steps):
    # Predict the next heart rate and respiration rate
    next_prediction = rf_model.predict(current_input)
    simulated_data.append(next_prediction[0])

    # Update the input for the next prediction
    # Shift the lag features and add the new prediction
    current_input = np.roll(current_input, shift=-2)
    current_input[0, -2:] = next_prediction

# Convert simulated data to DataFrame and save
simulated_df = pd.DataFrame(simulated_data, columns=target)
simulated_df.to_csv('simulated_vital_signs_rf_v4.csv', index=False)
```