Enhancing IMU-based smartphone context detection using Transformer

LE TRAN ANH DUC, University of Twente, The Netherlands

Deep learning models like Long Short Term Memory (LSTM) or Convolutional Neural Network (CNN) have yielded great results in the field of understanding smartphone context in recent years. In this paper, we propose using another deep learning model based on Transformer to tackle the same task with IMU sensors data. We will start by reviewing an overview of existing approaches for smartphone context detection. A new sensor-based dataset on smartphone context and step recognition is collected. Extensive experiments have been conducted to compare the accuracy of the Transformer model to other deep learning models like LSTM or CNN. Furthermore, the Transformer model used for smartphone context understanding also improves the efficiency of step counting.

Additional Key Words and Phrases: Transformer, Smartphone context detection, LSTM, CNN, IMU sensors, step detection.

1 INTRODUCTION

Smartphone context detection is the task of understanding the surrounding environment where the phone is placed, i.e, its relative locations and its postures using built-in sensors. Understanding smartphone context helps build a more responsive and adaptive system to the user's situation. This task is done by the usage of a wide range of provided sensors like inertial sensors (accelerometer, gyroscopes, and magnetometer), environmental sensors (pressure, temperature, light), cameras, or WiFi/Bluetooth signals. Understanding device placements can give great insights into real-world navigation behaviors [6]. A context recognition system can be incorporated into the software to deliver a better-personalized media recommendation system [11]. Moreover, the importance of this task has also been shown with a wide range of other applications like health tracking [22] or fall detection [4].

The Transformer [21] is a deep learning architecture introduced by the Google Brain team in 2017. Transformer consists of serial encoder and decoder blocks that are suitable for sequential inputs, which were intended for natural language processing but also usable for other kinds of data like image sequences or time-series data. Inside Transformer architecture, the positional encoding transforms the input into the vector with positional context information, and the self-attention mechanism helps to explore the relevance or attention of different parts of the input. These characteristics help the model understand the dependencies within the input space, which is not possible when using the CNN approach. At the same time, the attention mechanism also allows parallelization and reduces training time, unlike LSTM, which requires processing one word at a time. For those reasons, Transformer has been shown to yield great results in tasks outside natural language processing. With time-series data, Transformer works well with event forecasting [25, 27], anomaly detection [23] and classification [15]. Hence, this research proposes using Transformer architecture in the task of smartphone context detection with IMU sensor data.

In this research, we will investigate how efficient it is to use the Transformer deep learning model in the task of smartphone context detection. The contribution of this paper is as follows. First, an overview of existing smartphone context and step detection approaches will be reviewed as an introduction to fellow beginning researchers in the pervasive system. Secondly, a newly collected data set and training process will be introduced to highlight the accuracy of Transformer models compared to existing approaches like LSTM and CNN. Finally, the above Transformer model will be shown to have improved the accuracy of the task of step counting.

The following explains the outline of this paper. In section 2, an overview of existing approaches in smartphone context detection will be reviewed. Section 3 explains how the data is collected and pre-processed. Section 4 details the proposed Transformer model for smartphone context recognition. Section 5 describes the experiments. Finally, the paper ends with discussions and conclusions of the experiments in section 6.

2 EXISTING WORKS

The study of smartphone context detection is about classifying different relative positions of the smartphone on the human. Examples of these contexts are inside the bag, held and used in hand, or inside the purse. A study [3] in 2007 shows that women in major cities like Helsinki, New York, or Milan, on average, spend 61.06% of their carrying time putting their phone in the bag, while its counterpart has their phone inside their trousers for 60.10% of the time. The next commonly placed phone positions are in the upper body or used in hand. In existing research, smartphone context is used interchangeably with keywords like phone placement, phone position, phone location, pose, or user mode. Recognizing smartphone context is often studied as part of a larger topic of human activity recognition (HAR) or pedestrian dead reckoning (PDR). An early paper [13] classifies four different poses: bag, ear, hand, and pocket, using the Support Vector Machine (SVM) to build a pose-dependent model for predicting walking speed. A similar study [2] recognizes pockets, belts, hands, or bags and also uses SVM classifiers and Hidden Markov Chain (HMM) on hand-crafted features extraction to improve the accuracy of activity tracking. A study [18] for step detection algorithm uses a decision tree classifier for four classes of phone motion mode: static user, hand swinging, quasi-stable (texting, phoning, bag carrying), and irregular mode. This study also pre-processes the data by extracting features like signal energy, signal variances, or frequency. Smartphone carrying pattern is also used to improve indoor trajectory tracing with three modes of texting, phoning, or pocket [24]. A common strategy in early research in smartphone context understanding is to collect inertial measure sensor data like gyroscopes or accelerometers, extract various hand-crafted features like statistical features (mean, max, standard deviation) or time-domain features (skewness, magnitude, energy) and use traditional machine learning techniques to classify

TScIT 37, July 8, 2022, Enschede, The Netherlands

different phone positions. In recent years, the rise of deep learning techniques has outperformed existing approaches in this task, even when used with raw data. A study [9] on phone mode recognition shows that given the same training inputs, deep learning methods like CNN, Gated Recurrent Unit (GRU), or LSTM outperform in accuracy compared to tree-based techniques like the random forest, gradient boosting, or CatBoost. The task of classifying phones that are in the pocket, swinging, texting, or talking has shown high accuracy of around 92% with CNN or LSTM models even when training with only an accelerometer sensor [8]. Some studies have been really creative in combining different deep learning models to create complicated architecture that has shown excellent results on this task. Using a combination of the Inception module and bi-directional GRU model has yielded a high accuracy in studying human activity recognition, which is performed in different phone states [20]. A combination of CNN and LSTM outperforms the use of only LSTM or GRU models when being trained with accelerometer and gyroscope measurements [8].

The study of step detection using mobile phones or embedded devices IMU sensor information is a part of the larger study of Pedestrian Dead Reckoning (PDR). In PDR algorithms, step detection is the initial step before estimating step length, predicting the direction, and updating the position. Inertial sensor data like accelerometer or gyroscope [1, 14, 17, 19] are often used as the main input for detecting steps. Various algorithms like zero-crossing [16], flat zone detection [7] and most commonly peak detection [1, 14, 19] are used to solve this task. Melania shows that gyroscope peaks can be used to detect the swinging motion of the hand while holding the phone [19], which can be used to detect a step as the hand swinging motion is in sync with the leg stepping motion [12]. In other cases where the phone is being used for texting, calling, or being kept in the bag, the accelerometer's peak is an alternative for detecting a new step. Stéphane and Harald extracted time-series features from accelerometer data to train in a neural network for detecting steps. Itzik Klein et al. highlight that smartphone mode detection can help improve detecting steps. Hence it is an opportunity in this study to verify if the Transformer model can help detect steps.

3 DATASET

3.1 Data collection

The data collection experiments described in this section are conducted at the University of Twente, inside the open space of the Technohal building. The data collection is done by an Android application and Android phones. The decision to write a new Android application for data collection is to enable easy updates to the data collection process and help in deploying the trained model later for demo purposes. The Android is written in Kotlin, providing features like automatic labeling and a timer. This tooling is open source and can be found on https://github.com/anhanh11001/imu_transformer.

The data is collected through 5 phones from 3 brands (Samsung, Google Pixel, and HTC). There are five location context collected during the process:

- INSIDE THE PANT POCKET
- INSIDE THE BAG
- USING IN HAND (texting, surfing the internet)
- HOLDING IN HAND
- CALLING

At the same time, there are also 2 simple human-related labels: WALKING and STANDING. During the data collection process, each participant was asked to perform three 5-minute experiments of walking, standing, and a combination of walking and standing. The participant can walk in any direction and can change their walking speed. Moving around the open floor in the building includes walking in a straight line, turning left/right/around, avoiding other fellow students in the university or obstacles like tables, stopping at the coffee machine, or stopping to read an ad. At the beginning of the experiment, the participant will select the smartphone context and place the phone in the predefined context. When the timer starts counting, the participant will perform the corresponding action. When the timer finishes counting the experiment length, the application will notify by vibrating and playing a sound. The collected data will be extracted to the local storage of the mobile app in CSV format. Overall, five collected labels are evenly distributed and their length is around 75 minutes each.

The collected data come from the phone clock and inertial measurements sensors, which are accelerometers (including gravitational information), magnetometers, and gyroscopes. Besides the phone clock, each sensor provides data within three axes. These sensors provide information about relative changes in velocity or orientation of the phone body. The sampling rate is up to 20 milliseconds, depending on the phone state. For example, if the phone is turned off, the sampling rate will be reduced. However, we try to keep the phone on all the time to get the most data and remove any failed cases where the phone turned off accidentally.

During the data collection process, we also collect the information whenever a step is taken. This is done by manually recording whenever a new step is taken with another person using another Android application.

3.2 Data pre-processing

Even though the collected raw data is already labeled, it is still required to pre-process those data to use them for training with deep learning models. The first step of pre-processing is to visualize the data and find the error in the data to remove them. For example, there were a few experiments where the sampling rate dropped and produced a flat curve in the plots, requiring data recollection. For context detection, the formatted input is a fixed-size window of sensor data, and the formatted output is the numerical decimal representing the label. The raw input is divided into a window length of 2 seconds with 40 small chunks in each window. Each consecutive window has an overlap of 1 second. Each chunk represents a snapshot of IMU sensor information, and two consecutive chunks are 50 milliseconds in difference. The formatted input has the shape (40, 9), with 40 being the number of chunks and 9 being the number of input data (3 sensors with 3 axes each). After the input is formatted, a few techniques have been tried to improve the training process.

The first pre-processing technique used is normalization. In this project, normalization was done using the min-max scaler with the formula below. For every feature or column in the dataset, each value is scaled by taking its difference to the smallest value and comparing

Fig. 1. Top 10 extracted features using SVM.

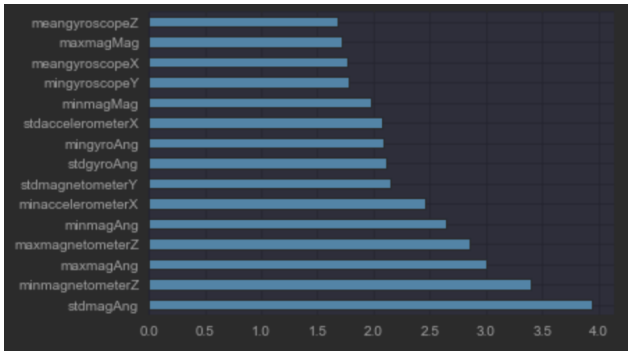


Table 1. Training accuracy with different number of extracted features using a simple CNN model.

Number of features	Accuracy
All features (60 features)	47.8%
Top 15 features	80.9%
Top 10 features	87.0%

it to the difference between the largest and the smallest values. The normalization formula is shown below. This scales all values into the range from 0 to 1. Other normalization methods had also been tried during the experiment, like L1 and L2, but did not yield good results compared to min-max normalization. Normalization aims to equalize the importance of different features while their value ranges differ.

$$min\_max\_scale(x) = \frac{x - min(data)}{max(data) - min(data)}$$

The second technique that has been tried is standardization. This technique scales each feature's value into a new data sample with a mean of 0 and a standard deviation of 1. Similar to normalization, standardization can be useful to avoid features with a higher range from dominating the decision during the training process.

$$standardized(x) = \frac{x - mean(data)}{std(data)}$$

Besides the two explained feature scaling techniques, the third pre-processing approach used is feature extraction and selection. From the existing features, we extracted two more features: the magnitude and the angle from the unit vector for each sensor (An IMU sensor data can be seen as a vector with 3 axes in 3D space). The magnitude is calculated as $magitude(x, y, z) = \sqrt{x^2 + y^2 + z^2}$. The angle is calculated with the formula $angle(x, y, z) = arcos((x + y + z)/(\sqrt{1 + 1 + 1} * \sqrt{x^2 + y^2 + z^2}))$. For each window of 2 seconds, the minimum, the maximum, mean, and standard deviation of its chunks' feature will be extracted. Some of its features can be useful to remove the noise in the data set. For example, the mean operation will smooth out the volatility and outlier in the data. The result is a list of 60 extracted features. The next step after feature extraction is feature selection using the Support Vector Machine (SVM). SVM helps to find the corresponding coefficient of each feature when mapping the dataset into the hyperplane that can classify different labels, and those coefficients show the importance of the features relative to the training process. We run a simple experiment of training a CNN model using a different number of extracted features and a subset of the data and found that using only the top 10 features shows the highest accuracy of 87% in table 1. Hence, we use those top 10 features in figure 1 in the next sections to compare with training on raw features.

4 TRANSFORMER-BASED MODEL

4.1 Model architecture

An overview of the model architecture is shown in figure 2 below. The design of the Transformer-based model closely follows the original design of Transformer [21] but removes the decoder part. Overall, the architecture consists of a stack of Transformer serial encoder blocks, followed by a few fully-connected layers and a softmax layer. The standard input of this architecture is a 1D-array sequence of numerical sensor values, and the standard output is a (5,) 1D-array of probabilistic results of labels. Inside each Transformer encoder block, there are an attention part and a feed-forward part, which is similar to the encoder part of the original Transformer model. The inputs, attention parts, and feed-forward parts are also connected by residual connections [5].

Attention Part: The attention part starts with a normalization layer, followed by a multi-head attention block and a dropout. The attention block is used to adjust the weights of each chunk inside the time-series window so that they will have an adjusted impact on the hidden states of downstream layers. This attention block is critical for the Transformer model to perform its long-range dependencies understanding property.

Feed-forward Part: The feed-forward part also starts with a layer normalization, followed by fully-connected layers and a dropout. This feed-forward part is used to better represent the output from the previous attention layer so that it can better fit the next attention part.

4.2 Training Transformer-based model

To train the proposed Transformer model, there are multiple variables to consider, like the number of transformer encoder blocks, the number of heads or head size in the multi-head attention block, or the number of fully-connected layers. To find an optimal configuration, we train the Transformer models on two small subsets, where subset 2 has twice the number of data compared to subset 1. Table 2 shows the detailed training configuration, containing the train-validation-test split of 70-15-15, batch size of 32, epoch number of 1000, and sparse categorical entropy for loss function. The models will be adjusted by changing one of these variables (number of heads, head size, number of Transformer encoder blocks) while keeping others the same. In these training trials, we observe that there is no specific set of (# heads, head size, # Transformer encoder block) that work best for all data, which is presented in figure 3. The training process shows that smaller models train faster and are less

Fig. 2. Transformer-based architecture for context detection.

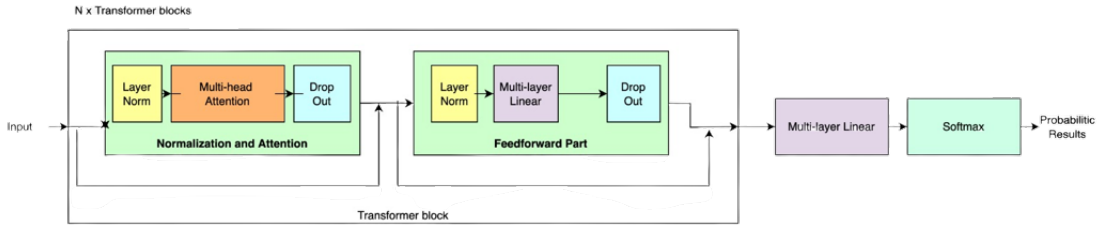


Table 2. Baseline configuration for training the Transformer-based context detection model.

Training Config	**Value**
Train/Val/Test Split	70/15/15
Optimizer	Adam
Batch size	32
Epoch	100 (subset) or 1000 (full dataset)
Loss function	Sparse Categorical-Crossentropy
Model Config	**Value**
Head size	32
# of heads	6
# of Transformer encoder block	2

Table 3. Test accuracy when using different sets of features on a small sub-dataset.

Features	**Test accuracy**
Acc only (raw) with CNN	98,9%
Acc + Mag (raw) with CNN	99,3%
All features (raw) with CNN	100%
Acc only (raw) with Transformer	32,3%
Acc + Mag (raw) with Transformer	75%
All features (raw) with Transformer	92.7%
Top 10 features (extracted) with Transformer	90.6%

likely to be over-fitted. Increasing model sizes by changing the number of Transformer encoder blocks or head sizes can improve the accuracy when being trained and tested with more data. Therefore, to find the best models for all collected training data, we decide to set the default configuration of (# heads, head size, # Transformer encoder block) to be (4, 64, 2) and keep increasing those variables to have bigger models in the experiment sections and observe the accuracy.

In smartphone context detection, selecting correct features can help speed up the training process by reducing the number of calculations. The baseline features used are accelerometers, gyroscopes, and magnetometers. The accelerometer can detect the changes caused by vibration or movement. The collected accelerometer also includes incorporated gravitational information, which can help understand the smartphone's relative posture compared to the earth. For instance, a phone lying on the desk will have its x-axis and y-axis values close to 0, while a phone inside the pant pocket might be perpendicular to the standing surface, hence making the x-axis value around 0. The gyroscope measures the angular motion and detects the rate of rotation of the phone around its axes, which can be helpful when the phone rotates in a certain pattern. For example, the phone swings in a pendulum-shaped motion when the person walks while holding the phone in their hand. We also test the accuracy of using different sets of features on a small subset of data (5 minutes of each label from one phone) with a simple Transformer model and a CNN model. The results show that models using all features from the three sensors significantly outperform those using one or two features. Moreover, using the top 10 extracted features from the proposed method in section 3.2 also shows high accuracy but is slightly worse than using all raw data. In table 3, CNN and Transformer models yield the highest accuracy results of 100% and 92.7% when being trained with all raw data. Hence, we decide to use all raw data as the main direction in the experiments section.

All training processes are performed with a Macbook Pro 2019, including a 2.4 GHz 8-Core Intel i9 and a graphic card of AMD Radeon Pro 5300M 4GB.

5 EXPERIMENTS

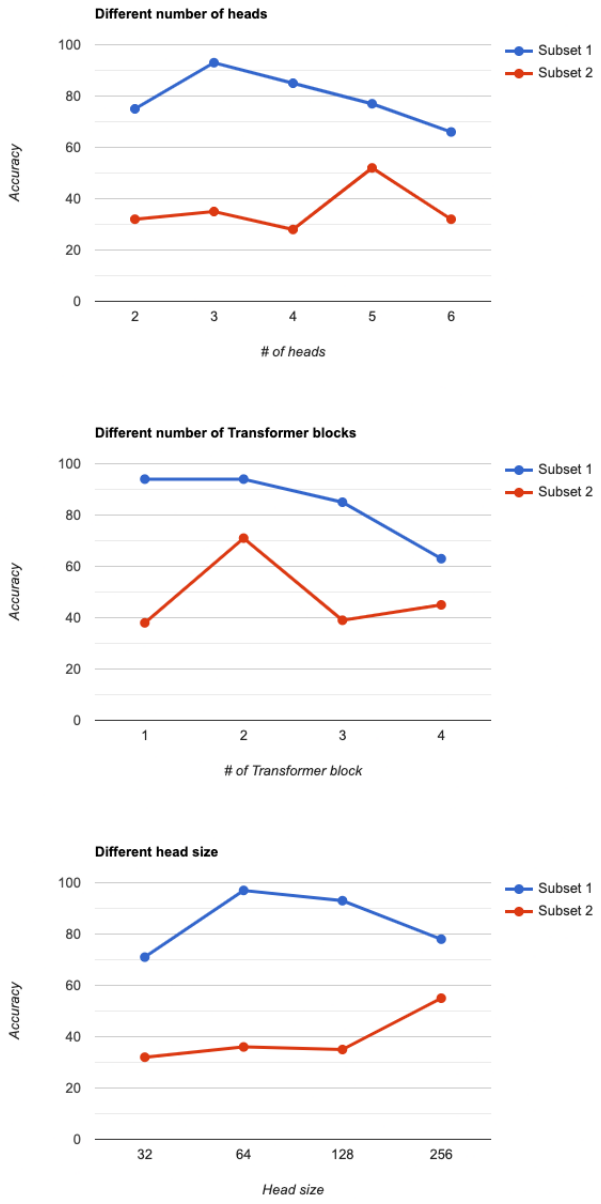
The experiments are set up to answer whether the Transformer-based model is feasible to be used in the task of smartphone context detection. To answer this, we set up simple CNN and LSTM models to compare with the proposed Transformer model. The CNN model contains a stack of three CNN modules, in which a module has a 1D-convolutional layer followed by batch normalization and ReLU activation. The LSTM model includes a simple LSTM layer with 60 units. Both models' architecture end with a fully-connected layer, dropout, and softmax to return probabilistic values of each label. We keep all three models to a similar size.

The experiments are set up with the collected data set to compare and gather the test accuracy. All experiments' detailed information, including the code for the training process, training logs, plots, and saved models, can be found on https://github.com/anhanh11001/context_-transformer.

5.1 Context detection while walking only

The first test conducted is done on the subset of context data collected when the participants only walk during the experiment. This data subset contains a total of 6120 2-second windows for training and 1076 windows for testing. These windows contain raw data

Fig. 3. Changing different configuration on the Transformer model



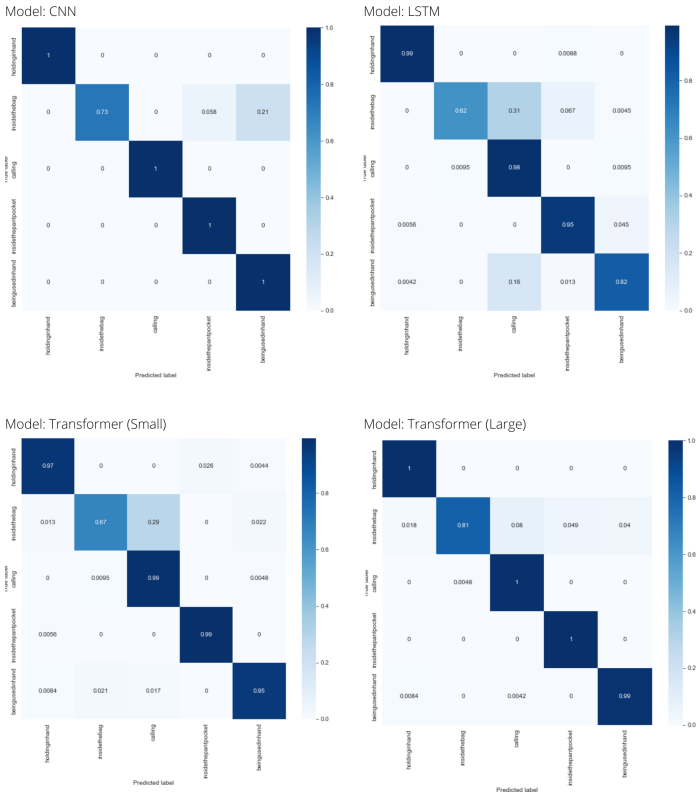
from all three sensors. All models perform processes with the same configuration.

In table 4, the result shows that CNN and Transformer models have outperformed the accuracy of the LSTM model with the results of 94.33% and 95.72% respectively. HOLDING IN HAND and CALLING are the two most easily detected positions, with all tested models showing accuracy above 97%, as shown in the figure 4. Among the five labels, INSIDE THE BAG is the label with the lowest accuracy rate of around 62%-81%. A possible explanation for why

Table 4. Test accuracy for context detection (walking only).

Model	Size (params)	Accuracy	F1 score
CNN	27,589	94,33%	94%
LSTM	23,405	86,62%	87%
Transformer (small)	9,767	91,17%	91%
Transformer (large)	**21,129**	**95,72%**	**96%**

Fig. 4. Confusion matrices for context detection (walking only)



INSIDE THE BAG is the most mispredicted label is due to the large amount of variances the phone position can have when being put inside the bag: the phone has more space to move around; different objects like paper, notebooks and pens can mix up the positions of the phone; the participant can wear the bag in different styles (using one or both straps).

5.2 Context detection while standing only

Similar to the subsection above, the second test is done on the subset of context data collected when the participants only stand during the experiments. This data subset contains 6377 windows for training and 1122 windows for testing. All models perform processes with the same configuration.

In table 5, the accuracy of detecting smartphone when standing ranges between 45,02% to 82,92%, which are lower than the scenario

Table 5. Test accuracy for context detection (standing only, 5 labels).

Model	Size (params)	Accuracy	F1 score
CNN	**27,589**	**82,92%**	**83%**
LSTM	23,405	67,7%	67%
Transformer (small)	9,767	57,92%	51%
Transformer (large)	21,129	45,02%	41%

Table 6. Test accuracy for context detection (standing only, 3 labels).

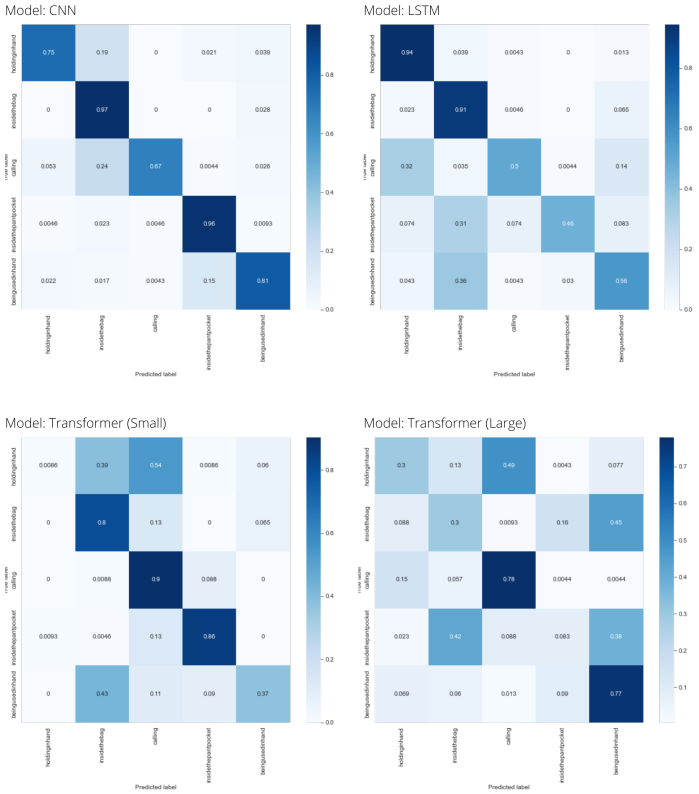
Model	Size (params)	Accuracy
CNN	**27,459**	**99,85%**
LSTM	23,203	96,02%
Transformer	9,509	94,01%

when the participants are walking. In figure 5, the Transformer and LSTM models performed high accuracy (around 80-90%) for labels INSIDE THE BAG, CALLING, and PANT POCKET while mispredicting the other two labels. Similarly, the CNN model shows a low error rate with INSIDE THE BAG and PANT POCKET. We observe that three pairs of labels tend to be mispredicted:

- Pair 1: HOLDING IN HAND and INSIDE THE BAG
- Pair 2: CALLING and INSIDE THE BAG
- Pair 3: USING IN HAND and PANT POCKET

Since the data is collected when the participants stand still, the changes in the values of the sensor are relatively small compared to the situation when the participant walks. As the participant does not move the smartphone much, there are not many significant changes in sensor values. The main distinguishing factor in recognizing the phone context is the angle created by the phone compared to the ground surface using accelerometer information (incorporated with gravitational information). All three pairs above are quite similar in how the phone is positioned. For instance, in pair 1, the phone's screen facing directions in HOLDING IN HAND and INSIDE THE BAG is parallel to the ground surface, making it difficult to predict the actual phone context. However, when the participant walks, the phone held in hand will swing a lot and create large variances in the gyroscope compared to being inside the bag, making it easy to distinguish. To confirm this, we run another test by comparing the accuracy of different models but only using data from three labels: HOLDING IN HAND (pair 1), CALLING (pair 2), PANT POCKET (pair 3). The result of this experiment is shown in table 6, which indicates that all three models yield high test accuracy above 94%, with CNN being the most performant model with 99,85% in accuracy. A possible explanation for why CNN is the best model is that because the data's changes over time are relatively small due to the stationary position of the participant, the advantage of understanding global dependencies and time translation invariance of LSTM and Transformer are not utilized. Moreoever, due to the locality inductive bias property, CNN is more suitable for finding local patterns among the sensors data.

Fig. 5. Confusion matrices for context detection (standing only)



5.3 Context detection with the combination of walking and standing

The third experiment is to compare the accuracy of the Transformer model with others when participants are allowed to walk and stop at their will. The data set contains 6457 2-second windows for training and 1048 windows for testing. These windows also have raw data from all three sensors.

In table 7, the CNN model is the most performant model with the highest accuracy of 99,85%, compared to the LSTM and Transformer models of around 93%. Training with a larger Transformer model slightly reduces the error rate of 0,28%. As explained in section 5.2, LSTM and Transformer models perform worse than the CNN model when the participant is stationary as the models' global dependencies and time translation invariance properties are not being utilized. Hence as part of the data having participant standing, the CNN model predicts better and yields a higher overall result.

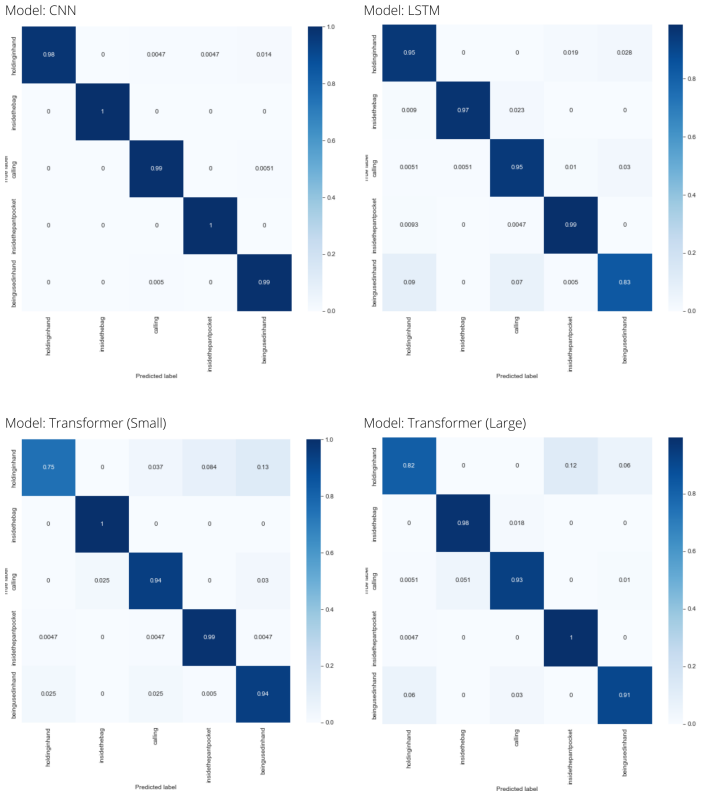
5.4 Using multi-task learning to improve the accuracy of Transformer model

In the fourth experiment, we investigate whether using multi-task learning can improve the accuracy of the Transformer model. By sharing similar hidden layers while keeping some output layers for specific tasks and training the model together with the same data,

Table 7. Test accuracy for context detection (standing and walking).

Model	Size (params)	Accuracy	F1 score
CNN	**27,589**	**99,33%**	**99%**
LSTM	23,405	93,00%	94%
Trans (small)	9,767	92,56%	92%
Trans(large)	21,129	92,84%	93%

Fig. 6. Confusion matrices for context detection (walking and standing)



this technique can improve the training accuracy as similar tasks can help avoid over-fitting and give better generalization [26]. In the dataset containing mixed data of both walking and standing, we manually label these two activity modes and train them together for activity mode recognition and context detection. The Transformer encoder blocks in the model are shared between these two tasks, while the final fully-connected layers are separated for each task. The loss function used for training is the weighted average of the two loss functions from those two tasks. In this experiment, we keep the Transformer models to have similar sizes and only adjust the loss weight ratio to observe the differences. For example, the weight 5-1 in the result table means the loss would be calculated as follows:

$$loss(total) = loss(context) * 5 + loss(activity) * 1$$

.

Table 8. Test accuracy of Transformer models trained with multi-task learning.

Model	Test accuracy
Multitask (weight: 5-1)	**91,44%**
Multitask (weight: 3-1)	89,90%
Multitask (weight: 3-2)	86,40%
Multitask (weight: 1-1)	86,75%
Without multitask	84,77%

In table 8, the Transformer models trained with multi-task learning have shown better results of 86,40%-91,44% in accuracy, compared to 84,77% of the model trained without multi-task learning. These results indicate that activity recognition task can be incorporated with training context recognition to generalize better. Another observation from the result is that increasing the loss weight of activity recognition too high can also decrease the accuracy of understanding smartphone context. For example, the most accurate multi-task learning model with the weight 5-1 is approximately 5% higher in accuracy compared to the weight 3-2 or 2-2.

5.5 Using Transformer model to improve step counting

In this experiment, we investigate whether understanding the smartphone context helps improve the accuracy of counting steps. We take a subset of input data from three following contexts: INSIDE THE PANT POCKET, INSIDE THE BAG, and HOLDING IN HAND. Similar to the context detection experiment, these input data will be divided into windows of 2 seconds, where each window contains 40 smaller 50ms chunks of sensor data. We determine whether each chunk is a step with the available labeled data. Two consecutive windows have an overlap of 250ms. The end output would be the number of steps counted in the 2-second-long window. With the provided input and output, we will train different models to compare the accuracy of counting steps, defined as the number and percentage of windows that have steps counted correctly. Finally, there are 9167 windows for training and 1626 windows for testing.

The baseline architecture used for comparing the task of counting steps is Shallow CNN architecture, inspired by the work of Long Luu et al. [10]. This architecture has one convolutional layer with 12 filters, a kernel of size 3, and a stride of 1. This layer is followed by a few drop out, max pooling 1D, and dense layers. The overview architecture is shown in figure 7. We train four models of this architecture with 4 sets of data: Pant Pocket (Model 1), Inside the Bag (Model 2), Holding in Hand (Model 3), and Combination of all those three data types (Model 4). Regarding the context detection, we use the same input data and follow a similar training process proposed in section 4.2 with the Transformer model. The overview of the nested model is shown in figure 8.

The first experiment we did was to compare the accuracy of step counting when the model is trained in one context and tested in another context. The second experiment we did was to compare the accuracy of step counting between a model trained with all data (Model 4) to a model using Transformer to detect the context and predict with the model trained with the corresponding context (Transformer + Model 1,2,3).

Fig. 7. Shallow CNN architecture for counting steps.

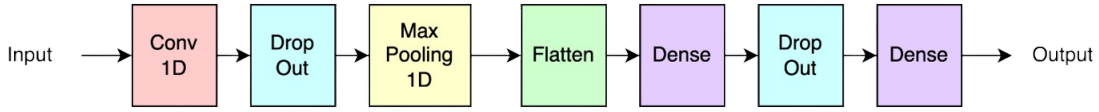


Fig. 8. Nested architecture of step counting using context detection with Transformer.

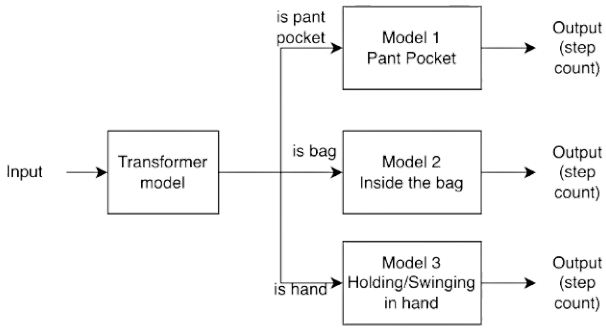


Table 9. Test accuracy for step counting when being trained on a different context.

Model	BAG test	HAND test	PANT test
Trained on BAG data	**75.24%**	61.65%	58.6%
Trained on HAND data	19.19%	**76.33%**	25.53%
Trained on PANT data	31.53%	23.12%	**75.63%**

Table 10. Test accuracy for step counting with/without context detection.

Model	Test accuracy
Without context detection	72.75% (1183/1626)
With context detection	**75.95% (1235/1626)**

In table 9, all three models perform with an accuracy of around 75% on the test set. However, when the models trained in one context are tested in a different context, they yield much higher errors. For example, a model trained with BEING HOLD IN HAND only predicts correctly once for every five windows of INSIDE THE BAG context. Similarly, the model trained with INSIDE PANT POCKET is only correct a quarter of all its prediction with the test dataset from BEING HOLD IN HAND context. Hence, this shows that step counting does not generalize greatly to different contexts and requires context understanding. In table 10, the accuracy of detecting steps by incorporating context detection is 75.95%, which is a slight improvement of 3% compared to the baseline model that does not use context detection. The trained Transformer model yields 96,8% accuracy in detecting context, where the most error comes from mis-detecting the HOLDING IN HAND to INSIDE THE BAG. Of the three contexts, counting steps while the phone is put inside the bag is the most difficult, with a success rate of 74.45%. From this experiment, it has been shown that using the Transformer context detection model can help improve the accuracy task of step counting.

6 CONCLUSIONS

In conclusion, we have collected a new dataset for the smartphone context and proposed a simple Transformer-based model that achieves high accuracy compared to existing approaches like LSTM and CNN. Even though the Transformer model takes a long time to train and fine-tune, it achieves high accuracy on the task of context detection in the scenario of the person walking, with the highest accuracy of 95.72%. The proposed Transformer model has also been shown to improve the accuracy of step counting. We conclude this study by publishing all the code and experiments log in https://github.com/anhanh11001/context_transformer. Further work can be done to improve the existing model by testing different variations of Transformer architecture to improve the accuracy and efficiency of using an embedded device.

7 ACKNOWLEDGEMENTS

I am immensely grateful to my supervisor Mr. Nguyen Minh Son, for the helpful feedback and comments that significantly improved my research project and paper. I thank Dr. Le Viet Duc and fellow student researchers in the Embedded Machine Learning track for the insightful weekly sessions and discussions. I also want to show my appreciation to Liping Yin, a fellow student in the Embedded Machine Learning track, for providing me with an excellent introduction and support for step detection. Finally, I would love to thank the support from the Pervasive System group from the University of Twente for providing me with the tooling (mobile phones) during the process of the data collection process.

REFERENCES

[1] Ahmad Alabadleh, Eshraq Hawari, Esra'a Alkafaween, and Hamad Alsawalqah. 2017. Step Detection Algorithm For Accurate Distance Estimation Using Dynamic Step Length. 324–327. https://doi.org/10.1109/MDM.2017.52

[2] Stephen A Antos, Mark V Albert, and Konrad P Kording. 2014. Hand, belt, pocket or bag: Practical activity tracking with mobile phones. *J. Neurosci. Methods* 231 (July 2014), 22–30.

[3] Yanqing Cui, Jan Chipchase, and Fumiko Ichikawa. 2007. A Cross Culture Study on Phone Carrying and Physical Personalization. In *Usability and Internationalization. HCI and Culture*, Nuray Aykin (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 483–492.

[4] Abdul Hakim, M. Saiful Huq, Shahnoor Shanta, and B.S.K.K. Ibrahim. 2017. Smartphone Based Data Mining for Fall Detection: Analysis and Design. *Procedia Computer Science* 105 (2017), 46–51. https://doi.org/10.1016/j.procs.2017.01.188 2016 IEEE International Symposium on Robotics and Intelligent Sensors, IRIS 2016, 17-20 December 2016, Tokyo, Japan.

[5] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 770–778.

[6] Robert Jackermeier and Bernd Ludwig. 2021. Smartphone-Based Activity Recognition in a Pedestrian Navigation Context. *Sensors* 21, 9 (2021). https://doi.org/10.3390/s21093243

[7] Hojin Ju and Chan Gook Park. 2018. A pedestrian dead reckoning system using a foot kinematic constraint and shoe modeling for various motions. *Sensors and Actuators A: Physical* 284 (2018), 135–144. https://doi.org/10.1016/j.sna.2018.09.043

[8] Itzik Klein. 2020. Smartphone Location Recognition: A Deep Learning-Based Approach. *Sensors* 20, 1 (2020). https://doi.org/10.3390/s20010214

[9] Itzik Klein, Yuval Solaz, and Rotem Alaluf. 2018. Robust Smartphone Mode Recognition. In *2018 IEEE International Conference on the Science of Electrical Engineering in Israel (ICSEE)*. 1–5. https://doi.org/10.1109/ICSEE.2018.8646011

[10] Long Luu, Arvind Pillai, Halsey Lea, Ruben Buendia, Faisal M. Khan, and Glynn Dennis. 2021. Accurate Step Count With Generalizable Deep Learning on Accelerometer Data. In *2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. 192–196. https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech52372.2021.00042

[11] Abayomi Moradeyo Otebolaku and Maria Teresa Andrade. 2016. User Context Recognition Using Smartphone Sensors and Classification Models. *J. Netw. Comput. Appl.* 66, C (may 2016), 33–51. https://doi.org/10.1016/j.jnca.2016.03.013

[12] Jaeheung Park. 2008. Synthesis of natural arm swing motion in human bipedal walking. *Journal of Biomechanics* 41, 7 (2008), 1417–1426. https://doi.org/10.1016/j.jbiomech.2008.02.031

[13] Jun-geun Park, Ami Patel, Dorothy Curtis, Seth Teller, and Jonathan Ledlie. 2012. Online Pose Classification and Walking Speed Estimation Using Handheld Devices. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (Pittsburgh, Pennsylvania) *(UbiComp '12)*. Association for Computing Machinery, New York, NY, USA, 113–122. https://doi.org/10.1145/2370216.2370235

[14] Cliff Randell, Chris Djiallis, and Henk Muller. 2005. Personal Position Measurement Using Dead Reckoning. 166– 173. https://doi.org/10.1109/ISWC.2003.1241408

[15] Marc Rußwurm and Marco Körner. 2020. Self-attention for raw optical Satellite Time Series Classification. *ISPRS Journal of Photogrammetry and Remote Sensing* 169 (11 2020), 421–435. https://doi.org/10.1016/j.isprsjprs.2020.06.006

[16] Jungryul Seo, Yutsai Chiang, Teemu Laine, and Adil Khan. 2015. Step counting on smartphones using advanced zero-crossing and linear regression. *ACM IMCOM 2015 - Proceedings* (01 2015). https://doi.org/10.1145/2701126.2701223

[17] Cristina Soaz and Klaus Diepold. 2016. Step Detection and Parameterization for Gait Assessment Using a Single Waist-Worn Accelerometer. *IEEE Transactions on Biomedical Engineering* 63, 5 (2016), 933–942. https://doi.org/10.1109/TBME.2015.2480296

[18] Melania Susi, Valérie Renaudin, and Gérard Lachapelle. 2013. Motion Mode Recognition and Step Detection Algorithms for Mobile Phone Users. *Sensors (Basel, Switzerland)* 13 (2013), 1539 – 1562.

[19] Melania Susi, Valérie Renaudin, and Gérard Lachapelle. 2013. Motion Mode Recognition and Step Detection Algorithms for Mobile Phone Users. *Sensors* 13, 2 (2013), 1539–1562. https://doi.org/10.3390/s130201539

[20] Lina Tong, Hanghang Ma, Qianzhi Lin, Jiaji He, and Liang Peng. 2022. A Novel Deep Learning Bi-GRU-I Model for Real-Time Human Activity Recognition Using Inertial Sensors. *IEEE Sensors Journal* 22, 6 (2022), 6164–6174. https://doi.org/10.1109/JSEN.2022.3148431

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[22] Praneeth Vepakomma, Debraj De, Sajal K. Das, and Shekhar Bhansali. 2015. A-Wristocracy: Deep learning on wrist-worn sensing for recognition of user complex activities. In *2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. 1–6. https://doi.org/10.1109/BSN.2015.7299406

[23] Hongchun Yuan, Zhenyu Cai, Hui Zhou, Yue Wang, and Xiangzhi Chen. 2021. TransAnomaly: Video Anomaly Detection Using Video Vision Transformer. *IEEE Access* 9 (2021), 123977–123986. https://doi.org/10.1109/ACCESS.2021.3109102

[24] Pengyan Zhang, Xiaojiang Chen, Xiaoqiang Ma, Yanyan Wu, Hongbo Jiang, Dingyi Fang, Zhanyong Tang, and Yang Ma. 2017. SmartMTra: Robust Indoor Trajectory Tracing Using Smartphones. *IEEE Sensors Journal* 17, 12 (2017), 3613–3624. https://doi.org/10.1109/JSEN.2017.2692263

[25] Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. 2020. Self-Attentive Hawkes Process. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 11183–11193. https://proceedings.mlr.press/v119/zhang20q.html

[26] Yu Zhang and Qiang Yang. 2018. An overview of multi-task learning. *National Science Review* 5 (01 2018), 30–43. https://doi.org/10.1093/nsr/nwx105

[27] Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. 2020. Transformer Hawkes Process. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 1084, 11 pages.