Peer-reviewed version available at Sensors 2020, 20; doi:10.3390/s20133803

Deep Sensing: Inertia and Ambient Sensing for Activity Context Recognition using Deep Convolutional Neural Networks

Abayomi Otebolaku[1], Timibloudi Enamamu[1], Ali Alfouldi[3], Augustine Ikpehai[2], and Jims Marchang[1]

[1] Department of Computing, Sheffield Hallam University, Sheffield, United Kingdom; a.otebolaku@shu.ac.uk, t.enamamu@shu.ac.uk, jims.marchang@shu.ac.uk
[2] Department of Engineering and Mathematics, Sheffield Hallam University, Sheffield, United Kingdom; a.ikpehai@shu.ac.uk
[3] College of Computer Science & Information Technology, University of Al-Qadisiyah, Iraq; a.salfouldi@qu.edu.iq

* Correspondence: a.otebolaku@shu.ac.uk;

Abstract: With the widespread of embedded sensing capabilities of mobile devices, there has been unprecedented development of context-aware solutions. This allows the proliferation of various intelligent applications such as those for remote health and lifestyle monitoring, intelligent personalized services, etc. However, activity context recognition based on multivariate time series signals obtained from mobile devices in unconstrained conditions is naturally prone to imbalance class problems. This means that recognition models tend to predict classes with the majority number of samples whilst ignoring classes with the least number of samples, resulting in poor generalization. To address this problem, we propose to augment the time series signals from inertia sensors with signals from ambient sensing to train deep convolutional neural networks (DCNN) models. DCNN provides the characteristics that capture local dependency and scale invariance of these combined sensor signals. Consequently, we developed a DCNN model using only inertial sensor signals and then developed another model that combined signals from both inertia and ambient sensors aiming to investigate the class imbalance problem by improving the performance of the recognition model. Evaluation and analysis of the proposed system using data with imbalanced classes show that the system achieved better recognition accuracy when data from inertial sensors are combined with those from ambient sensors such as environment noise level and illumination, with an overall improvement of 5.3% accuracy.

Keywords: Activity Context Sensing; Smartphones; Deep Convolutional Neural Networks; Smart devices

1. Introduction

According to the World Health Organization (WHO), insufficient physical activity is one of the leading risk factors for death worldwide [1]. This could lead to non-communicable illnesses such as cardiovascular diseases, cancer, diabetes, and many more. Physical activity is defined as "any bodily movement produced by skeletal muscles that require energy expenditure including activities undertaken while working, carrying out household chores, traveling and engaging in recreational pursuits"[1]. To improve the physical wellbeing of people and reduce pressure on health infrastructure and the cost of healthcare delivery, governments now encourage people to engage in various forms of physical activities. In this regard, various research works have been conducted to provide solutions that support physical activities. Besides, being able to predict or recognize user activity contexts is not only important in health monitoring applications but such information can

also be used in designing and implementing other intelligent applications in transportation, security, and intelligent recommendation systems [2,3,7,39]. This is now inevitable because recent advances in ubiquitous computing, Cloud computing, Artificial Intelligence (AI), and development in network solutions such as the 5G, etc. are opening up greater opportunities. Not only that most of us now carry or use devices that come with an incredible capability for sensing human activities and contexts but also these devices come with the capability to provide solutions that promote and improve human general wellbeing. Besides, there are smart objects everywhere that can interact with our living spaces, producing an incredible amount of data [2]. Exploiting this data for building more intelligent applications has seen keen academic and industrial interests [2, 7]. One of the key research interests is how to use this data to identify meaningful information not only about mobile users but also of the space in which we live. Therefore, to provide such solutions, researchers in the last decade have investigated various approaches for recognizing human activity contexts by collecting a large volume of data from body-worn devices or from smartphones and other sensory devices to develop automated solutions using various AI techniques [2,4-6].

Activity context recognition is one of the techniques that has been widely used to study human behaviors such as walking, running, driving, eating, jogging, running patterns, etc. [2-7]. Having an understanding of the patterns of these behaviors, more and more intelligent applications in the domain of mobile healthcare systems, information systems such as service recommendation systems, etc. are now a reality [3,7]. However, recognizing activity context despite the impressive efforts and results by enthusiastic researchers still has some significant challenges. One such challenge which has not been adequately addressed is that of class imbalance [2]. It is common with some human activities involving human behavioral monitoring, which occur often e.g. sleeping and others that occur infrequently such as climbing stairs at home. This problem particularly is common with an unconstrained collection of activity data. One of the approaches used to address this problem is duplication also referred to as oversampling of imbalanced classes [2].

Another challenge is that current approaches in the realm of human activity context recognition have largely focused on identifying individual activity by using handcrafted approaches to extract useful features from the collected data [8]. Feature extraction is one of the key steps in activity context recognition [9]. It captures such information that discriminates various activity contexts. In our previous work we have reported extensively on the traditional approaches used in our activity context recognition applications [3]. One of the key weaknesses of the approach used in that project and many others was based on laborious handcrafting of various statistical features from the data before these features are now fed into the traditional machine learning algorithms for context recognition or classification.

Recently, deep learning algorithms [4-6,8,16] have achieved unparalleled performance in several areas such as image processing, visual object recognition, natural language processing, driverless cars, robots, etc. DNN is now being used for the development of automatic human activity context recognition [4-6, 8, 10-15, 38, 40]. This is because it does not require the use of laborious and error-prone handcrafted feature extraction processes. It thus has led to the growing trend of representational learning from raw sensor data using DCNN [10-15]. Besides, deep learning algorithms have the capability for unsupervised and incremental learning because of its deep network structure compared to the traditional neural network. DCNN has become a well-known deep learning architecture among other deep learning models. DCNN is composed of multiple

building blocks such as convolutional layers, pooling layers, and fully connected layers [8]. It has been designed to automatically and adaptively learn spatial hierarchies of features, from low to high-level patterns, through backpropagation algorithm [15, 30]. Its attraction is due to its special architecture with a very powerful ability to learn filters and apply them to small-sub regions of data. This unsupervised feature learning, which is performed in the convolution layers, allows them to easily capture hidden local patterns and variations in the data. The resulting feature map is then passed to the fully connected layers for activity context classification. The convolutional layers are trained alongside other layers of the network as their outputs serve as the inputs of other CNN layers. The convolutional operation exploits effectively the local temporal dependency of time series data while its pooling operation cancels the impact of small translation of the input. With its weight sharing feature, the convolution operation of the DCNN allows reservation of scale invariance, which in activity context recognition can discriminate between two similar or identical classes. This operation also helps to capture local dependencies of the signals. For example, it would be able to capture the dependencies between inertia sensing signals and those of nearby ambient sensors. It also lowers the computational cost by reducing the number of connections between convolutional layers [4,8,10]. With the capability to be optimized using backpropagation, it is an excellent deep learning architecture that produces minimal prediction error.

The traditional activity context recognition system as depicted in Figure 1 consists of key processing steps. Data collection, data filtering, data pre-processing such as segmentation, handcrafted features extraction, model training, and activity context classification. One major problem of this approach is that this traditional approach struggles to identify accurately activity patterns that are very similar such as walking or running! Another problem is that because this traditional approach of activity context recognition relies on handcrafted feature extraction, it is prone to recognition errors as well as not having the ability to generalize.

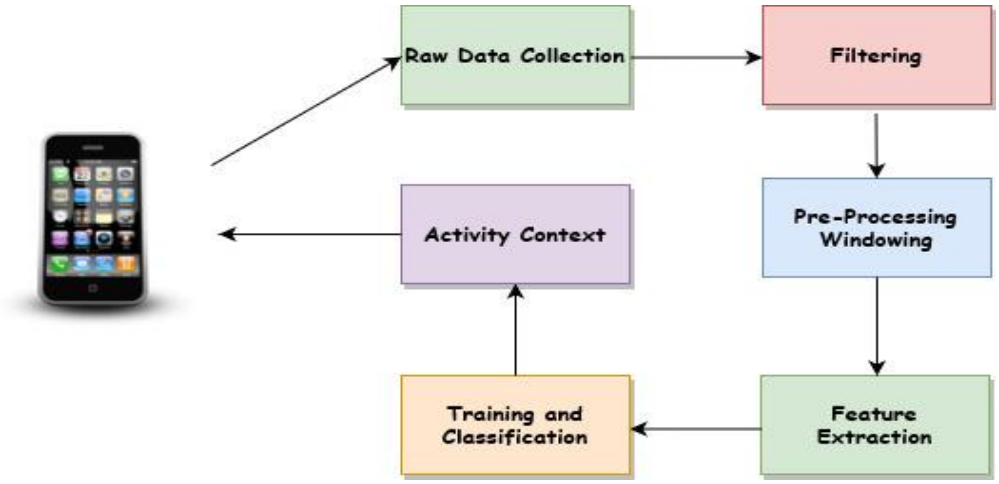

Figure 1. Classical Activity Context Recognition Processes

Deep learning-based approaches address this problem. Two key challenges have not been addressed by existing solutions that explore the Deep Neural Network for Human activity recognition. Firstly, DNN based activity recognition models are not flexible enough to recognize high-level activities. Current models can only recognize simple atomic daily activities. It is difficult

to determine the hierarchical structure of high-level activities because they contain more semantic and context information [9]. Secondly, the current solutions do not exploit ambient context information. To provide richer contextual information using both human activities and ambient information will improve the performance of such systems for human activity contexts in complex ambient environments. Most research works using Deep Neural Networks have focused on using data from video cameras [18] or inertia sensors such as accelerometers, and gyroscopes [8,10, 14-15]. Ambient sensing has been largely ignored. Ambient sensing is used to capture interactions between humans and the environment [12]. Belapurka et.al [2] made a very strong case for using ambient sensing for recognizing human activity contexts. However, they only proposed it as a means of tackling privacy-related problems of human activity context recognition. Ambient sensors are usually embedded in the environment and examples include temperature, light, sound, pressures sensors, etc. But modern mobile devices such as smartphones have these sensors and they are important sources of data that could be explored to improve the performance of human activity recognition models. We have therefore proposed to investigate the possibility of addressing class imbalance problem by enriching the classical inertia sensing data with ambient sensing data such as illumination and noise level. Figure 2 illustrates the combination of sensing data from both inertia and ambient sensor as inputs to the CNN algorithm.

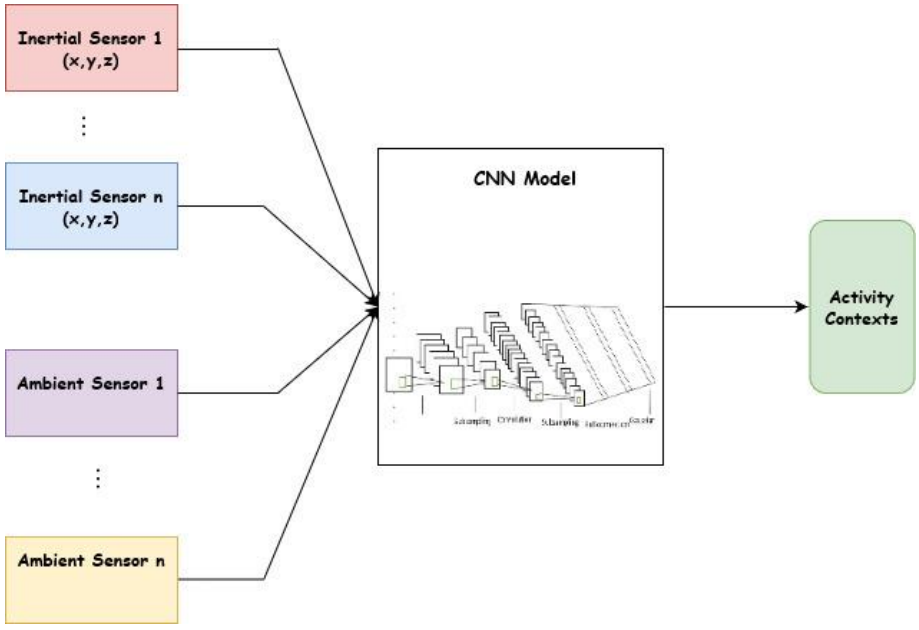

Figure 2. Inertia and Ambient Sensing Data

The key contributions of this article are threefold:

1) We demonstrate that inertia with ambient sensors namely, environment noise level and illumination could improve recognition performance using data with imbalanced classes
2) We performed extensive hyperparameter tuning to select optimal values to build the DCNN model
3) We demonstrate that DCNN can perform better recognition with raw sensing data without handcrafted features than with manually extracted features.

The rest of the paper is organized as follows. Section 2 presents relevant related work. In section 3, we present details of the proposed system for classifying context from raw sensor data. Section 4 presents our experiments and evaluation results. In section 5, we conclude and outline our future work.

2. Related Works

Human activity contexts are important contextual information especially in the new ubiquitous computing environments. This type of contextual information will play an important role in our daily life through various intelligent applications. Human activity contexts coupled with the emergence of the Internet of Things as the de facto means for gathering huge volumes of data relating to the human environment and their behaviors is revolutionizing how we engineer intelligent systems. Also, the new paradigm of emerging networked infrastructures that is enabling billions of interconnected devices to communicate and exchange information is the future of intelligent applications in various domains such as health monitoring, enhanced retail recommendation applications, smart homes, and smart cities. To engineer such systems, there is a need to provide an automatic way of recognizing and classifying human activity context. The process for automatic recognition of human activity context is generally known as Human Activity Recognition (HAR) [13, 14-15]. This is a typical pattern recognition problem based on using traditional algorithms such as support vector machines, K-Nearest Neighbor, naïve Bayes, decision trees, random forest, etc.

Research activities in human activity context can be broadly categorized into two. Video-based human activity recognition and sensor-based activity recognition [7, 15-17]. The sensor-based activity recognition processes have focused more on using data generated by inertial sensors such as accelerometer and gyroscope, for recognizing human locomotive activities by either placing these sensors in various parts of the body or using smartphones [15, 19-20,32]. The video-based human activity recognition has focused on using video surveillance data in the activity recognition processes [30]. In recent years, many research works have explored various algorithms whilst building new ones to automatically identify human activities. The conventional machine learning algorithms have been extensively explored and widely reported in the literature [2, 7, 15]. For example, in our previous work, we explored various traditional classification algorithms for automatic context recognition [3]. The result was applied in the development of a context model for an intelligent context-aware recommendation system. Other works based on classical machine learning algorithms and handcrafted future extraction processes have been extensively reported [7,9,17, 20, 22]. For example, authors in [18] proposed a new approach using a descriptor-based approach to human activity recognition. They have handcrafted time and frequency domain features extracted from accelerometer and gyroscope signals and used conventional support vector machines and k-nearest neighbor algorithms. In [21], Straczkiewicz & Onnela provide a comprehensive review of several human activity recognition research works using classical machine learning algorithms. The majority of the reported works using traditional machine learning algorithms are based on handcrafted feature extraction processes. Zeng et. al [8] report that although these works might have demonstrated good performance recognizing one activity however, they perform poorly recognizing others, due to class imbalance. They also noted that these works are not able to capture local dependencies of an activity signal as well as not able to preserve scale invariance. This explains why some models struggle to discriminate between jogging and running contexts.

In recent years, several works have also focused on using deep neural networks for activity recognition using signals from only inertia sensors [4-6, 8, 10-16, 36, 38, 40]. This new development is due to the incredible advancements in compute power, resulting in the resuscitation of the otherwise comatose field of neural networks. Several deep learning techniques have therefore been developed. For example, one of the earliest works is the one presented by Jiang &Yin [10]. In their paper rather than exploring handcrafted features from time-series sensor signals, they assembled signal sequences of accelerometers and gyroscopes into a novel activity image, and then, using this data to train Deep Neural Networks (DCNN) to automatically learn the optimal features from the activity for the activity recognition task. Another important and interesting work is the one presented by Zeng et. al [8]. They have also developed a system that automatically extracts features from raw sensing data using CNN with partial weight sharing. Another interesting work is the one presented by Zebin et.al [23], where signals from inertia sensors have been used to train DCNN for automatic feature extraction and activity recognition.

Some works combine statistical features with deep learning to automatically recognize human activities. For example, Hassan et. al [13] present a robust human activity recognition system using smartphone sensors and deep learning algorithms. In that work, from gyroscope and accelerometer data, they have extracted statistical features such as mean, median, autoregressive coefficients, etc., which are then fed into the Deep Neural Networks. Similarly, Ignatov [14] proposed a deep neural network architecture that combines shallow DCNN for unsupervised feature extraction together with statistical features to encode global characteristics of smartphone sensor data. Roan & Cho [15] proposed a DCNN based recognition system where they fed raw inertial sensor data into the DCNN model for automatic feature extraction. To improve the performance of the model, they combined manually extracted fast Fourier Transform of HAR dataset.

Researchers have also come up with an innovative way to identify human activity context using data from sensors other than the traditional inertial or motion sensors to augment these classical sensing data. The rationale is to address the problems associated with using inertial sensors to improve activity context recognition performance. Other researchers such as Belapurka et.al [28] made a very strong case for using ambient sensing for recognizing human activity contexts. However, they only proposed it as a means of tackling privacy-related problems of human activity context recognition. Some also used ambient sensing to address the computation complexity, power consumption, cost and recognition accuracy, or poor generalization issues. An example of such work is the one presented by Golestani & Moghaddam [24], where they introduced magnetic induction based human activity recognition to effectively detect physical movements using magnetic induction signals rather than inertial sensors signals. They compared the performance of their work using traditional machine learning algorithms such as SVM, KNN, etc. with deep learning algorithms such as deep long short-term memory (LSTM) and concluded that deep learning outperformed the traditional algorithms.

In terms of combining ambient data with inertial sensing data for activity recognition, only a few works have reported this approach. One of the latest reports is the one by Cruciani et. al [11] in which they have used audio and inertial datasets to pre-train a DCNN model for automatic human activity recognition. Another recent work is presented by Schrader et al [25], which uses audio signals and cameras as ambient sensors in addition to other sensors to recognize elderly people's activities for rehabilitation and early intervention. We have proposed a combination of 3 inertia sensors namely:

accelerometer, gyroscope, and magnetometer. This data is combined with ambient data from environment illumination and noise level data. We investigate the importance of ambient sensing in combination with inertia sensors to address the class imbalance problem of human activity context recognition. Like some of the works reviewed above, we have combined inertia and ambient sensing to recognize human activity contexts using deep convolutional neural networks for automatic feature extraction and a fully connected neural network for activity context classification.

3. Activity Context Recognition System Using Convolutional Neural Networks

In this section we present the CNN based model for activity context recognition using inertial and ambient sensing signals. We also present the architecture of the system, explaining each of the layers and the rationale of the architecture.

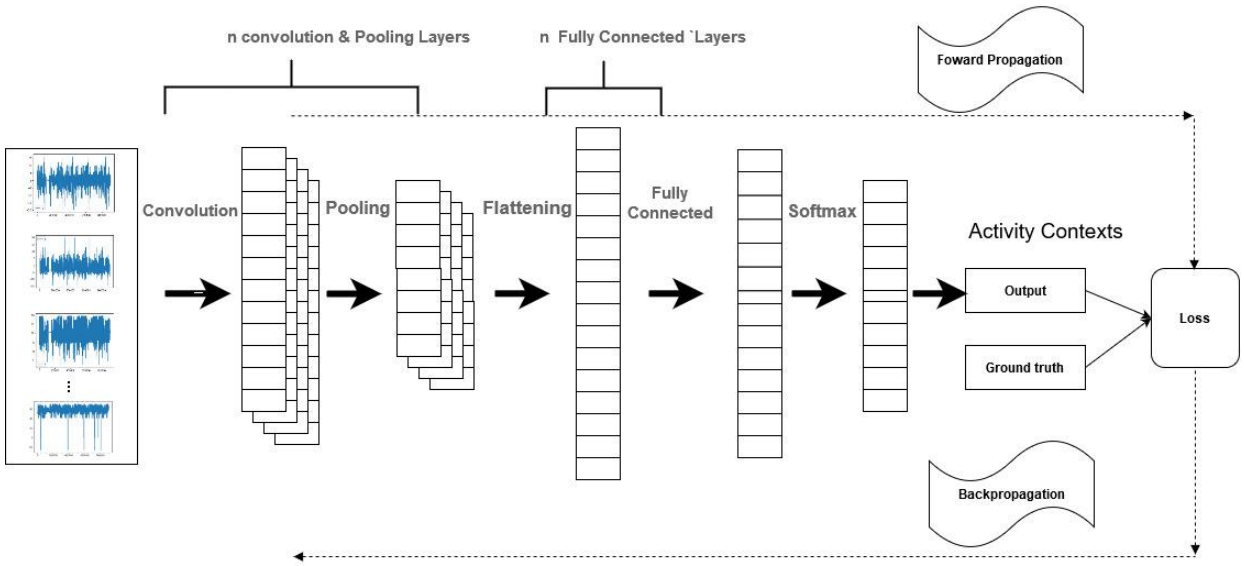

Figure 3. Architecture of the Proposed System

3.1 System Architecture

The architecture of the proposed DCNN model, as illustrated in Figure 3, consists of 3 types of layers namely: the convolutional and pooling layers, the flattened and the fully connected layer, and the output layer. Note that in Figure 3, we have denoted the number of layers with n. The value is determined later in section 4. In the first layer, data from both inertia and environment noise and illumination are fed into this layer consisting of 3 convolutional layers stacked with corresponding max-pooling layers and ReLU activation function [31]. Preprocessing such as filtering and feature extraction are executed in this layer, with the output being feature maps representing the activity context classes. We arrived at the number of convolution layers after a series of experiments to determine the number of convolutional layers with the best recognition accuracy, details are provided in section 4.3.1. The second layer consists of a flattened layer and a fully connected layer. The flattened layer accepts the feature maps from the previous layer (max-pooling layer) to convert the feature maps to a single column vector that is then passed to the fully connected layer to perform the classification process. The final layer, i.e., the output layer is a Softmax layer that receives the outputs of the fully connected layer to compute the probability distribution for each activity context

class [14-15]. The model has been trained to minimize cross entropy errors with L2 regularization and dropout probability to prevent overfitting. We have optimized the hyperparameters of the model using Adam modification of the stochastic gradient descent and backpropagation to compute the gradient of the loss function [29]. In the next section, we describe in detail starting with the data pre-processing, as well as the layers of the architecture.

3.2. Deep Convolutional Neural Network Model for Activity Context Recognition

In this section, we present our CNN-based feature extraction models for inertial and ambient data for activity context recognition. Starting with raw data standardization and data segmentation using a sliding window. Then we describe the CNN model. Figure 4 illustrates the activity context recognition process from data pre-processing to context classification using convolutional neural networks.

3.2.1 Raw data pre-processing

Before feeding the input data into the CNN algorithm, two important preprocessing were carried out, namely data standardization and segmentation using sliding windows with overlapping [27].

a. Data Standardization

Another pre-processing performed on the dataset is the data standardization. This is important especially when we are building models with gradient descent. Standardization helps to transform components of our input vector in such a way that the result has a unit variance of the entire dataset. This is done for each data point by subtracting the mean from the original value and then dividing the result by its standard deviation as illustrated in equation (1). The importance of this process is to minimize bias [31]. Thus each input 3D inertial signals and 1D ambient signals were standardized.

$$\frac{x-\bar{x}}{\sigma} \quad (1)$$

b. Segmentation

In the first phase, we first normalize the data. As explained above. This is then followed by the segmentation process known as a fixed-length temporal sliding window algorithm [7]. This algorithm splits the sensor data into data segments as illustrated in Figure 5 of fixed intervals of samples called "windows". A window contains a small part of the sensor signal [27]. Each window is "*overlapped*" or "*slid*" to form the next window, as illustrated in Figure 5, preserving a proportion of the previously sampled signal data for the beginning of the next sample [3, 9,25].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2020 doi:10.20944/preprints202005.0430.v1

Peer-reviewed version available at Sensors 2020, 20; doi:10.3390/s20133803

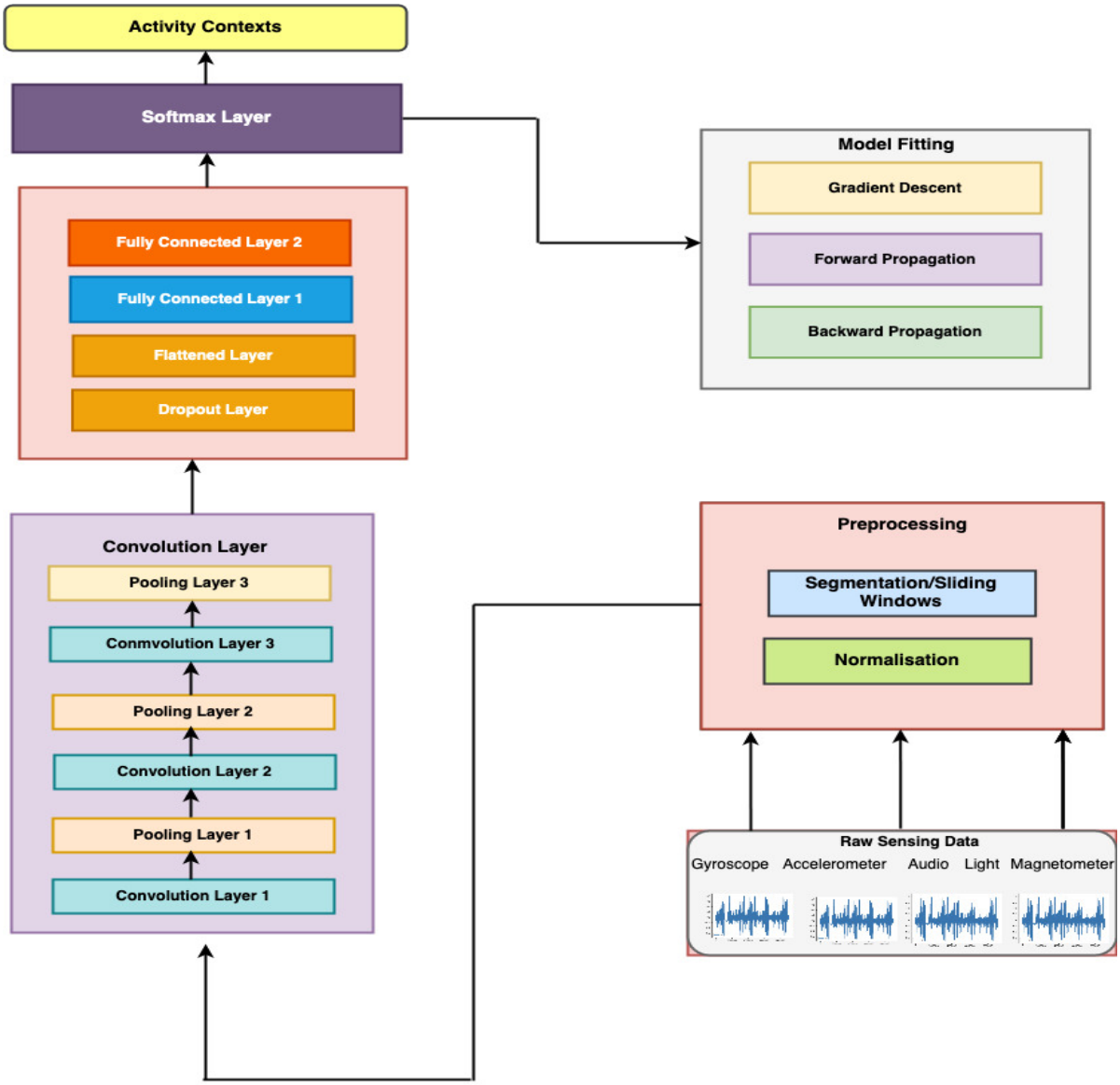

Figure 4. CNN Activity Context Recognition Processes

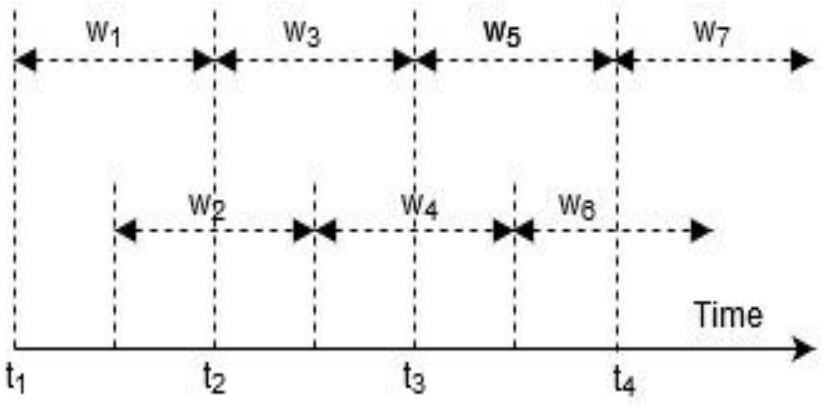

Figure 5. Sliding window with 50% Overlapping

3.2.2 Convolutional layer

In the deep convolutional neural network architecture, there are variant architectures with similar components. These components provide similar functionality. For example, one of the key and most important layers of CNN is the convolutional layer, where instead of performing matrix multiplication, it performs convolutional operations on the raw input data. The convolutional operation is a specialized linear operation combined with the non-linear operation of the activation function to extract discriminatory features from raw input data [10]. The convolutional layer is the fundamental component of CNN whose main function is to perform automatic feature extraction This layer contains several kernels for computing various feature maps. An element-wise product of the kernel and in the input matrix to generate the feature map. Every neuron in a convolution layer is connected to a region of neighboring neurons in the previous layer. The feature map is generated by convolving. Assuming that $x_i^a = [x_1, \dots, x_N]$ are inputs from the sensors, a represents the axis of the sensor. The output, depending on the number of convolutional layers, in the CNN architecture, of the lth convolutional layer is computed using equation 2.

$$z_i^{l,j} = \sigma\left(\sum_{k=1}^{K} w_k^j x_{i+k-1}^{l-1,j} + b_j^l\right) \quad (2)$$

Where $w_k^{l,j}$ and b_j^l are the weight and bias of the j-th term of the l -th layer.

$x_{i+k-1}^{l,j}$is the input patch l is the index of the current layer, and σ is the activation function. K represents the size of the filter/kernel. The activation function $a_j^l = \sigma(z_i^{l,j})$ introduces non-linearity to the CNN layer for detecting the non-linear features of the raw sensing data.

3.2.3 Pooling layer

The pooling layer is responsible for the downsampling operation of the generated feature maps from the convolutional layer [8,33]. Further, it helps to preserve the scale invariance property of the convolutional model. This is one of the key properties of CNN as earlier explained in section 1. The importance of the pooling layer is basically to reduce the spatial size of the feature map. It helps to reduce the number of parameters, computation requirements for the network, and memory footprints thus speeding up the training process. Different pooling operations can be applied for the downsampling of the feature maps [31]. However, in this article we have used Max pooling operation [39], which outputs a max value among those in a patch of inputs as illustrated in equation 3. Max pooling particularly performs well with sparse feature separation. Mathematically; max pooling is denoted by:

$$f_i^{l,j} = max_{s \in S}(z_{i*T+s}^{l,j}) \quad (3)$$

Where S is the pooling size, and its stride is denoted by T. $z_{i*T+s}^{l,j}$ is the value of the $i-th\ node\ in\ layer\ l$. The max pooling works by selecting the max value in a given patch and thereby trimming down the computation of upper layers by eliminating non-maximal components. Generally, several convolutional and pooling layers can be stacked hierarchically together not only to extract meaningful features from the signals but also to minimize the sensitivity of the signals. From bottom to top, we have the basic and more complex features respectively [15]. In our

architecture three layers of convolutional and pooling are stacked together to extract more meaningful features from the combination of inertial and ambient sensing data.

3.2.4 Non-Linear Activation function

The activation function is one of the most important parts of CNN. When we have binary outcomes to predict, there is a number activation function to decide the probability of the outcome. An example of such an activation function is the sign function, which is used with models with linear decision boundaries [16, 31]. However, for CNN with non-linear decision boundaries and multiclass classification, we use non-linear activation functions such as sigmoid or hyperbolic tangent functions. The sigmoid function returns a value in (0,1), which is used in computations that is interpreted as probabilities and also useful to derive functions loss functions derived from maximum likelihood models. The tanh function is similar to the sigmoid function but it horizontally rescaled and vertically rescaled to [-1,1]. However, these functions are very difficult to train. The recent activation functions such as ReLU are very popular nonlinear activation functions especially in deep neural networks such as CNN. ReLU brings with it the ease of training deep neural networks. In the architecture of our CNN model, each CNN layer and fully connected layers are followed by a ReLU activation function, equation (4), applied pointwise to outputs of their respective CNN layer.

$$ReLU = f(x) = max(0, x) \tag{4}$$

3.2.5 Fully Connected layer

Following the convolutional and max pooling layers, the outputs, i.e. the feature maps, of the last convolutional and max pooling layer are then flattened into a one-dimensional vector (1D) of features $f^l = [f_1, \dots, f_l]$, where l is the number of nodes in the last pooling layer. This is then passed into a fully connected layer, also known as the dense layer. This is where the classification is performed. To improve classification performance, additional features can be stacked. To perform non-linear classification with better recognition performance, the CNN architecture may have more than one fully connected layer.

The output of the pooling layer, which serves as the input to the fully connected layer is illustrated in equation (5)

.

$$f_i^l = \sum_j w_{ij}^{l-1}\left(\sigma(f_i^{l-1}) + b_i^{l-1}\right) \tag{5}$$

Where σ is the ReLU activation function, w_{ij}^{l-1} is the weight connecting the $i-th$ unit in layer $l-1$ and the $j-th$ unit in layer l and b_i^{l-1} is the bias.

The output of the fully connected layer is passed to the softmax layer to produce the inferred class. The softmax layer uses the softmax function, which computes the probability distribution for each class in each output node. If the activation function of j-th output neuron is:

$$a_j^l = \frac{e^{x_J^L}}{\sum_K e^{z_J^L}} \tag{6}$$

$$P(c|f) = argmax \frac{e^{f^{l-1}w^L b^L}}{\sum_{k=1}^{N} e^{f^{l-1}w_k}} = \quad (7)$$

Where N is the total number of classes or the number of neurons in the output layer, a_j^l is the activation function of output node j.

3.2.7 Regularization

If we have very large weights in the network, this might lead to weight vectors being stuck in their local minimum since the gradient descent only makes smaller moves in the direction of optimization [31]. Regularization allows the model to generalize to test or unseen data. It also helps to reduce variance in the training dataset by penalizing the network based on the size of the weights in the course of training the model. Rather than reducing the number of parameters in a hard way, a soft penalty is applied to the use of those parameters. Large absolute values of the parameters are also penalized more than those with smaller values. In L2 regularization is a suitable choice [8, 26]. The penalty is usually defined by the sum of squares of the values of the parameters, For the regularization parameter $\lambda > 0$, the objective function is defined as:

$$L = \sum_{(x,y)\epsilon D}(y - \hat{y})^2 + \lambda \sum_{i=0}^{d} w_i^2 \quad (8)$$

Where y & $\hat{y}$ represent the ground truth and the predicted class respectively.

We have used weight decay regularization to ensure that the proposed model generalizes, and Adam as our optimization algorithm after careful experimentations with popular optimization algorithms such as stochastic gradient descent (SGD) [29, 31].

a. Weight Decay

One of the keys to preventing complexity and overfitting in the convolutional neural network is the weight decay. It does this in two ways: by suppressing any irrelevant components of the weight vector by choosing the smallest vector that solves the learning problem. Second, if the right size is chosen for the weight decay, it can also suppress some of the effects of static noise on the target [31].

$$w_i \rightarrow \acute{w}_i = w_i - \alpha\lambda w_i - \alpha \frac{\partial L}{\partial w_i} \quad (9)$$

Where $\alpha\lambda w_i$ is the weight decay factor, which is first multiplied by the weight w_i before applying the gradient descent update. Note that α is the learning rate, and λ is the regularization constant.

b. Dropout

Deep neural networks contain several nonlinear hidden layers that allow them to learn very complicated patterns and relationships between the inputs and outs. To prevent overfitting due to large weights, which is a very common problem in deep neural networks, a dropout layer is added to the network to approximately combine exponentially many different network layers efficiently. Dropout means randomly and temporarily dropping some nodes including all their incoming and outgoing connections [26, 31]. To apply dropout, CNN uses node sampling to create an ensemble of networks. Whenever a node is dropped, all inputs and outputs of that node are dropped.

3.2.8 Backpropagation with shared weights

One other important operation in the deep learning model is the backpropagation, which is the process that follows the forward propagation (performed by equations 1-5). In each iteration, after the forward propagation, we then compute the value of the errors that is difference between the predicted class and the ground truth using the loss function L. The loss function is the negative likelihood using the gradient descent to update each weight in the softmax layer using weight decay and error cost minimization. This is done by Adam (a first-order gradient-based optimization of the stochastic objective function using adaptive estimates of lower-order moments) [23, 29]. The backpropagation iteration executes until a stopping criterion such as epochs has been satisfied, using the chain rule of derivative [14-15, 31].

In the fully connected layer, the gradient descent is typically computed using the classical partial derivatives and chain rules.

$$\frac{\partial L}{\partial w_{i,j}^{l}} = y_i^l \frac{\partial L}{\partial x_j^{l+1}} \tag{10}$$

Where in equation 10, L is the categorical cross entropy loss function that defines multiclass s logarithmic loss by comparing the distributions of the predictions with those of the ground truths setting the probability of the ground truth to[0,1], $y_i^l = \sigma(x_i^l) + b_i^l$ and $\sigma = Non-linear\ mapping\ function.$ $w_{i,j}^l$ is the weight connecting node n_i^l in the network layer l the network node n_i^{l+1} at layer $l+1$ the total number of input nodes n_i^{l+1} is x_j^{l+1}

$$L = -\sum_i^T \hat{y}_i \, log\,(softmax(y_i)) \tag{11}$$

Where softmax is the function (equation 7) that outputs the probability distributions of each class. In the 3 convolution layers of the model, the backpropagation is executed by computing the gradients of the layer's respective weights using equation 12 based on chain rule:

$$\frac{\partial L}{\partial w_{a,b}} = \sum_{i=1}^{N-M-1} \frac{\partial L}{\partial x_{i,j}^l} y_{i+a}^{l-1} \tag{12}$$

$y_{i+a}^{l-1} = \sigma(x_{i+a}^{l-1}) + b_i^{l-1}$ and $\sigma = Nonlinear\ mapping\ function$ of the convolution layer. $\frac{\partial L}{\partial x_{i,j}^l} = \frac{\partial L}{\partial x_{i,j}^l} \sigma^l(x_{i,j}^l)$. This process is repeated until the maximum epoch (the stopping criteria) is reached.

3.2.9 Hyperparameters

Bengio [33] defines hyperparameters for a learning algorithm as variables to be set before the actual application of the algorithm to the data, one that is not directly selected by the learning algorithm itself. We have a large number of hyperparameters to tune to get the optimal performance of the network. We have used greedy tuning of the hyperparameter as presented in [15]. We started with various values of epochs. The epoch is one cycle through the full training of the dataset. This is important because it gives CNN the chance to see previous data to adjust other parameters. The batch size helps to determine the number of batches or steps through the dataset to complete an epoch [31]. We also varied the number of CNN layers as well as the number of fully connected layers to

determine the number that gives the best recognition accuracy. We also tune the size of the pool, learning rates, sliding window size, number of CNN layers, number of fully connected layer, and the network dropout.

4. Experiments and Evaluation Results

In this section, we present our experimental setup including the analysis of the obtained results

4.1 Dataset and experiment setup

The data used in this experiment consists of those obtained from a combination of initial and ambient sensors. We have provided a full description of the process involved in our data gathering from the smartphone's built-in sensors in our previous publications [3, 32], where we used conventional machine learning algorithms and handcrafted statistical feature extraction processes. Unlike most existing works that use data from a 3D accelerometer, we have used data from 3D motion and position sensors(inertia). Figure 6 shows the numeric representation and corresponding classes in the dataset. We collected data from 2 motion sensors, namely

{1: Dowstairs;2: Riding in a car;3: Standing;4: Jogging; 5: Lying;6: Running;7: Walking; 8: Upstairs;9: Sitting}

Figure 6. Representation of Classes using numbers

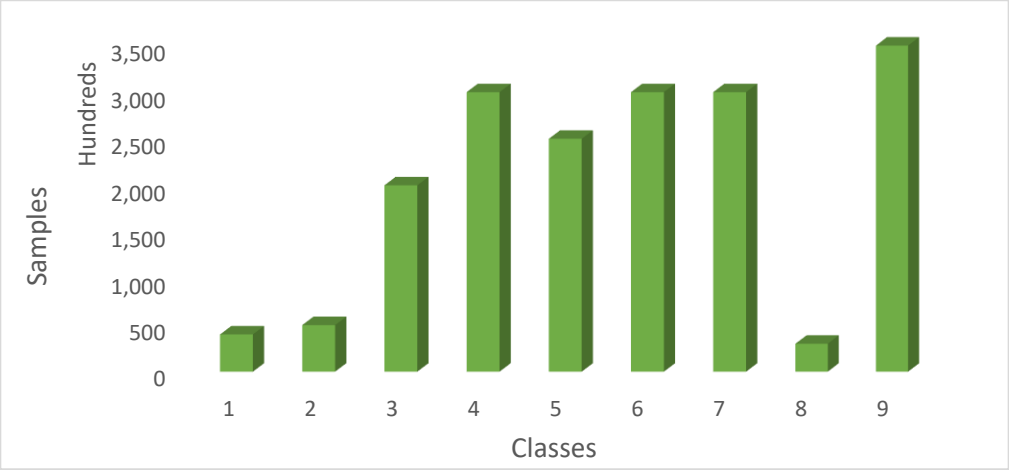

Figure 7. Distribution of classes by number of samples

accelerometer and gyroscope. For position sensors, we collected data from a magnetic sensor. Ambient data was collected from two sources: audio and light sensors. Figure 7 shows the distribution of classes with three classes with a significantly lower number of samples ("downstairs", "riding in a car", and "upstairs"). The dataset is divided into two. The first dataset consists of a dataset from all sensors. The second dataset consists of data from only motion and position sensors. This is done to allow us to investigate, if any, the importance of ambient sensors for recognizing human activity contexts with imbalanced classes.

In addition, we divided each of the two datasets into five sub-datasets, with window lengths of 32, 64, 128, 256, 512, and 1024 samples (approximately 0.75, 1.5, 3, 6, 12 and 24 seconds respectively). With the raw inertial and ambient sensors, we performed 17 and 15 channel 1D convolutions. Table 1 shows our experimental setup. Note that part of the pre-processing conducted was to standardize the data by subtracting the mean from each data and dividing it by the standard deviation as shown in equation (1). Following are the inertia and ambient sensors used in this project:

Accelerometer: It is a motion sensor that measures the acceleration in m/s2, along three axes. The measurement is the rate of change of velocity of the object. Figure 8 illustrates the accelerometer sensor's representation of the signals for the activity context classes in 3D.

Gyroscope: It is a sensor that measures the orientation and angular velocity of an object. A gyroscope is an advanced form of accelerometer that is about to capture the tilt and lateral orientation of an object, whereas the accelerometer only measures the change in linear velocity. Figure 9 illustrates the gyroscope sensor's representation of the signals for the activity context classes in 3D.

Magnetic sensor(magnetometer): This position sensor measures magnetic field strength and directions. Such magnetic results from the movement of charges or electrons. It is generally used to measure the induction. It is an important component of aircraft but now is being used as one of those sensors for detecting human activities. This is because several mobile devices now come with magnetometers. Normally it has been used to detect the orientation of the mobile phone relative to the Earth's magnetic north. Figure 10 illustrates the magnetometer's sensor's representation of the signals for the activity context classes in 3D.

Sound sensor: consisting of a microphone and speaker. modern smartphones usually have a pair of built-in speakers and a microphone. These can be used to recognize human activities and ambient contexts such as noise level. While the microphone receives the ultrasound signal the speaker is used to transmit the signals.

Light sensor: it generates an output signal that indicates the intensity of light by measuring the radiant energy that exists in a narrow range of frequencies, and which ranges from infrared, through visible light up to ultraviolet light spectrum. Figure 11 illustrates the environment noise and illumination representation of the activity context classes.

4.2 Evaluation Metrics

The metrics used in the context recognition experiments are, precision, recall, F-Score, and confusion matrix. These are the most widely used metrics to evaluate context recognition models [3, 9, 19].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2020 doi:10.20944/preprints202005.0430.v1

Peer-reviewed version available at Sensors 2020, 20; doi:10.3390/s20133803

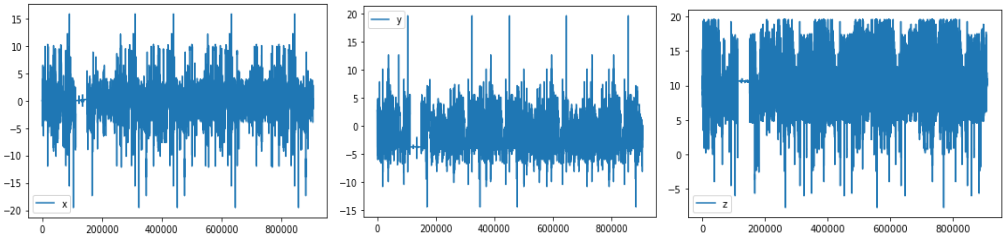

Figure 8. 3D Accelerometer Sensor Signals in

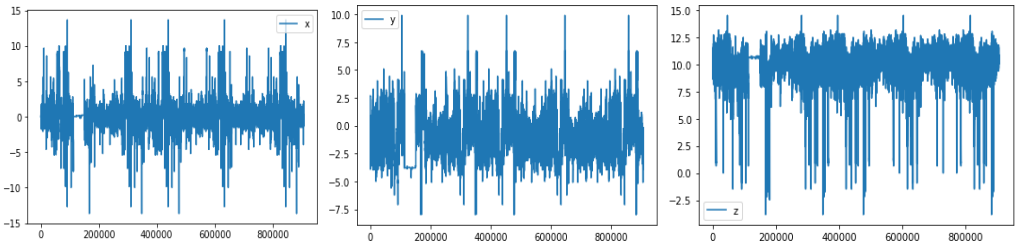

Figure 9. 3D Gyroscope Sensor Signals

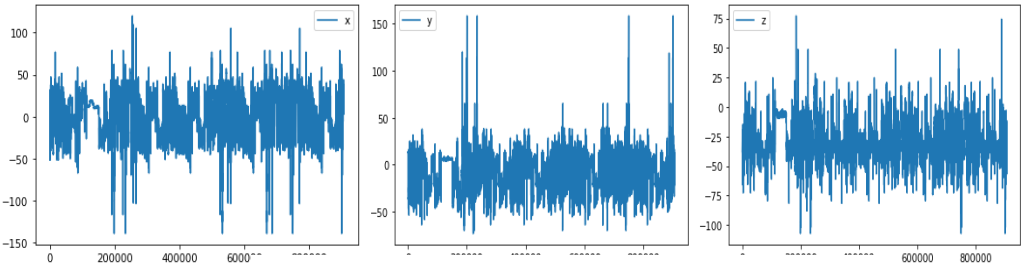

Figure 10. 3D Magnetic Sensor Signals

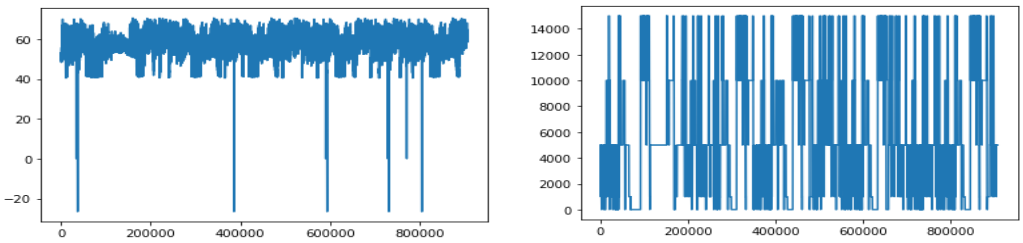

Figure 11. 1D Audio and Light Signals, representing the combined activity classes

F-Score is an often-used metric in information retrieval and natural language processing communities, and it is interpreted as the weighted average of precision (P) and recall (R). It is a measure of the statistical accuracy of the models given as follows:

$$\textbf{\textit{F-Score}}\ (R, P) = 2*RP/(R+P) \quad (13)$$

Where ***Recall*** (R) is the measure of the ability of a classification model to select instances of a certain class from a dataset. It is the sensitivity of the model defined as:

$$R = TP/\ (TP+FN) \quad (14)$$

TP is the number of true positive predictions and FN is the number of false negative predictions.

Precision *(P):* is the measure of the accuracy if a specific class is predicted; defined as:

$$P = TP/(TP+FP) \quad (15)$$

FP is the number of false positive predictions.

Confusion matrix: is a square matrix of order n number of classes used to present detailed results of a multiclass classification problem. The confusion matrix provides a more detailed and fine-grained analysis of both correctly and incorrectly classified classes of the supervised learning-based models. A given element $c_{i,j}$ of the matrix is the number of instances belonging to class i; classified as class j. Information about classification errors is also presented by the confusion matrix.

4.3 Experiments and Performance Evaluation

We have conducted extensive experiments to gain insights into various aspects of deep convolutional neural network-based activity context recognition using ambient and inertial sensors. The conducted experiments are broadly categorized into two. The first set of experiments investigates the impact of various hyperparameters on the recognition accuracy of the system. These experiments help us to determine the best/optimal values for the parameters to build the final recognition models. In the second set of experiments, first we investigate the recognition accuracy of the model using inertial sensors, and second, we investigate the recognition accuracy of combined data from both inertial and ambient sensing data.

4.3.1 Hyper-parameter Sensitivity Evaluation

Hyperparameter value selection is one of the most difficult parts of training an optimal learning model [31]. It is both an optimization problem (whereby we are looking for the hyper-parameter configuration that generates low validation error) and a generalization problem (whereby we at the same time looking for the configuration of the parameters that reduce estimation bias after optimizing validation performance) [33]. Our goal in this set of experiments is to carefully choose optimal configurations for the hyperparameters to have models with not only minimal test error but also with the least bias. We are looking for those values for the parameters that before evaluating the performance of the models with either inertial sensing data or with combined data from both inertial and ambient sensing data, we performed experiments to evaluate the sensitivity of the model's hyperparameters. In the first set of experiments, we investigated the impacts of various sliding windows sizes, decays, batch sizes, learning rates, dropout probabilities, number of nodes in the fully connected layer, and the number of CNN layers for feature extraction.

a. impact of sliding window size on the recognition accuracy

The segmentation of sensing data, especially time series data, is a crucial pre-processing mechanism. One of the key advantages of segmentation is that it allows us to overcome the limitation of a sample of a single time step to provide adequate information about the activity context using time series data [25]. The reason is that when a transition occurs particularly in the middle of a window, samples in a window might not always share the same label. Thus, sliding windows with overlapping can help to improve the accuracy of recognition. The sliding windows algorithms for segmentation of the sensing data overlaps with 50%. The details of the impact of sliding window

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2020 doi:10.20944/preprints202005.0430.v1

Peer-reviewed version available at Sensors 2020, 20; doi:10.3390/s20133803

sizes from 16 to 128 are summarized in Figure 12 depicting the accuracy in terms of loss errors. As shown in Figure 12, the initial window size of 16 with 50% overlapping generated the highest error followed by 64 and 128 with increasing loss error. Window size 32 has the lowest loss error, amounting to the highest recognition accuracy of the model.

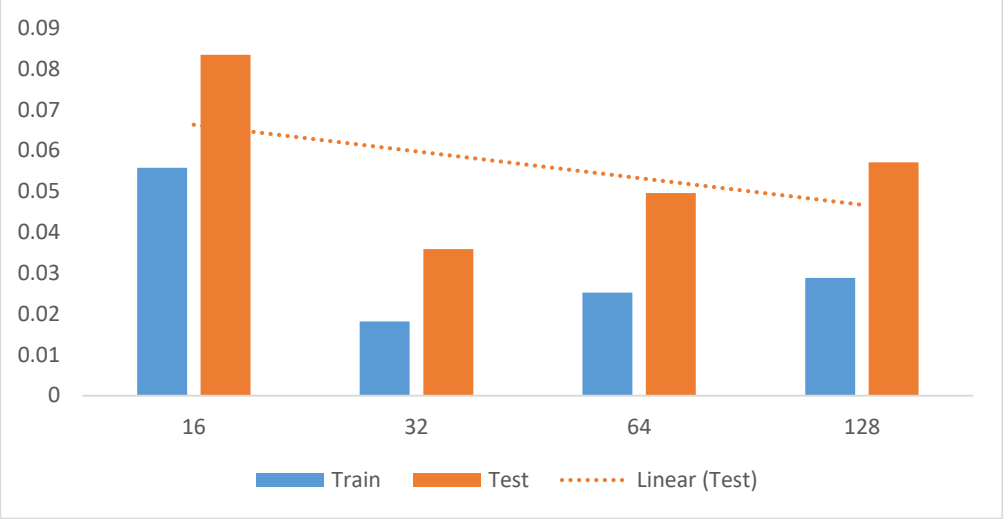

Figure 12. Model Performance with different window size overlapping at 50%

b. Influence of Pooling Size on the model's accuracy

We evaluated the impacts of various pooling sizes on the recognition performance of the CNN model configuration. We have used 3 convolutional layers, filter size 16, 32, and 64, one fully connected layer with 1024 nodes, sub-sampling factors of 2, 3, and 4 and a final softmax layer for generating the posterior probability of each class. The max-pooling values were increased from 1 to 10. Max-pooling size of 1 is equivalent to no max-polling process. Figure 13 shows the influence of the various pooling size on the accuracy of the model. The best performance was obtained between 3 and 5 although the loss at the pool size of 4 is higher than that of pool size of 5. However, the loss begins to increase from the value of 6.

c. impact of Learning Rates on CNN Context Recognition Accuracy

The initial learning rate is often considered as one of the most important hyperparameters to tune to obtain good performance. Normally the values of the learning rate are usually less than 1. Most practitioners rely on a value of 0.01 for standard multi-layer neural networks. To choose the optimal learning rate for our model, we evaluated the $\alpha = (10^{-6}, 10^{-1})$. Figure 14 shows the performance of the model with varied values of the learning rate. The performance improved from 10^{-6} 39.9% reaching the peak at 10^{-3} at 98.8% and then begins to decline and finally back to 39.9% at learning rate 0.1.

Figure 13. Performance of the model for various values of pool size

Figure 14. Performance of the model with various values of learning rates

d. Impact of Minibatch Size

Obtaining an optimal recognition performance of CNN models requires that some sensitive parameters have their values chosen appropriately. One of those parameters is the minibatch size, which controls the accuracy of the estimate of the loss function. It impacts the training process in terms of convergence time and the amount of overfitting [31,35]. Smaller batch sizes tend to lead to faster computation, but this requires visiting more examples to compute loss error during the training process. In this experiment, we have varied the values of batch sizes from 8 to 512 with increasing values of 8. Figure 15 shows the performance of each batch size. Initially,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2020 doi:10.20944/preprints202005.0430.v1

Peer-reviewed version available at Sensors 2020, 20; doi:10.3390/s20133803

the performance improves with the increasing value of the batch size. However initially batch size 8 tends to overfat, but from a mini batch size of 16, the model generalizes but the accuracy was declining with the increasing number of batch sizes. The best performance was achieved with a batch size of 32 and then dropped from batch size 64 The larger minibatch size of course will make greater gradient steps and thus produce poor performance.

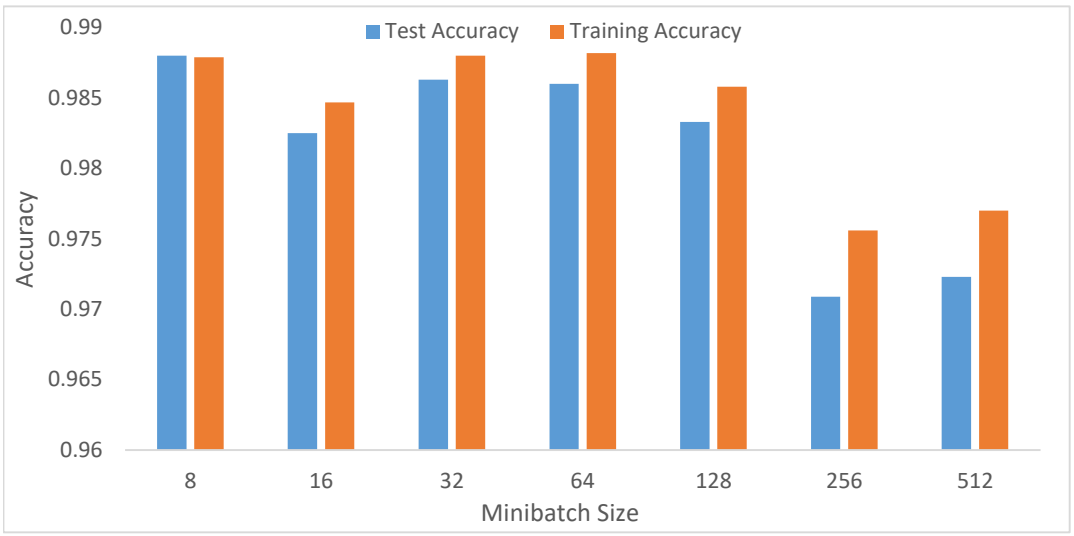

Figure 15. Comparison of model's performance with various batch sizes

e. impact of Decay on CNN Context Recognition Accuracy

Another important regularization mechanism to eliminate overfitting and improve generalization is weight decay, which is also known as L2 regularization [31]. The learning rate determines how much weight updating steps influence the current value of weights. Weight decay is used to cause the weight to decay exponentially to zero. To ascertain the optimal value of the decay for our CNN model, the hyperparameter needs to be tuned. We have varied the decay values between 10^{-6} to 10^{-1}. Figure 16 illustrates the impact of each decay value on the performance of the model. The accuracy generally increases with o.1 and then does not change from 0.001. This means a further reduction in the decay value does not improve the accuracy of the model. The significance is that a very small value of decay is required for the model to reach its best recognition performance.

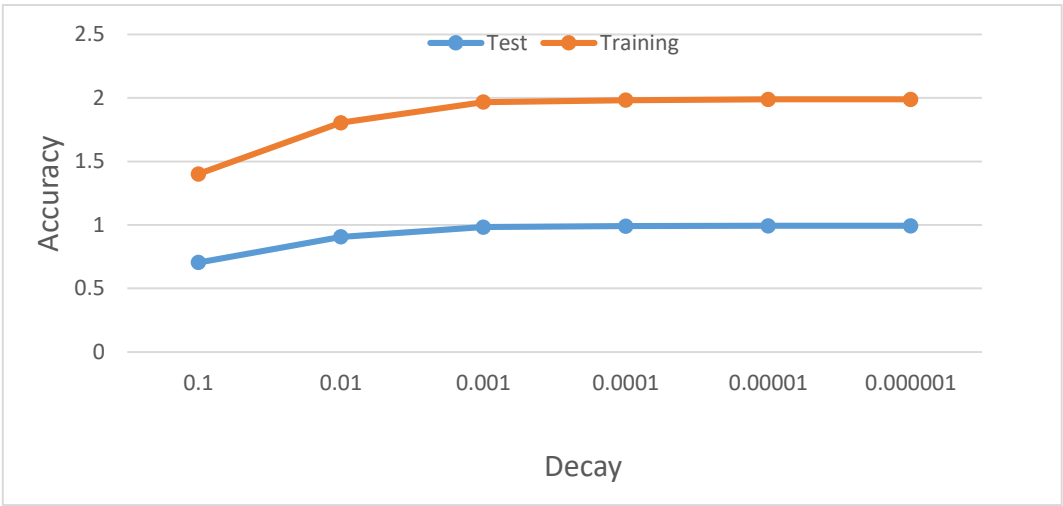

Figure 16. Performance of the model with various values of decay

f. impact of Dropout on the CNN Context Recognition Accuracy

One of the key regularization techniques is dropout as explained in section 3. The dropout represents the probability of retaining a hidden node in the network. The decision on which nodes to drop is random and dropping is done independently for each hidden node. Therefore, using the appropriate values for the dropout parameter helps the model to better learn redundant patterns in the time series input features. In the conducted experiment, we have varied the values of the dropout parameter(p) keeping the values of other parameters fixed, i.e., the number of hidden nodes and other parameters were kept constant just as in other experiments but only the values of dropout changes. Figure 17 shows the test and training accuracies. Note that the value ranges between 0.1 and 1.0. The best probability is 0.5 reaching 98.8% test accuracy. However, the values of p remain the same for both 0.9 and 1.0 but vary for other values of p. This result shows that by randomly dropping connections in the hidden layers and applying it in the top fully connected layer, the generalization errors can be reduced thereby preventing overfitting.

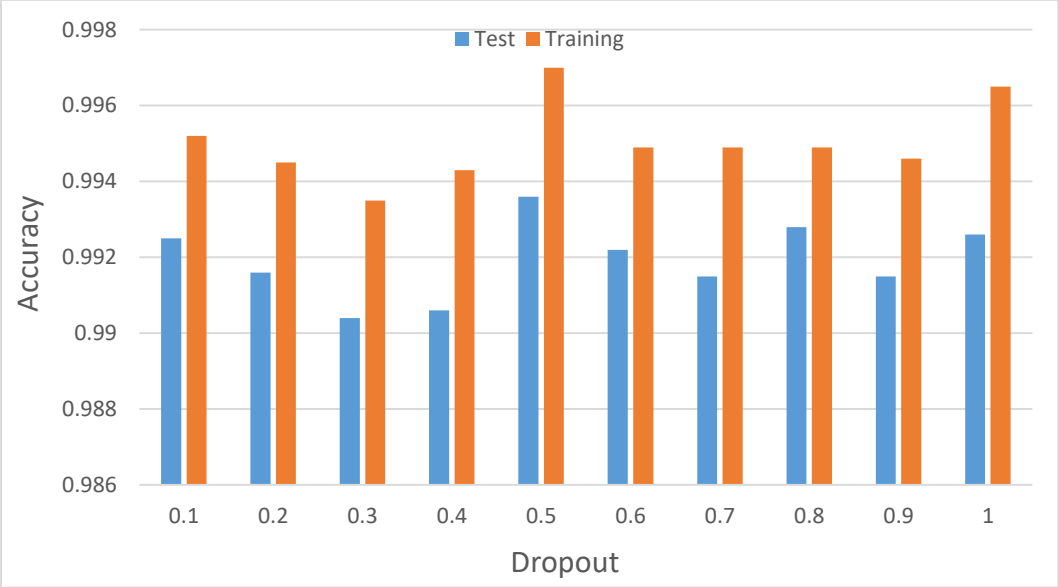

Figure 17. Performance with varying values of dropout probability

g. Influence of the number of CNN layers Context Recognition Accuracy

To determine the number of CNN layers required for the DCNN, we conducted experiments using the same values for other parameters as explained in section b above, we varied the values of l from 1 to 4. Figure 18 illustrates the improvement in performance from $l = 1$ to 3 and the accuracy begins to decline from $l = 4$. This result is expected since increasing the number of layers generally is expected to produce better performance. However, in the future, we would like to evaluate the computation cost of increasing the number of CNN layers running on real mobile devices.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2020 doi:10.20944/preprints202005.0430.v1

Peer-reviewed version available at Sensors 2020, 20; doi:10.3390/s20133803

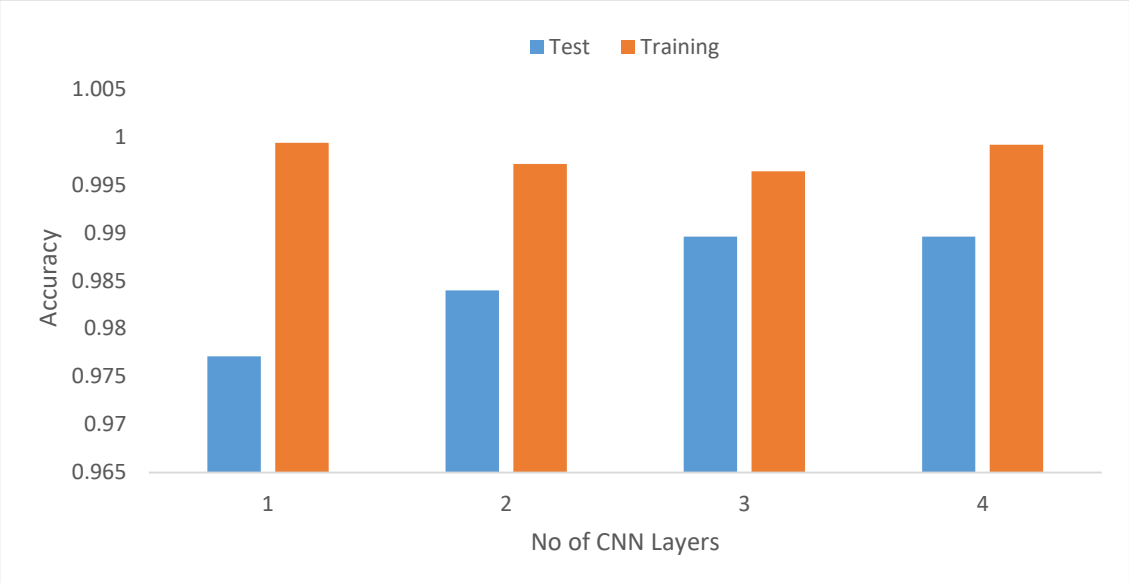

Figure 18. Performance with increasing number of CNN layers

h. Impact of the number of nodes in the fully connected layer

Another key experiment we performed is to determine the number of nodes in the fully connected layer. To determine the best value, we varied the values from 32 to 1024. As shown in Figure 19, the performance initially improves from 32 to 64 and declines before peaking at 512. This indicates that we do not need to use several numbers of nodes in the fully connected layer to achieve better performance.

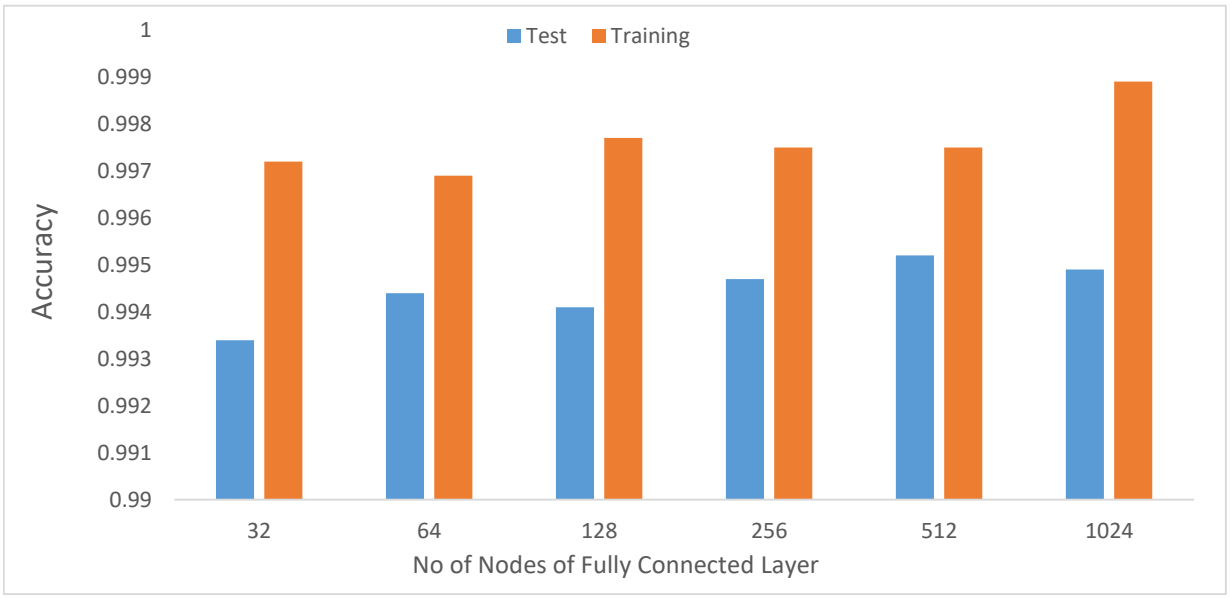

Figure 19. Model performance with increasing number of nodes in the fully connected layer

Table 1. Experiment Parameters

Hyperparameter	(Ambient + Inertial)	Inertial
Window size (size of input vectors)	32	32
Epochs from	25	25
Kernel size	1x3-1x5	1x3-1x5
Batch Size	32	32
Learning Rate	0.001	0.001
Decay	0.0001	0.0001
Dropout	0.5	0.5
Pooling Size	3x3	3x3
Activation Function	ReLU(CNN layer), Softmax	ReLU(CNN layer), Softmax
Input Channels	17	15
Fully connected layer(No of nodes)	512	512

3.2 Evaluating Recognition Accuracy

Having analyzed the impact of various hyperparameters on the accuracy of the proposed model, we now investigate the influence of combining data from ambient sensing with inertial sensors. First, we analyze the results obtained using only inertial sensing data. Secondly, we compare results from inertial only sensing data and those of inertial and ambient sensing data combined. Table 1 illustrates the values of hyperparameters used based on the results of section 4.3.1

a. Recognition Accuracy Using Only Inertial Sensing Data

The goal of the experiments in this section is to evaluate the performance of the model when trained with data signals from inertial sensors. The inertial sensors as explained in section 4.1 include motion and position sensors namely, accelerometer, gyroscope, and magnetometer. We have set the values of our hyperparameters to those optimal values obtained in the hyperparameter sensitivity evaluation as illustrated in Table 1.

Table 2 represents the confusion matrix of the results obtained showing the class-wise recognition accuracy of the model of each class. Note the performance of those classes with lower accuracy as measured by FScore. As expected, some of the classes have poor recognition performance. The result shows that *climbing upstairs(upstairs)*, followed by *climbing downstairs* performed worse than any other class reaching a poor value of 0.56 and 0.67 respectively. Compared to *Running* and *Jogging* classes shared the highest FScore value of 0.98. This result aligns with our initial hypothesis as reported in literature generally recognition models tend to have poor recognition performance when dealing with imbalanced classes and tend to perform well with classes that have majority number of samples. Besides, Figure 20(a&b) represents the overall accuracy of the model. The overall performance of the model based on inertia sensing reaching an accuracy of up to 93.6%.

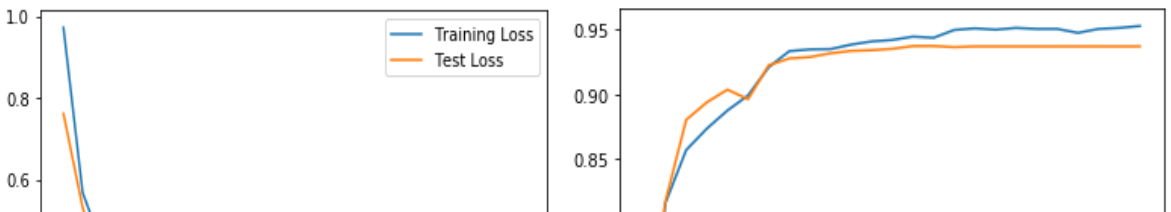

Table 2. Confusion Matrix for Inertial Sensing												
Predicted Class												
Actual Class		1	2	3	4	5	6	7	8	9	FScore	Label
	1	46	0	1	0	0	0	2	0	28	0.67	Downstairs
	2	0	70	0	0	0	1	1	2	2	0.70	Riding in car
	3	0	0	491	0	3	0	0	0	2	0.89	Standing
	4	0	0	0	437	0	6	2	1	4	0.98	Jogging
	5	0	0	5	0	180	0	0	1	1	0.77	Lying
	6	1	0	1	6	0	1397	26	1	12	0.98	Running
	7	2	0	1	3	0	4	362	6	19	0.90	Walking
	8	7	0	6	0	0	9	6	56	40	0.56	Upstairs
	9	4	0	4	0	2	4	7	9	1146	0.94	Sitting

b. Recognition Accuracy using both inertial and ambient sensing data

A key hypothesis of this article is that combining sensing data from both inertia and ambient sensors would produce a better performance of the CNN model. Having evaluated its performance with sensing data from inertial sensors, this section evaluates its performance with both sets of data. We also compared the results with the obtained in section 4.3.2(a). The results in Table 3 is the confusion matrix. This shows the performance of the model in terms of recognition accuracy for each activity class. The accuracy of the model reaches up to 98.9% as can be seen in Figure 21. This shows a marginal increase of 5.3 % in recognition accuracy compared to when we used only inertial sensing signals. This improvement is further elaborated in Figure 22 where we compared FScore for each of the activity context classes when using both datasets. The results indicate that models with inertial sensing struggled to recognize certain activity albeit considering good FScore value. For example, *climbing upstairs* and *sitting* activity Contexts. However, the model with both signals produced a far superior performance.

Table 2 is the confusion matrix showing the class-wise performance of the model of the new model. As can be seen, the recognition accuracy of those classes in the previous experiment has significantly improved. In section 4.3.2(a), downstairs and upstairs have 0.67 and 0.56 respectively. But in the current experiment, the new model achieved far better performance recognizing these activity contexts with 0.99 FScore value. This improvement is elaborated in Figure 22. For all classes model trained with both inertia and ambient data consistently performed better than model trained with only inertia data.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 May 2020 doi:10.20944/preprints202005.0430.v1

Peer-reviewed version available at Sensors 2020, 20; doi:10.3390/s20133803

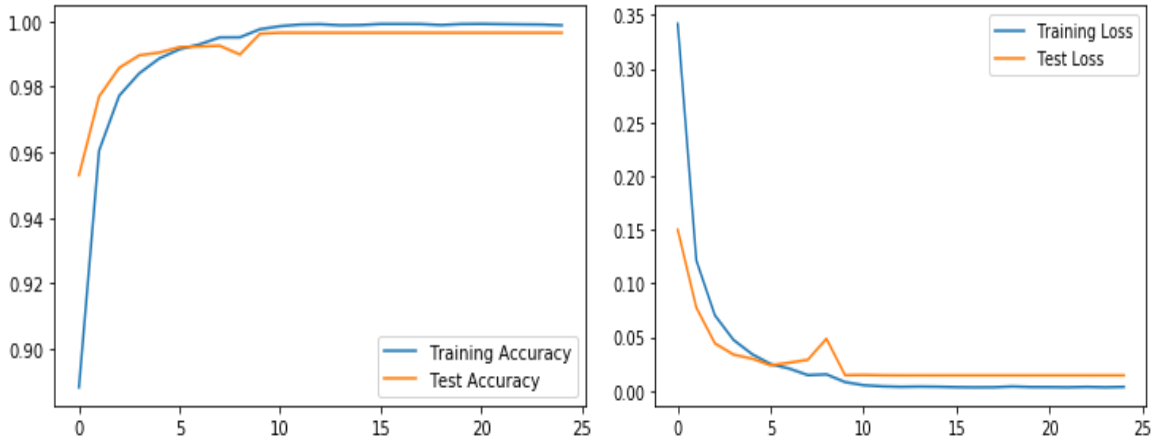

Figure 21. Train and test Accuracy and loss performances of Inertia + Ambient sensing data

Table 3. Confusion Matrix for Inertial + Ambient Sensing

Predicted Class												
Actual Class		1	2	3	4	5	6	7	8	9	FScore	Label
	1	629	0	2	1	0	0	0	1	2	0.99	Downstairs
	2	0	304	0	0	0	1	0	0	0	1.00	Riding in car
	3	0	0	821	1	3	0	1	0	1	0.99	Standing
	4	0	0	0	2561	0	0	0	0	0	1.00	Jogging
	5	0	0	4	0	136	0	0	0	0	0.97	Lying
	6	0	0	0	3	0	1892	10	1	3	0.99	Running
	7	1	0	1	0	0	1	2697	0	4	1.00	Walking
	8	1	0	0	0	0	1	4	898	4	0.99	Upstairs
	9	1	0	0	0	0	2	3	3	5725	1.00	Sitting

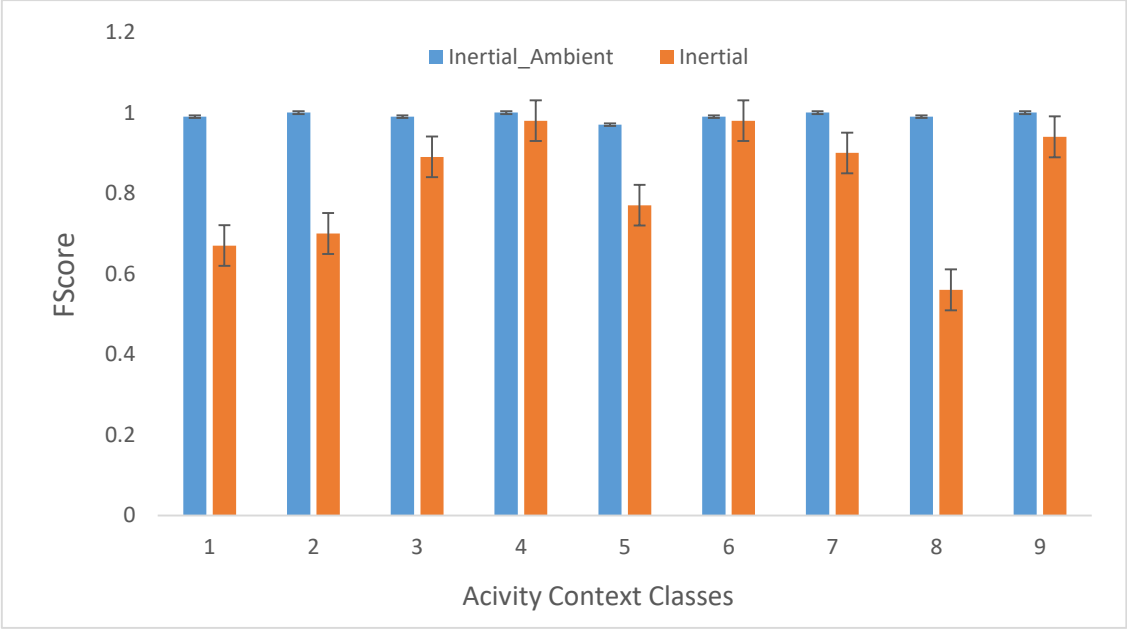

Figure 22. Performance Comparison of Inertia and Inertia + ambient sensing

5. Conclusion

In this article, we proposed to investigate the class imbalance problem of activity context recognition using deep convolutional neural networks to develop an effective and data-driven activity context recognition model using a combination of signals from inertial and ambient sensors. The signals from inertial sensors namely: gyroscope, accelerometer, and magnetometer and ambient sensing namely audio and light sensor have been used. In our previous work [3], we have used classical machine learning algorithms with handcrafted features as inputs. In the current work, we fed DCNN with raw sensing data without handcrafted features. We have used the DCNN to automatically extract features from the raw inertia and ambient sensing signals. Generally, time series data have inherent local dependency characteristics and activity contexts are naturally hierarchical as well as being translational [25]. The results show that with the CNN layer and with no additional statistical features as done by [14], our model can achieve very good performance.

We set out to investigate the class imbalance challenge of activity context signals by using additional signals from ambient sensors namely, environment noise level and illumination from audio and light sensors respectively. Experimental results confirm that combining these signals produce better recognition performance than inertia only data. The results also confirm significant improvement in the recognition performance of those classes with the least number of samples. Besides, we have used various techniques such as regularization techniques e.g. L2 regularization (aka weight decay) and dropout to prevent overfitting of the models. The developed DCNN model shows its capability to automatically extract features from the raw sensing data with better performance compared to the laborious and time-consuming handcrafted features and classical machine learning algorithms used in [3]. The developed DCNN model demonstrates the capability to capture local dependencies of the activity context signals using the correlation of both the inertial and ambient data signals. It also demonstrates that combining signals from ambient sensors produces better recognition performance than using signals from only inertia sensors. Finally, our experimental results demonstrate the influence of various hyperparameters on the eventual DCNN models.

The current activity context model can only classify simple classes; it only recognizes a single context in terms of the activity of the user i.e. it cannot combine certain contexts such as location to predict a much higher context. In the future we plan to update the system to integrate with a semantic model able to combine activity contexts with other ambient contexts as well as location information and user preferences to provide much higher level semantic contextual information. Finally, we will be investigating the computational cost of the model on resource-constrained devices considering the architecture of the model and the number of sensors involved when performing real-time activity context recognition.

References

1. Physical Activity. Available online: https://www.who.int/dietphysicalactivity/pa/en/ (accessed on 14th May 2020)
2. Bulling, A., Blanke, U. & Schiele, B. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Comput. Surv.* 2014, doi:10.1145/2499621
3. Otebolaku, A. M. & Andrade, M. T. User context recognition using smartphone sensors and classification models. *J. Netw. Comput. Appl.* 2016, doi: 10.1016/j.jnca.2016.03.013

4. Wang, J., Chen, Y., Hao, S., Peng, X. & Hu, L. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognit. Lett.* 2019, doi: 10.1016/j.patrec.2018.02.010
5. Chen, K. *et al.* Deep Learning for Sensor-based Human Activity Recognition: Overview, Challenges and Opportunities. arXiv 2020, arXiv:2001.07416.
6. Wan, S., Qi, L., Xu, X., Tong, C. & Gu, Z. Deep Learning Models for Real-time Human Activity Recognition with Smartphones. *Mob. Networks Appl. 2020* **25**, 743–755.
7. Lima, W. S., Souto, E., El-Khatib, K., Jalali, R. & Gama, J. Human activity recognition using inertial sensors in a smartphone: An overview. *Sensors (Switzerland)*2019, doi:10.3390/s19143213.
8. Zeng, M. *et al.* Convolutional Neural Networks for human activity recognition using mobile sensors. in *Proceedings of the 2014 6th International Conference on Mobile Computing, Applications and Services, MobiCASE 2014*, 2015, doi: 10.4108/icst.mobicase.2014.257786.
9. Figo, D., Diniz, P. C., Ferreira, D. R. & Cardoso, J. M. P. Pre-processing techniques for context recognition from accelerometer data. *Pers. Ubiquitous Comput.* 2010, doi:10.1007/s00779-010-0293-9.
10. Jiang, W. & Yin, Z. Human activity recognition using wearable sensors by deep convolutional neural networks. in *MM 2015 - Proceedings of the 2015 ACM Multimedia Conference*, 2015, doi:10.1145/2733373.2806333.
11. Cruciani, F. et al. Feature learning for Human Activity Recognition using Convolutional Neural Networks. *CCF Trans. Pervasive Comput. Interact. 2020*, 2, 18–32.
12. Rokni, S. A., Nourollahi, M. & Ghasemzadeh, H. Personalized Human Activity Recognition Using Convolutional Neural Networks. *Proc. 32nd Assoc. Advancement Artif. Intell. Conf. Aritf. Intell.*, 2018, pp. 8143-8144.
13. Hassan, M. M., Uddin, M. Z., Mohamed, A. & Almogren, A. A robust human activity recognition system using smartphone sensors and deep learning. *Futur. Gener. Comput. Syst.* 2018, doi: 10.1016/j.future.2017.11.029.
14. Ignatov, A. Real-time human activity recognition from accelerometer data using Convolutional Neural Networks. *Appl. Soft Comput. J.* 2018, doi: 10.1016/j.asoc.2017.09.027.
15. Ronao, C. A. & Cho, S. B. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Syst. Appl.* 2016. doi: 10.1016/j.eswa.2016.04.032.
16. Lecun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* ,2015, doi:10.1038/nature14539.
17. Vrigkas, M., Nikou, C. & Kakadiaris, I. A. A review of human activity recognition methods. *Frontiers Robotics AI* ,2015, doi:10.3389/frobt.2015.00028
18. Wang, J., Wang, W. & Gao, W. Fast and Accurate Action Detection in Videos with Motion-Centric Attention Model. *IEEE Trans. Circuits Syst. Video Technol.* 2020), doi:10.1109/TCSVT.2018.2887061
19. Kwapisz, J. R., Weiss, G. M. & Moore, S. A. Activity recognition using cell phone accelerometers. *ACM SIGKDD Explor. Newsl.*, 2011. doi:10.1145/1964897.1964918
20. Jain, A. & Kanhangad, V. Human Activity Classification in Smartphones Using Accelerometer and Gyroscope Sensors. *IEEE Sens. J.* 2018, doi:10.1109/JSEN.2017.2782492
21. Straczkiewicz, M. & Onnela, J.-P. A systematic review of human activity recognition using smartphones,.2019), arXiv:1910.03970[cs. HC]

22. Lara, Ó. D. & Labrador, M. A. A survey on human activity recognition using wearable sensors. *IEEE Commun. Surv. Tutorials* 2013, doi:10.1109/SURV.2012.110112.00192
23. Zebin, T., Scully, P. J. & Ozanyan, K. B. Human activity recognition with inertial sensors using a deep learning approach. in *Proceedings of IEEE Sensors*, 2017, doi:10.1109/ICSENS.2016.7808590
24. Golestani, N. & Moghaddam, M. Human activity recognition using magnetic induction-based motion signals and deep recurrent neural networks. *Nat. Commun.*2020, doi:10.1038/s41467-020-15086-2
25. Schrader, L. *et al.* Advanced Sensing and Human Activity Recognition in Early Intervention and Rehabilitation of Elderly People. *J. Popul. Ageing* ,2020, doi:10.1007/s12062-020-09260-z
26. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 2014. 15, 56:1929−1958
27. Wang, G. *et al.* Impact of sliding window length in indoor human motion modes and pose pattern recognition based on smartphone sensors. *Sensors* 2018, doi:10.3390/s18061965
28. Belapurkar, N., Shelke, S. & Aksanli, B. The case for ambient sensing for human activity detection. in *ACM International Conference Proceeding Series* 2018, doi:10.1145/3277593.3277628
29. Kingma, D. P. & Ba, J. L. Adam: A method for stochastic optimization. in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* 2015.
30. Yamashita, R., Nishio, M., Do, R. K. G. & Togashi, K. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging* 2018, doi:10.1007/s13244-018-0639-9
31. C. C. Aggarwal, Neural Network and Deep Learning, Springer, 2018, pp. 315-371.
32. Otebolaku, A. M. & Andrade, M. T. Recognizing high-level contexts from smartphone built-in sensors for mobile media content recommendation. in *Proceedings - IEEE International Conference on Mobile Data Management* 2013, doi:10.1109/MDM.2013.84.
33. Bengio, Y. Practical recommendations for gradient-based training of deep architectures. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)* (2012). doi:10.1007/978-3-642-35289-8-26
34. Yang, J. B., Nguyen, M. N., San, P. P., Li, X. L. & Krishnaswamy, S. Deep convolutional neural networks on multichannel time series for human activity recognition. in *IJCAI International Joint Conference on Artificial Intelligence*, 2015.
35. Masters, D. & Luschi, C. Revisiting Small Batch Training for Deep Neural Networks. *CoRR*, 2018, [online] Available: http://arxiv.org/abs/1804.07612.
36. Peng, L., Chen, L., Ye, Z. & Zhang, Y. AROMA: A Deep Multi-Task Learning Based Simple and Complex Human Activity Recognition Method Using Wearable Sensors. *Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol.* 2018, doi:10.1145/3214277
37. Tanberk, S., Kilimci, Z. H., Tukel, D. B., Uysal, M. & Akyokus, S. A Hybrid Deep Model Using Deep Learning and Dense Optical Flow Approaches for Human Activity Recognition. *IEEE Access* 2020, doi:10.1109/ACCESS.2020.2968529

38. Lee, J. & Bastos, N. Finding characteristics of users in sensory information: From activities to personality traits. *Sensors* 2020, doi:10.3390/s20051383

39. Nagi, J. et al. Max-pooling convolutional neural networks for vision-based hand gesture recognition. in *2011 IEEE International Conference on Signal and Image Processing Applications, ICSIPA 2011*, doi:10.1109/ICSIPA.2011.6144164

40. Moya Rueda, F.; Grzeszick, R.; Fink, G.A.; Feldhorst, S.; Ten Hompel, M. Convolutional Neural Networks for Human Activity Recognition Using Body-Worn Sensors. *Informatics* **2018**, *5*, 26.