**t**

**DA 221(M) COURSE PROJECT REPORT**

**Submitted by:-**

**Ankit Varshney** **Kovid Juneja**

**210121069 220101059**

**Comparing Physics-informed neural networks for modeling physiological time series for cuffless blood pressure estimation with CNN-LSTM based model**

**Literature review:-**

The advent of wearable physiological sensors has opened new avenues for precision medicine, particularly in monitoring cardiovascular health. Machine learning (ML) and deep learning (DL) techniques offer promising approaches to extract actionable insights from wearable time series data, facilitating personalized healthcare interventions. However, the efficacy of current ML and DL models heavily relies on large amounts of labeled ground truth data, which can be challenging to obtain due to the invasive or obtrusive nature of traditional measurement systems.

Addressing this challenge, there is a growing interest in leveraging domain knowledge and physics-based constraints to train deep neural networks (DNNs) more effectively. Physics-Informed Neural Networks (PINNs) emerge as a promising solution, integrating underlying physical principles into the learning process. PINNs have demonstrated success in various engineering domains but face challenges in cardiovascular applications due to inter-subject variations and dynamic physiological relationships.

In this literature review, the authors propose a novel framework for cuffless blood pressure (BP) estimation using PINNs with limited ground truth data. By approximating cardiovascular dynamics through Taylor's polynomials, the framework incorporates physiological features extracted from bioimpedance sensors into the learning process. This enables the optimization of neural network weights to minimize a loss function that integrates both standard supervised loss and physics-based constraints derived from Taylor's approximation.

The effectiveness of the proposed framework is demonstrated through a case study on continuous cuffless BP estimation from bioimpedance data. By leveraging PINNs, the models achieve high accuracy in BP prediction while significantly reducing the reliance on ground truth data compared to conventional ML models. The study involves a comprehensive evaluation on datasets from various BP elevation maneuvers, showcasing the robustness and scalability of the proposed approach.

Furthermore, the review provides insights into the architecture and training process of PINNs, emphasizing the integration of physiological features such as bioimpedance waveforms and hemodynamic parameters. Through meticulous feature selection and signal processing, the models capture relevant information for accurate BP estimation, laying the foundation for future advancements in wearable healthcare technologies.

Overall, the literature review highlights the potential of physics-informed approaches in overcoming data scarcity challenges in precision medicine. By bridging the gap between domain knowledge and machine learning, PINNs offer a promising pathway towards personalized and non-invasive monitoring of cardiovascular health, paving the way for transformative advancements in wearable healthcare technologies.

**Mathematical Foundations of Physics-Informed Neural Networks (PINNs) for Cuffless Blood Pressure Estimation**

**1. Neural Network Architecture**

**Let *f*<sub>NN</sub>(*x*<sub>^\*,</sub> *u*<sub>^\*,</sub> *Θ*) represent the output of a neural network with weights *Θ*. Here:**

* ***x*<sub>^\*,</sub> denotes the input time series bioimpedance data segmented based on cardiac cycles.**
* ***u*<sub>^\*,</sub> represents physiological features derived from the bioimpedance data.**

**2. Taylor's Approximation**

**The relationship between physiological features *u*<sub>^\*,</sub> and blood pressure (BP) is approximated using a Taylor's polynomial expansion:**

**Equation:**

**P\_i(*x*^\*, *u*^\*, *Θ*) = ∑\*\*(j=0)\**^N (1/j!) (∂^j f\_NN / ∂ u^{j}) (x^, u^*, Θ)**

**where:**

* ***P*<sub>i</sub>(*x*^\*, *u*^\*, *Θ*) is the *i*-th term of the Taylor polynomial expansion.**
* ***N* is the order of approximation.**
* **( \frac{\partial^j f\_{NN}}{\partial u^{*j}} ) represents the j-th partial derivative of the neural network output f<sub>NN</sub> with respect to physiological features u<sub>^*.</sub>**

**3. Physics-Based Loss Function**

**The physics-informed loss function, *L*<sub>physics</sub>, incorporates the difference between the neural network predictions and Taylor's approximation:**

**Equation:**

**For displaying in MS Word, you can use the Equation Editor or copy-paste the equation from a Math Equation Editor software.**

**L\_{physics} = ∑\*\*(i=1)*^M (f\_NN(x\_i^, u\_i^*, Θ) - P\_i(x\_i^\*, u\_i^\*, Θ))^2**

**where:**

* ***M* is the number of data points.**
* ***x\_i^* and *u\_i^* are the input bioimpedance data and physiological features for the *i*-th data point, respectively.**

**4. Modified Loss Function**

**The total loss function for training the PINN model is a combination of the conventional supervised loss *L*<sub>conventional</sub> and the physics-based loss *L*<sub>physics</sub>:**

**Equation:**

**For displaying in MS Word, you can use the Equation Editor or copy-paste the equation from a Math Equation Editor software.**

**L\_{total} = α L\_{conventional} + β L\_{physics}**

**where *α* and *β* are weighting factors.**

**5. Optimization**

**The PINN model is trained using gradient-based optimization techniques such as stochastic gradient descent (SGD) or Adam optimizer. The weights *Θ* of the neural network are updated iteratively to minimize the total loss *L*<sub>total</sub>.**

**By incorporating physics-based constraints into the training process, PINNs enable the neural network to learn from both labeled and unlabeled data, resulting in accurate predictions of blood pressure parameters while reducing the reliance on large amounts of ground truth data.**

**Methodology: Physics-Informed Neural Networks for Cuffless Blood Pressure Estimation**

**1. Data Collection and Preprocessing**

* Wearable bioimpedance sensors are placed on participants' wrists or fingers to capture physiological data.
* Bioimpedance waveforms are segmented based on cardiac cycles, and relevant physiological features (e.g., amplitude change, pulse wave velocity, heart rate) are extracted from the segmented data.

**2. Taylor's Approximation**

* The relationship between physiological features extracted from bioimpedance data and blood pressure (BP) is approximated using Taylor's polynomial expansion.
* Partial derivatives of the neural network output with respect to physiological features are calculated using auto-differentiation.

**3. Physics-Informed Loss Function**

* A physics-based loss function is defined based on the difference between the neural network predictions and Taylor's approximation.
* The loss function incorporates both supervised loss and physics-based constraints.

**4. Model Training**

* A modified loss function, combining conventional supervised loss and physics-based loss, is used for training the Physics-Informed Neural Network (PINN) model.
* Gradient-based optimization techniques such as stochastic gradient descent or Adam optimizer are employed to minimize the total loss function.

**5. Model Evaluation**

* The trained PINN model is evaluated on test datasets to assess its accuracy in cuffless BP estimation.
* Performance metrics such as mean error (ME) and standard deviation error (SDE) are calculated for systolic BP (SBP), diastolic BP (DBP), and pulse pressure (PP).

**6. Comparison with Conventional Models**

* The performance of the PINN model is compared with conventional machine learning models trained using larger amounts of labeled data (e.g., a CNN-based model).
* The reduction in the required amount of ground truth data for training the PINN model is quantified.

**7. Case Study and Validation**

* The effectiveness of the proposed approach is demonstrated through a case study involving participants undergoing BP elevation maneuvers.
* The PINN model's accuracy is validated across various BP ranges and categories defined by clinical guidelines.

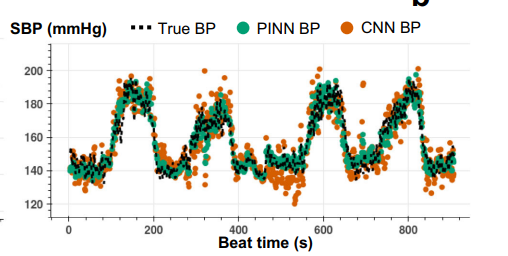
**8. Additional Proof-of-Concept Study**

* A proof-of-concept study is conducted to evaluate the PINN model's performance under varying amounts of training ground truth data.
* Consistency in the approximation of input-output relationships is assessed across different training scenarios.

This formatted text can be easily pasted into MS Word and will retain its structure with headings and bullet points.

**Comparative Results:-**

The authors made a comparision between CNN based model and their Physics informed neural network model. The results are as follows:-

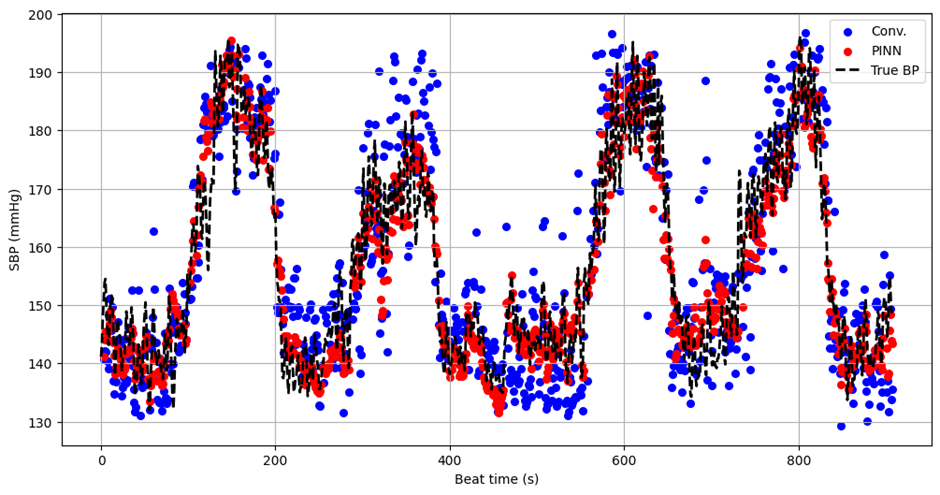


**The pearson corr coefficient for CNN based model is: 0.73**

**While with PINN it is 0.9**

**In our case, we trained a CNN-LSTM based model which gave a Corr coefficient of 0.89**

Which is much better than just CNN model but we were not able to beat the PINN model because of insufficient data as PINN have an advantage on small datasets.



(Our results where blue dots represent CNN-LSTM results and red one represent PINN results)

**Performance metrics:-**

1. Pearson correlation coefficient:-

The Pearson correlation coefficient (PCC), denoted by the symbol "r", is a statistical measure used to quantify the linear correlation between two variables. It ranges from -1 to +1, with the following interpretations:

* **+1:** Perfect positive correlation - As the value of one variable increases, the value of the other variable also increases proportionally.
* **0:** No correlation - There is no linear relationship between the two variables. Changes in one variable are not associated with predictable changes in the other.
* **-1:** Perfect negative correlation - As the value of one variable increases, the value of the other variable decreases proportionally.

Here's a breakdown of some key aspects of the Pearson correlation coefficient:

**What it Measures:**

* **Linear Relationship:** The PCC specifically measures the **linear** correlation between two variables. It does not capture non-linear relationships, even if a strong connection exists.

**Formula and Interpretation:**

* The PCC is calculated based on the covariance of the two variables and their standard deviations. It essentially standardizes the covariance to a range between -1 and +1.
* A positive PCC value indicates a positive linear relationship, while a negative PCC value indicates a negative linear relationship. The closer the value is to +1 or -1, the stronger the linear correlation.
* A PCC of 0 indicates no linear correlation between the variables. However, it's important to note that this doesn't necessarily imply there's no relationship at all, just that it's not linear.

1. **Mean squared error:-**

Mean squared error (MSE) is a common metric used to evaluate the performance of an estimator (a model or function that predicts a value). It measures the average squared difference between the predicted values (estimates) and the actual values.

Here's a breakdown of its key aspects:

**What it Measures:**

* **Error:** MSE focuses on the squared difference between predicted and actual values. Squaring the differences achieves two things:
  + Eliminates negative signs: Absolute values wouldn't tell you if an overestimation or underestimation occurred. Squaring ensures all errors contribute positively.
  + Weights larger errors more heavily: Larger differences are squared, giving them a more significant impact on the overall MSE compared to smaller errors.

**Interpretation:**

* **Lower MSE indicates better performance:** A model with a lower MSE generally produces predictions closer to the actual values on average. Ideally, you want the MSE to be as close to zero as possible.
* **Units:** MSE inherits the units of the squared variable being estimated. For example, if you're predicting house prices, MSE would be in squared units of currency (e.g., squared dollars).

**Applications:**

* **Regression Analysis:** Widely used to evaluate the performance of regression models that predict continuous values.
* **Machine Learning:** A common loss function used to train various machine learning models by minimizing the average squared error between predictions and ground truth.