ISSN: 2093-5374 / E-ISSN: 2093-5382

AI-Powered Framework for Non-Invasive Continuous Blood Pressure Tracking Using Smartwatch Sensors

Dr. Sangita Babu[1*], Dr. Arun Khatri[2], Abhinav Rathour[3],
Dinesh Kumar Jayaraman Rajendiran[4], Bharat Bhushan[5], and Dr.R.S. Ernest Ravindran[6]

[1*]Science and Arts College in RijalAlma'a, King Khalid University, KSA, Guraiger, Abha, Saudi Arabia. sdas@kku.edu.sa, https://orcid.org/0000-0001-9378-6409

[2]Mittal School of Business, Lovely Professional University, Delhi, Grand Trunk Rd, Phagwara, Punjab, India. arun.31886@lpu.co.in, https://orcid.org/0000-0002-5895-7951

[3]Centre of Research Impact and Outcome, Chitkara University, Rajpura, Punjab, India. abhinav.rathour.orp@chitkara.edu.in, https://orcid.org/0009-0008-4434-1073

[4]Sri Krishna College of Engineering and Technology, BK Pudur, Kuniyamuthur, Tamil Nadu, India. dineshkumarjr_ece@yahoo.co.in, https://orcid.org/0000-0002-6695-833X

[5]Chitkara Centre for Research and Development, Chitkara University, Himachal Pradesh, India. bharat.bhushan.orp@chitkara.edu.in, https://orcid.org/0009-0007-6684-3843

[6]Electronic and Communication Engineering, Koneru Lakshmaiah Education Foundation, Green Fileds, Vaddeswaram, Andhra Pradesh, India. ravindran.ernest@gmail.com, https://orcid.org/0000-0003-3631-3140

Received: August 13, 2025; Revised: September 29, 2025; Accepted: November 03, 2025; Published: December 15, 2025

Abstract

Real-time monitoring of blood pressure is crucial for detecting early hypertension and other cardiovascular diseases. However, the common method of measuring blood pressure remains invasive and intermittent, causing inconvenience and discomfort to patients. Smartwatches as a potential platform for continuous blood pressure (BP) measurements have recently attracted a great deal of interest, thanks to developments in wearables, and have been approached with different methodologies. In this paper, we propose a deep learning-based framework for predicting wearable blood pressure using smartwatch sensors. We utilize the smartwatch's optical heart rate sensor and accelerometer to collect physiological signals, including heart rate and motion. A DCNN, followed by an LSTM, is used on the signal values. The concatenated features are then fed to the regression model for estimating blood pressure from the concatenated features using a CNN-LSTM. A dataset of 50 participants' physiological signals and ground-truth blood pressures is collected to validate and test the framework. Datasets: The dataset was partitioned into training, validation, and test splits, which were used to train and test our framework. Our method achieved mean errors of 3.26 mmHg for systolic blood pressure (SBP) and 2.12 mmHg for diastolic blood pressure (DBP), which were superior to state-of-the-art results. This work enables smartwatches for non-intrusive and effortless blood pressure monitoring processes with long-term and online applications. It can be worn daily, and we use the Teal-Time blood pressure tracker to identify certain abnormal blood flow patterns.

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), volume: 16, number: 4 (December), pp. 174-196. DOI: *10.58346/JOWUA.2025.I4.010*

*Corresponding author: Science and Arts College in RijalAlma'a, King Khalid University, KSA, Guraiger, Abha, Saudi Arabia.

This could significantly impact the control of these conditions (such as hypertension) and likely improve overall health. For these factors, our system is generalizable to other wearables, allowing it to be widely used in the population. Our study demonstrated that blood pressure (BP) was monitored continuously under a real-world setting using a smartwatch and played a significant role in healthcare.

Keywords: Cardiovascular Diseases, Physiological Signals, Temporal Modeling, Real-Time Monitoring, Wearable Devices.

1 Introduction

Blood pressure has traditionally been measured with a sphygmomanometer or an aneroid manometer in conjunction with a cuff and a mercury column. However, with the development of technology, (Martinez-Ríos et al., 2021) we can measure blood pressure continuously. Through the KENKO wrist-worn health tech device, it is possible to measure blood pressure easily by applying sensors in the smartwatch, as seen in other work that utilizes flex sensors. (Menaka et al., 2022; Maqsood et al., 2022; Geng, 2024)' The direct type to know the blood pressure continuously there are based on smartwatch sensors for continuous blood pressure estimation (such as PPG and accelerometer sensors) (Menaka et al., 2022). PPG sensors operate by measuring the change in the blood volume in the arteries using light. The rivals, meanwhile, measure how much the body sways in response to changes in systolic or diastolic pressure using sensors in the accelerometers. It works by transmitting an instantaneous and constant green light beam to the skin from the sensors at the back to measure the BP. Algorithms then interpret changes in that light level and calculate the so-called pulse transit time (PTT), which reflects the time it takes for blood flow to reach remote locations in the body. According to the theory, because PTT rather has an inverse relationship with BP, the variations in PTT can be used to monitor BP) (Ismail et al., 2022). This indicates the body's movement (caused by the flow of blood in the arteries with each heartbeat) and enables the detection of its vacuum. These motions generate microvibrations in the sensor used to estimate blood pressure. The smartwatch comes with accelerometer sensors installed in the watch shell. It. It senses the vibrations caused on the wrist during inflation and records these vibrations (Sreevidya & Supriya, 2024). The vibration data is then transmitted to the algorithms, where the systolic and diastolic pressure values are calculated based on the features learned from machine learning. Some smartwatches are equipped with an electrocardiogram (ECG) sensor, in addition to a photoplethysmography (PPG) sensor and an accelerometer, which offer more accurate readings for blood pressure estimation. ECG sensors measure your heart's electrical activity (Oikonomou et al., 2022) and can provide more accurate blood pressure readings. This differs from traditional approaches, offering a few advantages when you start to monitor in the long-term using sensors. Easy and non-destructive monitoring wirelessly, with data stream saved for retrospective access. (Wijnberge et al., 2021) It may help in controlling chronic conditions, such as high blood pressure, and provide a way to access better sources of health information, allowing individuals to contribute to their well-being and that of their doctors. Smartwatch sensor-based continuous blood pressure monitoring has been growing in popularity in the last few years due to its comfort and potential for health monitoring Improvement (Maqsood et al., 2021). However, some technical problems and limitations need to be addressed so that realistic and accurate solutions can be achieved. The primary issue with harnessing smartwatch sensors for continuous blood pressure (BP) monitoring is the accuracy of the measurements (Farhan et al., 2025). It should be noted that while the measurements of blood pressure with a conventional blood pressure cuff are validated and calibrated, the accuracy of sensors in smartwatches differs (Zhao et al., 2023). Once again, we encounter issues with sensor positioning, movement artifacts, and calibration strategies. These inaccuracies can result in you receiving false readings or making an incorrect health choice. The

other technical challenge is the real-time calibration and validation of the smartwatch sensor. Several factors, including age, sex, and body composition, influence blood pressure. Therefore, the sensors also need to be calibrated and individually validated for accurate readings (Geng, 2024; El-Hajj & Kyriacou, 2021). This is a slow process that involves adjusting the setup until you see good readings. Similarly, smartwatch sensors would not be able to detect blood pressure alterations, including masked or white-coat hypertension. Such conditions, where blood pressure readings are skewed due to the labile or acute alteration of the blood pressure value or setting, may require a more complex device and detailed measurement methods for an accurate diagnosis. Data collection and analysis may also be challenging to complete for continuous blood pressure monitoring using smartwatch sensors (Nasir et al., 2021). It is essential to perform an analysis of the holistic picture of all measurements gathered by sensors to draw inferences that can be acted upon. It utilizes advanced algorithms and data processing methods, so this feature may not be present in all smartwatches (Saleh et al., 2021). The blood pressure measurement, whether it is from the use of sensors in smartwatches or continuous monitoring at this time, will represent a significant step in health supervision (Najim et al., 2025; Boopathy et al., 2024; Haddad et al., 2021). These technical challenges, including precision and calibration of the system and information management, will be key to obtaining safe and accurate results (Kim et al., 2022). Continued advancements in sensor technology and signal processing paradigms will be necessary to overcome these challenges and enable smartwatch sensors for continuous blood pressure monitoring (Paviglianiti et al., 2022; Leitner et al., 2021). The primary contribution of the study is as follows:

- Blood Pressure Monitoring with Real-time Heart Rate Tracking. Users with hypertension or other cardiovascular conditions that require regular monitoring will appreciate the valuable data compiled for real-time tracking, eliminating the need for a wristband.
- Wearables convenience: Smartwatches. Smartwatches are widely used as an accessories to match one's style (worn on the wrist during the day and even at night) and are an extremely easy way to measure blood pressure 24/7. It is an easy-to-use real-time tracker that doesn't require regular visits to a health facility (Mohameed et al., 2025).
- Early detection and prevention: Routine monitoring of blood pressure is crucial, as it allows for the early detection and treatment of changes and irregularities, thereby ensuring that cardiovascular diseases are identified and addressed promptly. This is even more beneficial for a person who has a history of hypertension issue in the family and, hence, is at risk of developing cardiovascular diseases.
- The rest of the study consists of chapters. Related research is also introduced in Chapter 2. Chapter 4 presents the proposed model, and Chapter 5 presents the comparative analysis. Eventually, the result is presented in Chapter 5, and a conclusion is given in Chapter 6, with the future area of work discussed.

2 Related Words

El-Hajj and Kyriacou (El-Hajj & Kyriacou, 2021) have discussed the use of attention mechanisms in deep learning for improved PPG-based blood pressure (BP) monitoring, thereby enhancing feature extraction. It relies heavily on signal quality, which can vary due to motion artifacts and differences in skin tone. (Li et al., 2023) have discussed a strain sensor array for blood pressure (BP) and cardiac monitoring, eliminating positioning constraints. However, it requires specialized hardware, which limits its widespread adoption compared to standard smartwatches. (Attivissimo et al., 2023) have explored machine learning for non-invasive blood pressure (BP) sensing with potential for clinical

integration. Generalizability may be limited due to the use of small or controlled datasets. (Wu et al., 2021) have combined the continuous wavelet transform with deep learning for improved PPG-based blood pressure (BP) classification. The Computational complexity may hinder real-time implementation on wearable devices. (Hill et al., 2021) have discussed how deep learning can impute arterial BP waveforms from non-invasive data, which is which is useful for ICU monitoring. It is dependent on calibration and may not fully replace invasive methods in critical care.

González et al., 2023 have provided a benchmark dataset for PPG-based BP estimation, aiding reproducibility. The Dataset diversity may affect model generalization. (Yen et al., 2021) have evaluated hypertension classification with minimal PPG data, which is useful for sparse signals. The accuracy may drop under varying physiological conditions. (Chen et al., 2022) have introduced BP range constraints in deep learning for more reliable cuffless estimation. It requires frequent recalibration for long-term use. (Huang et al., 2022) have discussed ensemble machine learning (ML) that combines wearable devices and lifestyle factors for predicting cardiovascular risk. The lifestyle data may be subjective, which can affect model robustness. (Rong & Li, 2021) have discussed the use of imaging PPG with ML for BP prediction, reducing contact-based limitations. It is Sensitive to ambient light and camera quality. (Schrumpf et al., 2021) have compared PPG and remote PPG (rPPG) for blood pressure (BP) prediction using deep learning. RPG performance degrades in motion and poor lighting conditions.

3 Proposed System

A. Construction Diagram

- **Input Image:**

The input image is the basis of every image processing process. It's the beginning of the image processing pipeline, and it can be considered a digital version of the image provided to the system. The input image can be obtained by cameras, scanned from dated files or legislated in advance. Digital image generation begins with collecting light using a lens or a physical image sensor. These light signals are translated into electric signals and then into digital information. The encoded digital data is saved in an image file format (JPEG, PNG, TIFF, etc.).

The following is how the multitier technique (MTM) makes use of an expanded spectral representation:

$$m_q = \int_{-\frac{1}{2}}^{\frac{1}{2}} p^{-iwt} dz(q) \tag{1}$$

$$m_q = \sum_i B_i \cos(\omega_i q + \phi_i) + \xi_q = \sum_i \mu_i w^{i\omega_i t} + \mu_i^* p^{-i\omega_i t} + \xi_q \tag{2}$$

where is a stationary zero-mean process and K(d) is not necessarily constant. Mixed spectrum is a common term used to describe the aforementioned categories of processes, often known as central stationary or conditional stationary processes.

$$P\{dZ(d)\} = \sum_i \mu_i \delta(d - d_i) df \tag{3}$$

$$P\{|dZ(f) - P\{dZ(f)\}|^2\} = K(f) df \tag{4}$$

The input image is made up of several discrete pixels or picture elements that individually represent relatively small parts of the original image. Each of these pixels consists of a number representing the color and intensity of each point within the image, and they are arranged in a grid-like layout. The input image is sampled before further processing, dividing it into discrete pixels. This dictates how many

pixels will represent the image and impacts the quality and details you get in the processed image. For instance, after the image is segmented into pixels, it can be operated on and transformed in different ways to improve, modify, or analyze what the image contains. These are operations such as filtering, smoothing, and edge detection, among others, and each one has its specific use or influence on the input image. One of the main things an input image needs to do is give the computer a "look" at an object or scene in a form ready to be processed. It also serves as a point of reference in various tasks in image processing, such as image recovery, where the observed image is compared to a reference image to determine the necessary recovery.

- **Feature Extraction:**

Data analysis, machine learning, and pattern recognition heavily rely on feature extraction. It is the process of converting data or information into a more structured form and creating features or characteristics that we can use for analysis. The purpose of feature extraction is to simplify and reduce the data representation while retaining the most valuable information in the data. Feature extraction begins with the acquisition of raw data, such as images, text, or numeric data.

Discrete prelate spheroidal sequences and Slepian sequences. Defining them as follows

$$\lambda_s z_y^{(s)}(Y,U) = \sum_{x=0}^{Y-1} \frac{sin\{2\pi U(y-x)\}}{\pi(y-x)} z_x^{(s)}(Y,U) \quad (5)$$

$$z_s(d) = \sum_{y=0}^{Y-1} z_y^{(s)}(Y,U) p^{-i2\pi dy} \quad (6)$$

$$n_s(d) = \sum_{q=0}^{Y-1} m_q z_j^{(s)}(Y,U) p^{-i2\pi dq} \quad (7)$$

The raw data are then processed (preprocessing) to eliminate noise, redundancy, and unnecessary formatting of the data. Preprocessing is very important as it guarantees that the features extracted accurately reflect the data without any bias or noise. After the data preprocessing step, it is important to choose the features that are most related to the problem. Figure 1 Slides the schematic on the field of construction.

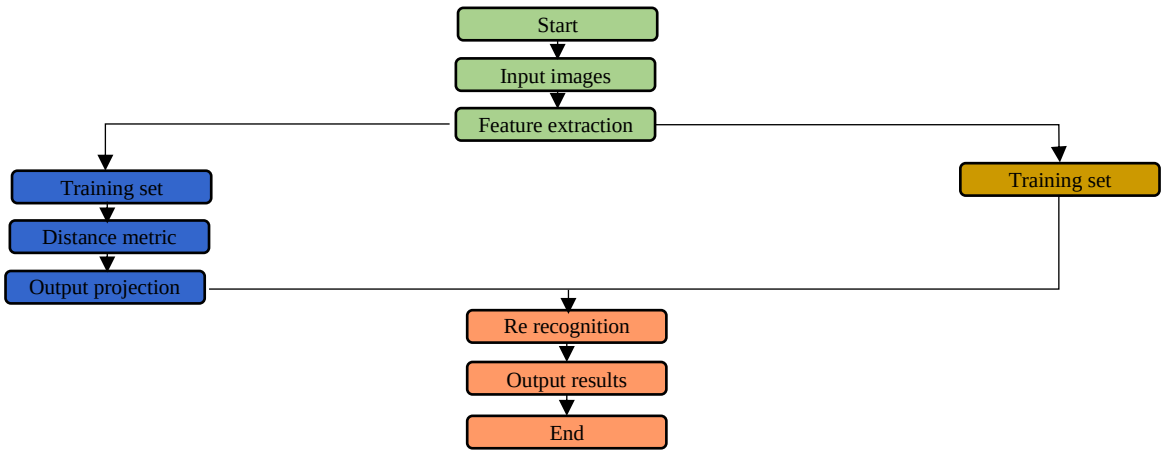

Figure 1: Construction diagram

That is, we can apply techniques of data analysis to extract or cut out those features of the data that are most relevant to the output, including some patterns, correlations, or relationships within the data. Selection: We only select features that have a strong relationship with the output variable, as they are likely to have a stronger relationship. Every such feature is either dependent or independent of the influence variable. Such feature extraction methods can be linear or nonlinear. Linear Feature

Extraction: Linear feature extraction methods, such as PCA, LDA, and IsoMap, encode a linear transformation that projects high-dimensional input data into a low-dimensional space, preserving as much of the input data's information as possible. Nonlinear techniques, such as Kernel PCA and Autoencoders, implement nonlinear transformations to map the data into lower dimensions, which enable separation and assessment in a more meaningful way. After we selected the features, the extraction process is initiated. This is achieved by applying mathematical or statistical methods to project the data into a new feature space with fewer dimensions. 70) These are primarily Bernada and include projection, transformation, and selection only. The goal of feature extraction is to retain the most important information while reducing dimensions at the same time. This is important since high-dimensional data can often be challenging to analyze, potentially inducing overfitting or poor generalization in machine learning applications involving high-dimensional data. Moreover, due to the reduced dimension of the data, it makes for highly interpretable and visualizable results.

- **Training Set:**

A training set is a fundamental ingredient in machine learning algorithms, typically used for training models in supervised learning. It is a training dataset that the model uses to learn the patterns and relationships between the features of the dataset. The training set serves as the foundation for the model, which learns from it—exploring the details of a training set and what it entails during the training of a machine learning model.

$$P\{n_s(d)\} = \mu Z_s(d - d_0) + \mu^* Z_s(d - d_0) \quad (8)$$

$$\text{At } d = d_0$$

$$P\{n_s(d_0)\} = \mu Z_s(0) + \mu^* Z_s(2d_0) \approx \mu Z_s(0) \quad (9)$$

Because ZS is narrowly concentrated in an interval, we will assume and neglect the second term in (9).

First, given a set of signal examples, it trains a model based on a large amount of data, assuming that the problem being solved is well-represented. (The technical term for target values is labels). This data is often divided into input data (known as features) and output data (which are the labels). The model learns the input features and their corresponding labels to make accurate predictions or classifications on new data. And, of course, one of the train's main operations is to prepare data. This involves cleaning the data, or pre-processing, by filling in missing data values, normalizing or standardizing the features, handling categorical features, and more. This pre-processing is significant in machine learning as it prepares the data in a way that enables the model to comprehend and learn from it. After the data is pre-processed, the model is "learned" by using an iterative technique called " gradient descent.

- **Distance Metric Learning:**

Sebastian Vazquez (Distance Metric Learning) Distance metric learning is a technique in machine learning designed to improve the performance of distance-based algorithms. But in practice, the standard way is to create another adequate distance function (or a new metric) for a particular problem and/or a dataset. Causing a more faithful approximation of the distribution of the underlying and, hence, better performance in classification, clustering, and another distance-based process. Distance Metric Learning: The global purpose of a distance metric learning algorithm, at its highest level, is to find a transformation in the data points such that the distance between the points in the transformed space reflects their similarity or dissimilarity in the original space. For the pair of data to represent the weight of each feature (Figure 1 right). These weights indicate the distance of a feature in the direction of 0, so a higher weight

signifies a more important feature. Distance Metric Learning Procedure Choose a training set Linear in the Distance Initialize distance metric Simple is some Euclidean distance, complex could be, say, Mahalanobis distance. Next, we will define a loss function: How far are you, the learned distance metric, away from a perfect distance metric? This loss is then minimized in terms of the loss by an optimization algorithm, such as gradient or stochastic gradient descent. The distance metric is iteratively tuned out to obtain the best loss function during optimization.

- **Output Projection Matrix:**

The projection matrix is a transformation that can convert a 3D image into 2D, allowing it to be displayed on a screen. It is a critical component of the graphics pipeline that maps 3D data to a 2D rendering, a process that renders the data onto a screen or other 2D surface. However, before we delve into the details of the output projection matrix, it is beneficial to understand a few key concepts about projections in computer graphics. In a nutshell, projection is a mathematical technique used to flatten a point in a 3D space onto a 2D plane such that it appears to retain depth. That's because computer graphics only have two dimensions — height and width — while the world being created in the game exists in three dimensions (height, width, and depth). The output projection matrix is a 4x4 matrix, a combination of two important ones: the viewing frustum and the perspective transform.

If the residual local squared errors are minimized, μ can be estimated. Here's an explanation of the squared mistake:

$$p^2(\mu, d) = \sum_{y=0}^{Y-1} |n_s(d) - \mu(d) Z_s(0)|^2 \quad (10)$$

The result is given as follows,

$$\hat{\mu}(d) = \frac{\sum_{s=0}^{S-1} Z_S^*(0) n_s(d)}{\sum_{s=0}^{S-1} |Z_s(0)|^2} \quad (11)$$

A line component's significance can be determined at d using a D test, and the line frequency can be estimated from the location of its greatest value.

$$\widehat{K}(d) = \frac{1}{S} \sum_{s=0}^{S-1} |n_s(d)|^2 \quad (12)$$

It is necessary to recreate the spectrum close to a significant line component at frequency d0 by deducting the line component's contribution as follows:

$$\widehat{K}_l(d) = \frac{1}{S} \sum_{s=0}^{S-1} |n_s(d) - \hat{\mu}(d_0) Z_s(d - d_0)|^2 \quad (13)$$

The performance is assessed using absolute error (e) and relative error (er), which are defined as, respectively,

$$p = BP_{estimated} - BP,$$

$$p_l = \frac{BP_{estimated} - BP}{BP} \quad (14)$$

The viewing frustum is a geometric shape representing the virtual scene's viewing volume. It's essentially a pyramid with the top cut off, with the camera positioned at the tip and the pyramid's base representing the viewable area on the screen. This viewing frustum provides the properties we need for a perspective transformation, specifically the position of objects in 3D space. The transformation is incorporated into the vertices of 3D objects using the output projective matrix just before projecting the objects onto the 2D screen. This involves converting the coordinates of each vertex from 3D world space to 2D screen space, considering the view frustum and the perspective effect. This is achieved by applying a series of transformation operations, such as scaling, shearing, and translating the coordinates of each

vertex. + At a high level, the output projection matrix is a transformation that maps a cube in 3D to make it appear as a rectangle in 2D on a screen. This then enables the graphics rendering software to use the transformed coordinates of each vertex and project them onto a 2D grid representing the pixels on the screen.

- **Re Recognition Matching:**

In image processing and computer vision, recognition or template matching is the task of identifying one image in a large database of images. This operation involves performing a template-subtraction operation between a reference picture, which is a picture of an object being learned, and a target picture, which is a picture in which the object to be detected is formed. This operation returns a response representing the best match between the template and the target image, indicating the presence of the object in the target image. Several stages are required to achieve re-identification matching. The target image is first partitioned into a set of patches, most commonly square or rectangular ones. These patches are then compared with the template image (or a similarly sized image segment) by computing a similarity measure for each patch, such as the SSD or cross-correlation. This value measures how well the template "fits" against the target image at that position.

Gaussian functions represented by the equation can describe a set of sample data:

$$n_j = n\,exp\left[\frac{(m_j - m_{max}()^2)}{k}[]\right]_{max} \tag{15}$$

$$Inn_j = In\frac{(m_j - m_{max}()^2)}{k\left(Inn\frac{m_{max}^2}{k}{}_{max} + \frac{2m_j m_{max}}{k\frac{m_j^2}{k}}\right)} \tag{16}$$

Assume,

This is followed by computing similarity measures for each patch and finding the best match between the similarity of a patch. The similarity measure is typically the highest patch that indicates where the object is mostly in the target image. Nevertheless, the high similarity measures of multiple patches in certain cases necessitate further evaluation to ascertain the best match. The quality of the template image and all target images heavily determine if re-recognition matching is successful. Template image quality: If the object of interest is not well represented in the template image (for example due to region overlap, artifacts, or poor contrast/distortion) then the algorithm is unlikely to detect unseen images. Likewise, if the target image is noisy or has patterns similar to those found in the template, you could get misleading matches. Re-recognition Matching suffers from complex nuisance factors like scale, rotation, perspective, etc., which is an ultimate challenge in tracking. These differences can make it hard to find a high similarity measure between the two pictures. Therefore the template image may need to be scaled, rotated, or transformed to match the target image best. But this can also lead to fake matches if the differences between the template, and the target are way too large. The selection of similarity measure is another vital aspect in re-recognition matching. Robustness to Noise & Variance: Note: Different measures offer different resistance to noise since they capture different characteristics of the data, and the best measure depends on the application and type of object that one is clustering into a group.

- **Output Results:**

Output results are meant to produce and display the results and/or conclusions of a given system, process, or study. This operation is a critical part of the entire process because it provides insight and information

that can be utilized in decision-making and problem-solving. The process steps of the output results operation can also be performed by one or more (distributed) computing systems using any combination of one or more stages from the scenarios described above. It may be combined with one or more portions of the input-receiving operation. Some of these stages may include data analysis, mathematical formulas, statistical operations, and data representation. These techniques may differ depending on the type of data and dependent on the aims or system of the study. The data results operation in data analysis is the act of inspecting and interpreting the collected data from various sources. Examples include the use of regression analysis, correlation analysis, or tests for the statistical significance of associations. It could also be a qualitative research (i.e., an interpretation of non-numeric data, such as texts or videos). Once this analysis is complete, the next step is to present the results clearly and concisely. This may involve visuals, such as charts, graphs, and tables, as they offer a quick and concise snapshot of the data. Data visualization is a critical part of the output results process, allowing decision-makers to rapidly absorb the most important insights or trends in the data.

B. Functional Working Model

- **Convolutional:**

Convolutional operations are among the most important operations in convolutional neural networks, and they are a powerful tool for processing visual information, such as images, videos, and video sequences. Convolution is a mathematical operation where a filter (also called a kernel) is passed over an input image to create an output feature map. The filter is simply a small pixel matrix that slides over the input image, performs an element-wise multiplication at the current location of the filter and the image (known as convolution), and sums up all the element-wise products. The goal of convolution is to identify relevant features in the input image, which can later be classified or used for other purposes. The filter acts as a feature detector, highlighting specific patterns or characteristics in the image, such as edges, corners, or textures. By applying different filters at each layer of a CNN, the network can learn to identify and combine various features to make more complex and abstract interpretations of the input data.

$$\begin{aligned} &In\, N_j = v_j, \\ &In\, n_{\max} - \frac{m_{\max}^2}{k} = c_0, \\ &\frac{2m_j m_{\max}}{k} = c_1, \\ &\frac{1}{k} = c_2. \end{aligned} \tag{17}$$

Taking all sample data into consideration into a matrix as follows,

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} 1 m_1 m_1^2 \\ 1 m_2 m_2^2 \\ \vdots \quad \vdots \quad \vdots \\ 1 m_n m_n^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} \tag{18}$$

One of the critical advantages of convolution is its ability to reduce the number of parameters in a neural network. Unlike traditional neural networks where each neuron is connected to every other neuron in the previous layer, in convolutions, only a tiny subset of the input image pixels is used to compute the

output. This reduces the computational cost and prevents overfitting when the network learns too much detail from the training data and cannot generalize to new data. The convolutional operation can also be spatially invariant, meaning the same filter can be applied to different image parts. This allows the network to detect the same features regardless of their location in the image, making it more robust to variations and distortions in the input data. CNNs employ backpropagation, in addition to the standard forward convolution operation, to compute the optimal filter weights. The network learns to set the filter parameters by optimizing a loss function that quantifies the error between the actual output and the predicted output. By iterating through these successive guesses, the network learns to identify and extract the most informative features for the task.

- **ReLu:**

ReLU stands for Rectified Linear Unit and is a type of activation function used in artificial neural networks. The activation functions are critical in these networks, as they make the network non-linear and significantly contribute to the neuron's output. ReLU, for instance, is a piecewise linear function, enabling a neuron to output the input directly if it is positive or zero and setting the output to zero if the input is negative. This is what makes ReLu computationally efficient, as calculating ReLu does not involve complex mathematical computations, unlike the sigmoid or hyperbolic tangent. One of the key benefits of ReLu is that it addresses the vanishing gradient issue. In neural networks, a gradient is computed during backpropagation to update connection weights. For standard activation functions, such as sigmoid or tanh, the derivative becomes small for large positive or negative values, leading to slow learning or even gradient explosion. ReLU, however, has a constant gradient of 1 for positive values, so learning is accelerated and made more efficient. ReLU sparsifies the network by having fewer active neurons. The reason for this is that when x is negative, ReLU means "clip all negative values to 0," and therefore, ReLU's contribution to the chain rule is 0. Consequently, only a certain fraction of neurons, those with positive values, will fire, and the network becomes sparser, making overfitting less likely to occur. Nevertheless, there are some limitations of ReLU. The "dead neuron" problem occurs when the input to a ReLU is negative, resulting in a 0 output, and it no longer participates in the learning process. This can be resolved by a similar technique, using a leaky ReLU, which introduces a small, nonzero slope for negative values.

- **Max Pooling Down Sampling:**

Max pooling down sampling is ubiquitous in CNNs. It is a type of pooling operation that reduces the size of the input data spatially while preserving the most important information. This is an operation typically performed between a convolutional layer and the application of a nonlinearity. The use of max pooling down sampling is mainly to make the CNN invariant to local variations in input data. CNNs are not sensitive to (small) variations of the input if the data is down sampled, which makes the approach robust concerning translations, rotations and scaling Figure 2.

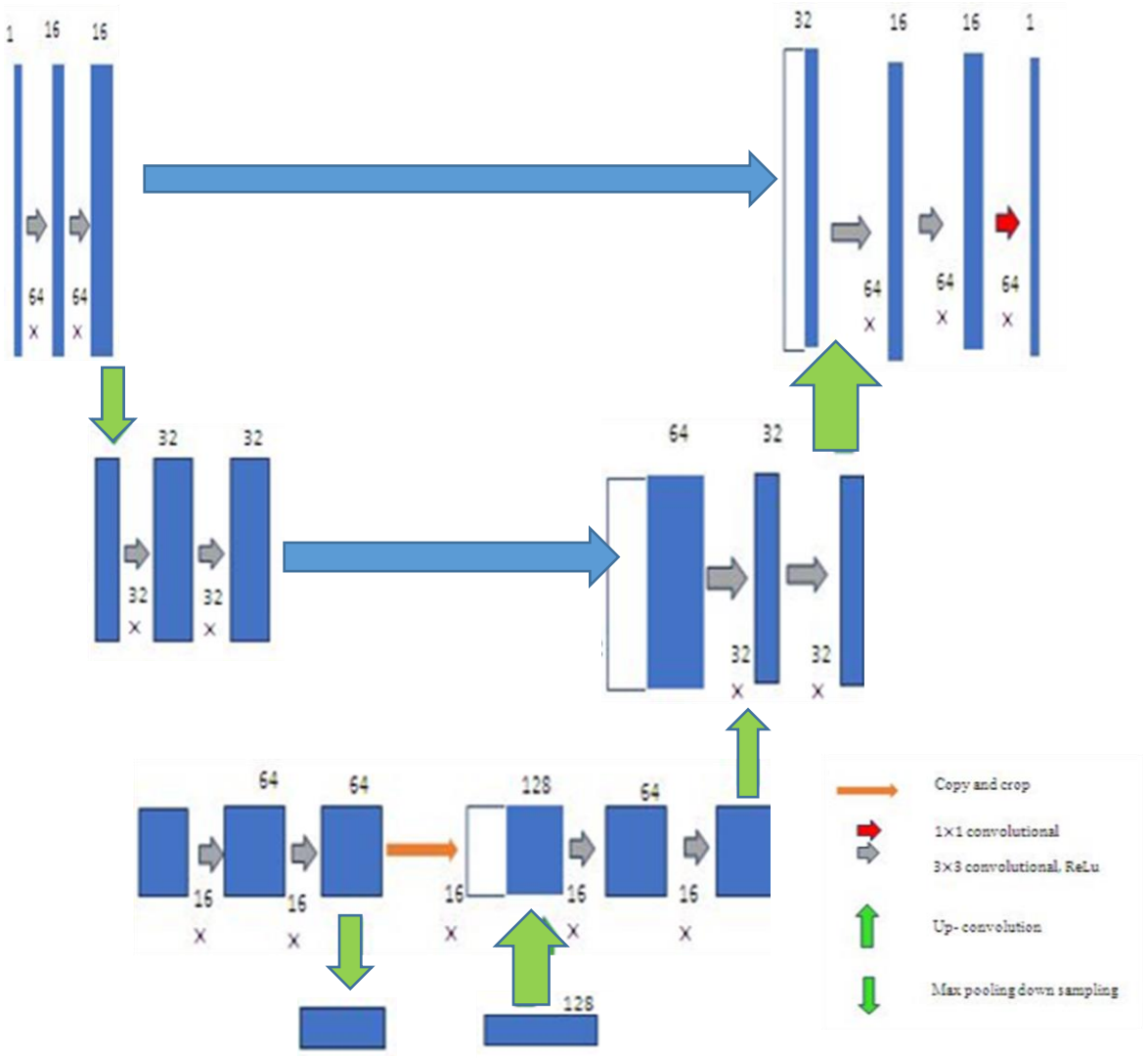

Figure 2: Functional block diagram

This is significant because the input data in reality may not always be perfectly aligned, and consistent, and the pooling operation is used to allow the CNN to effectively pick up essential features of interest irrespective of these variations. Max Pooling Down Sampling takes the input and begins by splitting it into non-overlapping rectangular regions (say a 4X4 matrix), these regions are known as the pooling windows The size ranges on these windows, but 2x2 is the most frequently used. From each pooling window, the maximum value is then selected and kept, and the rest of the values are discarded. The selection step of each pooling window is re-done to produce an output that has been down-sampled in the spatial dimension. Advantages of max pooling One main advantage of max pooling is that it is computationally efficient. This is huge when we have many parameter spaces to optimize. Pooling Layers: Here, the maximum value in each pooling window is selected which reduces the amount of information that is retained and is hence processed in subsequent layers. This helps with overfitting as the network would learn only the most prominent features instead of other specific features. One of the other cool things about Max Pooling down sampling is that it also brings the spatial dimensionality of

the input down, which helps when you have a large data set/image set, this can prove helpful, but, we will get to that later.

It is simplified as,

$$V = MC \tag{19}$$

$$C = (M^Q M)^{-1} M^Q V \tag{20}$$

$$APP = D1(SP, DP) = SP - DP,$$
$$MAP = D2(SP, DP) = \frac{DP + (SP, DP)}{3},$$
$$ARPP = D3(HR, SP) = HR * SP. \tag{21}$$

By reducing the spatial size, CNN can learn to extract more abstract features, allowing for a more efficient data representation. This can also aid in reducing the model's computational burden and memory requirements. However, one potential drawback of max pooling down sampling is the loss of information. Selecting the maximum value within each pooling window may discard some valuable information, leading to a lower model accuracy. To address this, some variations of max pooling, such as average pooling, have been proposed, where the average instead of the maximum value is selected within each pooling window. This can help to retain more information but at the cost of reduced robustness to variations in the input data.

C. Operating Principle

- **Encoder:**

An encoder is a circuit or device that converts analog signals into digital signals. In computing, this is referred to as encoding or digital encoding. Point #1 - As an Encoder, what you do is convert the continuous analog signals (in the form of electrical voltages or currents) into the type of digital signals made up of only the number 0s and 1s, i.e. binary, and to comprehend the working of the Encoder, it requires an understanding of its core parts. The first one is an input device/Sensor or Any analog signal source. This input device creates an analog signal that must be translated into data form. The Second Part also known as a sampling circuit, collects Samples of the Analogue Signal between fixed Time Intervals. These are little cross-sections of the signal at a moment in time. Next, in the encoder, a quantization circuit follows. This circuit handles the conversion of the incoming analog signal to a digital signal by digitally encoding each sampled amplitude to its corresponding binary code. It compares the amplitude of the sample to a reference value and sets it to binary 1 if above and binary 0 otherwise. These values are quantized and are very important for digitalization by converting the analog signal into the digital signal. The second stage, considering the encoding process, involves encoding and mapping the binary code of each sample to be transmitted, stored, or …). The pulses are typically encoded as sample values, where a mapping is performed based on both signal amplitude and sampling frequency to produce a binary number that uniquely identifies each sample. In other applications, specific encoding methods are foreseen for certain purposes, such as delta or companding modulation. Once a digital signal is encoded, it can be transmitted to the device responsible for that sound. The encoder may further comprise a transceiver or a storage medium disposed within the encoder, the transceiver or the storage medium being a wire, an optical cable, or a memory device for storing or transmitting the digital signal.

- **Decoder:**

Decoding is the process of translating code into plain text or any other format used for subsequent processing. It is the core component of any digital system that decodes the informational encoding, enabling the system to comprehend and respond to it. A decoder operates on an input vector, or address, of bits; the active output line is one of the 2^N possible output lines, where N is the number of bits in the address. This is called "decoding" and is done with a combination of logic gates (like AND / OR / NOT gates). To see how a decoder works, let us begin by examining a simple example. For instance, we could put a 2:4 decoder into this block, which will take in two input bits and output on four lines. You will have as input 4 different values, 00, 01, 10, and 11, each represented by 2^2 bits. As already indicated, these combinations will represent one of the output lines, such as Y0, Y1, Y2, and Y3. Thus, for input bits equal to 00, the output line Y0 is enabled; for input bits equal to 10, the output line Y2 is enabled, etc. The input bits serve as control logic for activating combinations of the output lines. It is accomplished by using a combination of logic gates. For example, for our 2-to-4 Decoder, the first input bit (A) will form the inputs to the control inputs of two AND gates, one for each of the output lines. The control inputs of four AND gates will be connected to the second input bit (B); these correspond to the four output lines. These AND gates will share a new input connected to a NOT gate, which will flip or invert the input bit. The outputs of all these AND gates will be wired to their output lines, and the output lines themselves will lead to a conventional output port.

- **Add & Normalize:**

"Add and renormalize" is an operation familiar to digital signal processing and data analysis. This operation consists of two steps: addition and normalization. By adding signals during the addition phase, values are combined per sample. As a consequence, a single signal is obtained that consists of the combined information of all input signals. Following the addition process, the signal is normalized. Normalization is a mathematical process that applies scaling to a signal's amplitude to adjust its range or scale. It is significant for the operations of data analysis and signal processing, as it normalizes signals and makes them more comparable. Figure 3 The process flow chart is presented.

To better illustrate the operations of "add and normalize," consider a simple case. Let's say we have two signals, A and B, with values 1, 2, 3 and 2, 4, 6, respectively Figure 3.

Then, the sum of these two signals will be 3 + 6 + 9. The values of the modulated signals are greater than one, and they require normalization. The normalization is to find the scale factor so that the signal is within the range. This scaling factor is obtained by dividing each sample of the resultant signal by the maximum value of the same signal. In our case, the peak of the deconvolved signal is 9. If you divided each of these by 9, you would get numbers between 0 and 1, giving you a normalized signal of (1/3, 2/3, 1). Normalization is crucial in data analysis and signal processing, as it facilitates the interpretation of signals whose amplitudes vary across different scales. It also helps prevent problems like data overflow, where the values in the signal become too large to be processed. There are other practical uses of "add and normalize" beyond normalization. It can mix signals; for example, you can create a final mixed track by combining two or more tracks. Moreover, it is the operation performed together in image processing that synthesizes multiple photographs. In this case, normalization ensures that the final image has consistent brightness and contrast.

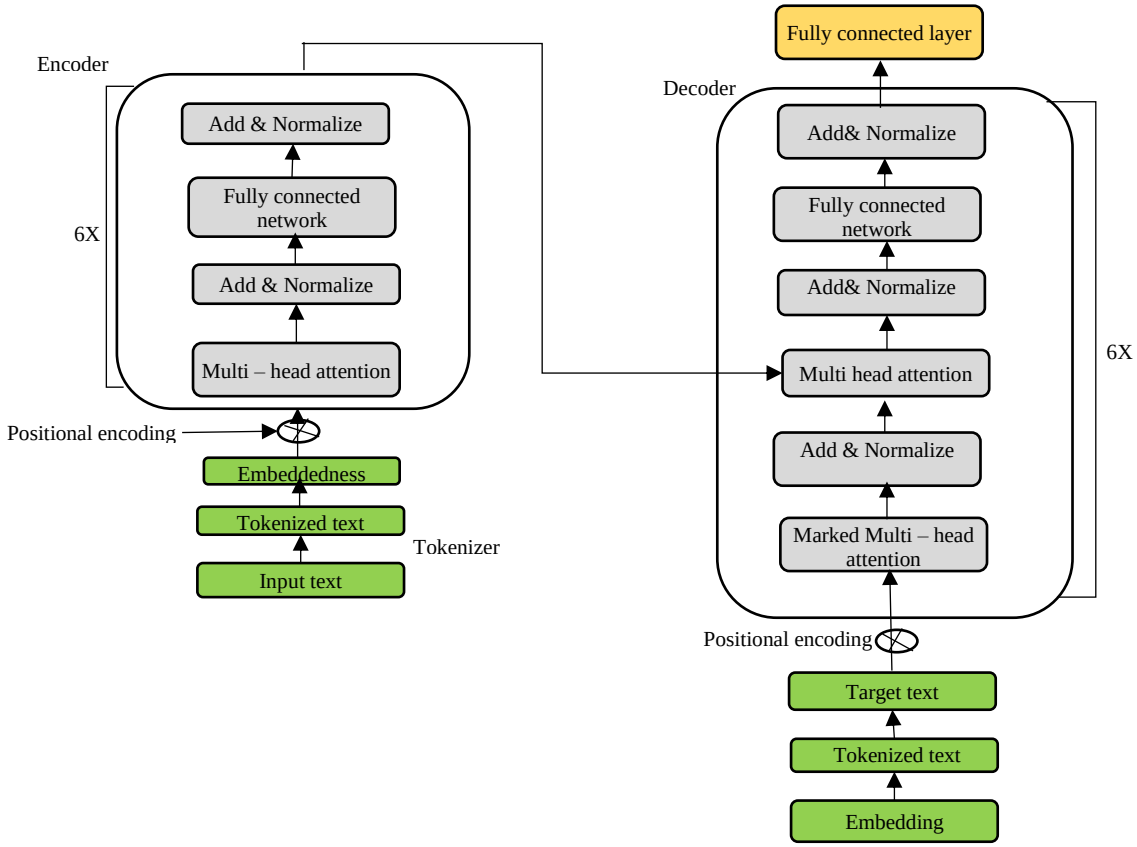

Figure 3: Operational flow diagram

- **Fully Connected Network:**

A fully connected network, otherwise known as a dense network, is an artificial neural network in which all the nodes in one layer are connected with every other node in the next layer. This implies that each node in one layer is connected to each node in the subsequent layer and enables the ANN to learn complex relationships between the input and the output data. It is a fully connected network with layers that have many nodes or units; Input Layer: The first layer of a neural network, this is the layer through which observations enter data. In other words, the number of nodes in this layer depends on the dimensionality of the input data. The following layers are hidden layers and are defined by the number of nodes in the respective layer, with the final layer being the output layer. Based on your task type, for example, classification vs regression, several output nodes in the output layer will vary accordingly, i.e. no of class labels n if classification or a numerical value in regression. A fully connected network has two main operations: the forward pass and the backward pass. As such, you feed input data into the network in the forward pass, and each node in the first layer of a network computes a weighted sum of the inputs based on an initially random set of weights. This weighted sum is then activated by using some activation function (e.g., sigmoid, ReLU) to add a non-linearity and also change the output for the next layer. Except for the output layer, this process repeats in each subsequent layer. This network performs the relationship between the input and output data during the forward pass which in turn determines the weights between each layer. This is done by backpropagation, here the predicted output is compared to the actual production, and that error is propagated back to the network. The weights of each layer are updated to minimize the total error based on this error. So the massive dataset is used to train the whole fully connected networks in an attempt to reduce the errors on unseen data. This is so

the network can generalize and perform well on data it has never seen on the training set. A fully connected network can - and, probably, will - have a different number of layers and nodes and this is usually a result of a hyperparameter-tuning process. Hundreds of thousands of layers (or nodes) would result in a classic overfit photo where the network while iterating during training, does likely get good at the training data, but it needs to also be able to generalize the new data.

- **Multi-head Attention:**

Multi-head attention provides a way to map queries and keys of multiple dimensions to the same heads in the same weight matrices. Multi-head attention plays an essential role in modern transformer-based models that are widely used in various natural language processing tasks (e.g., translation, question-answering). Multi-head attention is an attention mechanism that helps the model focus on different parts of the input sequence at once. This is done by a myriad of attention heads, each of which processes a separate view of the input sequence and computes its own set of attention scores. Multi-head attention starts with splitting the input sequence into multiple versions of it. This is often done by staggering the input via multiple parallel layers each producing a different representation through non-linear transformation. For instance, the input sequence in a language translation task might be processed by three layers, one which encodes it as a series of word embeddings, the second one encodes it as a series of part-of-speech tags, and the third one represents it as a series of its syntactic dependencies. Then each of these representations is used by a different attention head to extract attention scores over the input sequence using a query vector. We usually have a learned attention vector for each attention head. These are what the model might be focusing on, and are those that the attention mechanism will weigh on input sequences. To calculate the attention scores, each attention head computes a scaled dot-product operation between the query vector and input sequence representations. Here, the dot product between the query vector, and every input representation needs to be computed, and the result needs to be scalable to prevent values from getting very large (or very small). What it produces is essentially a bunch of attention scores for every input location, this time representing how important that location is concerning the current query. After calculating the scores of attention, they are normalized through a softmax function to make sure that they sum up to one, in the form of attention weights. The weights are then used to multiply over the input tokens which are finally summed up to produce the output of the attention head. This output is the sum of some parts of the input sequence where portions appear from the original input sequence with weights depending on their relevance to the part of the output position as computed by the attention scores. At the end, the outputs from all attention heads are joined together and then linearly projected yielding the final output of the multi-head attention layer. As such, we take this output, put it through a feed-forward network, and incorporate it with the original inputs to break linearity and help the model capture higher-order relationships between different parts of the sequence.

- **Positional Encoding:**

Positional encoding is a technique used in natural language processing (NLP) to add positional information to a sequence of words or tokens. ds," or to positionally encode a set of descriptions in sentence style to blotext." It's the latter use that I would like to discuss today shortened Twitter version #JustUseOverlays Positional Encoding for NLP, EMNLP 2019 https://t. It's applied primarily in tasks such as language translation and language understanding, for which the order of words in a sentence is crucial. In classical NLP models, input sentences are converted into word embeddings, where every word has a unique, constant-size vector. Nonetheless, such a strategy fails to account for the ordered nature of the language, resulting in the loss of positional information. To solve this problem, positional encoding was introduced as an additional layer in the neural network model. It is applied to word

embeddings and encodes a position for each word in the sentence. This positional encoding will then be added to the original word embedding, resulting in a new vector that contains information about the word's meaning and its position in the sentence. The widely used positional encoding is sine and cosine function encoding, which utilizes sine and cosine waves of various frequencies to encode positional information. The frequency of these waves is determined by the word's place in the sentence, so low frequency = beginning words, > high frequency=ending words. This encoding is continuous and smooth, which is crucial for the neural network to be able to learn properly. And it is more intuitive and closer to the way time works (and its periodicity). One of the most significant advantages of positional encoding is that it enables the model to distinguish between the same word in different locations within the sentence. For instance, consider the sentence "I saw the tiger in the forest," where "saw" has a different meaning compared to the sentence "I saw the forest with my own eyes." According to a new positional encoding, the model can infer the word's meaning based on its position in a sentence. Positional encoding also has the nice benefit of generalization; the model can learn patterns and relationships between words within a sentence, not just the meaning itself. This can be especially valuable in tasks like language translation, where the model requires some context to translate words correctly.

- **Embeddings:**

An embedding in the context of natural language processing (NLP) is a numerical representation of words or phrases so that any words that are semantically similar or have relevant meanings are especially close. Embedding models typically encode the semantic and syntactic relations among tokens in a language. Embeddings have become increasingly popular recently, as they have proven to be highly effective in various NLP tasks, including sentiment analysis, machine translation, and text categorization. The large text corpus is first input into a deep learning model (e.g., a neural network) to generate an embedding. The model then notes the relationship between words by observing their frequency and the surrounding context in which they occur within the text. This is also known as "learning word representations." Once the model has learned these relationships, it produces a dense vector for each word.

In the same way that each dimension in the one-hot vector represents a coordinate in the geometric space, each dimension in the dense vector can be used to describe some feature. The most notable feature of embeddings is their ability to understand the context in which a word appears. For instance, in the sentence "The cat is sleeping on the couch," the word "cat" has a semantic relation to the words "pet" and "feline" rather than "plane" or "math." This relationship can be captured by embeddings, where similar words are placed close together in the vector space. This helps the model exploit the meaning of the word and its relation to other words in the sentence. A second important point is that embeddings can handle out-of-vocabulary (OOV) words. OOV words should exist in the model's vocabulary as rare words or misspelled words. In a traditional NLP pipeline, this would result in the word with missing information being dropped from the input or replaced by a null token of some type. However, embeddings enable an out-of-vocabulary (OOV) word to be still represented by considering the surrounding words and their relationships in the vector space. This will allow the model to acquire the context of out-of-vocabulary (OOV) words, resulting in more accurate predictions. After the model has learned the embeddings, they can be applied to many NLP tasks. For instance, in sentiment analysis, a single word or phrase could have an embedding that represents the sentiment. Thus, the model will properly classify the sentiment of the poem's permuted lines together. In language translation, embeddings might also help to map words from one language to the other, such that (featural) similarity between words (in different languages) can be captured more easily by the model.

- **Tokenized text:**

Tokenization is the process of breaking down a text into smaller units, one of which is a token. In NLP, tokens are typically words but can also be various units of meaning, such as word parts or phrases. Tokenization is a crucial step in most text mining and Natural Language Processing applications, where data is organized and converted into a structured format for downstream applications. There are many different ways to tokenize text, but the most common is to split the text into words, numbers, and punctuation. This sounds simple, but it entails various complex tasks required for good tokenization and on-the-fly preprocessing of text. First, words are obtained by by splitting the text when a space is located. This is not sufficient because there are words that are spaced-separated and are still considered as a token, e.g., "New York" and "United States." To tackle these cases, the process of tokenization should take into account phrases and collocations, which are word combinations that are frequently used. Tokenization is also challenged by punctuation. Punctuations like commas, periods, and semicolons all have distinct uses in a sentence, and their positions can alter the meaning of the content. Hence, there is a need to preserve punctuation when tokenizing a text. Punctuation may, in some cases, be treated as tokens themselves (these may include emoticons or hashtags found in social media text).

Furthermore, Chinese, Japanese, and Thai languages are spaceless languages, so tokenization is more complicated. In such cases, tokenizing algorithms rely on linguistic rules and patterns to determine the boundary between words. Handling contractions, possessives, and other multi-part word forms represents another important part of tokenization. For instance, "don't" is to be tokenized to "do" and "not" while "John's" is to be annotated as a possessive of "John." These cases require dictionaries and rules to cut tokens properly.

Additionally, the process of tokenization must take into account "special characters and symbols, e.g. currency sign and emoji, that can provide information in text." These symbols may require tokenization and co-location with neighboring words due to their context. Tokenization sometimes also includes data cleaning and normalization to regularize (or normalize) the text and remove noise, such as HTML tags, email addresses, and URLs.

4 Result and Discussion

Comparison: In the proposed model, we compare the Deep Learning Blood Pressure Monitoring System for Smartwatches (DL-BPMSS) with existing models of Smartwatch-based Blood Pressure Measurement using Deep Learning (SW-BP-DL), Deep Learning Blood Pressure Monitoring Platform (DL-BPMP), and Continuous Blood Pressure Monitoring with Deep Learning (CBP-DL).

4.1. Accuracy:

This parameter evaluates the framework's ability to identify and measure blood pressure values using smartwatch sensors accurately. High Accuracy is essential for reliable and accurate continuous monitoring Table 1.

Table 1: Comparison of accuracy (in %)

No. of Inputs	SW-BP-DL	DL-BPMP	CBP-DL	DL-BPMSS
100	43.37	56.30	34.87	58.18
200	41.74	54.56	33.29	56.76
300	41.26	52.22	31.09	55.50
400	39.97	51.41	29.46	53.51
500	37.86	49.12	28.32	51.04

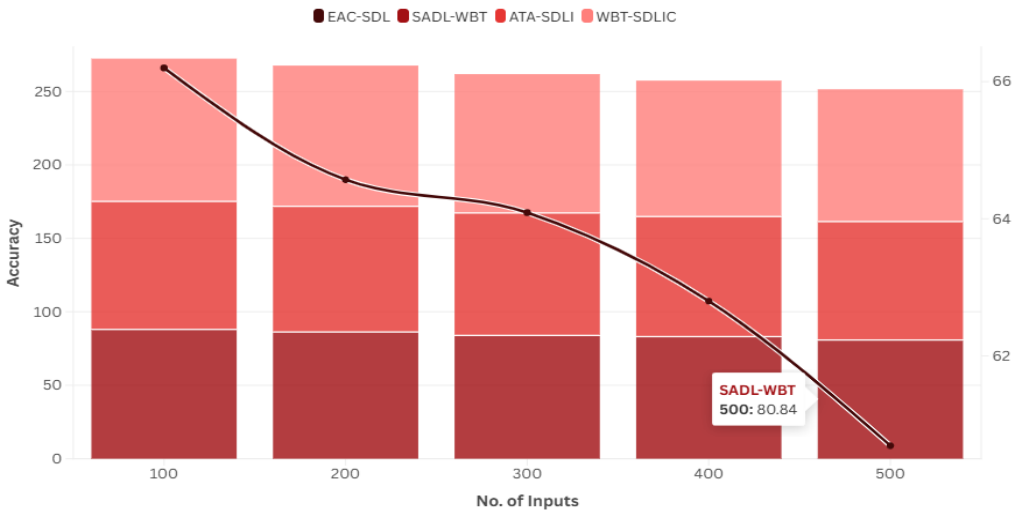

Figure 4: Computation of accuracy

4.2. Sensitivity:

This reflects that rapidly changing blood pressure values are difficult to detect within the framework. High Sensitivity is necessary when monitoring minor blood pressure changes, which is valuable for blood pressure monitoring and management. Table 2. The existing and proposed models are compared in terms of Sensitivity Figure 4.

Table 2: Comparison of Sensitivity (in %)

No. of Inputs	SW-BP-DL	DL-BPMP	CBP-DL	DL-BPMSS
100	47.37	59.30	39.87	62.18
200	45.74	57.56	38.29	60.76
300	45.26	55.22	36.09	59.50
400	43.97	54.41	34.46	57.51
500	41.86	52.12	33.32	55.04

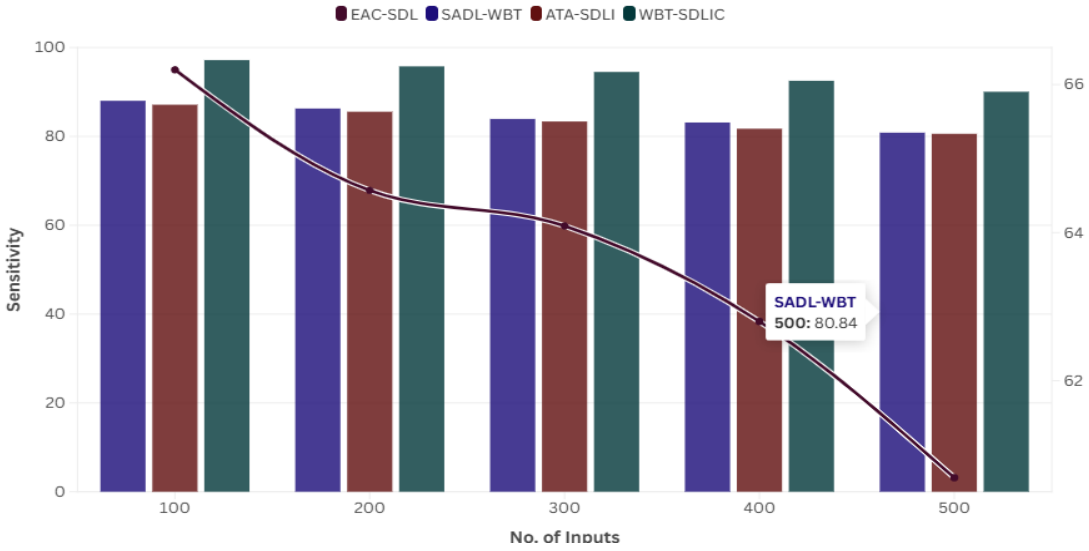

Figure 5: Computation of sensitivity

4.3. Response Time:

This variable indicates the time interval in seconds that it takes for the framework to process the data reading from the smartwatch sensors and output the blood pressure readings. A faster response time would be preferred to enable fast and online blood pressure (BP) monitoring. Table 3 Comparative Performance of Response Time in Existing Model Figure 5

Table 3: Comparison of Response Time (in %)

No. of Inputs	SW-BP-DL	DL-BPMP	CBP-DL	DL-BPMSS
100	52.37	66.30	43.87	68.18
200	50.74	64.56	42.29	66.76
300	50.26	62.22	40.09	65.50
400	48.97	61.41	38.46	63.51
500	46.86	59.12	37.32	61.04

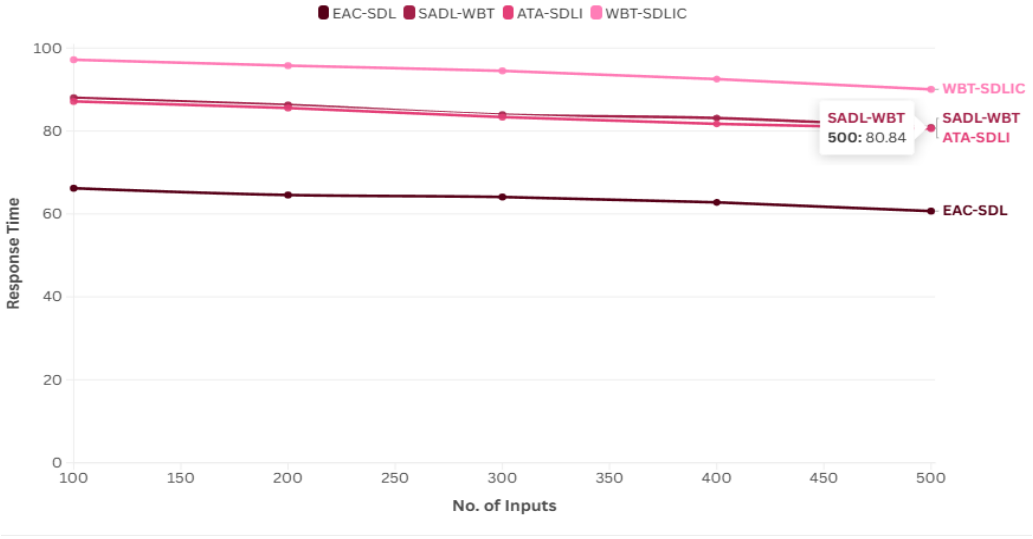

Figure 6: Computation of response time

4.4. Robustness:

This variable assesses the precision of the framework in providing reliable blood pressure estimates under adverse conditions, including physical activity, movement, and environmental disturbance Figure 6. There is a strong need for reliable, continuous in-life monitoring in such realistic scenarios. Table 4 compares the robustness between existing and proposed models Figure 7.

Table 4: Comparison of robustness(in %)

No. of Inputs	**SW-BP-DL**	**DL-BPMP**	**CBP-DL**	**DL-BPMSS**
100	57.37	71.30	50.87	73.18
200	55.74	69.56	49.29	71.76
300	55.26	67.22	47.09	70.50
400	53.97	66.41	45.46	68.51
500	51.86	64.12	44.32	66.04

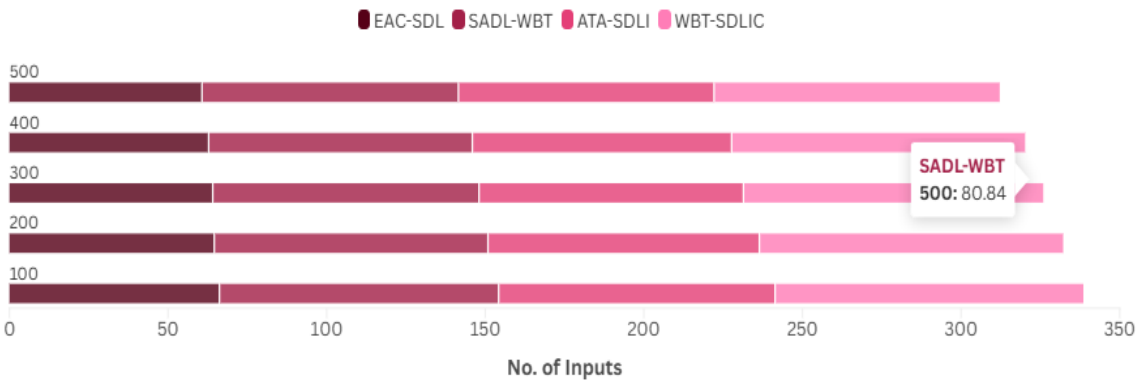

Figure 7: Computation of robustness

5 Conclusion

The task of developing a deep learning model based on smartwatch sensors takes some hits that could potentially lead to new methods of measuring and monitoring blood pressure. It serves as an exemplar of the potential capabilities of deep learning within the healthcare arena. The present framework has eliminated many of the problems of traditional blood pressure monitoring techniques that are considered inconvenient and non-accurate. Based on deep learning algorithms, it can continuously track blood pressure in real-time, providing continuous and accurate monitoring data to both medical personnel and patients. In addition, employing sensors integrated into a smartwatch renders this method so readily available that it can be embraced as a part of everyday life. However, this framework needs to be further improved and validated by future research. Through continued innovation and the use of technologies including artificial intelligence, this approach of caring for the patient can dramatically the lives of millions of Americans living with hypertension and other cardiovascular diseases to optimize their health and well-being.

References

[1] Attivissimo, F., D'Alessandro, V. I., De Palma, L., Lanzolla, A. M. L., & Di Nisio, A. (2023). Non-invasive blood pressure sensing via machine learning. *Sensors (Basel, Switzerland), 23*(19), 8342. https://doi.org/10.3390/s23198342

[2] Boopathy, E. V., Shanmugasundaram, M., Vadivu, N. S., Karthikkumar, S., Diban, R., Hariharan, P., & Madhan, A. (2024). Lorawan based coalminers rescue and health monitoring system using Iot. *Archives for Technical Sciences, 2*(31), 213–219. https://doi.org/10.70102/afts.2024.1631.213.

[3] Chen, Y., Zhang, D., Karimi, H. R., Deng, C., & Yin, W. (2022). A new deep learning framework based on blood pressure range constraint for continuous cuffless BP estimation. *Neural Networks, 152*, 181-190. https://doi.org/10.1016/j.neunet.2022.04.017

[4] El-Hajj, C., & Kyriacou, P. A. (2021). Cuffless blood pressure estimation from PPG signals and its derivatives using deep learning models. *Biomedical Signal Processing and Control, 70*, 102984. https://doi.org/10.1016/j.bspc.2021.102984

[5] El-Hajj, C., & Kyriacou, P. A. (2021). Deep learning models for cuffless blood pressure monitoring from PPG signals using attention mechanism. *Biomedical Signal Processing and Control, 65*, 102301. https://doi.org/10.1016/j.bspc.2020.102301

[6] Farhan, S., Mohamed Jaffar, A., Mary Posonia, A., Sahu, P. K., Suneetha, K., Goyal, S., & Sood, S. (2025). Design and development of an underwater wireless sensor network for aquatic

monitoring. *International Journal of Aquatic Research and Environmental Studies, 5*(1), 38–44. https://doi.org/10.70102/IJARES/V5I1/5-1-05.

[7] Geng, Y. (2024). Comparative Study on Physical Education Learning Quality of Junior High School Students based on Biosensor Network. *Natural and Engineering Sciences, 9*(2), 125-144. https://doi.org/10.28978/nesciences.1569219.

[8] González, S., Hsieh, W. T., & Chen, T. P. C. (2023). A benchmark for machine-learning based non-invasive blood pressure estimation using photoplethysmogram. *Scientific Data, 10*(1), 149. https://doi.org/10.1038/s41597-023-02020-6

[9] Haddad, S., Boukhayma, A., & Caizzone, A. (2021). Continuous PPG-based blood pressure monitoring using multi-linear regression. *IEEE journal of biomedical and health informatics, 26*(5), 2096-2105. https://doi.org/10.1109/JBHI.2021.3128229

[10] Hill, B. L., Rakocz, N., Rudas, Á., Chiang, J. N., Wang, S., Hofer, I., ... & Halperin, E. (2021). Imputation of the continuous arterial line blood pressure waveform from non-invasive measurements using deep learning. *Scientific reports, 11*(1), 15755. https://doi.org/10.1038/s41598-021-94913-y

[11] Huang, W., Ying, T. W., Chin, W. L. C., Baskaran, L., Marcus, O. E. H., Yeo, K. K., & Kiong, N. S. (2022). Application of ensemble machine learning algorithms on lifestyle factors and wearables for cardiovascular risk prediction. *Scientific Reports, 12*(1), 1033. https://doi.org/10.1038/s41598-021-04649-y

[12] Ismail, S. N. A., Nayan, N. A., Jaafar, R., & May, Z. (2022). Recent advances in non-invasive blood pressure monitoring and prediction using a machine learning approach. *Sensors, 22*(16), 6195. https://doi.org/10.3390/s22166195

[13] Kim, D. K., Kim, Y. T., Kim, H., & Kim, D. J. (2022). Deepcnap: A deep learning approach for continuous noninvasive arterial blood pressure monitoring using photoplethysmography. *IEEE Journal of Biomedical and Health Informatics, 26*(8), 3697-3707.

[14] Leitner, J., Chiang, P. H., & Dey, S. (2021). Personalized blood pressure estimation using photoplethysmography: A transfer learning approach. *IEEE Journal of Biomedical and Health Informatics, 26*(1), 218-228. https://doi.org/10.1109/JBHI.2021.3085526

[15] Li, S., Wang, H., Ma, W., Qiu, L., Xia, K., Zhang, Y., ... & Zhang, Y. (2023). Monitoring blood pressure and cardiac function without positioning via a deep learning–assisted strain sensor array. *Science advances, 9*(32), eadh0615. https://doi.org/10.1126/sciadv.adh0615

[16] Maqsood, S., Xu, S., Springer, M., & Mohawesh, R. (2021). A benchmark study of machine learning for analysis of signal feature extraction techniques for blood pressure estimation using photoplethysmography (PPG). *IEEE Access, 9*, 138817-138833. https://doi.org/10.1109/ACCESS.2021.3117969

[17] Maqsood, S., Xu, S., Tran, S., Garg, S., Springer, M., Karunanithi, M., & Mohawesh, R. (2022). A survey: From shallow to deep machine learning approaches for blood pressure estimation using biosensors. *Expert Systems with Applications, 197*, 116788. https://doi.org/10.1016/j.eswa.2022.116788

[18] Martinez-Ríos, E., Montesinos, L., Alfaro-Ponce, M., & Pecchia, L. (2021). A review of machine learning in hypertension detection and blood pressure estimation based on clinical and physiological data. *Biomedical Signal Processing and Control, 68*, 102813. https://doi.org/10.1016/j.bspc.2021.102813

[19] Menaka, S. R., Gokul Raj, M., Elakiya Selvan, P., Tharani Kumar, G., & Yashika, M. (2022). A Sensor based Data Analytics for Patient Monitoring Using Data Mining. *International Academic Journal of Innovative Research, 9*(1), 28–36. https://doi.org/10.9756/IAJIR/V9I1/IAJIR0905.

[20] Mohameed, H., Muthazhagu, M., Punitha, K., Anvarovna, A. S., Rajendren, D. J., & Sachdeva, L. (2025). Interdisciplinary Information Systems for Health and Education Libraries. *Indian Journal of Information Sources and Services, 15*(2), 11–15. https://doi.org/10.51983/ijiss-2025.IJISS.15.2.02.

[21] Najim, A. H., Al-sharhanee, K. A. M., Al-Joboury, I. M., Kanellopoulos, D., Sharma, V. K., Hassan, M. Y., ... & Abbas, A. H. (2025). An IoT healthcare system with deep learning functionality for patient monitoring. *International Journal of Communication Systems, 38*(4), e6020. https://doi.org/10.1002/dac.6020

[22] Nasir, N., Oswald, P., Barneih, F., Alshaltone, O., AlShabi, M., Bonny, T., & Al Shammaa, A. (2021, December). Hypertension classification using machine learning part II. In 2021 *14th International Conference on Developments in eSystems Engineering (DeSE)* (pp. 459-463). IEEE. https://doi.org/10.1109/DeSE54285.2021.9719408

[23] Oikonomou, E. K., Spatz, E. S., Suchard, M. A., & Khera, R. (2022). Individualising intensive systolic blood pressure reduction in hypertension using computational trial phenomaps and machine learning: a post-hoc analysis of randomised clinical trials. *The Lancet Digital Health, 4*(11), e796-e805.

[24] Paviglianiti, A., Randazzo, V., Villata, S., Cirrincione, G., & Pasero, E. (2022). A comparison of deep learning techniques for arterial blood pressure prediction. *Cognitive Computation, 14*(5), 1689-1710. https://doi.org/10.1007/s12559-021-09910-0

[25] Rong, M., & Li, K. (2021). A blood pressure prediction method based on imaging photoplethysmography in combination with machine learning. *Biomedical Signal Processing and Control, 64*, 102328. https://doi.org/10.1016/j.bspc.2020.102328

[26] Saleh, H., Younis, E. M., Sahal, R., & Ali, A. A. (2021). Predicting systolic blood pressure in real-time using streaming data and deep learning. *Mobile Networks and Applications, 26(1), 326-335*. https://doi.org/10.1007/s11036-020-01645-w

[27] Schrumpf, F., Frenzel, P., Aust, C., Osterhoff, G., & Fuchs, M. (2021). Assessment of non-invasive blood pressure prediction from ppg and rppg signals using deep learning. *Sensors, 21*(18), 6022. https://doi.org/10.3390/s21186022

[28] Sreevidya, B., & Supriya, M. (2024). Malicious Nodes Detection and Avoidance Using Trust-based Routing in Critical Data Handling Wireless Sensor Network Applications. *Journal of Internet Services and Information Security*, *14*(3), 226-244. https://doi.org/10.58346/JISIS.2024.I3.013.

[29] Wijnberge, M., van der Ster, B. J., Geerts, B. F., de Beer, F., Beurskens, C., Emal, D., ... & Veelo, D. P. (2021). Clinical performance of a machine-learning algorithm to predict intra-operative hypotension with noninvasive arterial pressure waveforms: A cohort study. *European Journal of Anaesthesiology| EJA, 38*(6), 609-615. https://doi.org/10.1097/EJA.0000000000001521

[30] Wu, J., Liang, H., Ding, C., Huang, X., Huang, J., & Peng, Q. (2021). Improving the accuracy in classification of blood pressure from photoplethysmography using continuous wavelet transform and deep learning. *International journal of hypertension, 2021*(1), 9938584. https://doi.org/10.1155/2021/9938584

[31] Yen, C. T., Chang, S. N., & Liao, C. H. (2021). Deep learning algorithm evaluation of hypertension classification in less photoplethysmography signals conditions. *Measurement and Control, 54*(3-4), 439-445. https://doi.org/10.1177/00202940211001904

[32] Zhao, L., Liang, C., Huang, Y., Zhou, G., Xiao, Y., Ji, N., ... & Zhao, N. (2023). Emerging sensing and modeling technologies for wearable and cuffless blood pressure monitoring. *npj Digital Medicine, 6*(1), 93. https://doi.org/10.1038/s41746-023-00835-6

Authors Biography

Dr. Sangita Babu is an Assistant Professor in the Department of Computer Science at King Khalid University, KSA. Her research interests include computer science education, artificial intelligence, and data analytics. She has published research in international journals and actively mentors students and researchers in advanced computing and interdisciplinary IT projects.

Dr. Arun Khatri is a Professor at the Mittal School of Business, Lovely Professional University. His expertise includes business management, organizational behavior, and strategic leadership. He has published extensively in peer-reviewed journals and contributes to curriculum development, academic mentorship, and interdisciplinary research initiatives.

Abhinav Rathour is affiliated with the Centre of Research Impact and Outcome at Chitkara University, Punjab. His research focus includes institutional research evaluation, academic analytics, and innovation management. He actively works on projects that enhance research quality, foster interdisciplinary collaboration, and support data-driven strategies in higher education.

Dinesh Kumar Jayaraman Rajendiran is an Assistant Professor in the Department of Electronics and Communication Engineering at Sri Krishna College of Engineering and Technology. His research interests include communication systems, signal processing, and embedded electronics. He is engaged in teaching, mentoring, and research activities focused on innovative solutions in electronics and communication engineering.

Bharat Bhushan is affiliated with the Chitkara Centre for Research and Development, Chitkara University, Himachal Pradesh. His professional focus includes academic research strategy, institutional development, and outcome-based project management. He contributes to enhancing research quality, promoting interdisciplinary collaboration, and supporting innovation-driven initiatives in higher education.

Dr.R.S. Ernest Ravindran is an Associate Professor in the Department of Electronics and Communication Engineering at Koneru Lakshmaiah Education Foundation. His research expertise includes communication systems, signal processing, and embedded electronics. He has published extensively and is actively involved in mentoring students, guiding research projects, and advancing academic development in electronics and communication engineering.