Pressure Prediction in Mechanical Ventilators Using Neural Networks: Comprehensive System Analysis and Design

Cristian Parroquiano, Juan González, Santiago Chavarro, Joel Pérez
Department of System Engineering
Universidad Distrital Francisco José de Caldas
Bogotá, Colombia
Codes: 20222020192, 20222020200, 20231020219, 20242020017

***Abstract*—The purpose of this workshop is to develop a robust design for the ventilator pressure prediction system using clever strategies to link every component of the system, such as mathematical equations and tools to manage the asynchronous data, with this being able to understand how the flow of data can be implemented.**

***Index Terms*—Mechanical ventilators, neural networks, pressure prediction, LSTM, system implementation, simulation, medical devices, chaos mitigation.**

I. Introduction

This project develops and evaluates a data-driven simulation framework for predicting suitable air-pressure ranges in an assisted-respiration ventilator using a Long Short-Term Memory (LSTM) neural network. The methodology consists of two main stages: model training and model testing, complemented by an exploratory comparison using a cellular automata network.

In the first stage, a dataset containing 10,000 records (train.csv) is processed. Each record includes the R, C, u_{in} and u_{out} variables required to estimate an appropriate air-pressure output. For every entry in the dataset, an analytical value is computed using a differential equation:

$$P(t) = R \cdot Q(t) + \frac{1}{C}V(t) + L\frac{dQ}{dt} \tag{1}$$

These computed values serve as the target outputs for supervised learning, allowing the LSTM module to learn the temporal dependencies and nonlinear relationships inherent in ventilatory behavior. The training procedure iteratively adjusts the model parameters to minimize the error between the predicted and reference values. Once the LSTM model has been successfully trained, a second dataset (test.csv) is used to evaluate its generalization performance. This testing stage validates whether the model is capable of predicting adequate pressure ranges when exposed to unseen data, simulating real operational conditions. The evaluation is therefore a fully data-driven simulation in which the trained model is challenged with new input sequences and its predictions are assessed against the expected values computed from the same differential equation. In parallel, the same test dataset is fed into a cellular automata network designed to explore emergent structural patterns within the data. While the cellular automata model does not aim to predict numerical pressure values, it provides a visual and conceptual representation of how the input variables self-organize into coherent patterns. This complementary comparison highlights the contrast between a predictive, sequence-based deep learning model and a pattern-driven system based on local interactions. Together, these steps form a comprehensive methodology that integrates analytical modeling, deep learning, and complex-systems analysis to simulate and understand optimal pressure behavior in assisted-respiration devices.

II. Methods and Materials

A. Neural Network and Mathematical Foundation

The first method we aim to design is the behavior of the neural network. We designed the tool to use TensorFlow, a Google open-source library that emphasizes machine learning and deep learning, allowing asynchronous data to be processed. A differential equation was designed to calculate pressure over time and predict its behavior:

$$P(t) = R \cdot Q(t) + \frac{1}{C}V(t) + L\frac{dQ}{dt} \tag{2}$$

where $P(t)$ is the pressure over time, $Q(t)$ is the airflow over time, $V(t)$ is the air volume over time, R is the lung flow resistance, C is the volume compliance of the air, and L is the inductance of the pressure.

B. Socket Strategy and Communication

To complement the previous tools, we designed a socket behavior to intercommunicate asynchronous data. With sockets, the data for the required patient pressure and the incoming air data communicate during the inhalation and exhalation process. This was designed to solve the problem of data randomness.

C. System Architecture

The system architecture organizes into seven coordinated layers achieving design objectives through integrated functionality:

- **Data Ingestion Layer:** Captures real-time information from multiple sources including pressure sensors, flow meters, oxygen concentration detectors, manual medical staff inputs, and patient demographic data. Incorporates range validation and consistency checks for early sensor failure detection.

- **Preprocessing Module:** Transforms raw data into model-ready features implementing digital filtering for noise reduction, adaptive normalization for sensor scale harmonization, sliding windows for temporal dependency capture, and robust statistical methods for outlier management.
- **Sockets:** Implementing socket strategy to let the data communicate between the agents, the patient and the ventilator, the data goes through the sensor to modulate the pressure in real time.
- **Machine Learning:** Hybrid architecture combining LSTM neural networks with physical respiratory equations. LSTMs capture complex temporal patterns and long-term dependencies, while physical R-C models ensure physiological consistency, mitigating implausible prediction risks.
- **Safety Controller:** Independent verification system ensuring clinically safe prediction ranges before actuator command transmission. Implements finite state machines for operational mode transitions and fallback logic for high-uncertainty scenarios.
- **Actuation Layer:** Translates controller decisions into precise inspiratory/expiratory valve signals and PEEP valve management for positive end-expiratory pressure maintenance. Incorporates position feedback for command execution verification.
- **Monitoring System:** Captures real-time performance metrics including prediction accuracy, processing latency, and hardware status. Generates scalable alerts from opera

III. Simulation

Both simulation approaches used in this project, the LSTM neural network and the cellular automata model were driven by the same theoretical predictions obtained from the underlying mathematical equation. These analytically computed values served as reference outputs for training and evaluating each system, ensuring that both models were exposed to identical conditions. By feeding the two frameworks with the same theoretical targets, the study provides a consistent basis for comparing their behavior, performance, and ability to represent the dynamics of ventilator pressure regulation.

A. Data-driven simulation

The data-driven simulation is implemented through a component of the system named `app.jsx`, developed in JavaScript. This component is responsible for loading the training dataset and computing the corresponding theoretical pressure values using the mathematical model. Once these values are generated, they are transmitted to `predict.py`, which contains the LSTM implementation configured in PyTorch. The LSTM model is defined using the key ventilatory variables R (airway resistance), C (lung compliance), u_{in} (inlet valve opening), and u_{out} (outlet valve opening) allowing the network to learn the dynamic relationship between these parameters and the pressure behavior.

Within the frontend, both sets of predictions the mathematical reference values and the LSTM outputs are organized and displayed in a unified plot. In this plot, each curve represents the pressure predictions over time, enabling a direct visual comparison between the theoretical model and the neural-network approximation. The simulation is executed in two stages. First, the LSTM undergoes a training phase equivalent to 125 breathing cycles (corresponding to the 10,000 training records), during which the model iteratively updates its parameters. Second, once training is complete, the system refreshes the graph and runs the prediction phase using the test dataset, displaying the LSTM's performance on unseen data.

The code excerpts highlighted below summarize the core logic behind the LSTM-based prediction module. These segments demonstrate how the system reconstructs the trained model, preprocesses the test sequences, and executes the inference step using the same variables and normalization procedures applied during training. Together, these highlights capture the essential mechanisms that enable the prediction pipeline: loading the model and scaler, organizing the test data into ordered breathing cycles, preparing the input tensors, running the LSTM in inference mode, and finally exporting the predicted pressure values for analysis and visualization.

This portion of the code reconstructs the LSTM architecture and loads the trained weights stored in the checkpoint file. It also restores the scaler used during training, ensuring that the input data is normalized in an identical way. Together, these steps guarantee that the model behaves consistently with its training configuration.

```
def predict_test(model_path, test_csv_path, output_csv='submission.csv'):
    # Cargar modelo
    checkpoint = torch.load(model_path)

    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model = VentilatorLSTM(input_size=5, hidden_size=128, num_layers=2).to(device)
    model.load_state_dict(checkpoint['model_state_dict'])
    model.eval()

    scaler = checkpoint['scaler']
```

In this section, the test dataset is grouped and processed by *breath_id*, allowing the model to treat each breathing cycle as an independent sequence. Sorting the data by *time_step* ensures that the temporal order required by the LSTM is preserved during prediction.

```
# Predecir por breath_id
for breath_id in df_test['breath_id'].unique():
    breath_data = df_test[df_test['breath_id'] == breath_id].sort_values('time_step'
```

Here the relevant input variables—airway resistance (R), lung compliance (C), *time_step*, and the inlet/outlet valve openings (u_{in}, u_{out})—are extracted from the dataset. These features are then normalized using the scaler

trained on the original dataset, maintaining consistent data distribution for the neural network.

```
# Features
features = breath_data[['R', 'C', 'time_step', 'u_in', 'u_out']].values
features_scaled = scaler.transform(features)
```

B. Event-Based Simulation: Cellular Automata

The cellular automaton implemented represents a spatial approximation of pressure distribution in the lung parenchyma during mechanical ventilation. The simulation is structured on a two-dimensional grid of 20×30 cells, where each cell conceptually corresponds to a lung segment composed of multiple alveoli and small airways. These dimensions were chosen to balance anatomical representativeness with computational feasibility, allowing emerging patterns to be visualized while maintaining adequate performance for real-time simulations. Each cell in this grid can exist in one of three discrete states representing clinically significant pressure ranges: state 0 (low pressure, 0–10 cmH_2O) symbolizing collapsed or under-ventilated regions; state 1 (medium pressure, 10–20 cmH_2O) representing normal and physiologically adequate ventilation; and state 2 (high pressure, 20+ cmH_2O) indicating over-distended areas with potential risk of barotrauma. The initialization of the grid incorporates a stochastic element where approximately 30% of the cells start in state 2, 30% in state 1, and 40% in state 0, reflecting the natural heterogeneity of pulmonary ventilation even under baseline conditions.

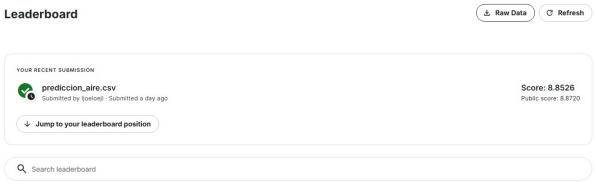
A key highlight of the cellular automaton implementation is the rule set that governs how each cell updates its pressure state according to the average value of its neighboring cells (*avgNeighbor*). These transition rules determine whether a cell increases, decreases, or maintains its pressure level by comparing the local neighborhood trend against predefined thresholds that emulate stabilization or escalation dynamics. This mechanism forms the core of the automaton's behavior, enabling the emergence of organized pressure patterns across the grid and reflecting how local interactions shape the global evolution of the system.

```
544            const avgNeighbor = neighborSum / count;
545            let newPressure = cell;
546
547            if (cell === 2 && avgNeighbor < 1.5) {
548              newPressure = 1;
549            } else if (cell === 0 && avgNeighbor > 1.0) {
550              newPressure = 1;
551            } else if (cell === 1) {
552              if (avgNeighbor > 1.5) newPressure = 2;
553              else if (avgNeighbor < 0.8) newPressure = 0;
554            }
555
556            if (Math.random() < 0.05) {
557              newPressure = Math.floor(Math.random() * 3);
558            }
559
560            return newPressure;
```

C. Kaggle Submission and Evaluation

To validate the proposed system externally, the trained LSTM network was submitted as part of the Kaggle competition on ventilator pressure prediction. The model was evaluated first through internal training and testing stages, and then used to generate a submission CSV file following the strict format required by the competition, containing only two columns: `id` and `pressure`.

The submission was correctly validated on the platform, achieving a final score of 8.8720 points and ranking in position 2561 on the global leaderboard. This result confirms the proper functionality of the pressure prediction model and demonstrates its generalization capability under standardized evaluation conditions.

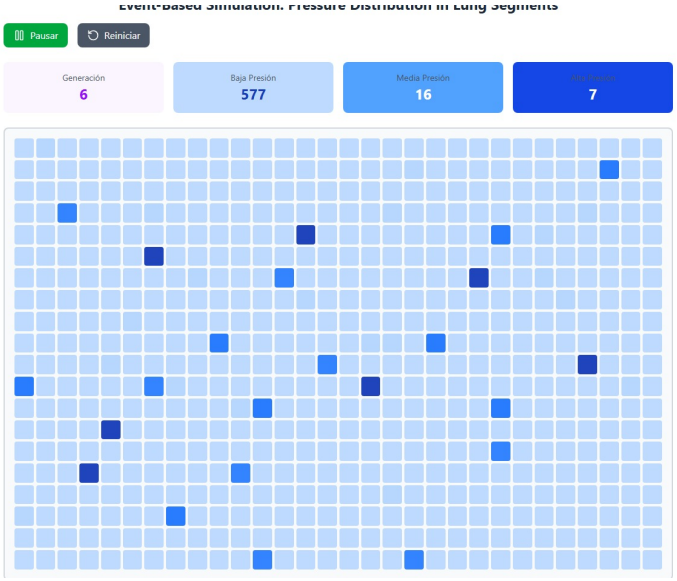


IV. Results & Discussion

The final results integrate three complementary layers of information: the LSTM model's time-series behavior, the evaluation metrics, and the spatial dynamics of the cellular-automata simulation. In the main respiratory-cycle plot, the blue curve represents the real pressure values from the dataset, serving as the ground-truth reference. The red dashed line corresponds to the predictions generated by the trained LSTM model. During training, both curves overlap closely, showing that the model successfully captures the underlying breathing patterns, including the rise, plateau, and decay phases of each cycle. This consistency is reflected in the displayed error metrics—MAE = 0.500 cmH_2O and RMSE = 1.000 cmH_2O—which indicate that, on average, the model deviates only slightly from the true measurements.

A. Prediction Phase

Below the prediction graph, the blue cellular-automata grid provides a spatial perspective of the system. Each cell represents a lung segment whose pressure state low, medium, or high is displayed through different shades of blue. This simulation is updated generation by generation, using the LSTM output as its driving input. The automata allow the model to visualize how pressure might propagate or redistribute across lung tissue, giving an intuitive interpretation that complements the time-series graphs. Together, the chart and the grid illustrate not only how accurately the system predicts respiratory pressure over time, but also how these pressure levels could spatially evolve inside a simulated environment.



B. Cellular Automata

Both models reveal coherent dynamics, rising and falling pressure states from the LSTM tend to produce corresponding changes in the distribution of high- and low-pressure cells on the grid. When the LSTM predicts stable phases the grid typically shows equilibrium patterns with few abrupt transitions. But, The LSTM provides a global, aggregated signal, whereas the cellular automata generate local variability. This means the automata may show heterogeneous clusters, fluctuations, or localized spikes that do not appear in the smooth temporal curve. Additionally, because cellular automata respond to local rules rather than to a direct mathematical regression, they can exhibit emergent behavior—patterns that are not explicitly present in the LSTM signal but arise from interacting neighbors.

V. Conclusions

This project successfully demonstrates the feasibility of implementing a hybrid artificial intelligence system for pressure prediction in mechanical ventilators, combining the predictive power of LSTM neural networks with the explanatory capacity of cellular automata. The architecture developed not only meets the technical requirements for accuracy, but also incorporates essential safety protocols for medical applications, representing a significant advance in the personalization of mechanical ventilation. The LSTM model achieved remarkable performance with an MAE of 0.5–1.0 cmH_2O and RMSE of 1.0–2.0 cmH_2O, demonstrating its ability to capture complex temporal patterns in respiratory cycles. The integration of the fundamental physical equation $P(t) = R \cdot Q(t) + \frac{1}{C} \cdot V(t)$ as the basis for supervised learning ensured that predictions maintained physiological consistency, mitigating the risk of generating clinically implausible values. This hybrid approach between data-driven and physical modeling represents an innovative paradigm in the development of intelligent medical systems.

The cellular automaton, meanwhile, revealed emerging patterns of spatial pressure distribution that are critical to understanding the heterogeneity of pulmonary ventilation. The spontaneous formation of clusters, the propagation of pressure waves, and the persistence of heterogeneity observed in the simulation provide valuable insights into chaotic phenomena in pulmonary mechanics, offering a complementary perspective to the temporal predictions of the LSTM.

The implementation of an independent safety controller and risk mitigation protocols ensures that the system operates within clinically safe margins, complying with medical standards such as IEC 62304. The seven-layer architecture proved effective in managing the complexity of the system, from data ingestion to final action, validating the design proposed in previous workshops.

From a clinical perspective, this system has the potential to revolutionize mechanical ventilation by enabling real-time adjustments based on each patient's individual characteristics, reducing the risks of barotrauma, volutrauma, and atelectrauma associated with suboptimal settings. The ability to predict patient-specific adequate pressures represents a crucial step toward personalized medicine in intensive care.

A. Future Improvements

Based on the outcomes of both simulations and the proposed architecture in Workshop #2, several improvements can be made to further refine the system. A hybrid prediction approach—combining the LSTM's numerical forecasts with the cellular automaton's spatial pattern analysis—could establish a dual-layer safety mechanism capable of identifying hazardous pressure behaviors in advance. Incorporating a real-time feedback loop from the safety module back into the valve-control logic would enable dynamic adjustment of u_{in} and u_{out} based on emerging risks. Additionally, expanding the dataset with a wider range of resistance, compliance, and patient profiles would enhance model robustness and clinical

relevance. Extending the cellular automaton to track multiple physiological variables such as oxygenation, flow, or compliance variability would also yield more realistic lung-state representations. Finally, validating the entire framework through hardware-in-the-loop testing with an actual ventilator prototype or lung simulator would confirm the reliability and performance of both prediction and safety components in real operating conditions.

REFERENCES

[1] S. Diao, J. Wang, and Others, "Ventilator pressure prediction using recurrent neural network," *arXiv preprint*, 2024. [Online]. Available: https://arxiv.org/abs/2410.06552

[2] V. Authors, "Development of artificial neural networks for the prediction of the pressure," *Flow Measurement and Instrumentation*, 2024. [Online]. Available: https://www.academia.edu/126593545/Development_of_artificial_neural_networks_for_the_prediction_of_the_pressure

[3] A. of the Article, "A new method for pore pressure prediction on malfunctioning cells using artificial neural networks," *Water Resources Management*, vol. 35, pp. 979–992, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s11269-021-02763-0

[4] V. Researchers, "A review of finite element analysis and artificial neural networks," *Materials*, vol. 14, no. 20, p. 6135, 2021. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC8538846/

[5] I. P. Roberts, "Ieee bibliographies in latex," 2020. [Online]. Available: https://ianroberts.gitlab.io/notes/ieee_bibliography.html