International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 11s
DOI: https://doi.org/10.17762/ijritcc.v11i11s.8081
***Article Received:** 25 June 2023 **Revised:** 18 August 2023 **Accepted:** 04 September 2023*

TinyML based Deep Learning Model for Activity Detection

Kayarvizhy N[1], Bharath Mahesh Gera[2], Bhavya Sharma[3], Dakshinamurthy S[4]
[1] Department of Computer Science,
B.M.S College of Engineering, Bangalore, India
kayarvizhyn.cse@bmsce.ac.in
[2] Department of Computer Science,
B.M.S College of Engineering, Bangalore, India
bharathmahesh.cs19@bmsce.ac.in
[3] Department of Computer Science,
B.M.S College of Engineering, Bangalore, India
bhavyas.cs19@bmsce.ac.in
[4] Department of Computer Science,
B.M.S College of Engineering, Bangalore, India
dakshinamurthy.cs19@bmsce.ac.in

Abstract— Our physical and emotional well-being are directly impacted by our body positions. In addition to promoting a confident, upright image, maintaining good body posture during various activities also ensures that our musculoskeletal system is properly aligned. On the other side, bad posture can result in a number of musculoskeletal conditions, discomfort, and reduced productivity. Accurate systems that can detect posture in real time, activity detection, are required due to the rising use of wearable technology and the growing interest in health and fitness tracking. The goal of this project is to create a TinyML model for wearable activity detection that will allow users to assess their posture and make necessary corrections in order to improve their health and general well-being. The project intends to contribute to the creation of useful posture detection technologies that can be quickly implemented on wearable devices for widespread usage by leveraging machine learning algorithms and wearable sensor data. For reliable posture categorization, the model architecture combines deep neural networks (DNN) and LSTM layers. With the development and implementation of the TinyML model, a significant decrease in the model's power consumption, memory, and latency was achieved without any compromise in the accuracy. This work can be used in the fields of health, wellness, rehabilitation, corporate life, sports and fitness to keep track of calories burned, activity duration, distance traveled, posture analysis, and real-time tracking.

Keywords- TinyML, DNN, LSTM, Activity Detection, Wear OS, TensorFlow Lite, Accelerometer, Gyroscope

I. INTRODUCTION

Activity Detection, Real-time posture detection, involves the automatic identification of human body positions and movements. This technology recognizes different postures or activities by evaluating data from sensors providing applications in health monitoring, fitness tracking, ergonomic assessment, and more. Posture has an impact in maintaining musculoskeletal health and preventing various disorders and injuries. With the increasing prevalence of sedentary lifestyles and the rise in musculoskeletal issues, the need for accurate and real-time posture detection systems has become more significant. Wearable devices equipped with sensors and machine learning algorithms have shown promise in monitoring and analyzing human posture. In recent years, the emergence of TinyML [5], which focuses on deploying machine learning models on resource-constrained devices, has opened up new possibilities for developing efficient and low-power posture detection solutions.

This paper aims to explore the application of TinyML models for posture detection on wearables. We present a comprehensive analysis of the existing literature and research advancements in this domain, focusing on the development and deployment of TinyML models specifically designed for posture detection. The challenges associated with posture detection on wearable devices, including limited computational resources, power constraints, and the need for real-time inference is overcome by converting the ML model into TinyML [49].

To develop a TinyML model for posture detection on wearables, we utilize a combination of deep neural networks (DNN) and long short-term memory (LSTM) models [20]. Once the ML model is trained, it is converted to a TinyML model using TensorFlow Lite. TensorFlow Lite [9] is a lightweight framework specifically designed for deploying machine learning models on resource-constrained devices, such as wearables. It enables efficient execution of models with

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 11s
DOI: https://doi.org/10.17762/ijritcc.v11i11s.8081
Article Received: *25 June 2023* ***Revised:*** *18 August 2023* ***Accepted:*** *04 September 2023*

minimal memory footprint and low computational requirements, making it an ideal choice for implementing TinyML models for posture detection. With TensorFlow Lite, we can leverage the power of deep learning models while ensuring optimal performance and energy efficiency on wearables. By integrating TensorFlow Lite into our posture detection system, we can take advantage of its capabilities to run the converted ML model seamlessly on wearable devices. This conversion process optimizes the model for deployment on resource-constrained wearable devices while maintaining its performance.

The converted model is then integrated into a Java-based Wear OS application developed using Android Studio.To set up the Wear OS application, we include the TensorFlow Lite interpreter and TensorFlow Select OP dependency [49] in the Gradle build file. Additionally, we create an emulator for the Wear OS application to run on, simulating the functionality on a virtual device.

During runtime, the application takes input, such as sensor readings from wearable devices. The TensorFlow Lite model performs inference on this input, predicting the posture based on the learned patterns and features. The output of the inference, which represents the detected posture, is then displayed on the Wear OS watch through text and animation, providing real-time detection to the user.

By implementing this pipeline, we enable real-time posture detection on wearable devices, leveraging the power of TinyML models and the versatility of Wear OS. This implementation showcases the feasibility and practicality of deploying posture detection systems on resource-constrained wearables, empowering users to monitor and improve their posture for healthcare purposes.

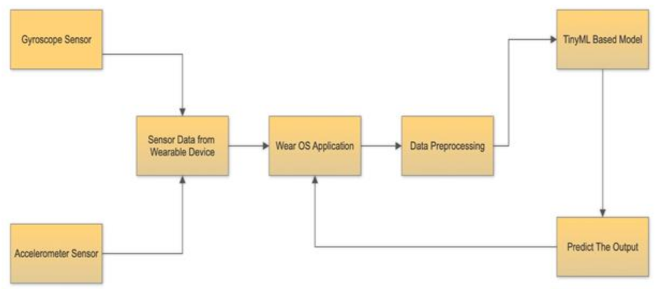

Figure 1. High level design for posture detection using wearable data.

By investigating the advancements and challenges in the field of TinyML-based posture detection on wearables, this paper aims to contribute to the growing body of knowledge and provide insights for researchers, developers, and practitioners interested in developing efficient and effective posture detection systems. The findings presented in this paper can serve as a foundation for further research and innovation in the field, ultimately leading to improved posture monitoring solutions that promote musculoskeletal health and enhance overall well-being.

Our key contributions are stated as follows:

A. Model Design (Posture Detection): The model design incorporates a DNN architecture [20] for posture detection. The input data includes accelerometer and gyroscope readings. LSTM layers are used to establish temporal dependencies, addressing the gradient problem. The neural network consists of two LSTM layers, a dense layer, and employs activation functions such as tanh and SoftMax. The model is trained using Stochastic Gradient Descent optimizer [20], with a loss function of Sparse Categorical Cross Entropy [20]. Accuracy is monitored during training, which spans 52 epochs.

B. Data Fusion: The dataset required preprocessing before training the model. The text files were transformed into a pandas DataFrame, and data fusion was performed by combining accelerometer and gyroscope readings based on the common timestamp parameter. This fusion allowed for a larger and more diverse dataset from multiple users. For the implementation, only five main activities were selected, including Jogging, Sitting, Climbing stairs, Standing, and Walking. The data was further transformed from a pandas DataFrame to a numpy array to serve as input for the neural network model in TensorFlow.

C. Conversion to TinyML: The trained TensorFlow model is converted to TinyML using TensorFlow Lite, which is well-suited for performing inference on the Android platform. Since the regular TensorFlow runtime does not satisfy the dependencies of the RNN, an advanced conversion technique is used. Two additional libraries, TFLITE_BUILTINS and SELECT_TF_OPS OpsSets, are added to enable the conversion. Once the conversion is complete, the file is saved for future use.

D. Performance Measure: The performance of the project was evaluated using various metrics to assess the accuracy and effectiveness of the posture detection model. The metrics used were precision, recall and F1 score. The accuracy reported by our model was 87.4%.

II. RELATED WORK

A. *ML in wearables*

ML techniques have gained significant attention in the field of wearables due to their ability to process sensor data and extract meaningful insights. One study focused on utilizing ML in the development of a multi-label TinyML machine learning model for greenhouse microclimate control. The model was trained on multivariate sensed data and successfully optimized the greenhouse environment for better plant growth. Another research work explored the application of ML in healthcare wearables, providing a big picture overview of the use of ML for wearable devices in the healthcare domain. The study highlighted the potential of ML algorithms for analyzing health data collected by wearables and facilitating personalized healthcare monitoring.

B. TinyML in wearables

The emergence of ultra-low-power IoT edge devices has opened up possibilities for deploying TinyML models on resource-constrained wearables. Several studies focused on enabling TinyML inference on these devices. For instance, one study developed an algorithm for compressing TinyML models specifically for IoT environments. The algorithm effectively reduced model size without compromising accuracy, making it suitable for deployment on low-power wearables. Another study explored the use of TinyDL, employing wearable sensors for edge computing and hand gesture recognition. The research demonstrated the feasibility of using TinyML for real-time gesture recognition, paving the way for intuitive human-computer interactions through wearables.

C. Wearables in healthcare

Wearable devices have revolutionized healthcare by providing continuous monitoring and personalized insights. One study reviewed wearable IoT devices for healthcare, emphasizing the potential of these devices in improving heart health through data sharing and telehealth engagement. The research highlighted the benefits of commercial wearable devices in collecting data for healthcare applications, enabling better management of conditions like atrial fibrillation. Another research work investigated the performance of wearable motion sensors for fall detection and daily activity recognition using machine learning. The study demonstrated the effectiveness of wearable devices in accurately detecting falls and monitoring daily activities, thereby enhancing the safety and well-being of individuals, particularly the elderly.

D. Low power consumption in wearables

The power consumption of wearable devices is a critical factor in their usability and practicality. To address this challenge, one study focused on training neural networks at the edge, considering the practical implications of power consumption in wearable ML applications. The research discussed strategies for reducing power consumption during model training, making it feasible to deploy ML models on low-power wearables. Another study explored the use of TinyML systems for real-time quality inspection in smart manufacturing. The integration of wearable smart devices and deep learning enabled real-time monitoring and quality control, minimizing energy consumption while ensuring high accuracy.

E. Posture detection

Posture detection using wearable devices has gained attention as a means to promote musculoskeletal health. One study investigated the use of deep learning-based models for complex activity recognition using wrist-worn wearable sensor data. The research demonstrated the effectiveness of deep convolutional neural networks (CNNs) and recurrent neural networks (RNNs) in accurately recognizing different activities related to posture. Another study focused on contactless posture detection using smartwatch sensor data and deep learning techniques in an Internet of Things (IoT) environment. The research proposed a model based on restricted Boltzmann machines (RBMs) for detecting human activities, including posture, and showcased the potential of IoT and wearable devices in monitoring and improving posture.
These literature surveys provide an overview of the research conducted in each categorized topic, showcasing the advancements and potential applications of ML, TinyML, wearables, and posture detection. The studies highlight the benefits of using ML and TinyML techniques in wearable devices, particularly in healthcare monitoring, gesture recognition, quality inspection, and musculoskeletal health. They also emphasize the need for further research in optimizing power consumption, exploring different sensor configurations.

III. METHODOLOGY

A. Sensor Description

Two sensors were used to record the data. The description of the sensors is as follows -
Accelerometer: A sensor called an accelerometer [42,43] measures appropriate acceleration, or the acceleration that an item experiences in relation to free fall. Static (due to gravity) and dynamic (due to movement) linear acceleration variations are detected and measured. The mass of the accelerometer is coupled to a spring, and its displacement varies according to the applied acceleration. The accelerometer can offer real-time measurements of acceleration in three axes: X, Y, and Z since this displacement is turned into an electrical signal.

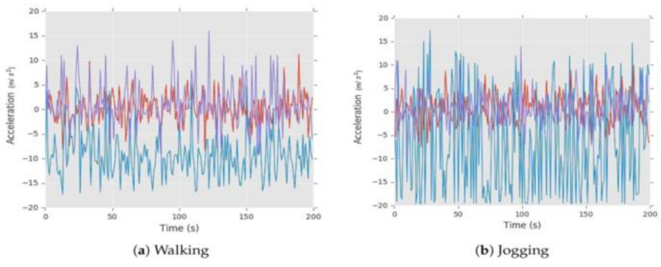

Fig 2. Example of Accelerometer Graphs for various activities

Gyroscope: A gyroscope [42,43] is a sensor that detects rotational motion or angular velocity. It can recognize changes in orientation and can tell you how quickly the three axes (roll, pitch, and yaw) are rotating. A rotating disc or a vibrating structure that maintains its rotational axis in the presence of outside forces makes up a gyroscope. The Coriolis effect generates a deflection in the vibrating structure when the gyroscope is rotating, producing an electrical output that is proportional to the angular velocity.

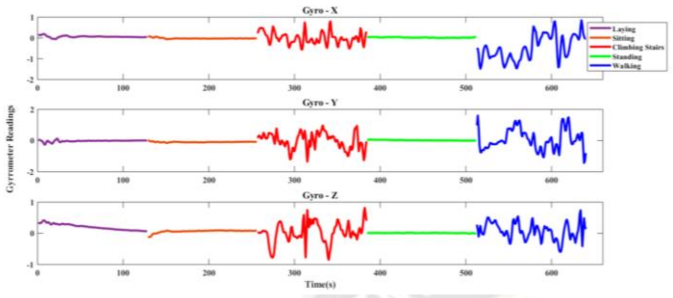

Fig 3. Example of Gyroscope Graph for various activities

B. Dataset description

The dataset that is being used is the WISDM dataset [51] which contains the data from sensors on both watches and mobile phones. The main data collected were from 2 main sensors in the watch, that is the gyroscope and the accelerometer. The dataset had collected the data from each sensor for a total of 51 persons or subjects and had recorded for 18 activities each. The dataset was well structured to differentiate the data for each user and on which device the sensor values were recorded from. The files related to the accelerometer sensor includes the following columns:

- First column: Subject ID of each user.
- Second column: Activity code which identifies each activity uniquely.
- Third column: Timestamp of each data acquired in milliseconds (ms).
- Fourth column: Value of the x-axis of the accelerometer (m/s2).
- Fifth column: Value of the y-axis of the accelerometer (m/s2).
- Sixth column: Value of the z-axis of the accelerometer (m/s2).

Next, the files related to the gyroscope sensor includes the following columns:

- First column: Subject ID of each user.
- Second column: Activity code which identifies each activity uniquely.
- Third column: Timestamp of each data acquired in milliseconds (ms).
- Fourth column: Value of the x-axis of the accelerometer (m/s2).
- Fifth column: Value of the y-axis of the accelerometer (m/s2).
- Sixth column: Value of the z-axis of the accelerometer (m/s2).

C. Data Fusion

The dataset was not available for direct use in the csv format. There was some pre-processing and data fusion that had to be done in order to make it fit for the training of the model. The files were in text document format and had to be formatted and transformed to a pandas Dataframe. The accelerometer and gyroscope readings also were in separate files on which data fusion had to be done in order to combine the sensor data for a single user. The time stamp was used as a common parameter to perform the data fusion. This way the data fusion was performed for multiple users to get a good volume and variety of data. In our implementation we had decided to only pick 5 main and common activities to train the Recurrent Neural Network. The data therefore had to be transformed to pick only 5 main activities. These activities include Jogging, Sitting, Climbing stairs, Standing and Walking. The data input to the neural network had to be transformed from a pandas data frame to an numpy array in order to train the TensorFlow model.

D. Recurrent Neural network design

The data input for the neural network is of the format accel[X], accel[Y], accel[Z], gyro[X], gyro[Y], gyro[Z]. The Recurrent neural network was used to provide a connection between one node to another which allows the output of one node to influence the input of the next node. In the project we are using LSTM or Long short-term memory layers [20] for the recurrent neural network as they provide a solution to the vanishing or exploding gradient problem. The RNN experiences the weight of the nodes to become very small or very large which limits the effectiveness of the model. The neural network mainly consists of 3 layers. The first 2 layers are the LSTM layers with each layer having 6 nodes each.

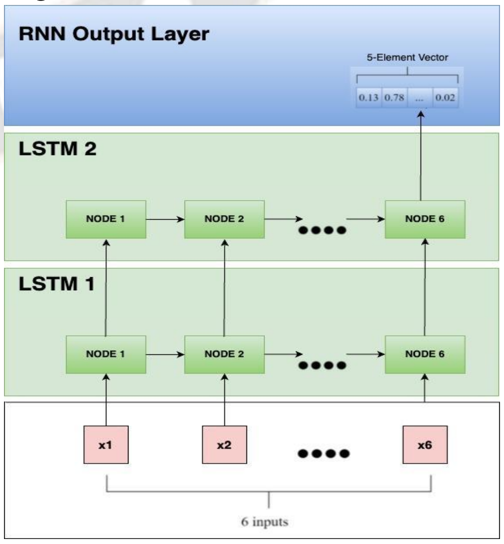

Fig 4. Architecture of Neural Network

Each of the LSTM layers have the tanh activation function which has been found to converge to the correct output. LSTM layers also have GlorotNormal initialization which are used to randomize weights which is essential for the training process. The last layer is a dense layer with 5 nodes as output with the SoftMax activation function which provides the probability of the output classes which corresponds to the number of activities. The entire model is compiled with the Stochastic Gradient Descent optimizer [20] function with the learning rate and momentum set at 0.08 and 0.85 respectively. The loss function used was the Sparse Categorical Cross Entropy which is mainly used when there are more than two labels. The accuracy of the model is used as the main metrics to monitor the training of the model. The model is then trained for a total of 52 epochs and there is a validation dataset that is provided which ensures that the model is not being overfitted. The model then is saved as a TensorFlow saved model to be used in the future.

E. Conversion of DNN to TinyML

The TensorFlow model saved model trained previously is used for the conversion to TinyML [49]. The TinyML framework used is TensorFlow lite as it is very compatible with the android platform to perform inference. The conversion of the model was not possible directly as the RNN had various dependencies which were not satisfied by the regular TensorFlow runtime. Therefore, the advanced conversion technique had to be employed and two additional libraries were added, namely the TFLITE_BUILTINS and SELECT_TF_OPS OpsSets. The file is then saved for future use.

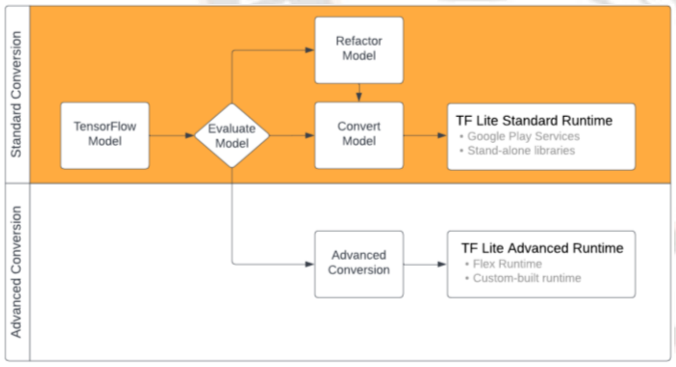

Fig 5. TensorFlow Lite conversion

F. Deployment of TinyML model on wear OS

The converted tensorflow based TinyML model is to be used on the Android based wear OS device. An Java based Wear OS Application is to be written and an emulator was used to run inference on the same.

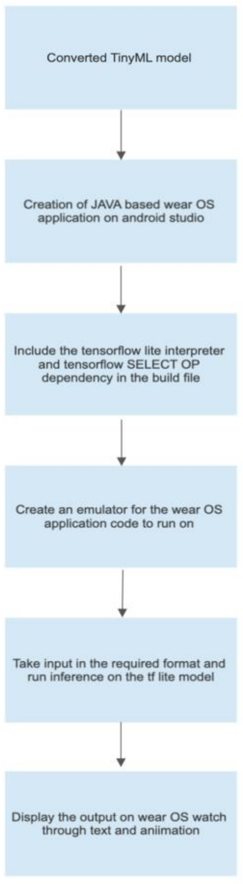
Converted TinyML model
Creation of JAVA based wear OS application on android studio
Include the tensorflow lite interpreter and tensorflow SELECT OP dependency in the build file
Create an emulator for the wear OS application code to run on
Take input in the required format and run inference on the tf lite model
Display the output on wear OS watch through text and aniimation

Fig 6. Deployment of TinyML model on wear OS

The Wear OS application is then split into the layout file and Java file. Since the conversion to TinyML was based on the advanced runtime we had to add an additional dependency: the TensorFlow SelectOps dependency along with the regular TensorFlow lite interpreter [9].

The Wear OS application functions were to take input values from the accelerometer and gyroscope sensors that are based on the watch and provide that as an input to perform inference on the same TinyML model. The output from the tensorflow lite model is an array of probability of each of the activities and there in order to provide the appropriate output we need to pick the output with the highest probability and then map the same to activities. To make the interface of the watch attractive we have added animations matching each activity. Once the output from the model is ready, we then display the result on the watch interface.

IV. RESULTS AND DISCUSSION

A. Evaluation of Neural Network

The whole dataset is divided in the ratio of 80:20 which corresponds to the training and validation set for the purpose of training. The validation set determines the performance of the neural network each epoch. For later use, the model that produces the highest validation accuracy is kept. Additionally,

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 11s
DOI: https://doi.org/10.17762/ijritcc.v11i11s.8081
***Article Received:** 25 June 2023 **Revised:** 18 August 2023 **Accepted:** 04 September 2023*

after training, the accuracy of each class is then determined using the validation set. Each class accuracy is determined by using the below formulas:
Accuracy: (TP +TN)/ (TP +FP +FN +TN)
TP: True Positive
TN: True Negative
FP: False Positive
FN: False Negative

Validation accuracy: In the validation phase, a machine learning model's performance on untried data is assessed using a statistic called validation accuracy. It gauges the model's accuracy in predictions produced using a different validation dataset. Better generalization and the capacity to correctly forecast outcomes on novel, untested data are shown by higher validation accuracy. It aids in comparing and choosing the best model among several versions at the same time, tracking the model's development throughout training. The model's performance on new data can be improved and overfitting or underfitting problems can be avoided by optimizing the model based on validation accuracy.

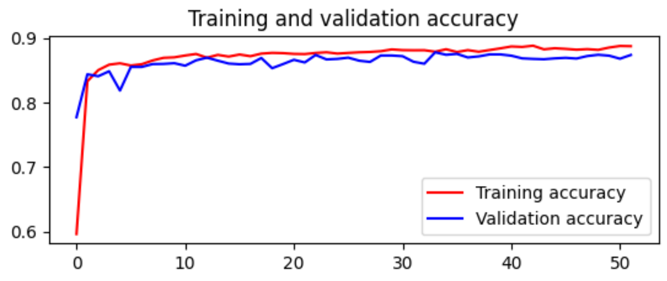

Fig 7. Training accuracy vs Validation Accuracy

Training accuracy: The training accuracy is used to monitor the model's performance while the model is in the training phase. The training data is used to count the proportion of accurate predictions. The model's training accuracy reveals how well it has internalized or learnt the input set of training data and how well it can categorize or predict the target variable. A model that can fit training data well and generate accurate predictions based on data it has already seen is said to have a greater training accuracy. It's crucial to remember that measuring training accuracy alone is insufficient to determine how well the model performs with fresh, untested data. Overfitting, when the model performs well on the data with which it was trained but fails to generalize to new data, can result from overemphasizing training accuracy without taking validation accuracy into account or testing on unknown data.

Validation loss: A machine learning model's projected outputs and the actual values in the validation dataset are measured by a statistic called validation loss. The model's effectiveness and generalizability to unobserved data can be evaluated by using the validation loss. The training loss can be reduced by changing the model's parameters training. A validation dataset, which contains data the model hasn't seen during training, is used to separately calculate the validation loss. We can identify overfitting, which happens when the model becomes overly focused on identifying the patterns within the training dataset and struggles to generalize, by keeping an eye on the validation loss. The objective is to strike a balance between a low validation loss, which indicates strong generalization, and a low training loss, which provides an insight into the learning of the model.

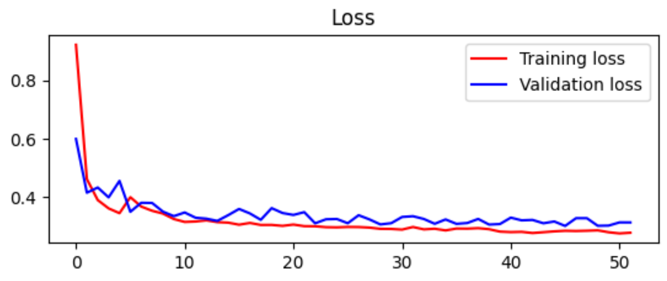

Fig 8. Training Loss vs Validation Loss

Training loss: A machine learning model's anticipated outputs and the actual values in the training dataset are measured using a metric called training loss. It displays how well the model fits the training set of data during training. To increase the model's capacity for precise prediction, the parameters are altered through the use of an optimization procedure like gradient descent. The training loss, which represents the average error made by the model on the training examples, is derived by comparing the model's predictions with the true labels in the training dataset. The model's learning progress is monitored by the training loss, but overfitting must be avoided by assessing the model's performance on a separate validation or test datasets.

Precision: TP / (TP + FP)
A classification model's precision, or ability to accurately identify positive examples, is evaluated as a performance metric. It determines whether the percentage of the total anticipated positive occurrences are real positive predictions (positive examples that were successfully categorized). In other words, precision assesses how accurate or precise the model's optimistic forecasts are. A high precision score means that the model reliably detects positive cases and has a low rate of false positives. In instances when the cost of false positives is large, like in medical diagnostics or spam email identification, precision is crucial. Since false negatives are not taken into consideration, accuracy alone might not give a full view of a model's performance.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 11s
DOI: https://doi.org/10.17762/ijritcc.v11i11s.8081
***Article Received:** 25 June 2023 **Revised:** 18 August 2023 **Accepted:** 04 September 2023*

Recall: TP / (TP + FN)
Recall, sometimes referred to as sensitivity or the percentage of true positives in a dataset, is a performance indicator used to assess how well a classification model distinguishes between positive and false positive occurrences. Out of the total real positive occurrences, it determines the percentage of true positive predictions (positive examples that were correctly categorized). Recall thus assesses the comprehensiveness or scope of the model's optimistic predictions. With a low incidence of false negatives and a high recall value, the model is able to properly identify a high percentage of positive examples. In situations where the cost of false negatives is large, such as in medical diagnostics or fraud detection, where it is critical not to miss positive instances, recall is particularly important. A high recall value, meanwhile, may also be associated with a higher likelihood of false positives. To have a thorough evaluation of the model's performance, recall is frequently required in conjunction with other measures, such as precision or F1 Score.

F1 Score: 2 * TP / (2 * TP + FP + FN)
The F1 Score is a performance indicator used to assess a classification model's efficacy by combining precision and recall into a single number. Calculating the harmonic mean of precision and recall yields a fair assessment of the model's accuracy. While recall reflects the model's capacity to properly identify positive cases, precision reflects the accuracy of positive predictions. An increase in the F1 Score from 0 to 1 denotes improved model performance. As a benchmark indicator for model comparison and improvement, it is especially helpful in datasets with imbalances.

TABLE I. METRICS FOR NEURAL NETWORK

Accuracy	**Precision**	**Recall**	**F1 Score**
0.874029	0.875194	0.872000	0.872719

B. Comparison between ML models and TinyML models

Model Size: Due to the use of complex frameworks and parameters, ML models frequently have higher sizes. On the other hand, TinyML models are designed to be compact, light, and to consume very little space. This enables them to function on devices with constrained memory and processing capability.

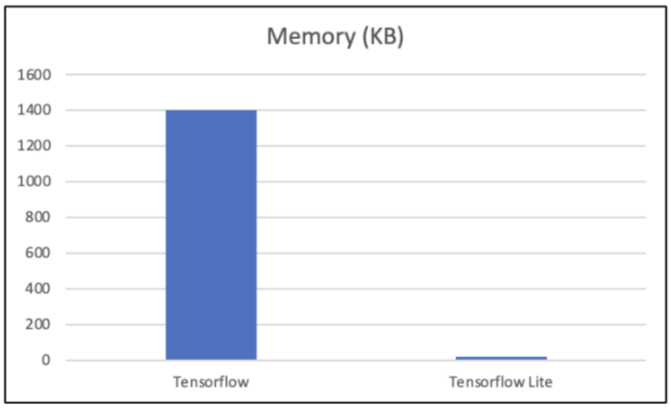

Fig 9. Model Size comparison

Power Consumption: ML models frequently use greater computing resources, which leads to higher power usage. TinyML models, on the other hand, are optimized for low-power consumption, allowing them to operate effectively on battery-powered or IoT edge devices without rapidly depleting the battery.

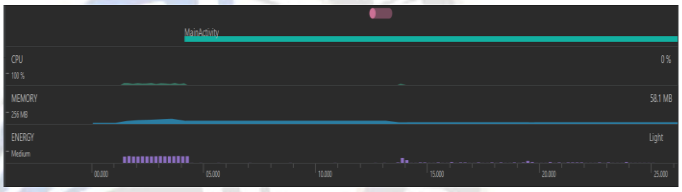

Fig 10. Wear OS application energy consumption to run TinyML model.

Latency: Due to their greater size and processing demands, ML models may have a higher latency. TinyML models are made to be less latency-intensive, allowing for real-time or almost real-time inference on edge devices without a lot of delays.

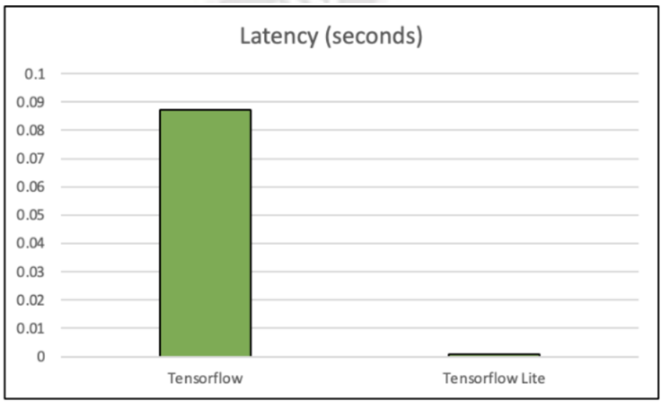

Fig 11. Latency comparison

Deployment flexibility: ML models are frequently deployed and inferred using powerful hardware or cloud infrastructure. On the contrary, TinyML models can be installed directly on edge devices, eliminating the requirement for ongoing cloud contact and ensuring data protection and privacy.

TABLE II. COMPARISON: ML MODEL VS TINYML

	ML Model (RNN - Tensorflow)	**TinyML (Tensorflow Lite)**
Memory	1400 KB	19 KB
Accuracy	0.874029	0.874029
Latency	0.08731245995 sec	0.0009715557098 sec

Development Process: Working with high-level frameworks and libraries when developing ML models usually requires machine learning knowledge and computational resources. TinyML models need specialized tools and frameworks that concentrate on model optimization, deployment, and compression which is essential for devices that have low computational energy and power.

V. CONCLUSION

Posture of a person critical aspects for one's well-being. In our project we would like to help a person correct their posture before any serious adversities. This project has demonstrated the potential of using a TinyML model for posture detection on wearables. By leveraging deep learning techniques, specifically DNN and LSTM models, we were able to develop an accurate posture detection system. The implementation of the model using TensorFlow Lite allowed for efficient deployment on wearable devices, making it practical for real-time monitoring.

VI. FUTURE WORK

In future work, the project can focus on enhancing model accuracy for all users by gathering a diverse dataset, using other sensors present in the wearable device and refining the model architecture. Increasing the subset of postures would make the system more versatile and adaptable to different activities. Adding more sensors can improve performance by providing comprehensive data on posture. Customization and features for the healthcare sector can be developed to aid healthcare professionals and provide personalized guidance. Implementing a recommendation system would give users feedback to improve posture and overall health.

ACKNOWLEDGMENT

We would like to extend our sincere gratitude to our college, B.M.S College of Engineering for giving us the tools and infrastructure we needed to complete this project.

REFERENCES

[1] Surantha, Nico, Prabadinata Atmaja, and Maulana Wicaksono. "A review of wearable internet-of-things device for healthcare." Procedia Computer Science 179 (2021): 936-943.
[2] Ihoume, Ilham, Rachid Tadili, Nora Arbaoui, Mohamed Benchrifa, Ahmed Idrissi, and Mohamed Daoudi. "Developing a multi-label tinyML machine learning model for an active and optimized greenhouse microclimate control from multivariate sensed data." Artificial Intelligence in Agriculture 6 (2022): 129-137.
[3] Awad, Ahmed, Mostafa M. Fouda, Marwa M. Khashaba, Ehab R. Mohamed, and Khalid M. Hosny. "Utilization of mobile edge computing on the Internet of Medical Things: A survey." ICT Express (2022).
[4] Rüb, Marcus, and Axel Sikora. "A Practical View on Training Neural Networks in the Edge." IFAC-PapersOnLine 55, no. 4 (2022): 272-279.
[5] Ray, Partha Pratim. "A review on TinyML: State-of-the-art and prospects." Journal of King Saud University-Computer and Information Sciences (2021).
[6] Altayeb, Moez, Marco Zennaro, and Ermanno Pietrosemoli. "TinyML Gamma Radiation Classifier." Nuclear Engineering and Technology (2022).
[7] Signoretti, Gabriel, Marianne Silva, Pedro Andrade, Ivanovitch Silva, Emiliano Sisinni, and Paolo Ferrari. "An evolving tinyml compression algorithm for iot environments based on data eccentricity." Sensors 21, no. 12 (2021): 4153.
[8] de Prado, Miguel, Manuele Rusci, Alessandro Capotondi, Romain Donze, Luca Benini, and Nuria Pazos. "Robustifying the deployment of tinyml models for autonomous mini-vehicles." Sensors 21, no. 4 (2021): 1339.
[9] David, Robert, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger et al. "Tensorflow lite micro: Embedded machine learning for tinyml systems." Proceedings of Machine Learning and Systems 3 (2021): 800-811.
[10] Bringmann, Oliver, Wolfgang Ecker, Ingo Feldner, Adrian Frischknecht, Christoph Gerum, Timo Hämäläinen, Muhammad Abdullah Hanif et al. "Automated HW/SW co-design for edge AI: state, challenges and steps ahead." In Proceedings of the 2021 International Conference on Hardware/Software Codesign and System Synthesis, pp. 11-20. 2021.
[11] Banbury, Colby, Chuteng Zhou, Igor Fedorov, Ramon Matas, Urmish Thakker, Dibakar Gope, Vijay Janapa Reddi, Matthew Mattina, and Paul Whatmough. "Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers." Proceedings of Machine Learning and Systems 3 (2021): 517-532.
[12] Reddi, Vijay Janapa, Brian Plancher, Susan Kennedy, Laurence Moroney, Pete Warden, Anant Agarwal, Colby Banbury et al. "Widening access to applied machine learning with tinyml." arXiv preprint arXiv:2106.04008 (2021).
[13] Banbury, Colby R., Vijay Janapa Reddi, Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang et al. "Benchmarking tinyml systems: Challenges and direction." arXiv preprint arXiv:2003.04821 (2020).

[14] Surantha, Nico, Prabadinata Atmaja, and Maulana Wicaksono. "A review of wearable internet-of-things device for healthcare." Procedia Computer Science 179 (2021): 936-943.

[15] Nuvvula, Sri, Eric Y. Ding, Connor Saleeba, Qiming Shi, Ziyue Wang, Alok Kapoor, Jane S. Saczynski et al. "NExUS-Heart: Novel examinations using smart technologies for heart health—Data sharing from commercial wearable devices and telehealth engagement in participants with or at risk of atrial fibrillation." Cardiovascular Digital Health Journal 2, no. 5 (2021): 256-263.

[16] Sarivan, I-M., Johannes N. Greiner, D. Díez Álvarez, Felix Euteneuer, Matthias Reichenbach, Ole Madsen, and Simon Bøgh. "Enabling real-time quality inspection in smart manufacturing through wearable smart devices and deep learning." Procedia Manufacturing 51 (2020): 373-380.

[17] S., E. ., & Kasturi, K. . (2023). Evaluation of Linkage Between the Corporate Economic Value-Added Analysis and Information Technology Using Big Data. International Journal of Intelligent Systems and Applications in Engineering, 11(4s), 109–117. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2577

[18] Gopinath, Sridhar, Nikhil Ghanathe, Vivek Seshadri, and Rahul Sharma. "Compiling KB-sized machine learning models to tiny IoT devices." In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 79-95. 2019.

[19] Pardamean, Bens, Haryono Soeparno, Bharuno Mahesworo, Arif Budiarto, and James Baurley. "Comparing the accuracy of multiple commercial wearable devices: a method." Procedia Computer Science 157 (2019): 567-572.

[20] Godfrey, Alan, Victoria Hetherington, Hubert Shum, Paolo Bonato, N. H. Lovell, and Sam Stuart. "From A to Z: Wearable technology explained." Maturitas 113 (2018): 40-47.

[21] Tan, Song Lim. "Posture detection of smartphone users using deep learning." PhD diss., UTAR, 2018.

[22] Huynh-The, Thien, Cam-Hao Hua, Nguyen Anh Tu, and Dong-Seong Kim. "Physical activity recognition with statistical-deep fusion model using multiple sensory data for smart health." IEEE Internet of Things Journal 8, no. 3 (2020): 1533-1543.

[23] Coffen, Brian, and Md Shaad Mahmud. "TinyDL: edge computing and deep learning based real-time hand gesture recognition using wearable sensor." In 2020 IEEE International Conference on E-health Networking, Application & Services (HEALTHCOM), pp. 1-6. IEEE, 2021.

[24] Mekruksavanich, Sakorn, and Anuchit Jitpattanakul. "Sport-related activity recognition from wearable sensors using bidirectional gru network." Intelligent Automation & Soft Computing 34, no. 3 (2022): 1907-1925.

[25] Bian, Sizhen, and Paul Lukowicz. "Capacitive sensing based on-board hand gesture recognition with TinyML." In Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers, pp. 4-5. 2021.

[26] Islam, Md Milon, Sheikh Nooruddin, Fakhri Karray, and Ghulam Muhammad. "Human activity recognition using tools of convolutional neural networks: A state of the art review, data sets, challenges, and future prospects." Computers in Biology and Medicine (2022): 106060.

[27] Ambati, Loknath Sai, and Omar El-Gayar. "Human activity recognition: a comparison of machine learning approaches." Journal of the Midwest Association for Information Systems (JMWAIS) 2021, no. 1 (2021): 4.

[28] Jain, Rahul, Vijay Bhaskar Semwal, and Praveen Kaushik. "Deep ensemble learning approach for lower extremity activities recognition using wearable sensors." Expert Systems 39, no. 6 (2022): e12743.

[29] Burq, Maximilien, and Niranjan Sridhar. "Human Activity Recognition Using Self-Supervised Representations of Wearable Data." arXiv preprint arXiv:2304.14912 (2023).

[30] Ricotti, Valeria, Balasundaram Kadirvelu, Victoria Selby, Richard Festenstein, Eugenio Mercuri, Thomas Voit, and A. Aldo Faisal. "Wearable full-body motion tracking of activities of daily living predicts disease trajectory in Duchenne muscular dystrophy." Nature Medicine (2023): 1-9.

[31] Issa, Mohamed E., Ahmed M. Helmi, Mohammed AA Al-Qaness, Abdelghani Dahou, Mohamed Abd Elaziz, and Robertas Damaševičius. "Human activity recognition based on embedded sensor data fusion for the internet of healthcare things." In Healthcare, vol. 10, no. 6, p. 1084. MDPI, 2022.

[32] Alsareii, Saeed Ali, Mohsin Raza, Abdulrahman Manaa Alamri, Mansour Yousef AlAsmari, Muhammad Irfan, Hasan Raza, and Muhammad Awais. "Artificial Intelligence and Internet of Things Enabled Intelligent Framework for Active and Healthy Living." Computers, Materials and Continua 75, no. 2 (2023): 3833-3848.

[33] Dowd, Kieran P., Deirdre M. Harrington, Alan K. Bourke, John Nelson, and Alan E. Donnelly. "The measurement of sedentary patterns and behaviors using the activPAL™ professional physical activity monitor." Physiological measurement 33, no. 11 (2012): 1887.

[34] Ms. Mohini Dadhe, Ms. Sneha Miskin. (2015). Optimized Wireless Stethoscope Using Butterworth Filter. International Journal of New Practices in Management and Engineering, 4(03), 01 - 05. Retrieved from http://ijnpme.org/index.php/IJNPME/article/view/37

[35] Schrader, Lisa, Agustín Vargas Toro, Sebastian Konietzny, Stefan Rüping, Barbara Schäpers, Martina Steinböck, Carmen Krewer, Friedemann Müller, Jörg Güttler, and Thomas Bock. "Advanced sensing and human activity recognition in early intervention and rehabilitation of elderly people." Journal of Population Ageing 13 (2020): 139-165.

[36] Granat, Malcolm, Andreas Holtermann, and Kate Lyden. "Sensors for Human Physical Behaviour Monitoring." Sensors 23, no. 8 (2023): 4091.

[37] Manimaran, M., A. Sasi Kumar, N. V. S. Natteshan, K. Baranitharan, R. Mahaveerakannan, and K. Sudhakar. "Detecting the Human Activities of Aging People using Restricted Boltzmann Machines with Deep Learning Technique in IoT." In 2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS), pp. 105-110. IEEE, 2023.

[38] Thu, Nguyen Thi, To-Hieu Dao, Bao Bo Quoc, Duc-Nghia Tran, Pham Van Thanh, and Duc-Tan Tran. "Real-time wearable-device based activity recognition using machine learning

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 11s
DOI: https://doi.org/10.17762/ijritcc.v11i11s.8081
***Article Received:** 25 June 2023 **Revised:** 18 August 2023 **Accepted:** 04 September 2023*

methods." International Journal of Computing and Digital Systems 11, no. 1 (2022): XXXX-XXXX.

[39] Paraschiakos, Stylianos, Ricardo Cachucho, Matthijs Moed, Diana van Heemst, Simon Mooijaart, Eline P. Slagboom, Arno Knobbe, and Marian Beekman. "Activity recognition using wearable sensors for tracking the elderly." User Modeling and User-Adapted Interaction 30, no. 3 (2020): 567-605.

[40] Jayasinghe, Udeni, Balazs Janko, Faustina Hwang, and William S. Harwin. "Classification of static postures with wearable sensors mounted on loose clothing." Scientific Reports 13, no. 1 (2023): 131.

[41] Bonnet, Cédrick T., and Boris Cheval. "Sitting vs. standing: an urgent need to rebalance our world." Health Psychology Review (2022): 1-22.

[42] Natalia Volkova, Machine Learning Approaches for Stock Market Prediction , Machine Learning Applications Conference Proceedings, Vol 2 2022.

[43] Balli, Serkan, Ensar Arif Sağbaş, and Musa Peker. "Human activity recognition from smart watch sensor data using a hybrid of principal component analysis and random forest algorithm." Measurement and Control 52, no. 1-2 (2019): 37-45.

[44] Hung, Wei-Chih, Fan Shen, Yi-Leh Wu, Maw-Kae Hor, and Cheng-Yuan Tang. "Activity recognition with sensors on mobile devices." In 2014 International Conference on Machine Learning and Cybernetics, vol. 2, pp. 449-454. IEEE, 2014.

[45] Ferreira, José M., Ivan Miguel Pires, Gonçalo Marques, Nuno M. Garcia, Eftim Zdravevski, Petre Lameski, Francisco Flórez-Revuelta, Susanna Spinsante, and Lina Xu. "Activities of daily living and environment recognition using mobile devices: a comparative study." Electronics 9, no. 1 (2020): 180.

[46] Pires, Ivan Miguel, Nuno M. Garcia, Eftim Zdravevski, and Petre Lameski. "Activities of daily living with motion: A dataset with accelerometer, magnetometer and gyroscope data from mobile devices." Data in brief 33 (2020): 106628.

[47] Vähä-Ypyä, H., P. Husu, J. Suni, T. Vasankari, and H. Sievänen. "Reliable recognition of lying, sitting, and standing with a hip-worn accelerometer." Scandinavian journal of medicine & science in sports 28, no. 3 (2018): 1092-1102.

[48] Mekruksavanich, Sakorn, and Anuchit Jitpattanakul. "Deep convolutional neural network with rnns for complex activity recognition using wrist-worn wearable sensor data." Electronics 10, no. 14 (2021): 1685.

[49] Gupta, Saurabh. "Deep learning based human activity recognition (HAR) using wearable sensor data." International Journal of Information Management Data Insights 1, no. 2 (2021): 100046.

[50] Oluwalade, Bolu, Sunil Neela, Judy Wawira, Tobiloba Adejumo, and Saptarshi Purkayastha. "Human activity recognition using deep learning models on smartphones and smartwatches sensor data." arXiv preprint arXiv:2103.03836 (2021).

[51] Mekruksavanich, Sakorn, Anuchit Jitpattanakul, Phichai Youplao, and Preecha Yupapin. "Enhanced hand-oriented activity recognition based on smartwatch sensor data using lstms." Symmetry 12, no. 9 (2020): 1570.

[52] Tensorflow lite conversion, URL: https://www.tensorflow.org/lite/models/convert

[53] Extrasensory dataset, URL: https://www.kaggle.com/datasets/yvaizman/the-extrasensory-dataset

[54] Wisdm dataset - Weiss,Gary. (2019). WISDM Smartphone and Smartwatch Activity and Biometrics Dataset . UCI Machine Learning Repository. https://doi.org/10.24432/C5HK59