Chapter 3

An unseen locomotion mode identification model using multiple semantic matrices

This chapter presents a deep learning-based model that can recognize both seen and unseen locomotion modes using data instances from inertial sensors. We adopt the concept of zero-shot learning to improve the capability of the traditional classifier by using multiple semantic metrics.

3.1 Introduction

Identification of human locomotion activities has been a prominent area of research for many years. Literature indicates several significant contributions towards building a pertinent identification approach, for identifying locomotion activities using the sensory data of smartphone or wearable [1, 2]. Recent smartphones consist of variety of sensors (*e.g.,* accelerometer, gyroscope, magnetometer, *etc.*), which are able to capture a crucial information about basic and complex human locomotion activities [87]. As the smartphone has become an integral part of our daily routine, these activities can be

recognized by using the sensory data (obtained when user is performing the activity). Activity recognition has several potential applications such as user authentication [24], locomotion or transportation mode recognition [5, 73], detecting vehicle-riding activities [74], and so on [75].

Information about the locomotion mode helps in travel time estimation, traffic management, journey planning, and so on [5]. In locomotion mode recognition, the main objective of a classifier is to identify a transportation mode (*e.g.,* bicycle, bike, car, *etc.*) by using the smartphone sensors. Such a classifier first learns a mapping between instances (*i.e.,* sensory data) and class labels (or locomotion modes) by using the training dataset. Later, the classifier predicts a class label of a new instance. Most of the prior studies either employed machine learning models [29, 30, 88] or deep network models [26–28] to learn the mapping from the training dataset.

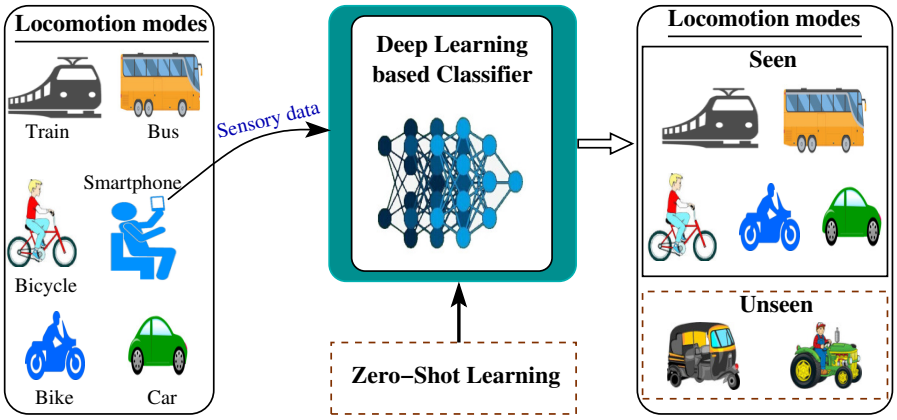


Figure 3.1: An example of locomotion mode recognition using a deep learning based classifier and zero-shot learning.

A traditional classifier can recognize only seen classes (locomotion modes) that are given with training instances. In other words, such a classifier is incapable to identify an unseen class that appears first time during testing. Such unseen class or locomotion mode can be identified by using the semantic information of the seen classes [9, 10]. Zero-Shot Learning (ZSL) [11] is a concept that extends the capability of the traditional classifier for identifying the unseen classes. Figure 3.1 illustrates an

example of locomotion mode recognition using deep learning based classifier and ZSL, where the smartphone provides sensory data which is used to identify modes.

3.1.1 Motivation of this work

Previous studies have following major limitations which motivated this work:

- The existing recognition approaches [5, 26–28] can identify only seen locomotion modes by extracting the deep features from the training instances. As it is impractical to acquire prior knowledge (*i.e.,* labeled data) about every type of locomotion mode, the recognition approach should identify a new locomotion mode without any corresponding training instance. It indicates that the approaches [5, 26–28] can not be employed for identifying an unseen locomotion mode.
- In prior studies [29,30], the authors employed machine learning based recognition models for identifying the locomotion modes, which heavily rely on the knowledge of domain related features. Such dependency can be obviated by utilizing the automatic feature extraction capabilities of deep network models.
- The existing work [10, 66] construct a semantic matrix by using word2vec [89], GLoVe [90] and human-annotated characteristics to identify unseen images or text. The semantic matrix captures meaningful information from seen classes, which is later utilized for recognizing the unseen classes. Fusion of multiple semantic matrices can enrich this information remarkably.

In this chapter, we address the problem: *how to identify a locomotion mode (seen or unseen) using semantic information and deep learning features of labeled training instances?* To solve this problem, this work proposes, DeepZero, a sensors based Deep learning model by incorporating the concept of Zero-shot learning for identifying unseen locomotion modes. It first extracts features from training data using a sequential combination of CNN and LSTM. Later, the model builds a classifier by learning a mapping between extracted features and semantic information.

3.1.2 Major contributions

To best of our knowledge, this is the first work to address the identification of unseen locomotion modes using deep learning models. This work makes following major contributions:

- We propose a recognition model DeepZero to recognize the locomotion modes using labeled training dataset. DeepZero model is capable enough to identify an unseen locomotion mode by incorporating the concept of ZSL.
- DeepZero model develops a feature extraction framework to extract the features automatically from the given dataset and uses them during construction of the classifier. The framework consists of a sequential combination of CNN and LSTM, to capture spatial as well as temporal dependencies among the sensory data points of the instances and thus provides information rich features.
- Next, this work constructs a novel attribute matrix by using multiple semantic matrices including human-annotated, one-hot encoding, and word2vec for identifying unseen locomotion mode. DeepZero model builds a classifier by learning a mapping between the extracted features and the attribute matrix. This work considers various characteristics of locomotion modes to obtain the human-annotated semantic matrix.
- Finally, we develop an android application to collect a Locomotion Mode Recognition (LMR) dataset using acceleration, gyroscope, and magnetometer sensors. This work conducts various experiments to evaluate DeepZero model using LMR dataset along with an existing Sussex-Huawei Locomotion (SHL) dataset [77].

The rest of chapter is structured as follows. Next section describes the terminologies and notations used in this chapter. Section 3.3 proposes DeepZero model for locomotion mode identification. In Section 3.4, we discuss a feature extraction framework. Next, Section 3.5 presents the experimental evaluation of DeepZero. Finally, Section 3.6 concludes the chapter.

3.2 Preliminary

In an application of locomotion mode recognition, dataset $\mathcal{D}$ contains sensory measurements of n different types of sensors. An i^{th} instance of $\mathcal{D}$ is denoted by $\mathbf{x}_i$, where $1 \leq i \leq N$. Each instance $\mathbf{x}_i$ belongs to one of k class labels that are denoted as $L_1, L_2, \cdots, L_k$. Let $\mathcal{D}_j$ denotes a dataset that contains data of j^{th} sensor, where $1 \leq j \leq n$. An instance of $\mathcal{D}_j$ is a series of m-dimensional measurements. The length of the instances in $\mathcal{D}$ is denoted by M. As $\mathcal{D}$ contains labeled instances corresponding to various locomotion modes, we refer locomotion modes as class labels.

Definition 3.1 (Unseen class) *A class is said to be unseen if there exists no training instances of the class in the dataset.*

Definition 3.2 (Attribute matrix) *It contains semantic information of all possible classes of the dataset $\mathcal{D}$. The semantic information is obtained in terms of attributes (i.e., most identifiable features). The attribute matrix is denoted by $\mathcal{A} \in \mathbb{R}^{k \times a}$, where a is the number of attributes.*

3.2.1 Problem statement and overview of solution

Locomotion mode recognition using smartphone sensors has several user-centric applications including travel time estimation, journey planning, *etc.*, as discussed in the introduction. As it is impractical to have information about all types of locomotion modes, the recognition approach should be able to identify a new locomotion mode even if no corresponding training instances are given. This work therefore addresses problem of identifying an unseen locomotion mode.

Overview of the solution: DeepZero model constructs an attribute matrix $\mathcal{A}$ from a given set of class labels of the dataset $\mathcal{D}$. The model also obtains a feature matrix $\mathcal{F} \in \mathbb{R}^{N \times a}$ corresponding to the N instances of $\mathcal{D}$, using a combination of CNN and

LSTM. Later, a classifier $\mathbf{\Pi}$ is constructed by learning a mapping between $\mathcal{F}$ and $\mathcal{A}$. Finally, the classifier $\mathbf{\Pi}$ is used to predict a locomotion mode for a given test instance.

3.3 DeepZero model

In this section, we propose a deep learning based recognition model (DeepZero) for identifying a locomotion mode using sensory data. The objective of DeepZero model is to recognize a locomotion mode even when training data does not have any corresponding instances. DeepZero employs ZSL to identify an unseen locomotion mode by using the semantic information of seen classes. The model consists of following major components: 1) construction of attribute matrix, 2) feature extraction, 3) classifier construction, and 4) prediction of class label. Figure 3.2 illustrates a block diagram of DeepZero model, where the training dataset consists of labeled instances of sensory data for different locomotion modes. The new instance is an unlabeled testing instance that belongs to either seen or unseen locomotion mode. Algorithm 3.1 illustrates all the steps of DeepZero model for locomotion mode recognition.

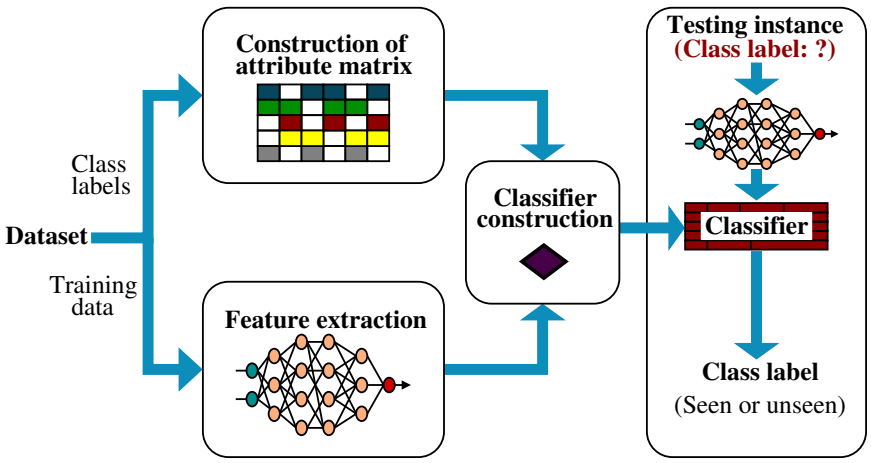


Figure 3.2: Overview of DeepZero model.

3.3.1 Construction of attribute matrix

This work initially constructs an attribute matrix to capture the semantic information of the classes. DeepZero model uses this matrix for identifying the unseen locomotion modes. The attribute matrix is obtained from the fusion of three semantic matrices:

3.3.1.1 One-hot encoding matrix

It is a diagonal matrix of size $k \times k$, where k is the number of classes in dataset $\mathcal{D}$. The matrix contains binary representation of the classes in such a way that only diagonal entries are 1. For example, for three classes (bike, car, and bus) in $\mathcal{D}$ then their corresponding binary representation can be given as $100, 010$, and 001. In this work, one-hot encoding matrix is denoted by $\mathcal{Z}_k$.

3.3.1.2 Word2vec matrix

It is a matrix of size $k \times v$, where v is the number of most similar words corresponding to a word (class label). Such similar words can be obtained by using word embeddings [89] along with a vector of probabilities for those obtained words. We utilize the probabilities of 300 words (*i.e.,* $v = 300$) corresponding to each of k class labels in the dataset $\mathcal{D}$, which gives a word2vec matrix of size $k \times 300$. The word2vec matrix is denoted by $\mathcal{Z}_v$.

3.3.1.3 Human-annotated semantic matrix

Understanding the importance of domain knowledge in recognition model, we create a human-annotated semantic matrix based on common characteristics of locomotion modes. For the matrix, we have considered all the relevant characteristics that can be visually observed by humans. Following steps illustrate the procedure for creating this semantic matrix.

(a) *Characteristics identification:* First of all, we find some common characteristics of the class labels that can provide sufficient information for their unique identification.

This work considers following characteristics of the locomotion modes: speed, capacity, power, fuel, pathway, and wheel count. The characteristics such as capacity, wheel count, pathway, and fuel can be easily determined. Moreover, the characteristics such as speed and power are obtained by taking the experts opinion. Further, to build a classifier for identifying unseen classes, it is prerequisite to have a semantic matrix with sufficient information about domain. We therefore develop a human-annotated attribute matrix that can provide desirable distinguishable information to the classifier.

(b) *Integer value representation:* This step creates a matrix of size $k \times c$, where c is the number of characteristics taken into consideration for each of the k class labels. Figure 3.3 illustrates an example of such matrix for four locomotion modes including bike, car, bus, and train. To illustrate the matrix conveniently, we considered only four characteristics including speed, capacity, power, and fuel, as shown in Figure 3.3. The values in the matrix (given on the left side of Figure 3.3) are taken from the practical knowledge of different locomotion modes for the considered characteristics. In order to represent each characteristic using an integer value, the values in power characteristic are shown after multiplying the horse power (hp) with 100. For the same reason, the values in fuel characteristic indicated as follows: 0 for petrol, 1 for diesel, 2 for electric energy, and 3 for human energy.

	Speed (km/hr)	Capacity (Seats)	Power (hpx100)	Fuel
Bike	80	2	500	0
Car	120	4	17000	0
Bus	100	45	24000	1
Train	110	1464	700000	2

Speed (7 bits)	Capacity (11 bits)	Power (20 bits)	Fuel (2 bits)
1010000	00000000010	···	00
1111000	00000000100	···	00
1100100	00000101101	···	01
1101110	10110111000	···	10

Figure 3.3: An example of human-annotated semantic matrix.

(c) *Binary representation:* Next, the integer values of the matrix are converted into binary. The converted binary matrix is shown on right side of Figure 3.3. Each characteristic is represented by a fixed number of bits, which is equal to the number of bits required to represent the maximum value in the column. We call this matrix as

human-annotated semantic matrix which is denoted by $\mathcal{Z}_h \in k \times h$, where h denotes the number of bits in a row obtained after binary conversion. If we directly use non-binary values of human-annotated matrix then the multiplication operations require higher temporary storage and computations [91]. Further, normalization is another way to handle non-binary values of matrix but it results in floating values which demands for higher computational resources.

3.3.1.4 Fusion of semantic matrices

After obtaining the semantic matrices $\mathcal{Z}_k$, $\mathcal{Z}_v$, and $\mathcal{Z}_h$, this work performs fusion of these matrices to construct the attribute matrix. As the matrices are of different dimensions, we first pass each of them through a separate Fully Connected (FC) layer consisting equal number of neurons, as shown in Figure 3.4. The FC layers result the matrices of equal dimension. Next, the resultant matrices are passed through a Rectified Linear Unit (ReLU) function to eliminate the negative values. In the fusion, we perform an element-wise sum of resultant matrices to get a single semantic matrix that incorporates the different semantic spaces. Finally, the obtained semantic matrix is passed through an activation function and the resultant matrix is called as *attribute matrix*. This work uses Tanh activation function to add non-linearity in the matrix, which helps the classifier to learn better mapping [92].

Let $f_i(\cdot)$ denotes a ReLU function that takes a resultant matrix at FC layer as input, where $i \in \{k, v, h\}$. It is mathematically expressed as:

$$f_i(\mathbf{W}_i \mathcal{Z}_i + \mathbf{b}_i), \tag{3.1}$$

where, $\mathbf{W}_i$ and $\mathbf{b}_i$ denote the weight matrix and bias vector for $\mathcal{Z}_i$, respectively. The weight matrix and bias vector are estimated during mapping. Now, the fusion of the

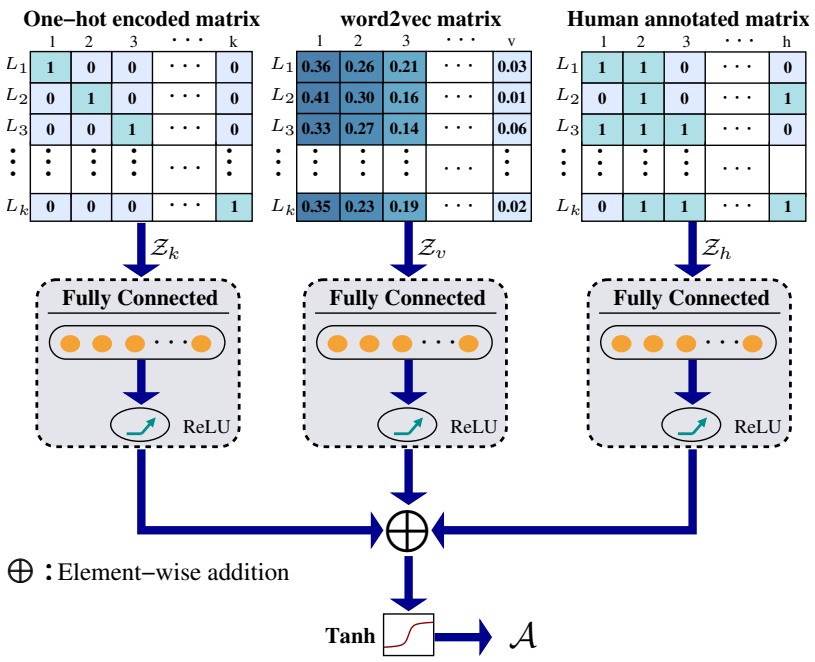


Figure 3.4: Construction of attribute matrix by fusing three semantic matrices.

three semantic matrices can be given as:

$$\mathcal{F} = f_k(\mathbf{W}_k \mathcal{Z}_k + \mathbf{b}_k) \oplus f_v(\mathbf{W}_v \mathcal{Z}_v + \mathbf{b}_v) \oplus f_h(\mathbf{W}_h \mathcal{Z}_h + \mathbf{b}_h), \tag{3.2}$$

where, symbol $\oplus$ denotes an element-wise sum. After fusion, we apply Tanh activation function (denoted by $g(\cdot)$) on the output matrix $\mathcal{F}$ to get the attribute matrix as:

$$\mathcal{A} = g(\mathcal{F}). \tag{3.3}$$

The attribute matrix $\mathcal{A} \in \mathbb{R}^{k \times a}$, where a denotes the number of attributes representing a class label (locomotion mode).

3.3.2 Feature extraction

We extract two types of features: hand-crafted and deep. The hand-crafted features include minimum, maximum, autocorrelation (four lags), below mean, and above mean results in total 8 features. In order to extract deep features, this work uses CNN and

LSTM. Later, both hand-crafted and deep features are combined and output of this layer is used as features in DeepZero model. Let $\mathcal{F} \in \mathbb{R}^{N \times a}$ denotes a feature matrix obtained for the given dataset $\mathcal{D}$, where N is the total number of instances in $\mathcal{D}$ and a is the number of features extracted against a single instance. The details of feature extraction is presented in Section 3.4.

3.3.3 Classifier construction

In this section, DeepZero model builds a classifier by using the obtained feature matrix ($\mathcal{F}$) and attribute matrix ($\mathcal{A}$). As the features are extracted using the data and the attributes are obtained using only the class labels, the classifier needs to learn a mapping between them by minimizing the loss.

3.3.3.1 Objective function

This work first defines loss and regularization terms to formulate the objective function of the classifier, denoted by $\mathbf{\Pi}$. Let $\mathcal{F}_i$ denotes a feature vector (*i.e.,* i^{th} row of $\mathcal{F}$) which is obtained corresponding to an instance $\mathbf{x}_i \in \mathcal{D}$. Let $\mathcal{T}_i$ denotes an attribute vector obtained as $\mathcal{T}_i = \mathcal{A}_l$, where l is an index corresponding to class label of $\mathbf{x}_i$ and $1 \leq l \leq k$. We define a loss term using vector l_2-norm as:

$$\mathcal{L}_{term} = \frac{1}{N} \sum_{i=1}^{N} ||\mathcal{F}_i - \mathcal{T}_i||_2^2 = \frac{1}{N} \sum_{i=1}^{N} ||\mathcal{F}_i - g_i(\boldsymbol{\mathcal{F}})||_2^2, \tag{3.4}$$

where, function $g_i(\cdot)$ returns an attribute vector (from $\mathcal{A}$) corresponding to the class label of $\mathbf{x}_i$. By converting vector norm into matrix Frobenius norm (denoted by $||\cdot||_F$) and substituting $\boldsymbol{\mathcal{F}}$ from Equation 3.2, Equation 3.4 can be written as:

$$\mathcal{L}_{term} = \Big\| \mathcal{F} - g\Big(f_k(\mathbf{W}_k \mathcal{Z}_k + \mathbf{b}_k) \oplus f_v(\mathbf{W}_v \mathcal{Z}_v + \mathbf{b}_v) \oplus f_h(\mathbf{W}_h \mathcal{Z}_h + \mathbf{b}_h) \Big) \Big\|_F^2. \tag{3.5}$$

As the attribute matrix is obtained from three different matrices by passing them

through a separate FC layer, this work adds a separate regularization term to each of the FC layers. By the regularization, DeepZero model prevents overfitting of the classifier to make it resilient enough for unseen class prediction. This work formulates the regularization term as:

$$\mathcal{R}_{term} = \lambda_1 \left(\frac{||\mathbf{W}_k||_F^2}{2} + \frac{||\mathbf{b}_k||_2^2}{2} \right) + \lambda_2 \Big(\frac{||\mathbf{W}_v||_F^2}{2} + \frac{||\mathbf{b}_v||_2^2}{2} \Big) + \lambda_3 \left(\frac{||\mathbf{W}_h||_F^2}{2} + \frac{||\mathbf{b}_h||_2^2}{2} \right), \tag{3.6}$$

where, λ_1, λ_2, and λ_3 are regularization parameters for the respective FC layers. The objective function of the classifier $\mathbf{\Pi}$ can be expressed as:

$$\min_{\mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_h} (\mathcal{L}_{term} + \mathcal{R}_{term}). \tag{3.7}$$

By solving Equation 3.7, DeepZero model learns the weight matrices $(\mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_h)$ and corresponding bias vectors $(\mathbf{b}_k, \mathbf{b}_v, \mathbf{b}_h)$. We consider the values in bias vectors as constant in order to simplify the optimization problem.

3.3.3.2 Optimization

The optimization problem expressed in Equation 3.7 is convex for $\mathbf{W}_k, \mathbf{W}_v$, and $\mathbf{W}_h$ individually but not convex for all of them together. We therefore solve for one weight matrix by fixing two others (as constant). This solution of optimization is motivated from a work in [76]. From Equation 3.7, let $\boldsymbol{\mathcal{L}} = \mathcal{L}_{term} + \mathcal{R}_{term}$. Using Equations 3.5 and 3.6, we get:

$$\begin{aligned} \boldsymbol{\mathcal{L}} = & \Big\| \mathcal{F} - g\Big(f_k(\mathbf{W}_k \mathcal{Z}_k + \mathbf{b}_k) \oplus f_v(\mathbf{W}_v \mathcal{Z}_v + \mathbf{b}_v) \oplus f_h(\mathbf{W}_h \mathcal{Z}_h + \mathbf{b}_h) \Big) \Big\|_F^2 \\ & + \frac{\lambda}{2} \Big[||\mathbf{W}_k||_F^2 + ||\mathbf{b}_k||_2^2 + ||\mathbf{W}_v||_F^2 + ||\mathbf{b}_v||_2^2 + ||\mathbf{W}_h||_F^2 + ||\mathbf{b}_h||_2^2 \Big], \end{aligned} \tag{3.8}$$

where, $\lambda = \lambda_1 = \lambda_2 = \lambda_3$. As the main objective of the classifier is to obtain weight matrices, the bias vector terms can be omitted from Equation 3.8, which gives:

$$\begin{aligned}\boldsymbol{\mathcal{L}} =& \Big\| \mathcal{F} - g\Big(f_k(\mathbf{W}_k\mathcal{Z}_k) \oplus f_v(\mathbf{W}_v\mathcal{Z}_v) \oplus f_h(\mathbf{W}_h\mathcal{Z}_h)\Big)\Big\|_F^2 \\ &+ \frac{\lambda}{2}\Big[||\mathbf{W}_k||_F^2 + ||\mathbf{W}_v||_F^2 + ||\mathbf{W}_h||_F^2\Big]. \end{aligned} \tag{3.9}$$

This work first optimizes Equation 3.9 for $\mathbf{W}_k$ by fixing $\mathbf{W}_v$ and $\mathbf{W}_h$. For optimization, we take first order derivative of $\boldsymbol{\mathcal{L}}$ with respect to $\mathbf{W}_k$ and equate it to zero. As the function $g(\cdot)$ does not influence the learning of weights, it can be omitted from Equation 3.9 while computing its derivative for simplicity. The derivative is as follows:

$$\frac{\mathrm{d}\boldsymbol{\mathcal{L}}}{\mathrm{d}\mathbf{W}_k} = 2\Big\{\mathcal{F} - \beta\Big\}\Big\{-\frac{\mathrm{d}}{\mathrm{d}\mathbf{W}_k}\beta\Big\} + \lambda\mathbf{W}_k,$$

where, $\beta = f_k(\mathbf{W}_k\mathcal{Z}_k) \oplus f_v(\mathbf{W}_v\mathcal{Z}_v) \oplus f_h(\mathbf{W}_h\mathcal{Z}_h)$. As $f_i(\cdot)$ is ReLU function for $i \in \{k, v, h\}$, we get:

$$\frac{\mathrm{d}\beta}{\mathrm{d}\mathbf{W}_k} = \begin{cases} 0, & \mathbf{W}_k\mathcal{Z}_k < 0. \\ \mathbf{W}_k\mathcal{Z}_k, & \mathbf{W}_k\mathcal{Z}_k > 0. \end{cases} \tag{3.10}$$

Considering the case of $\mathbf{W}_k\mathcal{Z}_k > 0$ for $f_i(\cdot)$ and substituting $\frac{\mathrm{d}\beta}{\mathrm{d}\mathbf{W}_k}$, Equation 3.10 can be written as:

$$\begin{aligned} 2(\mathcal{F} - \mathbf{W}_k\mathcal{Z}_k - \mathbf{W}_v\mathcal{Z}_v - \mathbf{W}_h\mathcal{Z}_h)(-\mathbf{W}_k\mathcal{Z}_k) + \lambda\mathbf{W}_k = 0. \\ \mathbf{W}_k = \frac{2\mathcal{Z}_k(\mathcal{F} - \mathbf{W}_v\mathcal{Z}_v - \mathbf{W}_h\mathcal{Z}_h) - \lambda}{4\mathcal{Z}_k^2}. \end{aligned} \tag{3.11}$$

For case $\mathbf{W}_k\mathcal{Z}_k > 0$, Equation 3.10 can be written as:

$$\frac{\mathrm{d}\boldsymbol{\mathcal{L}}}{\mathrm{d}\mathbf{W}_k} = 2\Big\{\mathcal{F} - \beta\Big\}\Big\{ - \mathbf{W}_k\mathcal{Z}_k\Big\} + \lambda\mathbf{W}_k. \tag{3.12}$$

Estimating the second-order derivative,

$$\frac{\mathrm{d}^2\boldsymbol{\mathcal{L}}}{\mathrm{d}^2\mathbf{W}_k} = 2\Big\{ -\frac{\mathrm{d}\beta}{\mathrm{d}\mathbf{W}_k}\Big\}\Big\{ -\mathbf{W}_k\mathcal{Z}_k\Big\} + 2\Big\{\mathcal{F}-\beta\Big\}\Big\{ -\frac{\mathrm{d}}{\mathrm{d}\mathbf{W}_k}\mathbf{W}_k\mathcal{Z}_k\Big\} + \lambda\frac{\mathrm{d}}{\mathrm{d}\mathbf{W}_k}\mathbf{W}_k,$$
$$= 2\Big\{\mathbf{W}_k\mathcal{Z}_k\Big\}^2 + 2\mathcal{Z}_k\Big\{\beta-\mathcal{F}\Big\} + \lambda. \tag{3.13}$$

Next, to prove the convexity of the optimization problem in Equation 3.8, $\frac{\mathrm{d}^2\boldsymbol{\mathcal{L}}}{\mathrm{d}^2\mathbf{W}_k} > 0$. For simplifying the representation, Equation 3.13 is rearranged as follows:

$$\frac{\mathrm{d}^2\boldsymbol{\mathcal{L}}}{\mathrm{d}^2\mathbf{W}_k} = \underbrace{2\Big\{\mathbf{W}_k\mathcal{Z}_k\Big\}^2 + \lambda}_{\mathbf{T1}} + \underbrace{2\mathcal{Z}_k\Big\{\beta-\mathcal{F}\Big\}}_{\mathbf{T2}}. \tag{3.14}$$

In Equation 3.14, **T1** is always greater than zero, as it is a summation of a square term ($\{\mathbf{W}_k\mathcal{Z}_k\}^2$) and regularization parameter λ. The value assigned to λ is always greater than zero. β lies in the range between 0 and ∞. There are two possible cases in order to show $\mathbf{T1}+\mathbf{T2} > 0$, which are as follows:

1. $\{\beta \geq 1\}$: In this case, the value of $\mathbf{T1}+\mathbf{T2} > 0$ only if $\{\beta-\mathcal{F}\} > 0$. Here, the feature matrix $\mathcal{F}$ contains only floating point numbers less than 1, as it is obtained by applying a Tanh function in the deep leaning models. Now, since $\beta \geq 1$ and $\mathcal{F} < 1$, we get $\{\beta-\mathcal{F}\} > 0$.
2. $\{0 < \beta < 1\}$: In this case, as the value of β is less than 1, there may be a chance that $\mathbf{T2} < 0$. However, **T1** is having a sufficiently high positive value as it incorporates a square term and summation with regularization λ. It means that **T1** can adjust the negative value of **T2** while preserving the positivity of $\mathbf{T1}+\mathbf{T2}$. Additionally, as the value of β decreases the weight $\mathbf{W}_k$ also decreases, which in turn will increase the regularization λ. It also helps to hold $\mathbf{T1}+\mathbf{T2} > 0$.

We can now conclude that the second order-derivative is non-negative, which is

required to prove the convexity. Similarly, the classifier $\mathbf{\Pi}$ can obtain $\mathbf{W}_v$ by fixing $\mathbf{W}_k$ and $\mathbf{W}_h$, and $\mathbf{W}_h$ by fixing $\mathbf{W}_k$ and $\mathbf{W}_v$.

3.3.3.3 Distance threshold computation

To identify an unseen class, the classifier $\mathbf{\Pi}$ needs to learn a distance threshold that can separate an unseen class from the seen classes. For each training instance, $\mathbf{\Pi}$ first obtains an attribute vector by using the feature matrix ($\mathcal{F}$) and weight matrices ($\mathbf{W}_k$, $\mathbf{W}_v$, and $\mathbf{W}_h$). Later, this attribute vector is compared with known attribute vector of each class label in $\mathcal{A}$ (constructed previously). Finally, the class label of nearest attribute vector is assign to that training instance. Let $\mathcal{T}$ denotes a matrix of size $N \times a$ where a row $\mathcal{T}_i$ corresponds to an attribute vector, obtained by $\mathbf{\Pi}$, for $\mathbf{x}_i \in \mathcal{D}$. This work computes a separate distance threshold for each class label L_l, where $1 \leq l \leq k$. Let δ_l denotes a distance threshold for L_l, which is calculated as:

$$\delta_l = \underset{1 \leq i \leq N_l}{max} \Big\{ \sum_{j=1}^{a} |\mathcal{T}_{i,j} - \mathcal{T}'_j| \Big\}, \tag{3.15}$$

where, N_l is the number of instances with class label L_l in $\mathcal{D}$ and $\mathcal{T}'$ is a median attribute vector for L_l.

3.3.4 Prediction of class label

DeepZero model first extracts features from the given testing instance, using the trained deep learning models. Later, the classifier $\mathbf{\Pi}$ uses these features to predict an attribute vector. Finally, it predicts a class label by using computed distance thresholds.

Figure 3.5 illustrates the process of prediction of a class label for a testing instance $\mathbf{x}^t \notin \mathcal{D}$, using the built classifier $\mathbf{\Pi}$. The extracted feature vector for $\mathbf{x}^t$ is denoted by $\mathcal{F}^t$. In the figure, $\mathcal{A}^t$ is an attribute vector and δ is a set of computed distance thresholds for the seen class labels as $\delta = \{\delta_1, \delta_2, \cdots, \delta_k\}$. The classifier compares the

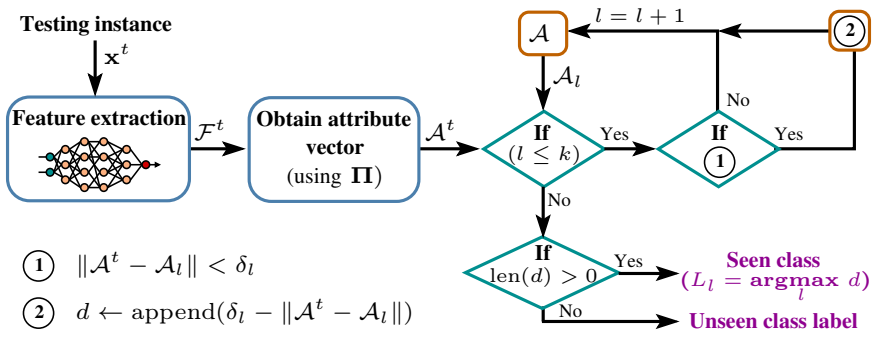


Figure 3.5: Prediction of class label of a testing instance in DeepZero model.

testing attribute vector $\mathcal{A}^t$ with the attribute vectors of seen classes in $\mathcal{A}$. For a class label L_l, if $\|\mathcal{A}^t - \mathcal{A}_l\| < \delta_l$ holds, value $(\delta_l - \|\mathcal{A}^t - \mathcal{A}_l\|)$ is appended into a distance list d. Next, after estimating distance from k classes, if length of d is null then an unseen class label is assigned to $\mathbf{x}^t$. However, if d is not empty then the class having highest value of $\delta_l - \|\mathcal{A}^t - \mathcal{A}_l\|$ is the seen class L_l for $\mathbf{x}^t$.

The time complexity of Algorithm 3.1 mainly depends on CNN and LSTM models. The CNN model takes $O\big(\sum_{i=1}^{l} p_{i-1} \cdot s_i^2 \cdot f_i \cdot o_i^2\big)$ per time step of the input sequence [93], where l is the number of convolution layers, p is the number of input channels, f is number of filters, s is filter size, and o is the size of output feature, at i^{th} convolution layer. Next, the time complexity of LSTM model is $O(W)$ [94], where W is the number of weight parameters. As DeepZero uses sequential combination of CNN and LSTM, total time complexity can be given as $O\big((ps^2fo^2 + W)M\big) = O(WM)$, where M is the length of the instance and $\{p, s, f, o\} \ll W$.

3.4 Feature extraction using deep learning

DeepZero model extracts features from the given dataset $\mathcal{D}$ using a sequential combination of two deep learning models, *i.e.,* CNN and LSTM. The model first computes deep features and then combines them with hand-crafted features to obtain a feature matrix (denoted by $\mathcal{F}$). CNN model initially extracts the spatial features from the sensory data instances. These extracted features act as input to the LSTM model. Since we

Algorithm 3.1: DeepZero model

Input: A labeled dataset $\mathcal{D}$ with N instances. Each instance consists of data from n sensors. A test instance $\mathbf{x}^t \notin \mathcal{D}$;

Output: An attribute matrix ($\mathcal{A}$), a feature matrix ($\mathcal{F}$), distance thresholds δ, and predicted label for $\mathbf{x}^t$;

Build a one-hot encoding matrix $\mathcal{Z}_k$.
Build a word2vec matrix $\mathcal{Z}_v$.
Build a human-annotated semantic matrix $\mathcal{Z}_h$.
Compute $\boldsymbol{\mathcal{F}}$ from $\mathcal{Z}_k, \mathcal{Z}_v$, and $\mathcal{Z}_h$ using Equation 3.2.
Obtain $\mathcal{A} \leftarrow g(\boldsymbol{\mathcal{F}})$, as shown in Equation 3.3.
/* Obtain feature matrix ($\mathcal{F}$) using training dataset $\mathcal{D}$ */
for *each sensor* $j \leftarrow 1$ *to* n **do**
    Pass $\mathcal{D}_j$ through a sequence of 5 convolution layers.
    Flatten the output of last convolution layer.
    Pass the flattened output through a FC layer.
Concatenate the n outputs of FC layers and reshape.
Obtain deep features using LSTM model.
$\mathcal{F} \leftarrow$ merge deep and hand-crafted features.
Find $\min_{\mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_h} (\mathcal{L}_{term} + \mathcal{R}_{term})$, as given in Equation 3.7.
Obtain $\mathbf{W}_k, \mathbf{W}_v$, and $\mathbf{W}_h$ using Equation 3.11.
Compute thresholds $\delta = \{\delta_1, \delta_2, \cdots, \delta_k\}$ using Equation 3.15.
Extract features $\mathcal{F}^t$ and obtain attribute vector $\mathcal{A}^t$.
for *each class* $l \leftarrow 1$ *to* k **do**
    if $\|\mathcal{A}^t - \mathcal{A}_l\| < \delta_l$ **then**
        Assign class label L_l to $\mathbf{x}^t$.
        break.
if x^t *is not recognized* **then**
    Assign an unseen or new class label to $\mathbf{x}^t$.

are using sensory data sequences; thus, the LSTM model can easily fetch temporal features. Additionally, the memory cell of LSTM can store previous time step information to learn the temporal dependencies. Furthermore, the LSTM does not require fine-tuning the multiple parameters and work well over a broad range of parameters, such as learning rate, input gate biases and output gate biases. Thus, the DeepZero model uses both spatial and temporal features, which improves the recognition performance.

For a given locomotion dataset $\mathcal{D}$, DeepZero model first splits $\mathcal{D}$ into the smaller datasets $\mathcal{D}_j$ which consists of the data of j^{th} sensor only, where $1 \leq j \leq n$. Each dataset

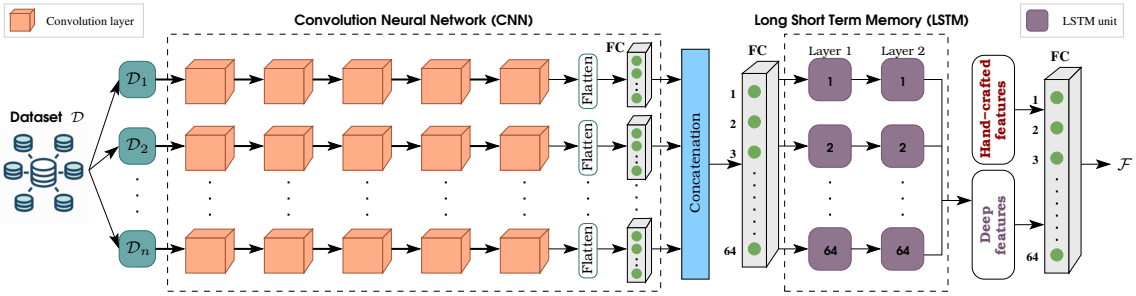


Figure 3.6: Feature extraction framework using CNN and LSTM.

$\mathcal{D}_j$ is first passed through a sequence of five convolution layers followed by a flatten layer. The vectors of features (for $\mathcal{D}_j$) are concatenated and passed through FC layer to get a fixed number of features. Later, these features are given as input to a sequence of two LSTM units that provide a vector of features called as deep features. Finally, the deep features are combined with hand-crafted features and are passed through a FC layer to obtain a desired number of features corresponding to each instance $\mathbf{x}_i$ of $\mathcal{D}$, where $1 \leq i \leq N$. A feature matrix of size $N \times a$ is obtained, where a denotes the number of features obtained for each $\mathbf{x}_i$. The features are generated instance wise during training and the weights are updated after each instance. It also indicates that CNN and LSTM layers do not generate features among different instances. Figure 3.6 shows the feature extraction framework for dataset $\mathcal{D}$ using CNN and LSTM.

Let us consider an example scenario for the feature extraction process with a single input instance. The dataset $\mathcal{D}$ consists of 100 data points of two sensors, *i.e.,* 3-axis accelerometer and 3-axis gyroscope. The instance is first partitioned into two parts ($\mathcal{D}_1$ for accelerometer and $\mathcal{D}_2$ for gyroscope) to get sensor-wise data. Next, data of each of three axes is channelized to make an input of shape $1 \times 3 \times 100$, which is given to the first convolution layers. As we used 64 3-channel 1×1 filters in the convolution layer, it produces an output of 1×64, which is further given as input to the next convolution layer. After five convolution operations on five different layers, an output of dimension 1×64 is obtained for each sensor. The convolutional output passed through a separate fully connected layer, and the outputs are concatenated to generate

an output of dimension 128×1. Further, this output (128×1) is supplied as input to a fully connected layer of 64 neurons that produce input for 2-LSTM layers (64 cells each). Finally, we obtain the output of size 1×64 as features against single input.

3.4.1 Convolution layer

The dataset $\mathcal{D}_j$ contains total N instances with length of sensory measurements M , where each instance is a m-dimensional (*e.g.,* $m = 3$ for gyroscope data with x, y, and z axis). The dataset is therefore first channelized and then given to the convolution layer where m-channel filter of size 1×1 is applied on each instance of the channelized data with one stride. As a result, a feature matrix of size $N \times m$ is obtained. In this work, the convolution layer uses 64 filters which results 64 matrices of size $N \times 64$.

3.4.2 LSTM unit

LSTM unit comprises of four main components: a cell and three gates (input, forget, and output). At any time t, the LSTM unit takes three inputs: 1) current input vector, denoted by x_t, 2) output (or hidden) state at time $t-1$, denoted by h_{t-1}, and 3) cell state at $t-1$, denoted by c_{t-1}. Let $\mathcal{W}_k$ and $\mathcal{U}_k$ denote a weight matrix for a component k for x_t and h_{t-1}, respectively, where $k = \{c, i, f, o\}$ and c for cell, i for input gate, f for forget gate, and o for output gate are mathematically represented as:

- **Input gate:** In the input gate, the current input x_t and hidden state h_{t-1} are passed through a sigmoid and a Tanh function (denoted by $g(\cdot)$).

$$u_t = \sigma(\mathcal{W}_i x_t + \mathcal{U}_i h_{t-1}) \otimes g(\mathcal{W}_c x_t + \mathcal{U}_c h_{t-1}), \tag{3.16}$$

  where, symbol $\otimes$ denotes an element-wise product.

- **Forget gate** It decides what information of x_t and h_{t-1} should be kept or forgot

using sigmoid function ($\sigma(\cdot)$).

$$f_t = \sigma(\mathcal{W}_f x_t + \mathcal{U}_f h_{t-1}). \tag{3.17}$$

- **Cell** It updates the cell state by using previous cell state c_{t-1}, f_t, and u_t as follows:

$$c_t = f_t \otimes c_{t-1} + u_t. \tag{3.18}$$

- **Output gate** It operates on input vector x_t, hidden state h_{t-1}, and updated cell state c_t, to decide the next hidden state.

$$h_t = \sigma(\mathcal{W}_o x_t + \mathcal{U}_o h_{t-1}) \otimes g(c_t). \tag{3.19}$$

3.5 Evaluation of DeepZero model

In this section, we first discuss the setup to collect a Locomotion Mode Recognition (LMR) dataset and then evaluate DeepZero model on this dataset along with a publicly available Sussex-Huawei Locomotion (SHL) dataset [77]. We carried out experiments to answer the following questions:

- What is the minimum number of epochs, where the performance of DeepZero model stabilizes? (Section 3.5.4.1)
- What is class-wise performance of DeepZero model on seen classes? (Section 3.5.4.2)
- How does the performance influence with one unseen class? (Section 3.5.4.3)
- What is the accuracy with varying number of unseen classes? (Section 3.5.4.4)
- How does the attribute matrix impact on the performance? (Section 3.5.4.5)

3.5.1 Data collection

In this work, we developed an android application to collect the sensory data for six locomotion modes ($k = 6$) including bicycle ($\mathbf{a_1}$), bike ($\mathbf{a_2}$), car ($\mathbf{a_3}$), auto rickshaw

($\mathbf{a_4}$), bus ($\mathbf{a_5}$), and train ($\mathbf{a_6}$). The application uses three sensors ($n = 3$): accelerometer, gyroscope, and magnetometer. The sampling rate of each of the sensors is set to 100 Hz. For each locomotion mode, the data is collected from 10 participants (5 males and 5 females). The application provides a menu to the participant for selecting the locomotion mode, as shown in Figure 3.7. Once a mode is selected, the application records measurements for 60 seconds. As each participant has provided 200 instances of each locomotion mode, the dataset contains total 12000 instances.

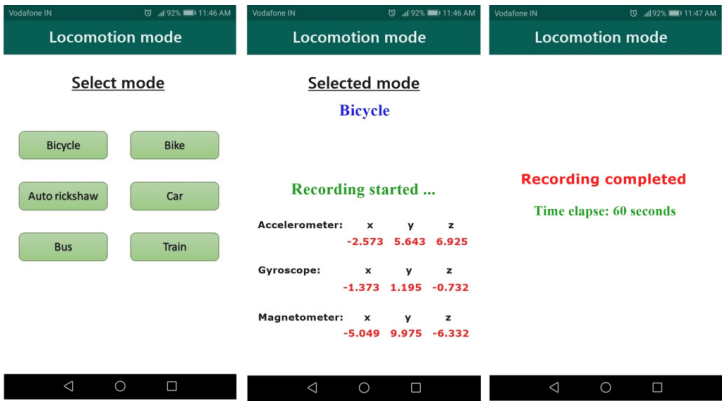


Figure 3.7: User interface of developed application for data collection.

Preprocessing: As the mode selection needs user interaction with the application, some unwanted sensory measurements are recorded in the first 10 seconds which is the time elapsed in putting the smartphone back to the pocket. This work, therefore, removed the measurements of first 10 seconds from all the instances. Now, each instance in the dataset contains sensory measurement for 50 seconds, *i.e.*, $50 \times 100 = 5000$ data points. Further, the instances are reduced to 250 data points by taking a mean value of non-overlapping windows of size 20. At this point, the dataset contains 12000 labeled instances where each instance consists of 250 data points ($M = 250$). We call the preprocessed dataset as LMR.

3.5.2 SHL dataset

The SHL dataset was given in locomotion activity recognition challenge 2018 [55]. It contains two sub-datasets: training with 16310 instances and testing with 5698 instances. The dataset consists the data of eight locomotion modes ($k = 8$) including bike ($\mathbf{b_1}$), car ($\mathbf{b_2}$), bus ($\mathbf{b_3}$), subway ($\mathbf{b_4}$), train ($\mathbf{b_5}$), still ($\mathbf{b_6}$), walk ($\mathbf{b_7}$), and run ($\mathbf{b_8}$). We use the data of the following five sensors ($n = 5$): linear acceleration, accelerometer, orientation, gyroscope, and magnetometer. As each instance consists of 6000 samples, we first preprocess the dataset to reduce the length of instances to 300 samples ($M = 300$) by taking a mean value of window size 20.

3.5.3 Parameter settings during implementation

We have implemented DeepZero model in Python language using functional API of Keras. The feature extraction model is a lightweight model, where values of different sensors are input to different CNN layers. It achieves significant accuracy in less time and preserves system energy. We come up with this feature extraction model after a rigorous experimental analysis, where we compared it with two widely adopted feature extraction techniques, *i.e.,* Bidirectional LTSM [95] and Time-gated LSTM [96]. Further, values of different characteristics, used for obtaining human-annotated matrix, are given in Table 3.1.

3.5.4 Experimental results

In this section, several experiments are carried out to evaluate DeepZero model on LMR and SHL datasets and the obtained results are discussed in detail.

3.5.4.1 Number of epochs for evaluation

In this experiment, the performance of DeepZero model is reported on the training data during construction of classifier. Figure 3.8 illustrates the accuracy and F_1-score

Table 3.1: Characteristics used in human-annotated semantic matrix.

Locomotion mode	Characteristics					
	Speed (km/hr)	Capacity (Seats)	Power (hp ×100)	Fuel	Pathway	Wheel count
Bicycle	15	1	70	3	0	2
Bike	80	2	500	0	0	2
Auto rickshaw	30	4	700	0	0	3
Car	120	4	17000	0	0	4
Bus	100	45	24000	1	0	6
Subway	105	360	303600	2	1	24
Train	110	1464	700000	2	1	56
Still	0	0	0	3	0	0
Walk	5	0	37	3	0	0
Run	12	0	67	3	0	0

results on LMR and SHL datasets using different number of epochs. Figure 3.8, we observe a rapid increment in the accuracy up to 25 epochs and a marginal increment afterwards. It is also observed that the model achieved maximum accuracy ($\approx 99\%$) using 50 epochs for both datasets. Similar observations can be made for F_1-score. This work therefore performs all the subsequent experiments with 50 epochs.

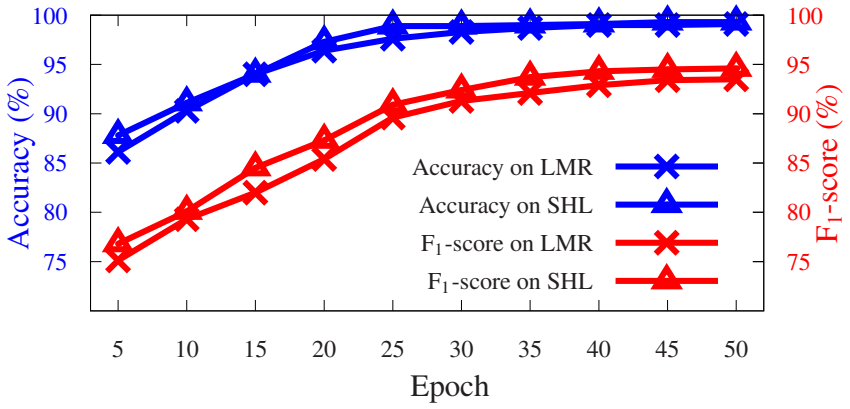
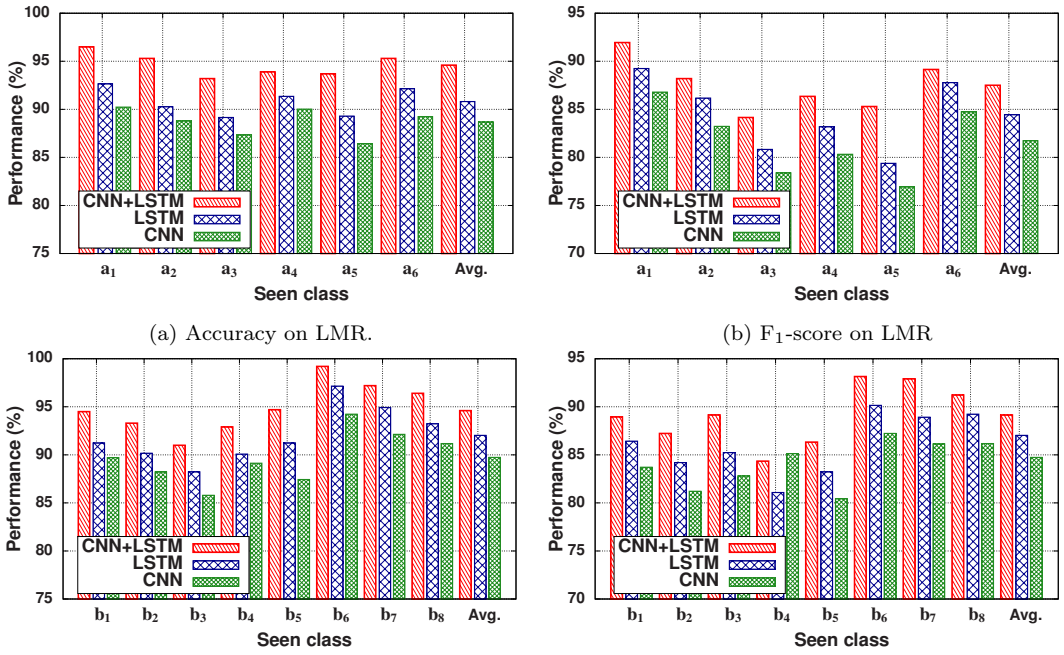


Figure 3.8: Performance results on LMR and SHL datasets during training.

3.5.4.2 Class-wise performance on seen classes

This work evaluates performance of the model for seen classes by considering all locomotion modes during training. Figure 3.9(a) and Figure 3.9(b) illustrate that DeepZero

model achieves maximum accuracy and F_1-score on $\mathbf{a_1}$ (bicycle) class as it provides better identifiable patterns in the sensory measurements. For SHL dataset, the instances of $\mathbf{b_6}$ (still) class are recognized with an accuracy of 99.3% and F_1-score of 93.2%, as shown in Figure 3.9(c) and Figure 3.9(d). Figure 3.9(a) and Figure 3.9(b) also illustrate accuracy and F_1-score on LMR dataset, respectively, where features are extracted using CNN-LSTM, LSTM, and CNN models. From the results, we observe that the features extracted from CNN-LSTM outperforms LSTM with $\approx 3\%$ and CNN with $\approx 5\%$ for all the classes.



(a) Accuracy on LMR.

(b) F_1-score on LMR

(c) Accuracy on SHL.

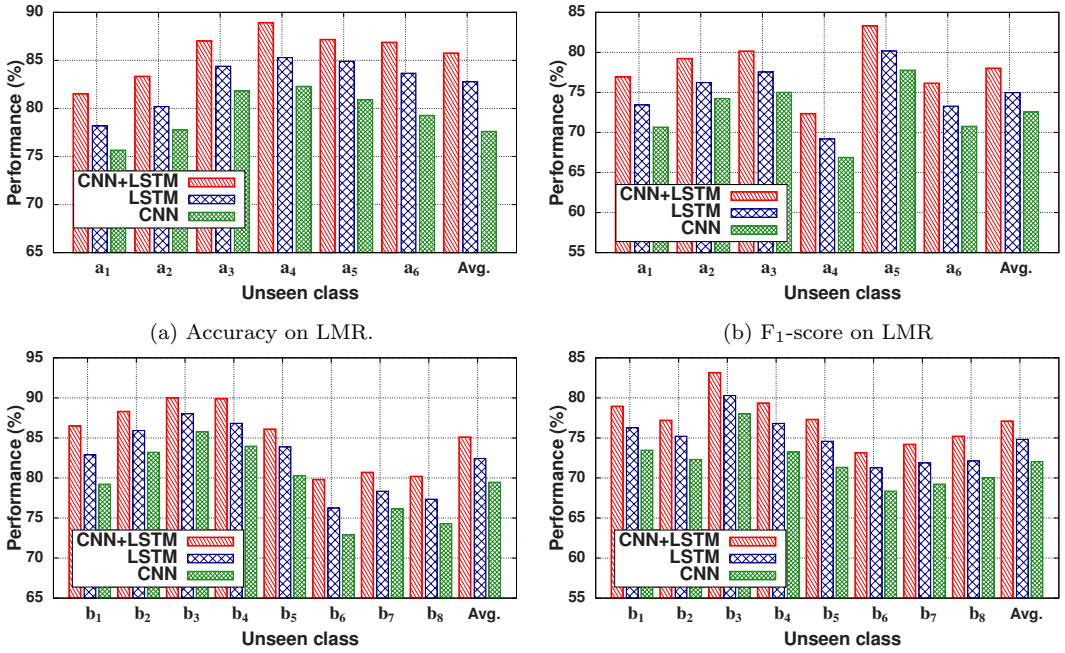
(d) F_1-score on SHL.

Figure 3.9: Performance of the proposed approach for seen classes where features are extracted from combination of CNN-LSTM, LSTM, and CNN models.

3.5.4.3 Performance of DeepZero with one unseen class

Next, DeepZero model is evaluated by keeping one class as unseen. Figure 3.10(a) shows the results for LMR dataset where the classifier is trained on five classes (one class unseen) and tested on all the six classes. The classifier obtained an average accuracy of 83.7%, which indicates that DeepZero model is able to capture significant

amount of identifiable information. Further, in Figure 3.10(b) the F_1-score is greater than 70% for all the classes. Similarly, Figure 3.10(c) and Figure 3.10(d) shows the performance of the classifier for SHL dataset.



(a) Accuracy on LMR.

(b) F_1-score on LMR

(c) Accuracy on SHL.

(d) F_1-score on SHL.

Figure 3.10: Performance of the proposed approach for unseen classes where features are extracted from combination of CNN-LSTM, LSTM, and CNN models.

3.5.4.4 Accuracy results with multiple unseen classes

Figure 3.11 illustrates the accuracy results where the classifier is trained and tested on different combinations of seen and unseen classes. As LMR dataset consists of six classes, at most four of them can be unseen as shown in Figure 3.11. The result shows that the classifier is able to achieve an accuracy of 36.4% even with the four unseen classes. Similarly, at most six of the eight classes can be unseen for SHL dataset and the result is shown in Figure 3.11. It is interesting to observe that the classifier achieve the accuracy of 35.2% when six out of eight classes are unseen. The main reason for such a drop in accuracy is directly depend on the number of seen classes in the dataset.

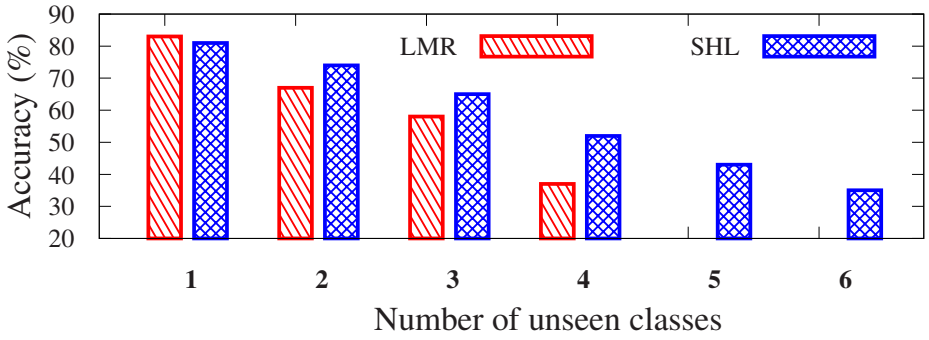


Figure 3.11: Accuracy results of DeepZero with multiple unseen classes.

3.5.4.5 Impact of attribute matrix on the performance

This work considers two variants of DeepZero model to evaluate the impact of attribute matrix. These two variants of model are: V_1 and V_2. V_1 is same as DeepZero model while V_2 is different as it does not use attribute matrix during recognition. Figure 3.12 shows the performance comparison between V_1 and V_2 models on LMR dataset using accuracy and F_1-score, respectively. The result shown a substantial drop in the accuracy from V_1 to V_2 model, which clearly indicates the efficacy of fusion based attribute matrix in the locomotion mode recognition. We also observe similar drop in F_1-score values, as shown in Figure 3.12.

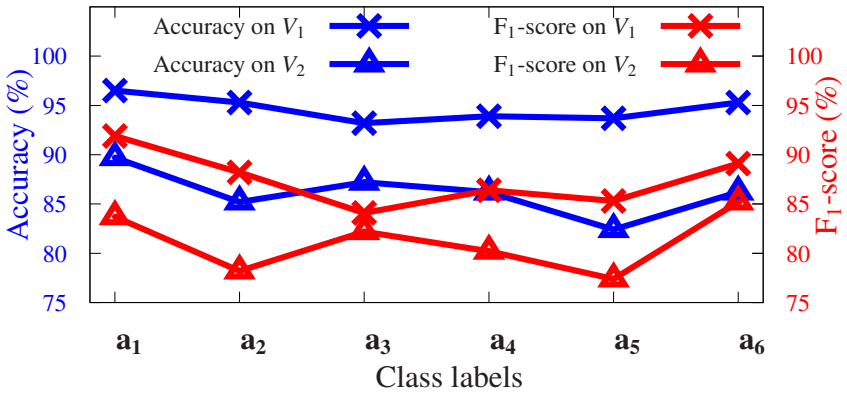


Figure 3.12: Impact of attribute matrix on performance for LMR dataset.

3.5.5 Comparison with existing approaches

We compare DeepZero model with the existing approaches including Learning Transportation Mode (LTM) [26], Human Activity Recognition with FFT (HARF) [28], TEDLSTM [27], and Nuactiv [66]. First three approaches use deep learning models for motion based activity recognition but can not identify the unseen classes. On the other hand, Nuactiv can identify the unseen classes and uses machine learning during identification. As there exists no prior work for the unseen locomotion mode identification using deep learning, we first compare DeepZero model with LTM, HARF, and TEDLSTM approaches by considering all the classes during training and testing. Later, the model is also compared with Nuactiv approach for unseen classes.

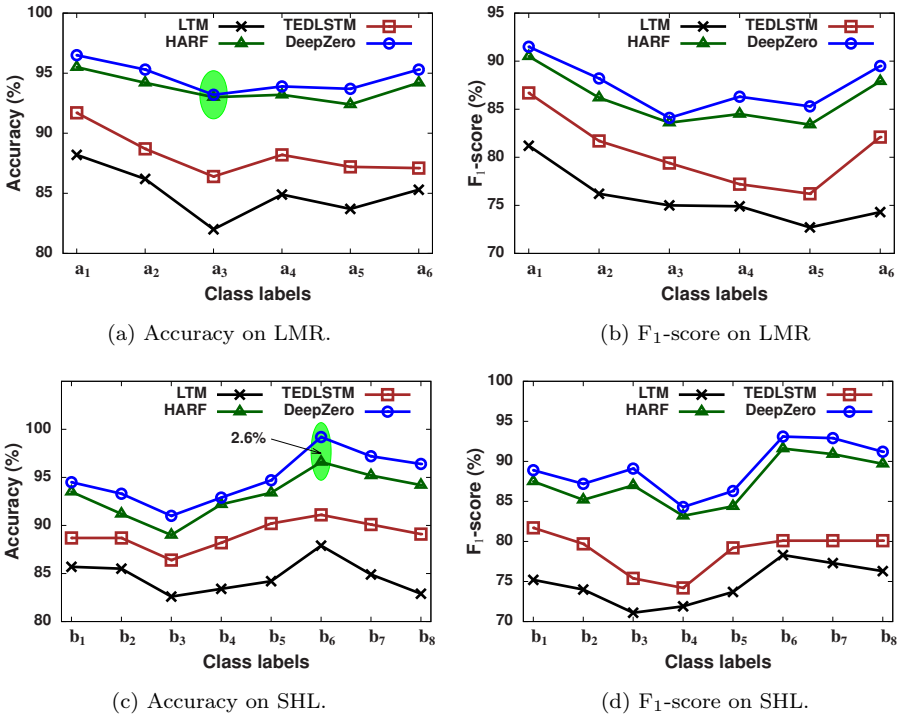
3.5.5.1 Performance comparison on seen classes

Figure 3.13 illustrates the performance comparison of DeepZero model with the existing approaches using accuracy and F_1-score.

- DeepZero model outperforms all existing approaches for both LMR and SHL datasets on accuracy and F_1-score. Though DeepZero model shows small performance gain of $(1.6 \pm 1)\%$ compared to HARF approach but able to identify the unseen classes as well.
- As all the existing approaches employed deep learning models for recognition, they are able to obtain more than 85% of average accuracy and 75% of F_1-score.
- The existing approach HARF provided more accurate results compared to LTM and TEDLSTM, which indicates that it extracts better identifiable features.

3.5.5.2 Performance comparison on unseen classes

Next, this work compares the performance of DeepZero model with Nuactiv. Figure 3.14(a) and Figure 3.14(b) show the obtained accuracy results for LMR and SHL datasets, respectively. The accuracy shown in the results is a median accuracy obtained



(a) Accuracy on LMR.

(b) F_1-score on LMR

(c) Accuracy on SHL.

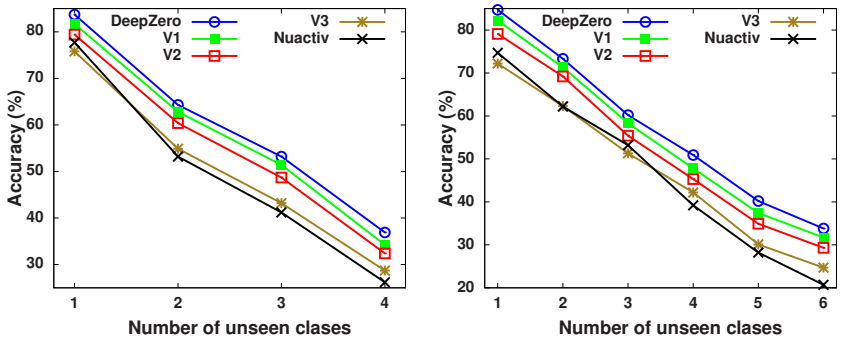
(d) F_1-score on SHL.

Figure 3.13: Comparison of DeepZero model with existing approaches using LMR and SHL datasets.

for all combinations of unseen classes. DeepZero model gains maximum accuracy 12% to that of Nuactiv when number of unseen classes is three, as shown in Figure 3.14(a). Similar observations can be made about Figure 3.14(b). To illustrate the impact of different semantic matrices individually, we perform the experiments using three versions of DeepZero: *V1*, *V2*, and *V3*. The version *V1* denotes DeepZero model with human-annotated matrix only. Similarly, *V2* and *V3* represent the model with word2vec and one-hot encoding, respectively.

3.6 Conclusion

This chapter proposed DeepZero model to identify a locomotion mode using smartphone sensors. Unlike existing approaches, DeepZero model utilized the concept of ZSL to build a classifier that has the capability to identify an unseen locomotion mode for which no training instances are given. The model constructed an attribute matrix by



(a) LMR dataset. (b) SHL dataset.

Figure 3.14: Comparison of accuracy results for DeepZero model with its three versions *V1*, *V2*, *V3*, and Nuactive using LMR and SHL datasets.

fusing three semantic matrices. The attribute matrix is constructed by using only the class labels of datasets. A deep learning based feature extraction framework is also developed to extract the features from the dataset. The attribute and features are used during the construction of the classifier. This work carried several experiments to evaluate the performance of DeepZero model using two locomotion mode datasets. The experimental results shown that the model provided an accuracy of more than 94% for seen classes and more than 80% on one unseen class.