Chapter 4

A locomotion mode recognition approach with noisy labels

The previous chapter presented a deep learning-based model capable of recognizing both seen and unseen locomotion modes. The incorporation of zero-shot learning using the fusion of available and hand-crafted semantic metrics has enhanced the robustness of the built classifier against the unseen class labels. This chapter considers the problem of noisy labels in the dataset that adversely affect the performance of the model.

4.1 Introduction and major contributions

Locomotion activity recognition has been a field of great research interest that helps to understand the movement patterns of human [24,25]. Most of the recent studies [25] use sensory data for recognizing locomotion activities due to easier availability of sensors with smartphone and wearable devices. Information about the locomotion activities (or modes) helps to estimate routine transportation expenditure, travel time, traffic congestion, and journey planning. Prior work employed two widely used techniques, *i.e.,* machine learning [38, 40, 97] and deep learning [39, 98, 99] for locomotion activity recognition using sensory data. The main objective of locomotion mode recognition is

to build a classifier that can predict a transportation mode.

Deep learning techniques have shown impressive performance on the labeled dataset because of their automatic feature extraction capabilities from the raw data [13, 31, 32]. However, these techniques require a large amount of training data with correctly annotated labels. Acquiring such annotated dataset is expensive and time-consuming. The data analysts, therefore, employ different annotation techniques such as labeling through crowdsourcing, web-based queries, and so on [12]. However, these annotation techniques may introduce noisy labels in the dataset as the labeling is carried out by non-expert volunteers. If noisy labels are present in the dataset then the classifier learns a wrong mapping which results in performance diminution [13]. Figure 4.1 illustrates an example of locomotion mode recognition. A deep learning-based classifier is built by learning a mapping between raw sensory data and their annotated labels with some wrong labels of car and auto-rickshaw as bike and bus (in red color), respectively.

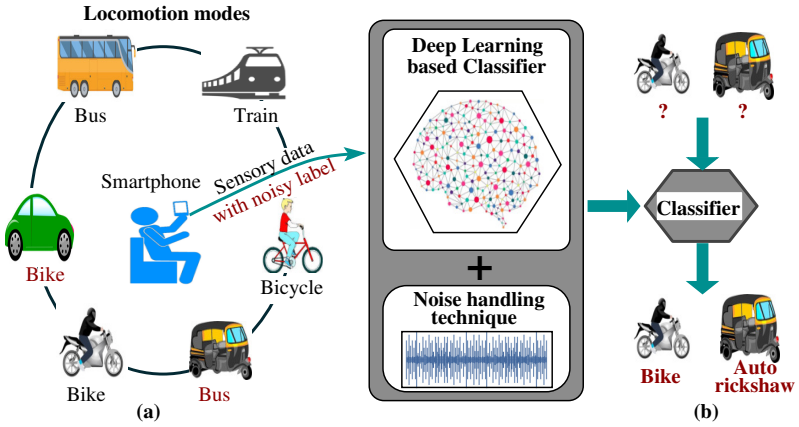


Figure 4.1: An example of locomotion mode recognition with noisy labels using the deep learning-based classifier. The wrong labels of the car as bike and auto rickshaw as bus (in red color) indicate noisy labels in training data.

• **Motivation of this work:** The existing approaches [31–37] require prior information about the concentration of noisy labels in the dataset. The prior information helps in the easier handling of noisy labels by setting parameters accordingly. However, the approaches [31–41] do not provide any mechanism to determine the concentration. Thus, it generates the requirement for a noisy handling approach that can work with-

out prior information and determine noisy labels concentration after training. The information about noisy labels helps in improving the annotation mechanism for the upcoming training processes. Additionally, some of the prior studies [33, 39, 40] can handle a low concentration. Therefore, in the real-world scenario for recognizing locomotion modes, where the chance of noisy labels are high, we need a mechanism that can achieve adequate performance on a higher concentration.

In this chapter, we address the problem: *how to recognize a locomotion mode using deep learning models in the presence of noisy labels without prior information of noise concentration?* To solve the problem, this work proposes a deep learning-based approach called **L**ocomotion mode **R**ecognition with **N**oisy **L**abels (LRNL). The approach builds an ensemble model by developing three different models (build separately) *i.e.,* conventional, noise adaptive, and noise corrective.

• **Major contributions** To the best of our knowledge, this is the first work to address the recognition of locomotion modes with noisy labels using an ensemble model. The contributions are:

- This work proposes an approach, namely LRNL, that builds an ensemble model using deep learning models to recognize a locomotion mode in the presence of noisy labels. It does not require any prior information about noisy labels concentration.
- Next, we propose three different models, *i.e.,* conventional, noise adaptive, and noise corrective, which work as input components of the ensemble model. The conventional model extracts feature using deep learning and provide robustness against the low concentration of noisy labels. The noise adaptive model introduces a new loss function incorporating two dynamic variables whose values are adjusted to handle a moderate concentration. Later, the noise corrective model uses the low-rank estimate of the noisy labels matrix during training to provide robustness against high noise concentration.
- Further, in the absence of prior information about noisy labels, we select an

appropriate noise handling model by associating weights to each model using the ensembling technique. These weights adjust their values for prioritizing one model over others, provided the summation of the weights is unity. In addition, the fractional weights also help in figuring out the concentration of noisy labels.

- Finally, we conduct various experiments to evaluate the performance of the LRNL approach using collected LMR dataset along with existing datasets [55, 56].

The rest of chapter is organized as follows. Next section describes the preliminaries. Section 4.3 proposes an approach for locomotion mode recognition with noisy labels. Later, Section 4.4 presents the experimental evaluation of the LRNL approach. Finally, the chapter is concluded in Section 4.5.

4.2 Preliminaries

Let $\mathcal{D} = \{\mathbf{X}_{tr}, \mathbf{Y}\}$ represents training data of locomotion modes, where $\mathbf{X}_{tr} \in \mathbb{R}^{N \times M}$ holds N labeled instances and each of which consists of M data points, and $\mathbf{Y} \in \mathbb{R}^{N \times k}$ is one-hot encoded label matrix for k class labels in the dataset. Each element of matrix $\mathbf{Y}$ is either 0 or 1. The value of an element $\mathbf{Y}_{(ij)}$ is 1 if i^{th} instance belongs to j^{th} class label ($\forall j \in \{1, 2, \ldots k\}$) and 0 otherwise. An instance of $\mathbf{X}_{tr}$ is a vector of length M, denoted by $\mathbf{x}_{tr}$. Similarly, an instance of $\mathbf{Y}$, denoted by $\mathbf{y}$, is a vector of length k. Let $\mathbf{X}_{te}$ denotes a matrix of testing data, and $\mathbf{Y}_{te}$ is a prediction matrix obtained by ensemble model. As the class labels may be noisy in the training dataset $\mathcal{D}$, we also use $\mathbf{Y}'$ for representing a noisy label matrix. In deep learning models, it is crucial to determine correct mapping between training data and true labels, *i.e.,* H: $\mathbf{X}_{tr} \rightarrow \mathbf{Y}$. The mapping H can be obtained from various intermediate transformations (denoted by o) at different layers of the deep neural networks. For example, if there are l layers in the neural network, then H can be estimated as $H = H_1(\mathbf{X}_{tr})\text{o}H_2(\mathbf{X}_{tr})\text{o}\ldots\text{o}H_l(\mathbf{X}_{tr})$.

Definition 4.1 (True label probability) *For a given noisy label vector* $\mathbf{y}'_i$ *of* i^{th} *training instance* ($\mathbf{y}'_i \in \mathbf{Y}', \forall 1 \leq i \leq N$) *against true label vector* $\mathbf{y}_i \in \mathbf{Y}$*, the probability*

of true label ϕ_i of i^{th} instance can be given as [100]:

$$\phi_i = \begin{cases} 1 & \textit{if } \mathbf{y}_i = \mathbf{y'}_i, \\ (k-2)/(k-1) & \textit{otherwise.} \end{cases} \tag{4.1}$$

Initially, we set the probability of true label $\phi_i = 1$ ($\forall i \in 1 \leq i \leq N$). Next, the loss $\mathcal{L}_i$ for each instance i is estimated to obtain mean loss $\bar{\mathcal{L}}$, where, $\bar{\mathcal{L}} = 1/N \sum_{i=1}^{N} \mathcal{L}_i$. Further, if $\bar{\mathcal{L}}$ is low and the variance of loss against all the instances, *i.e.,* Variance $(\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_N) \approx 0$ then the training dataset is free from noisy labels. Thus, we can use $\phi_i = 1$, where, $\forall i \in 1 \leq i \leq N$. However, if the variance is high then the dataset possess noisy labels. So, we set $\phi_i = (k-2)/(k-1)$ for instance i, where, $\mathcal{L}_i > \bar{\mathcal{L}}$.

The training on noisy labels degrades the performance due to the coercive memorizing capacity of neural networks [32, 101]. Thus, it is required to either suppress the concentration of noise or estimate true labels from training data. The noise suppression can be achieved by improving the loss functions of model [34] given information about noise concentration. We can also use matrix completion [102] to obtain a true label matrix ($\mathbf{Y}''$) from the noisy label matrix ($\mathbf{Y}'$). The true label matrix ($\mathbf{Y}''$) obtained from noisy label matrix ($\mathbf{Y}'$) has a lower rank in contrast matrix ($\mathbf{Y}$) in noise-free scenario, *i.e., rank*$(\mathbf{Y}'') <$ *rank*$(\mathbf{Y})$. This work computes a low-rank estimate of $\mathbf{Y}'$ to obtain true labels matrix $\mathbf{Y}''$.

4.3 LRNL approach

This section proposes an LRNL approach using deep learning models. The main objective of LRNL is to recognize locomotion modes in the presence of noisy labels with minimal accuracy compromise. In the LRNL approach, we first develop three deep learning-based models for handling different concentrations of noisy labels. Later, the approach builds an ensemble model by combining these models in a proper ratio. Fi-

nally, the ensemble model is used to recognize a locomotion mode for a given testing instance. Figure 4.2 illustrates the block diagram of the LRNL approach.

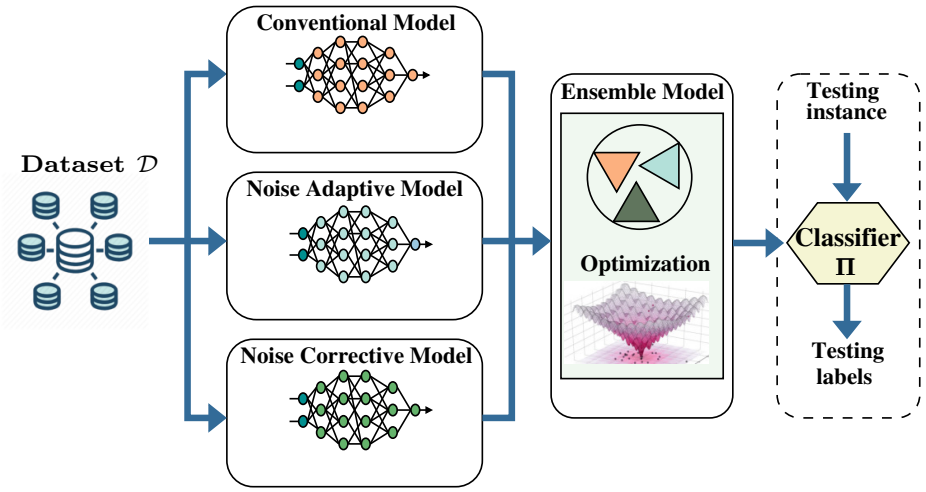


Figure 4.2: Block diagram of LRNL approach with its components.

4.3.1 Conventional model

First, this work constructs a conventional model using deep neural networks. The model extracts spatial and temporal features from raw sensory data using Convolution Neural Network (CNN) and Long Short Term Memory (LSTM), respectively. Spatial features refer to the relation among different sensory instances. The temporal features are defined as the time-dependent relation between data points in an instance. The conventional model extracts these features parallelly and then combines them to form an intermediate output. Later, the intermediate output is passed through the same configuration of CNN and LSTM to calculate the final features.

We use existing deep learning-based models (CNN and LSTM) for feature extraction in the conventional model. However, the constructed conventional model has a lightweight architecture that can run on a resource-constrained device like a smartphone. Due to the ability of CNN to mine hidden information in data and the ability of LSTM to process information in time series, we propose the conventional model based

on CNN and LSTM. The CNN part extracts features within each sensor modality, and the LSTM part extracts temporal dependencies. The conventional model also acts as a baseline for proposed noise adaptive and noise corrective models. These models train conventional model after performing matrix manipulations for handling noisy labels.

The proposed architecture possesses resemblance with the presented architecture in [103, 104], where, parallelly extracted features from CNN or Recurrent Neural Network (RNN) are either input to a CNN or RNN (*e.g.,* LSTM) to obtain final features for predicting labels. It indicates further refinement of the intermediate features is beneficial. We can use the sequential operations involving CNN and LSTM, but it increases the training time. So, we utilize the architectural configuration in the first stage to exploit the inherent parallelism. The proposed model is able to combine spatial and temporal features and ignores the correlations among them if stopped at the intermediate stage. In locomotion mode recognition, cross-features correlation into the deep learning framework can recognize more general patterns and improve performance.

Let S represents a set of sensors (accelerometer, gyroscope, magnetometer, *etc.*) used for data collection. Each sensor $s_j \in S$ has q_j axes, for $3-$axis accelerometer sensor $q_j = 3$. The dimension of the CNN input is $N \times M \times Q$, where, $Q = \sum_{j=1}^{S} q_j$. The input passes through 5 convolution layers and each of which contains 64 filters. The shorthand representation of CNN is: $C(64) - C(64) - C(64) - C(64) - C(64) - FC(64)$. $C(64)$ denotes a convolution layer with 64 filters and $FC(64)$ denotes a fully connected layer with 64 neurons. Each $C(64)$ performs $64 \times N \times M \times Q$ element-wise multiplications and $64 \times N \times M$ additions to generate an output of size $64 \times N \times M$. Finally, FC layer performs non-linear transformation.

In LSTM, the input instances of dataset $\mathcal{D}$ are reshaped in size $M \times Q$ and pass through two sequential layers of LSTM units. The output of the sequential layers is supplied as input to the FC layer. The shorthand representation of LSTM is: $LSTM(64) - LSTM(64) - FC(64)$, where, $LSTM(64)$ and $FC(64)$ denote LSTM

unit and fully connected layer with 64 neurons, respectively. The neurons in the LSTM unit are called memory cells consisting of input, output, and forget gates. Figure 4.3 illustrates an overview of the conventional model, where $\mathcal{D}$ is supplied as two parallel inputs to CNN and LSTM.

The features extracted from CNN and LSTM are of same size, *i.e.,* $N \times 64$. The conventional model combines these features at the concatenation layer, and the combined features are passed through $FC(64)$ to obtain an intermediate feature matrix of size $N \times 64$. The intermediate feature is passed through a similar configuration of CNN and LSTM to get a feature matrix $\mathbf{X}$. Later, the model built a classifier $\mathbf{\Pi_1}$.

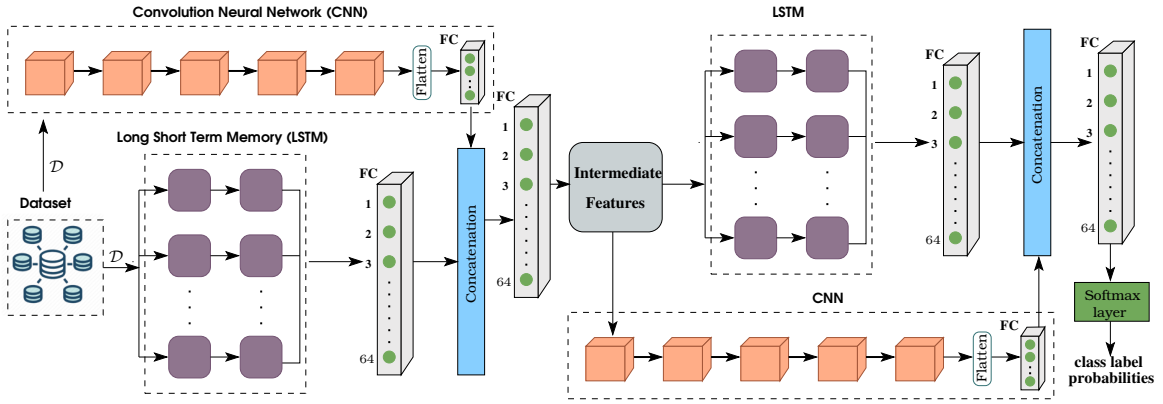


Figure 4.3: Overview of conventional model using CNN and LSTM.

4.3.2 Noise adaptive model

This section presents a noise adaptive model to alleviate the noisy labels by adopting a dynamic learning technique. The noise adaptive model formulates a loss called Noise Adaptive Loss (NAL). The conventional model is not robust towards noisy labels due to the coercive memorizing capacity of the deep learning models [105]. Loss function captures discrepancy between the true label matrix $\mathbf{Y}$ and predicted matrix $\mathbf{Y}_{tr}$. The value of discrepancy (loss) determines the update in the weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$ during backpropagation. The default loss functions used in various models are inefficient to handle the noisy labels. In addition, there exist some methods, *e.g.,* [36],

which modified the default loss function for handling noise, but their application is limited to image datasets.

Let $x_{ij} \in \mathbf{X}$, $w_{ij} \in \mathbf{W}^T$, and $b_j \in \mathbf{b}$ represent an element of feature matrix, weight matrix, and bias vector for j^{th} class of i^{th} training instance, respectively, for $1 \leq i \leq N$ and $1 \leq j \leq k$. We estimate an element z_{ij} as: $z_{ij} = w_{ij}x_{ij} + b_j$. Later, $\mathbf{z}_i = \{z_{ij} | 1 \leq j \leq k\}$ passes to a softmax function to compute predicted class label probability ρ_{ij}.

The predicted probability vector for i^{th} instance is a set of probabilities $\{\rho_{i1}, \ldots, \rho_{ik}\}$, against each class label k and a class label with highest probability value is said to be the predicted class label. The loss functions in the deep learning models consider only the class label with the highest predicted probability during training. This consideration is valid only when class labels are noise-free. The probabilities of other class labels (except highest) are nearly zero (or negligible) but can provide significant information in the case of noisy labels. Therefore, the noise adaptive model considers all k-class predicted probabilities by assigning the proper weight to each of them. These weights are estimated using the variables α and β. The NAL considers the following two terms:

1. Loss term to conquer true predictions:

$$\mathcal{L}_a = -\frac{1}{N \times k} \sum_{i=1}^{N} \sum_{j=1}^{k} \left[\alpha\beta(\phi_i \log \rho_{ij})\right], \tag{4.2}$$

   where, ϕ_i can be obtained using Equation 4.1.

2. Loss term to conquer false predictions:

$$\mathcal{L}_b = -\frac{1}{N \times k} \sum_{i=1}^{N} \sum_{j=1}^{k} [(1-\alpha)\rho_{ij} \log \phi_i + (1-\beta)(1-\phi_i) \log(1-\rho_{ij})]. \tag{4.3}$$

The loss term $\mathcal{L}_b$ (given in Equation 4.3) conquers false predictions. In other words, the loss term helps in assigning some weight to the wrongly predicted class labels. This weight assignment helps in modifying the simple cross-entropy loss using two dynamic

variables α and β, as given in Equation 4.4. This modified loss function increases the value of the loss when noise is present in the training data. Further, after sufficient epochs for training, the built classifier can recognize locomotion modes with higher accuracy. Combining Equations 4.2 and 4.3, we can obtain the noise adaptive loss as:

$$\mathcal{L}_{NAL} = -\frac{1}{N \times k} \sum_{i=1}^{N} \sum_{j=1}^{k} [\alpha\beta(\phi_i \log \rho_{ij}) + (1-\alpha)\rho_{ij} \log \phi_i + (1-\beta)(1-\phi_i) \log(1-\rho_{ij})]. \tag{4.4}$$

Equation 4.4 holds two terms, *i.e.,* $\phi_i \log \rho_{ij}$ and $\rho_{ij} \log \phi_i$. The term $\phi_i \log \rho_{ij}$ is the definition of cross-entropy loss that captures the difference between two probability distributions (*i.e.,* actual class probability and predicted class probability) for instance i. In other words, if we consider actual class labels distribution A and an approximation of actual class label distribution B, then the cross-entropy of B from A is the number of additional bits to represent an event using B instead of A. Similarly, the term $\rho_{ij} \log \phi_i$ captures the difference between predicted class probability and actual class probability for given data instance i. It also defines the number of additional bits to represent an event using A instead of B. While estimating $\phi_i \log \rho_{ij}$ we are having both A and B, thus, we can easily calculate $\rho_{ij} \log \phi_i$. The term $\rho_{ij} \log \phi_i$ also capture significant information while estimating loss in case of noisy labels in the training data. Further, NAL assigns dynamic weights α and β to *loss term to conquer true predictions* and *loss term to conquer false predictions* to specify their contributions. Additionally, if α is assigned to true class prediction then for false prediction $(1-\alpha)$ is assigned to make the sum of losses to unity.

We initialize the dynamic variables in NAL as $\alpha = 1$ and $\beta = 1$. This initialization converts NAL into cross-entropy loss. During the training, α and β decrease asynchronously if training data possess noisy labels. We estimate $\mathcal{L}_{NAL}$ for all possible

α and β in each training epoch. Later, the minimum $\mathcal{L}_{NAL}$ and corresponding α and β are selected after each epoch. It is continued for maximum iterations to obtain a trained model with minimum $\mathcal{L}_{NAL}$ and optimal α and β. Algorithm 4.1 illustrates all the steps for estimating the dynamic variables α and β.

Algorithm 4.1: Estimation of dynamic variables α and β.

Input: $\mathbf{X}_{tr}$, noisy label matrix $\mathbf{Y}'$ and weight matrix $\mathbf{W}$;
Output: Optimal value of parameters α and β;
Initialize $max\_iter = 200$, $\alpha = \beta = 1$, $r = 1 \times e^{-2}$.
for $i \leftarrow 1$ *to* N **do**
  for $j \leftarrow 1$ *to* k **do**
    Calculate ρ_{ij}, append ρ_{ij} in $\boldsymbol{\rho}_i$.
  Calculate ϕ_i.
Calculate $\mathcal{L}_{NAL}$ at $\alpha = \beta = 1$.
while *epoch* $\leq$ *max_iter* **do**
  $\alpha' = \alpha$, $\beta' = \beta$.
  $\alpha_1 = \alpha - r, \beta_1 = \beta - r$.
  $\alpha_2 = \alpha, \beta_2 = \beta - r$.
  $\alpha_3 = \alpha - r, \beta_3 = \beta$.
for $i \leftarrow 1$ *to* 3 **do**
  $\mathcal{L}_{NAL_i} = checker(\alpha_i, \beta_i, \mathcal{L}_{NAL})$.
$i \leftarrow \text{argmin}([\mathcal{L}_{NAL_1}, \mathcal{L}_{NAL_2}, \mathcal{L}_{NAL_3}])$.
$\alpha = \alpha_{i+1}$, $\beta = \beta_{i+1}$.
if $(\alpha' == \alpha$ ***and*** $\beta' == \beta)$ ***or*** $(\mathcal{L}_{NAL_i} == 0)$ **then**
  break.
$\mathcal{L}_{NAL} = \mathcal{L}_{NAL_i}$, $epoch = epoch + 1$.
return α, β.
Function $checker(\alpha, \beta, \mathcal{L}_{NAL})$
  begin
    $\mathcal{L} = \mathcal{L}_{NAL}$.
    Calculate $\mathcal{L}_{NAL}$ at α and β.
    if $\mathcal{L}_{NAL} \leq \mathcal{L}$ **return** $\mathcal{L}_{NAL}$.
    else return $\mathcal{L}$.
  end

4.3.3 Noise corrective model

This section presents a noise corrective model to handle a high order concentration of noisy labels in training data. The noise corrective model estimates the true label matrix

$\mathbf{Y}$ from given feature matrix $\mathbf{X}$ and noisy label matrix $\mathbf{Y}'$. The model handles noisy labels by adopting a concept discussed in [102]. Let $\mathbf{T}$ be a projection matrix of size $M \times k$ on feature matrix $\mathbf{X}$ and $\mathbf{W}$ denotes a sparse matrix of size $N \times k$ that captures information about the noisy labels. Let $\mathbf{T}^*$ and $\mathbf{W}^*$ be the optimal value of $\mathbf{T}$ and $\mathbf{W}$, respectively. Now, the projection matrix $\mathbf{T}$ can be estimated using the two terms:

1. Loss term:
$$\sum_{i=1}^{N}\sum_{j=1}^{k} L((\mathbf{XT}(\mathbf{Y}')^T)_{ij} + \mathbf{W})_{ij}, \mathbf{Y}_{ij}). \tag{4.5}$$

2. Regularization term to avoid overfitting:

$$\lambda_T||\mathbf{T}||_* + \lambda_W||\mathbf{W}||_*, \tag{4.6}$$

   where, λ_T and λ_W are trade-off parameters, and $||\cdot||_*$ is the trace norm of matrix. The value of parameter $\lambda_T = k^{-\frac{1}{2}}$ and $\lambda_W = k^{-\frac{1}{2}}$, $\lambda_T > 0$ and $\lambda_W > 0$.

The projection matrix $\mathbf{T}$ can obtain by minimizing the summation of loss and regularization terms. Noisy label matrix $\mathbf{Y}'$ can be mathematically given as follows:

$$\mathbf{Y}' = \mathbf{XT} + \mathbf{W}. \tag{4.7}$$

By aggregating Equations 4.5, 4.6, and 4.7, we define an optimization problem resembling rank minimization [106].

$$\begin{aligned} &\min_{\mathbf{T},\mathbf{W}} \ rank(\mathbf{T}) + \lambda_T||\mathbf{T}||_F^2 + \lambda_W||\mathbf{W}||_F^2, \\ &\text{s.t. } \mathbf{Y}' = \mathbf{XT} + \mathbf{W}, \ \mathbf{XT} \in \{0,1\}^{N\times k}, \end{aligned} \tag{4.8}$$

where, $||\cdot||_F$ is Frobenius norm that captures the distribution of noisy labels in $\mathbf{Y}'$.

The matrices $\mathbf{T}^*$ and $\mathbf{W}^*$ can be obtained by solving the optimization problem in Equation 4.8. The solution to the optimization problem is difficult due to non-

continuous nature of rank function. We therefore replace the *rank* function with trace norm (*i.e.,* $rank(\mathbf{T}) \approx ||\mathbf{T}||_*$), to convert Equation 4.8 into a convex optimization problem. Let $\mathbf{E}, \mathbf{F}, \mathbf{G},$ and $\mathbf{H}$ are the matrices of size $a \times b, a \times c, c \times b,$ and $a \times b$, respectively. From [107], an equation of form $\mathbf{E} = \mathbf{FG} + \mathbf{H}$ is said to have unique solution $\mathbf{G}^*$ with $\mathbf{F}^\dagger$ (pseudo-inverse of $\mathbf{F}$), if and only if a condition $\mathbf{F} \neq 0$ holds. Thus, the obtained solution is of form: $\mathbf{G}^* = \mathbf{F}^\dagger(\mathbf{E} - \mathbf{H})$.

As the feature matrix can not be zero (*i.e.,* $\mathbf{X} \neq 0$), Equation 4.8 should have a unique solution. In this work, we adopted a widely used technique called Alternating Direction Method of Multipliers (ADMM) [108] for solving the optimization problem (given in Equation 4.8). ADMM introduces an auxiliary matrix $\mathbf{A}$ in the optimization problem to ensure a close form solution. Now, the optimization problem is:

$$\begin{aligned} &\min_{\mathbf{T},\mathbf{W},\mathbf{A}} ||\mathbf{T}||_* + \lambda_T ||\mathbf{T}||_F^2 + \lambda_W ||\mathbf{W}||_F^2, \\ &\text{s.t. } \mathbf{Y}' = \mathbf{A} + \mathbf{W},\ \mathbf{XT} = \mathbf{A},\ \mathbf{A} \in [0,1]^{N \times k}. \end{aligned} \tag{4.9}$$

Augmented Lagrangian form of Equation 4.9 with multipliers, L_1 and L_2, is:

$$\begin{aligned} \mathcal{L} = &||T||_* + \lambda_T ||\mathbf{T}||_F^2 + \lambda_W ||\mathbf{W}||_F^2 + tr(L_1^T(\mathbf{Y}' - \mathbf{A} - \mathbf{W})) \\ &+ tr(L_2^T(\mathbf{XT} - \mathbf{A})) + \frac{\mu}{2}(||\mathbf{Y}' - \mathbf{A} - \mathbf{W}||_F^2 + ||\mathbf{XT} - \mathbf{A}||_F^2), \end{aligned} \tag{4.10}$$

where, $\mu(> 0)$ is a penalty coefficient and $tr(\cdot)$ is trace of a given matrix. Further, we estimate the value of $\mathbf{T}, \mathbf{W},$ and $\mathbf{A}$ as follows:

Estimating $\mathbf{T}$*:* To estimate $\mathbf{T}$, we set $\mathbf{T}$ as a variable with fixed values of $\mathbf{W}$ and $\mathbf{A}$. Next, the projection matrix $\mathbf{T}$ is obtained by solving the following problem:

$$\min_{\mathbf{T}} \mathcal{L}_T = ||\mathbf{T}||_* + \lambda_T ||\mathbf{T}||_F^2 + tr(L_2^T(\mathbf{XT} - \mathbf{A})) + \frac{\mu}{2}||\mathbf{XT} - \mathbf{A}||_F^2. \tag{4.11}$$

For any matrix $\mathbf{B}$, Frobenius norm is defined as $||\mathbf{B}||_F^2 = tr(\mathbf{B}^T.\mathbf{B})$ and nuclear norm

is defined as $||\mathbf{B}||_* = tr(\sqrt{\mathbf{B}^T\mathbf{B}})$. By using these properties, Equation 4.11 is given as:

$$\mathcal{L}_T = tr(\sqrt{\mathbf{T}^T\mathbf{T}}) + \lambda_T tr(\mathbf{T}^T.\mathbf{T}) + tr(L_2^T(\mathbf{XT} - \mathbf{A})) - \frac{\mu}{2}tr(\mathbf{A}^T(\mathbf{XT})). \tag{4.12}$$

Equation 4.12 can be solved by taking derivative *w.r.t.* $\mathbf{T}$ and equating to zero as:

$$\sqrt{\mathbf{T}^T\mathbf{T}}(\mathbf{T}^T + \mathbf{T}) + \lambda_T(\mathbf{T}^T + \mathbf{T}) + L_2\mathbf{X} - \mu(\mathbf{A}^T\mathbf{X}) = 0.$$

$$\implies \mathbf{T} = \frac{1}{2}\left(\sqrt{\lambda_T^2 + 2(\mu\mathbf{A}^T - L_2\mathbf{X})} - \lambda_T\right). \tag{4.13}$$

Estimating $\mathbf{W}$*:* It can be obtained by setting fixed values of $\mathbf{T}$ and $\mathbf{A}$ with $\mathbf{W}$ as a variable. To obtain sparse matrix $\mathbf{W}$, the minimization problem is:

$$\min_{\mathbf{W}} \mathcal{L}_W = \lambda_W||\mathbf{W}||_F^2 + tr(L_1^T(\mathbf{Y}' - \mathbf{A} - \mathbf{W})) + \frac{\mu}{2}tr((\mathbf{Y}' - \mathbf{A} - \mathbf{W})^T(\mathbf{Y}' - \mathbf{A} - \mathbf{W})). \tag{4.14}$$

Upon solving Equation 4.14, we get sparse matrix $\mathbf{W}$ as:

$$\mathbf{W} = \frac{L_1^T - \mu(\mathbf{A} - \mathbf{Y}')}{2(\lambda_W + \frac{\mu}{2})}. \tag{4.15}$$

Estimating $\mathbf{A}$*:* Similar to the estimation of $\mathbf{T}$ and $\mathbf{W}$, here, we set $\mathbf{A}$ as a variable with the fixed value of $\mathbf{T}$ and $\mathbf{W}$. Thus, the auxiliary matrix $\mathbf{A}$ can be obtained by solving Equation 4.10 with respect $\mathbf{A}$, as given below:

$$\min_{\mathbf{A}} \mathcal{L}_A = tr(L_1^T(\mathbf{Y}' - \mathbf{A} - \mathbf{W})) + tr(L_2^T(\mathbf{XT} - \mathbf{A})) + \frac{\mu}{2}(||\mathbf{Y}' - \mathbf{A} - \mathbf{W}||_F^2 + ||\mathbf{XT} - \mathbf{T}||_F^2). \tag{4.16}$$

By differentiating Equation 4.16 with respect to $\mathbf{A}$, we get:

$$\mathbf{A} = \frac{L_1^T + L_2^T + \mu((\mathbf{Y}')^T + (\mathbf{XT})^T - \mathbf{W}^T))}{2\mu}. \tag{4.17}$$

Using estimated matrices $\mathbf{T}$, $\mathbf{W}$, and $\mathbf{A}$ from Equations 4.13, 4.15, and 4.17, respectively, we can solve the optimization problem in Equation 4.10. Algorithm 4.2 summarizes the steps involved in estimation of true class labels from given matrices $\mathbf{X}$ and $\mathbf{Y}'$. Noise corrective model that takes sensory data $\mathbf{X}_{tr}$ and noisy label matrix $\mathbf{Y}'$ as input and then builds a classifier ($\mathbf{\Pi}_3$).

Algorithm 4.2: Estimation of true class labels.

Input: Feature matrix $\mathbf{X}$ and noisy label matrix $\mathbf{Y}'$;
Output: True label matrix $\mathbf{Y}$;

Initialize $\mathbf{T}, \mathbf{W},$ and $\mathbf{A}$ with zero matrix $\mathbf{0}$.
$\mu = 10^{-6}, \mu_{max} = 10^6,$ and $\epsilon = 1 \times e^{-7}$.
while $iter \leq max\_iteration$ **do**
    Calculate $\mathbf{T}_{iter}$, $\mathbf{W}_{iter}$, and $\mathbf{A}_{iter}$ (Equations 4.13, 4.15, and 4.17).
    if $(||\mathbf{Y}' - \mathbf{A} - \mathbf{W}|| < \epsilon$ ***and*** $||\mathbf{XT} - \mathbf{A}|| < \epsilon)$ **then**
        $\mathbf{T} = \mathbf{T}_{iter}$, $\mathbf{W} = \mathbf{W}_{iter}$, $\mathbf{A} = \mathbf{A}_{iter}$.
        $L_1 := L_1 + \mu(\mathbf{Y}' - \mathbf{A} - \mathbf{W})$.
        $L_2 := L_2 + \mu(\mathbf{XT} - \mathbf{A})$.
    $iter = iter + 1$.
Compute $\mathbf{Y} = \mathbf{XT}$ ($i.e., \mathbf{Y}' - \mathbf{W}$).
return $\mathbf{Y}$

4.3.4 Ensemble model

The section presents an ensemble model that combines three classifiers ($\mathbf{\Pi_1}, \mathbf{\Pi_2},$ and $\mathbf{\Pi_3}$) discussed in previous sections. It is impractical to determine whether the class label against a sensory instance is noisy or not? Therefore, selecting an appropriate model for noise handling is also tedious. Ensemble model ensures to predict the best possible outcome by iteratively verifying the prediction probabilities for all the three classifiers. The classifiers $\mathbf{\Pi_1}, \mathbf{\Pi_2},$ and $\mathbf{\Pi_3}$ generate prediction matrices $\mathbf{Y}^1_{te}, \mathbf{Y}^2_{te},$ and $\mathbf{Y}^3_{te}$ against the same input $\mathbf{X}_{te}$. The ensemble model multiplies a fraction weight (λ_i) with each prediction matrix $\mathbf{Y}^i_{te}$ to calculate a matrix $\mathbf{Y}_{te}$ where, $\sum_{i=1}^{3} \lambda_i = 1$. The optimal value of the fraction weights are estimated using differential evolution technique [109, 110]. The differential evolution solves following problem:

$$\begin{aligned} &\min \mathcal{L}(\lambda_1 \mathbf{Y}^1_{te}, \lambda_2 \mathbf{Y}^2_{te}, \lambda_3 \mathbf{Y}^3_{te}), \\ &s.t., \ \lambda_1 + \lambda_2 + \lambda_3 = 1, \end{aligned} \tag{4.18}$$

where, $\mathcal{L}(\cdot)$ is a loss function that captures the discrepancy between true and predicted labels. Initially, the differential evolution technique sets the value of fractional weights to $\frac{1}{3}$ (or 0.3333), which generates a prediction matrix $\mathbf{Y}_{te}$ as $\mathbf{Y}_{te} = \sum_{i=1}^{3} \lambda_i \mathbf{Y}^i_{te} = \frac{1}{3}(\mathbf{Y}^1_{te} + \mathbf{Y}^2_{te} + \mathbf{Y}^3_{te})$, where, λ_1^*, λ_2^* and λ_3^* are the optimal fractional weights and $\lambda_1^* + \lambda_2^* + \lambda_3^* = 1$. The fractional weights assigned to the predictions of each model helps in selecting the preferable class label. For example, let **Model 1** achieves an accuracy of 92.5%. However, it may lag in predicting class labels of some instances. These instances may be correctly predicted by a **Model 2** achieving an accuracy of 86%. However, determining which instance is correctly predicted by which model is tedious; therefore, the ensemble techniques is beneficial.

Algorithm 4.3 illustrates all the steps involve in the locomotion mode recognition with noisy labels. Further, the optimal value of the weight parameters and individual

parameters associated with each model helps in predicting the correct class label even when we do not know the noise concentration.

Algorithm 4.3: LRNL approach.

Input: Training data $\mathbf{X}_{tr}$ with noisy labels $\mathbf{Y}'$;
Output: Prediction matrix $\mathbf{Y}_{te}$ against testing data $\mathbf{X}_{te}$;
for $i \leftarrow 1$ *to* 3 **do**
  $\mathbf{X}_i \leftarrow$ *feature_extractors*$(\mathbf{X}_{tr})$.
  $\mathbf{X}_i \leftarrow$ *feature_extractors*$(\mathbf{X_i})$.
  if $i = 1$ **then**
    Compute $\mathbf{Y}^i_{tr}$ using softmax with $\mathbf{X}_i$ and $\mathbf{Y}'$.
  if $i = 2$ **then**
    Estimating loss by $\mathcal{L}_{NAL}$ using Equation 4.4.
    Compute $\mathbf{Y}^i_{tr}$ using softmax with $\mathbf{X}_i$ and $\mathbf{Y}'$.
  else
    Estimate true label matrix $\mathbf{Y}$ using Algorithm 4.2.
    Compute $\mathbf{Y}^i_{tr}$ using softmax with $\mathbf{X}_i$ and $\mathbf{Y}$.
  Build a classifier $\mathbf{\Pi_i}$, $\mathbf{Y}^i_{te} \leftarrow \mathbf{\Pi_i}(\mathbf{X}_{te})$.
$\lambda^*_1, \lambda^*_2, \lambda^*_3 \leftarrow$ *differential_evolution*$(\mathbf{Y}^1_{te}, \mathbf{Y}^2_{te}, \mathbf{Y}^3_{te})$.
return $\mathbf{Y}^*_{te}$.
Function *feature_extractors*$(\mathbf{X}_{tr})$
  begin
  Pass $\mathbf{X}_{tr}$ through a sequence of 5 convolution layers.
  Pass $\mathbf{X}_{tr}$ through a sequence of 2 LSTM layers.
  Pass the output of 2^{nd} LSTM layer through FC layer.
  Pass concatenated output through FC layer to get $\mathbf{X}$.
  return X.
  end

Time complexity of LRNL approach: Algorithm 4.3 takes $O(Nk+q)$ time for N input instances and q iterations. As $\{k, c\} \ll N$, the time complexity of Algorithm 4.1 becomes $O(N)$. Similarly, the complexity of Algorithm 4.2 is $O(1)$ as it depends on the iterations. Next, the time complexity of *feature_extractor()* function depends on the computations involved in CNN and LSTM models. The CNN model takes $O\big(\sum_{i=1}^{l} c_{i-1} \cdot s_i^2 \cdot u_i \cdot v_i^2\big)$ per time step of the input sequence [93], where, l is the number of convolution layers, c is the number of input channels, u is number of filters, s is the size of a filter, and v is the size of output feature vector, at i^{th} convolution layer. Further, complexity of

LSTM model per time step can be given as $O(W)$ [94], where, W is the number of weight parameters that are learned during training. Time complexity of *feature_extractor()* can be given as $O((cs^2uv^2 + W)M) = O(WM)$. Next, the time complexity of *differential_evolution()* function is $O(NLR_{max})$ [109], where, L is the number of variables taken for optimization and R_{max} is the maximum number of iterations. Now, the time complexity of LRNL approach can be given as $O(N + WM + NLR_{max})$.

4.4 Experimental evaluation of LRNL approach

This section first discusses the data collection setup to collect the Locomotion Mode Recognition (LMR) dataset. It next covers the evaluation of the LRNL approach on the LMR dataset with Sussex-Huawei Locomotion (SHL) [55] and Transportation Mode Detection (TMD) [56] datasets.

4.4.1 Data collection

In this work, sensory data of six locomotion modes including bicycle ($\mathbf{a_1}$), auto-rickshaw ($\mathbf{a_2}$), bike ($\mathbf{a_3}$), car ($\mathbf{a_4}$), bus ($\mathbf{a_5}$), and train ($\mathbf{a_6}$) were collected using the sensors of the smartphone, as discussed in Chapter 3.

4.4.2 Existing dataset

This work also evaluates LRNL approach on two existing datasets: SHL [55] and TMD [56]. The sampling rate during SHL data collection was 100 Hz. The dataset consists the data of eight locomotion modes ($k = 8$) including still ($\mathbf{b_1}$), walk ($\mathbf{b_2}$), run ($\mathbf{b_3}$), bike ($\mathbf{b_4}$), car ($\mathbf{b_5}$), bus ($\mathbf{b_6}$), train ($\mathbf{b_7}$), and subway ($\mathbf{b_8}$). The dataset contains 2294, 2195, 688, 2093, 2480, 2089, 2513, and 1958 instances of classes $\mathbf{b_1}$, $\mathbf{b_2}$, $\mathbf{b_3}$, $\mathbf{b_4}$, $\mathbf{b_5}$, $\mathbf{b_6}$, $\mathbf{b_7}$, and $\mathbf{b_8}$, respectively, detailed in Chapter 3. We first preprocess the dataset to reduce the length of instances from 6000 to 300 samples ($M = 300$) by taking a mean value of non-overlapping windows of size 20.

Next, TMD dataset was collected for detecting the locomotion modes including still ($\mathbf{c_1}$), walk ($\mathbf{c_2}$), car ($\mathbf{c_3}$), bus ($\mathbf{c_4}$), and train ($\mathbf{c_5}$). The classes $\mathbf{c_1}$, $\mathbf{c_2}$, $\mathbf{c_3}$, $\mathbf{c_4}$, and $\mathbf{c_5}$ hold 967, 930, 891, 744, and 193 instances, respectively. In the provided dataset, the sensory data is recorded at 20 Hz (20 data samples per second) by using different smartphone sensors. We use the data of accelerometer, magnetometer, gyroscope, and orientation sensors. The length of each instance (M) is 1200, which is reduced to 240 by taking a mean value of non-overlapping windows of size 5.

4.4.3 Parameter settings during implementation

This section discusses the settings of various parameters used in implementing the LRNL approach. We have implemented the ensemble model in Python language using the functional API of Keras. For extracting the features from raw sensory data, we have used CNN and LSTM models, as discussed in Section 4.3.1. This work considers the random distribution of noisy labels in the training dataset as obtaining the uniform distribution for different classes is stringent in the real-world scenario. Next, to perform the random selection of training and testing sub-datasets, we have incorporated the function *sklearn.model_selection.train_test_split*().

Further, we compared proposed conventional model with two widely adopted feature extraction models, *i.e.,* DeepFusion [104], Deepsense [103]. The experimental results indicate that the proposed conventional model achieved higher accuracy and F_1-score within the given time frame than DeepFusion and Deepsense while consuming limited computational resources. The $<$ parameters (for single instance), FLOPs (for single instance), training-time (in minutes) $>$ of DeepFusion, Deepsense, proposed conventional model on LMR dataset is given as follows: DeepFusion $< 5.4 \times 10^8, 8.5 \times 10^{11}, 92 \pm 2 >$, Deepsense $< 3.1 \times 10^7, 7.6 \times 10^{10}, 73 \pm 2 >$, and proposed conventional $< 1.9 \times 10^7, 4.3 \times 10^{10}, 67 \pm 2 >$. It indicates that the proposed conventional model requires fewer parameters and FLOPs in contrast with DeepFusion and Deepsense. The computational

complexity of a model directly depends upon the required FLOPs [111].

4.4.4 Experimental results

This work carried out several experiments to answer following questions:

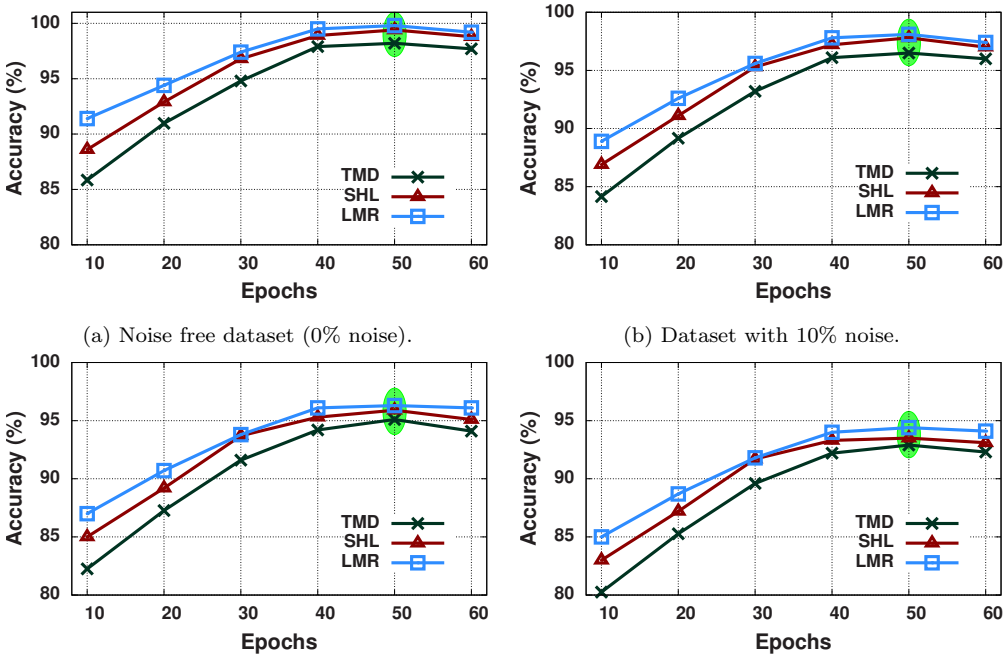
- What is the appropriate number of epochs to achieve stabilized performance? (Section 4.4.4.1)
- What is the impact of noise on fractional weights in the ensemble model? (Section 4.4.4.2)
- What is the impact of different noise concentration? (Section 4.4.4.3)
- What is the class-wise accuracy on different datasets? (Section 4.4.4.4)
- How do different models of LRNL approach perform on accuracy? (Section 4.4.4.5)

4.4.4.1 Number of epochs for evaluation

At first, this work carried out experiments on three datasets, namely LMR, SHL, and TMD, to identify a suitable number of epochs for evaluating the LRNL approach. In the experiments, the performance of LRNL is reported on the training data during the construction of the classifier. Figure 4.4 illustrates the training accuracy results on LMR, SHL, and TMD datasets using a different number of epochs and noise concentrations. We observe a rapid increment in the accuracy up to 40 epochs and a marginal increment afterwards. It is also observed that the ensemble model achieved maximum accuracy on 50 epochs. The accuracy difference between datasets with 0% and 30% noisy labels is around 6%, as shown Figure 4.4(a) and Figure 4.4(b).

4.4.4.2 Impact of noise on fractional weights

This work also conducted an experiment to find the impact of noise concentrations on fractional weights: λ_1 (conventional model), λ_2 (noise adaptive model), and λ_3 (noise corrective model) at 50 epochs. The experiments are conducted on all three datasets at



(a) Noise free dataset (0% noise).

(b) Dataset with 10% noise.

(c) Dataset with 20% noise.

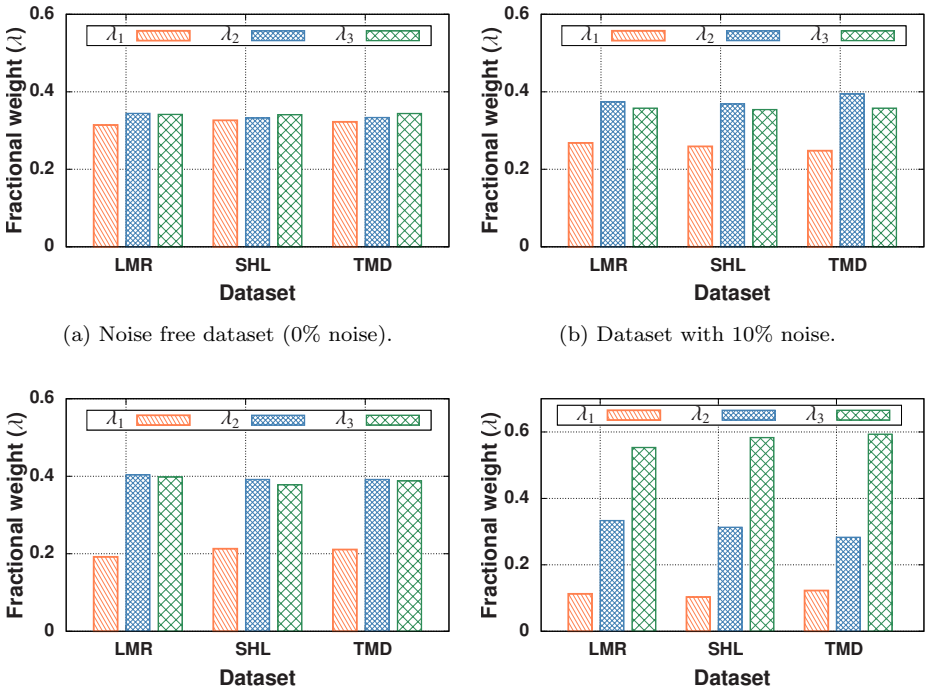
(d) Dataset with 30% noise.

Figure 4.4: Performance results on LMR, SHL, and TMD datasets during training with different noise concentrations (*i.e.,* 0%, 10%, 20%, and 30%).

0%, 10%, 20% and 30% noise concentrations. The optimal value of the fraction weights are estimated using the differential evolution technique at different concentrations of noisy labels. Figure 4.5 illustrates the impact of noise concentrations on fractional weights (λ_1, λ_2 and λ_3) using LMR, SHL, and TMD datasets. Figure 4.5(a) shows that the value of λ_1, λ_2, and λ_3 are nearly equal which indicates that all three models perform well on noise free data. Figure 4.5(b), Figure 4.5(c), and Figure 4.5(d) illustrate that with increment in noise concentration the contribution of conventional model (λ_1) decreases whereas it increases for noise corrective model (λ_3). If $\lambda_1 \approx \lambda_2 \approx \lambda_3$ then concentration of noisy labels is low, if $\lambda_1 < \lambda_2 \approx \lambda_3$ then concentration of noisy labels is medium, and if $\lambda_1 < \lambda_2 < \lambda_3$ then the concentration of noisy labels is high.

4.4.4.3 Performance of LRNL approach

Next, the LRNL approach is evaluated at varying noise concentrations from 0% to 40% at 50 epochs. Table 4.1 illustrates the performance results using accuracy and F1-score.



(a) Noise free dataset (0% noise).

(b) Dataset with 10% noise.

(c) Dataset with 20% noise.

(d) Dataset with 30% noise.

Figure 4.5: Value of fractional weights on LMR, SHL, and TMD datasets during training with different noise concentrations (*i.e.,* 0%, 10%, 20%, and 30%).

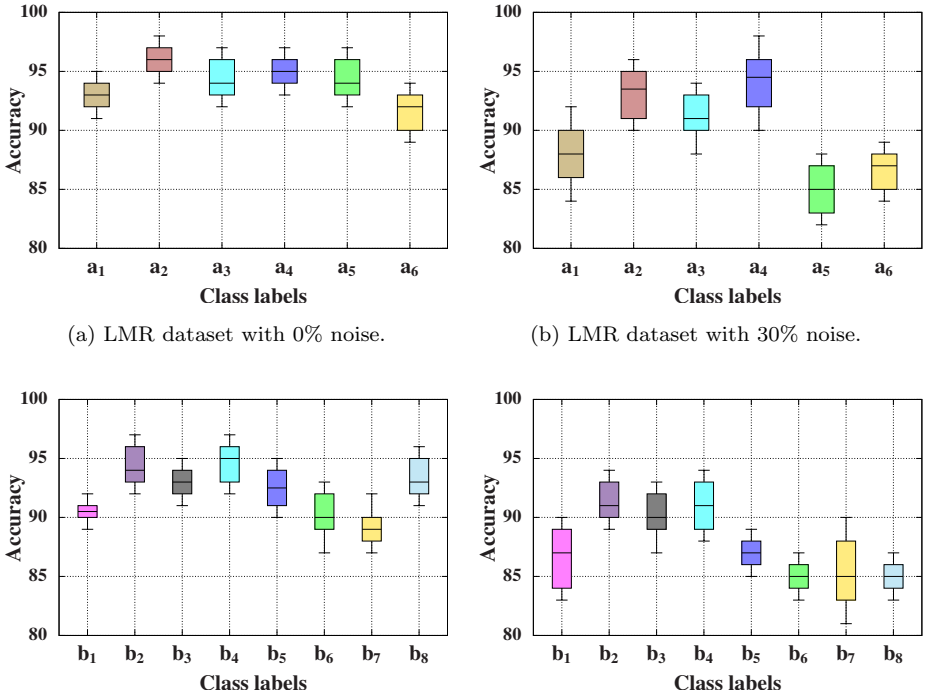
We use the micro-average F1-score, where, the contributions of classes are determined via their available instances. The reported results illustrate that the LRNL approach can handle noise up to 30% with a marginal compromise of accuracy (from 3% to 5%) for all the datasets. When the noise concentration reaches 40%, the accuracy drops suddenly by more than 18%. The sudden drop in the accuracy indicates that the data of certain classes have nearly all their labels incorrect at 40% noise concentration.

Table 4.1: Performance results of LRNL approach on LMR, SHL, and TMD datasets at different noise concentrations.

Dataset	**Metric**	**Concentration of noisy labels**					
		0%	10%	20%	30%	40%	50%
LMR	Accuracy	94.21	92.23	90.55	89.76	72.25	63.17
	F_1-score	94.12	92.41	90.96	89.17	72.42	62.73
SHL	Acc	92.53	91.56	90.19	88.13	71.32	61.44
	F_1-score	92.61	91.73	90.85	88.22	71.79	60.73
TMD	Acc	91.24	88.07	87.57	86.19	69.42	59.41
	F_1-score	91.61	88.23	87.74	86.60	69.64	59.07

4.4.4.4 Class-wise accuracy

This work also evaluates the class-wise accuracy of the LRNL approach using LMR and SHL datasets, and the obtained results at 50 epochs are illustrated in Figure 4.6. Figure 4.6(a) and Figure 4.6(b) illustrate that LRNL approach achieves maximum class-wise accuracy for $\mathbf{a_5}$ class at 0% noise and $\mathbf{a_4}$ class (car) at 30% noise concentration. It is due to the availability of better identifiable patterns in the sensory measurements of class $\mathbf{a_4}$. We observe that the accuracy of $\mathbf{a_5}$ class (bus) is highly affected with the increase in the noise concentration (*i.e.,* from 95.30 to 84.90). It is due to the presence of a large number of noisy labels for $\mathbf{a_5}$ at 30% noise. Similar results are observed for the SHL dataset, as illustrated in Figure 4.6(c) and Figure 4.6(d).



(a) LMR dataset with 0% noise.

(b) LMR dataset with 30% noise.

(c) SHL dataset with 0% noise.

(d) SHL dataset with 30% noise.

Figure 4.6: Class-wise accuracy of LRNL approach on LMR and SHL datasets with different noise concentrations.

4.4.4.5 Performance analysis of different models in LRNL approach

Table 4.2 illustrates the accuracy of all models with different noise concentrations using LMR, SHL, and TMD datasets. The noise corrective model shows a marginal decrement in the accuracy, whereas the conventional model shows a substantial decrement due to the absence of a noise handling mechanism. The noise adaptive model can handle moderate noise up to 20%, indicating a minimal accuracy drop, up to 20% of noise concentration. Table 4.2 also illustrates that the concatenation of three models provides better accuracy.

Moreover, at lower concentrations, the contributions of models are approximately the same. Thus, for all instances, the performance of models are comparable and deterministic. Thus, the addition of the conventional model at lower concentrations results in marginal improvement. Further, at higher concentrations of noise levels, the uncertainty in the prediction of conventional model is high, which results in its performance deterioration. However, for some instances, performance of the conventional model outperforms NA and NC.

Table 4.2: Performance (in %) of conventional (Conv), noise adaptive (NA), and noise corrective (NC) models on different datasets with variable noise concentrations. Acc=Accuracy and F_1= F_1-score.

Dataset	**Models**	**Concentration of noisy labels**					
		10%		20%		30%	
		Acc	F_1	Acc	F_1	Acc	F_1
LMR	Conv	82.37	82.86	77.93	78.34	72.95	73.31
	NA	86.92	87.14	83.13	83.47	77.91	78.22
	NC	90.19	89.29	88.71	88.92	86.95	87.17
	NA+NC	91.53	91.83	89.22	89.71	88.05	88.48
	LRNL	92.23	92.41	90.55	90.96	89.76	90.17
SHL	Conv	80.18	80.67	77.30	77.74	71.10	71.67
	NA	85.57	85.92	80.65	81.12	76.81	77.41
	NC	88.39	88.74	86.48	86.82	85.70	86.21
	NA+NC	90.89	90.34	89.42	89.87	87.19	87.41
	LRNL	91.56	91.73	90.19	90.85	88.13	88.22
TMD	Conv	78.67	79.22	75.93	75.81	69.27	69.22
	NA	83.23	83.71	79.78	80.37	73.11	73.67
	NC	86.38	86.69	84.26	84.72	82.10	82.44
	NA+NC	87.41	87.76	85.82	86.23	84.35	84.73
	LRNL	88.07	88.23	87.57	87.74	86.19	86.60

4.4.4.6 Random versus uniform distribution of noisy labels

In the random distribution, the noisy labels in the training dataset are randomly dispersed means we cannot figure out the percentage of noisy labels for a particular class. Further, the uniform distribution of the noisy labels signifies that an equal proportion of noisy labels disperse in the training dataset for all class labels. Figure 4.7 shows that the confusion matrices for the LMR dataset with random and uniform distribution of noisy labels at 30% noise concentration. It is interesting to observe that the uniform distribution of noisy labels improves the prediction accuracy marginally, *i.e.,* 1.2%.

• **Adding Noisy labels:** In the random distribution of noisy labels, we use "pandas.DataFrame. sample(frac=0.x)" to randomly pick x% training data instances (S) with its labels. Next, flip the class labels of selected S with the label other than the true class labels. It generates a set of data instances S' with all its labels are noisy. Finally, we replace all the class labels corresponding to the data instances of S in training dataset with the class labels of S'. Similarly, in the uniform distributions of noisy labels, we pick x% training data instances of all the class labels in S and perform the noisy labels insertion uniformly.

(a) Random distribution of noisy labels. (b) Uniform distribution of noisy labels.

Figure 4.7: Impact of noisy labels distribution on class-wise accuracy with 30% noise concentration.

4.4.4.7 Impact of window size

Finally, we perform an experiment to study the impact of window size on the accuracy of the LRNL approach on LMR and SHL datasets. From the results, we observed the highest accuracy is achieved at window size 20, as shown in Table 4.3. It is because at window size 20, most distinguishable characteristics are obtained to learn a mapping between features and labels. However, if the window size is more than 20, the features overlap, and if the window size is less than 20, there are only a few distinguishable features for training. Similarly, while performing the experimental evaluation on the TMD dataset having sample length 1200, we select a window size of 5. It indicates that the window size is specific and changes with the datasets.

Table 4.3: Accuracy (in %) variations of the proposed approach on collected LMR and SHL datasets with different window sizes.

Window size	**LMR dataset**			**SHL dataset**		
	Noise concentration			*Noise concentration*		
	0%	20%	30%	0%	20%	30%
5	73.45	69.23	63.27	68.63	60.98	59.23
10	83.20	75.47	69.49	79.42	71.96	63.38
15	89.23	84.29	78.66	86.23	80.33	72.97
20	**94.21**	**90.55**	**89.76**	**92.53**	**90.19**	**88.13**
25	92.39	88.75	86.45	90.17	86.32	84.85
30	90.45	86.23	85.29	89.22	85.73	84.21
35	90.2	85.17	84.79	89.03	84.34	83.91
40	89.7	84.86	84.21	88.25	83.78	82.71

4.4.5 Comparison with existing approaches

This section compares LRNL approach with existing approaches, including DenseNetX and GRU (DNG) [6], Locomotion Mode recognition based on Gaussian Mixture Model (LMGMM) [79], Locomotion Activity Recognition (LAR) [31], Automatic Annotation for human Activity Recognition (AAAR) [40], PENCIL [35], and CopyNet [37].

4.4.5.1 Performance comparison using accuracy

Table 4.4 illustrates the performance comparison of the LRNL approach with the existing approaches at different noise concentrations. Following observations can be made from the result:

- LRNL approach outperforms the existing approaches on all the datasets in all the settings of noisy labels concentrations. This is due to the fact that the LRNL approach incorporates the noise corrective model, which helps in handling high order noisy labels with minimal accuracy compromise.
- The accuracy of the noise adaptive model is nearly similar to LAR and PENCIL for LMR and SHL datasets when the concentration of noisy labels is 0%. It is because of the stack encoder in the LAR approach, which performs well on noise-free data. However, the LAR approach suffers from performance compromise on the high concentration of noise.
- The accuracy of the existing approaches falls rapidly as the concentration of noisy labels increase and reach the minimum at 30% noisy labels. On the contrary, the proposed LRNL approach can maintain accuracy around 88% for all the datasets even at 30% noisy labels.
- At 0% noisy label concentration, DNG slightly exceeds LRNL. However, the complexity of DNG architecture is more than LRNL. Additionally, with increasing noisy label concentration, performance of the LRNL approach exceeds DNG.

4.5 Conclusion

This chapter proposed a deep learning-based LRNL approach for recognizing the locomotion modes using sensory data with noisy labels. Unlike existing approaches, LRNL approach built an ensemble model to enhance the recognition capability of the classifier, without having any prior information about the concentration of noisy labels.

Table 4.4: Accuracy (in %) comparison of LRNL approach with existing approaches using different datasets.

Datasets	Approaches or Models	Noise Concentration			
		0%	**10%**	**20%**	**30%**
LMR	LAR [31]	88.20	81.35	80.13	57.73
	AAAR [40]	86.49	79.37	77.10	69.17
	PENCIL [35]	90.49	83.07	80.97	74.22
	CopyNet [37]	85.21	79.27	76.53	67.39
	LMGMM [79]	88.25	82.21	80.65	63.81
	DNG [6]	**94.73**	87.42	74.23	54.39
	Conventional	84.37	82.37	77.93	72.95
	Noise adaptive	88.10	86.92	83.13	77.91
	Noise corrective	92.50	90.19	88.17	86.95
	LRNL	94.21	**92.23**	**90.55**	**89.76**
SHL	LAR [31]	87.10	80.53	77.53	59.67
	AAAR [40]	85.17	76.67	73.10	63.13
	PENCIL [35]	89.29	82.57	78.29	72.87
	CopyNet [37]	84.29	75.83	73.23	62.81
	LMGMM [79]	87.41	81.17	78.25	59.44
	DNG [6]	**93.07**	84.44	72.28	52.57
	Conventional	83.20	80.18	77.30	71.10
	Noise adaptive	88.40	85.57	80.65	76.81
	Noise corrective	91.20	88.39	86.48	85.70
	LRNL	92.53	**91.56**	**90.19**	**88.13**

The ensemble model incorporates three models, *i.e.,* conventional, noise adaptive, and noise corrective, for handling different concentration of noisy labels. The noise adaptive model proposed a noise adaptive loss function that reduces the discrepancy between true labels and predicted labels using dynamic variables. Next, the noise corrective model used a low rank estimate of true labels for handling noisy labels. We also carried out several experiments to validate the effectiveness of the proposed approach using a collected and two existing datasets. The experimental results showed that LRNL approach achieved an accuracy of around 93% even datasets have 10% noisy labels.