Chapter 4

Jointly Prediction of Activities, Locations, and Starting Times for Isolated Elderly People

This chapter presents a multi-task activity prediction system that jointly predicts labels, locations, and starting times of future activities. The observed sequence of previous activities characterizes future activities. The activity prediction system consists of recurrent neural networks to capture temporal dependencies. This work also carries out several experiments on collected and existing real datasets to evaluate the system's performance.

4.1 Introduction

Coronavirus disease (COVID-19) has caused a global pandemic and has become the most urgent threat to the entire world. Governments have adopted various policies such as social distancing, social isolation, and quarantine to reduce infection rates. Although all age groups are at risk of contracting COVID-19, older people face a significant risk of developing severe illness if they contract the disease. Therefore, older people have

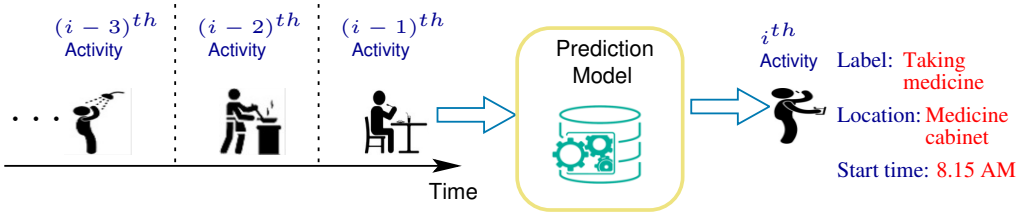


Figure 4.1: Illustration of prediction of activity labels, locations, and starting times using the sequence of previous activities.

been specifically required to stay home in isolation [110, 111].

The development of smart environments to accommodate healthcare services and assistive technologies in the elderly home is one-way such people can be helped. Activity prediction is an essential module to assist people confronting troubles to live independently in their homes [68–70]. For example, using an activity prediction module, a smart home can automate the predicted activity or prompt inhabitants to start essential activities. In a home environment, inhabitants do numerous tasks, and the majority of the tasks are repeatedly done in a specific period, thus generate a temporal pattern. Although each inhabitant has a different choice, the same inhabitant does some common tasks routinely. For example, taking medication is very likely followed by taking meal activity, and exercise is usually preceded by breakfast activity. Figure 4.1 illustrates an example scenario, where previous activities are used to predict future activity.

Deep learning has dramatically pushed artificial intelligence in many tasks [112]. Human activity prediction is one area that has not yet to gain from deep learning techniques [7]. These techniques can achieve considerable success but require a considerable amount of resources for its execution. Therefore, to reduce the resource requirement of multiple deep learning tasks, we can use a multi-task learning system that targets to achieve generalization by leveraging the inter-relatedness of multiple problems/tasks [92]. The intuition behind multi-task learning is that if two or more tasks are correlated, the joint-model can learn effectively from the shared representations. In

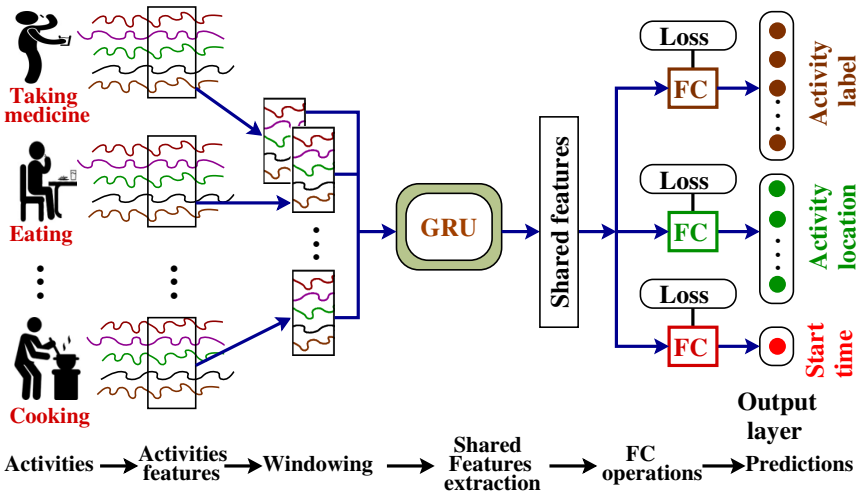


Figure 4.2: Overview of the proposed ADLP system.

comparison with the single-task system (tasks executed in isolation), a multi-task system offers three main advantages *i.e.,* i) better generalization, ii) improve performance through shared representation, and iii) a single unified model reduces the complexity of task execution.

This chapter presents an **A**ctivities of **D**aily **L**iving **P**rediction (ADLP) system that can answer three important questions: ***which activity will happen next?, where will it happen?, and when will it happen?***, *i.e.*, we predict *labels*, *locations* and *starting times* of future unobserved activities. The ADLP incorporates a multi-tasking system consisting of Recurrent Neural Network (RNN) that uses Gated Recurrent Unit (GRU) units to capture temporal dependencies. Figure 4.2 illustrates the overview of the ADLP system.

4.1.1 Motivation

The work proposed in this chapter is motivated by the following limitations in the existing literature.

- The existing literature on activity label [68,70], location [71–73], and starting time prediction [74, 75] use individual models for predicting the upcoming activity,

location, and its starting time. These models do not employ the advantage of multi-task architecture.

- The datasets for ADL [45, 46, 48] requires a huge cost of sensors deployment and data collection. Thus, a cost-effective technique is required for collecting ADL datasets.

4.1.2 Major contributions

To the best of our knowledge, this is the first work to address the problem of jointly recognizing the activity labels, location, and starting time. Our major contributions are as follows

- We propose a novel GRU recurrent neural network-based multi-task system that jointly models sequential relationships of the activities to predict the future activity labels, locations, as well as its starting times.
- We introduce three variables α, β, and γ to achieve an equilibrium of classification losses of activity label and location prediction and regression loss of starting time prediction to minimize the combined loss by solving an optimization problem.
- The experimental results demonstrate that multi-task learning outperforms that of building prediction models separately. We carry out several experiments on collected and existing datasets to evaluate the system's performance. We use [44] real well-established publicly available existing datasets.

The rest of this chapter is organized as follows. In the next section, we present the activity prediction system. The experimental results are presented in Section 4.3. Finally, we conclude the chapter in Section 4.4.

4.2 Activities of Daily Living Prediction system

This section presents an Activities of Daily Living Prediction (ADLP) system, which incorporates a multi-task system that jointly predicts labels, locations, and starting

times of future activities. First, we describe the role of different context features that are considered to solve the activity prediction problem. Next, this section presents the network architecture for the multi-tasking system. Finally, we defined the optimization problem to obtain a minimum loss.

4.2.1 Role of different features

The description of all features contributing to the proposed ADLP system is as follows:

- **Previous activities labels:** In real-life scenarios, we observe the activities follow fixed temporal sequences. Therefore, previous activities can provide useful information about future activity, which is also termed a sequential activity context.
- **Previous activities locations:** The locations of the previous activities play a vital role in the estimation of future activity locations. The location is specified in terms of the location of the motion sensor. The locations we consider in the smart home include Kitchen, Bathroom, Bedroom, Living room, Dining room, Medical room, and Front door.
- **Previous activities time of day:** Time of previous activities occurrence in a day contributes significantly in the determination of future activity time as people's behavior follows a daily rhythm, *i.e.*, activities are correlated with its timestamp.
- **Previous activities day of week:** It represents the day of a week at which previous activities happened. This feature captures the weekly rhythm, allowing the model to learn the activities that frequently occur on a particular day of the week.

These features are reported in [70]. For future activity prediction, we can then build and train a many-to-one recurrent neural network with input that consists of vectors of the features mentioned above. We can empirically select the window length of preceding

activity features for a given dataset. Recurrent neural networks are a popular choice for sequential context incorporation.

4.2.2 Network architecture

The proposed ADLP system consists of two shared GRU layers, each having 100 neurons to process sequential data, as shown in Figure 4.3. Three task-specific fully-connected layers follow the GRU layers for each task of predicting label, location, and starting time of future activities. For each task, the first two task-specific layers have 200 and 100 neurons, respectively. The third task-specific layer for label prediction task and location prediction task has 12 and 7 neurons, respectively, and is activated by the softmax. For the starting time prediction task, the third task-specific layer has one neuron activated by the linear function. All other layer's activation is set to the rectified linear unit. We have experimented with different network configurations (e.g., recurrent units type (GRU/LSTM), the number of layers, *etc.*), and observed better performance with the configuration as mentioned above. The ADLP system takes previous activities features as input, for example, features of $(k-1)^{th}$, $(k-2)^{th}$, and $(k-3)^{th}$ activities, as shown in Figure 4.3. These activity features pass through two sequential GRU units, whose output and shared features are passed through different FC layers.

4.2.3 Objective loss formulation

The proposed system jointly performs prediction of three tasks, *i.e.,* activity label, activity location, and activity starting time, as illustrated in Figure 4.3. Therefore, we minimize the summation of three losses associated with these tasks. Let $\mathbf{X} = \{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}\}$ represent the set of input feature vectors in the training dataset. Then, categorical cross-entropy loss function $\mathcal{L}_a(\cdot)$ of label prediction task can be given as

$$\mathcal{L}_a(\mathbf{Y}_a, \widehat{\mathbf{Y}}_a) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{C_a}\mathbf{1}(y_a^{(i)} = j)\log p(y_a^{(i)} = j \mid \mathbf{x}^{(i)}), \tag{4.1}$$

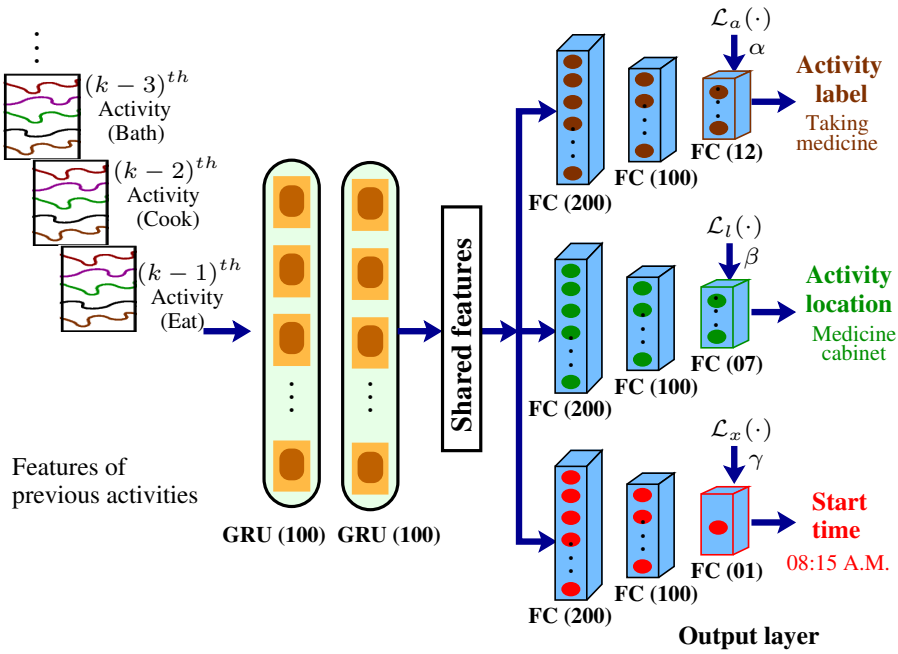


Figure 4.3: Architecture of the ADLP system.

where $\mathbf{Y}_a = \{y_a^{(1)}, \cdots, y_a^{(n)}\}$ and $\widehat{\mathbf{Y}}_a = \{\hat{y}_a^{(1)}, \cdots, \hat{y}_a^{(n)}\}$ correspond to set of future activities true and predicted labels. n and C_a denote number of samples and activities classes, respectively. For an i^{th} training instance, $\mathbf{x}^{(i)}$ represents the sequential activity features extracted from the previous N activities. The symbol $\mathbf{1}(\cdot)$ represents an identity function, where $\mathbf{1}(\cdot) = 1$, if condition $(\cdot)$ is true and 0 otherwise. Similarly, we can find the categorical cross-entropy loss function $\mathcal{L}_l(\cdot)$ of location prediction task as

$$\mathcal{L}_l(\mathbf{Y}_l, \widehat{\mathbf{Y}}_l) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{C_l}\mathbf{1}(y_l^{(i)} = j)\log p(y_l^{(i)} = j \mid \mathbf{x}^{(i)}), \tag{4.2}$$

where $\mathbf{Y}_l = \{y_l^{(1)}, \cdots, y_l^{(n)}\}$ and C_t represent set of future activity true locations and number of location classes, respectively. $\widehat{\mathbf{Y}}_l$ denotes the set of predicted locations of the activities. Further, for the regression part *i.e.* future activity starting time prediction task, we choose Mean Absolute Error (MAE) for estimating difference between actual

Procedure 4.2: Training procedure of single-task models.

Input: $\mathbf{X}$, $\mathbf{Y}_a$, $\mathbf{Y}_l$, $\mathbf{Y}_x$, η_a activity label learning rate, η_l location learning rate, η_x starting time learning rate;

Output: Trained model parameter Θ_a,Θ_l,Θ_x;

while *not converge* **do**
  $t = t + 1$;
  Forward propagation and compute activity loss $\mathcal{L}^t_{a_s}(\mathbf{Y_a}, f_a(\mathbf{X}, \Theta_a))$ over $\mathbf{X}$ by using Equation 4.1;
  Backward propagation of the model output and generate loss function gradient $g^t_a = \bigtriangledown_{\Theta_a} \mathcal{L}^t_{a_s}(\mathbf{Y_a}, f_a(\mathbf{X}, \Theta_a))$;
  Update weight parameters Θ_a by $\Theta^{t+1}_a = \Theta^t_a - \eta_a g^t_a$;
while *not converge* **do**
  $t = t + 1$;
  Forward propagation and compute location loss $\mathcal{L}^t_{l_s}(\mathbf{Y_l}, f_l(\mathbf{X}, \Theta_l))$ over $\mathbf{X}$ by using Equation 4.2;
  Backward propagation of the model output and generate loss function gradient $g^t_l = \bigtriangledown_{\Theta_l} \mathcal{L}^t_{l_s}(\mathbf{Y_l}, f_l(\mathbf{X}, \Theta_l))$;
  Update weight parameters Θ_l by $\Theta^{t+1}_l = \Theta^t_l - \eta_l g^t_l$;
while *not converge* **do**
  $t = t + 1$;
  Forward propagation and compute time loss $\mathcal{L}^t_{x_s}(\mathbf{Y_x}, f_x(\mathbf{X}, \Theta_x))$ over $\mathbf{X}$ by using Equation 4.3;
  Backward propagation of the model output and generate loss function gradient $g^t_x = \bigtriangledown_{\Theta_x} \mathcal{L}^t_{x_s}(\mathbf{Y_x}, f_x(\mathbf{X}, \Theta_x))$;
  Update weight parameters Θ_x by $\Theta^{t+1}_x = \Theta^t_x - \eta_x g^t_x$;
return $\{\Theta_a,\Theta_l,\Theta_x\}$;

and predicted output. The loss function $\mathcal{L}_x(\cdot)$ of starting time prediction is given as

$$\mathcal{L}_x(\mathbf{Y}_x, \widehat{\mathbf{Y}}_x) = \frac{1}{n} \sum_{i=1}^{n} \left| y_t^{(i)} - \hat{y}_t^{(i)} \right|, \tag{4.3}$$

where, $\mathbf{Y}_x$ and $\widehat{\mathbf{Y}}_x$ denotes the set of true and predicted starting time of the activities.

Procedure 4.2 summarizes the steps involved in the training of the single task prediction models to estimate the model parameters, Θ_a, Θ_l, and Θ_x for activity label, location, and starting time predictions models, respectively. It involves a forward pass for calculating loss and backward pass for updating the model parameters. The procedure runs up till the convergence point, where the value of loss reaches near to zero, or

the difference between successive losses is negligible.

In a multi-tasking system, the contribution of each loss-function is not equal. Therefore, to estimate the fractional contribution of these loss functions, we introduce three variables α, β, and γ for label prediction, location prediction, and starting time predictions tasks, respectively. Let $f(\mathbf{X}, \Theta) = \{\widehat{\mathbf{Y}}_a, \widehat{\mathbf{Y}}_l, \widehat{\mathbf{Y}}_x\}$, denotes prediction function for multi-tasking model, where $\mathbf{X}$ is the input vector and Θ is the model parameter. Then by combining the loss functions given in Equation 4.1, Equation 4.2, and Equation 4.3 with their fractional contribution, we obtain combined loss function $\mathcal{L}(\mathbf{Y}_a, \mathbf{Y}_l, \mathbf{Y}_x, f(\mathbf{X}, \Theta))$, which is defined as

$$\mathcal{L}(\mathbf{Y}_a, \mathbf{Y}_l, \mathbf{Y}_x, f(\mathbf{X}, \Theta)) = \alpha\mathcal{L}_a(\mathbf{Y}_a, \widehat{\mathbf{Y}}_a) + \beta\mathcal{L}_l(\mathbf{Y}_l, \widehat{\mathbf{Y}}_l) + \gamma\mathcal{L}_x(\mathbf{Y}_x, \widehat{\mathbf{Y}}_x). \tag{4.4}$$

To obtain the value of the three variables α, β, and γ, we define the following optimization problem:

$$\begin{aligned}
&\arg\min_{\Theta} \ \mathcal{L}(\mathbf{Y}_a, \mathbf{Y}_l, \mathbf{Y}_x, f(\mathbf{X}, \Theta)) \\
&s.t. \quad C_1: \ \alpha + \beta + \gamma = 1, && \text{(4.5a)} \\
&\qquad C_2: 0 < \{\alpha, \beta, \gamma\} < 1, && \text{(4.5b)} \\
&\qquad C_3: \mathcal{L}_a(\mathbf{Y}_a, \widehat{\mathbf{Y}}_a) \leq \mathcal{L}_{a_s}(\mathbf{Y}_a, f_a(\mathbf{X}, \Theta_a)), && \text{(4.5c)} \\
&\qquad C_4: \mathcal{L}_l(\mathbf{Y}_l, \widehat{\mathbf{Y}}_l) \leq \mathcal{L}_{l_s}(\mathbf{Y}_l, f_l(\mathbf{X}, \Theta_l)), && \text{(4.5d)} \\
&\qquad C_5: \mathcal{L}_x(\mathbf{Y}_x, \widehat{\mathbf{Y}}_x) \leq \mathcal{L}_{x_s}(\mathbf{Y}_x, f_x(\mathbf{X}, \Theta_x)). && \text{(4.5e)}
\end{aligned}$$

• **Objective:** The main objective of the optimization problem is to determine the optimal value of variables α, β, and γ such that the combined loss $\mathcal{L}(\mathbf{Y}_a, \mathbf{Y}_l, \mathbf{Y}_x, f(\mathbf{X}, \Theta))$ is minimized and we obtain an optimal trained model parameter Θ for the multi-tasking system.

• **Constraints:** Constraint C_1 ensures the values assigned to the variables α, β, and

γ sums to unity. The values of α, β, and γ individually do not exceed 1 is a condition imposed by C_2. Constraints C_3, C_4, and C_5 ensures that the values of losses in multi-tasking model $(\mathcal{L}_a(\cdot), \mathcal{L}_l(\cdot), \mathcal{L}_x(\cdot))$ should not exceed losses $(\mathcal{L}_{a_s}(\cdot), \mathcal{L}_{l_s}(\cdot), \mathcal{L}_{x_s}(\cdot))$ in single task models, respectively.

Algorithm 4.1 depicts the steps involved in solving the optimization problem given in Equation 4.5 and training ADLP system.

Algorithm 4.1: Solving optimization problem in Equation 4.5 and training ADLP system.

Input: $\mathbf{X}$, $\mathbf{Y}_a$, $\mathbf{Y}_l$, $\mathbf{Y}_x$, η learning rate, trained model parameter Θ_a,Θ_l,Θ_x from Procedure 4.2;
Output: Trained model parameter Θ;

```
Initialize α, β, γ that satisfy constraints C1 and C2;
while True do
    Initialize model parameter Θ;
    while not converge do
        t = t + 1;
        Forward propagation and compute loss L(Y_a, Y_l, Y_x, f(X, )) over X by
        using Equation 4.4;
        Backward propagation of the model output and generate loss function
        gradient g^t = ▽^t_Θ L(Y_a, Y_l, Y_x, f(X, ));
        Update weight parameters Θ by Θ^{t+1} = Θ^t − ηg^t;
    if (constrains C3, C4, and C5 are true) then
        Θ_f=Θ;
        break;
    else
        Update α, β, γ that satisfy constraints C1 and C2;
return Θ_f;
```

4.2.3.1 Time complexity

The time complexity of Algorithm 4.1 mainly depends on the computation of trained parameters, *i.e.,* Θ_a, Θ_l, and Θ_x using Procedure 4.2 and optimization steps for estimating final trained parameter Θ_f. Procedure 4.2 involves three **while** loops for computing, Θ_a, Θ_l, and Θ_x. Each loop involves the computation of loss function that includes feature extraction through RNN layers, contributing to the major portion of the

time complexity. The time complexity of extracting features using RNN is $O(W)$ [113], where W is the number of weight parameters learned during training. The time complexity of each loop is $O(I_k W)$, where I_k is the number of iterations after which the convergence is achieved, $\forall k \in \{a, l, x\}$. Therefore, the time complexity of computation of trained parameters is $O(IW)$ with I, representing the maximum iterations for convergence. Similarly, the time complexity for computing trained parameters Θ is $O(I'W')$, where I' is the number of iterations for convergence and W' is the number of weight parameters. Therefore, we finally come up with the time complexity of Algorithm 4.1 as $O(IW) + O(nI'W')$, where n represents the number of iterations taken in finding optimal value of variables α, β, and γ.

4.3 Experiments and Results

The experimental analysis on the proposed ADLP system is performed on collected and existing real datasets to evaluate the systems performance. This section first describes different schemes of deployment by considering three tasks, *i.e.,* activity prediction ($\mathbf{T}_1$), location prediction ($\mathbf{T}_2$), and starting time prediction ($\mathbf{T}_3$). Next, we elaborate a detailed description of the collected dataset for human activity prediction. Further, this section presents the parameter setting during the implementation, followed by the experimental results. Finally, we present a comparison of the proposed work with different machine learning and deep learning techniques.

4.3.1 Deployment schemes

We consider the following five schemes for the deployment of multi-task configuration; Scheme $\mathbf{S}_1$: all three tasks *i.e.,* $\mathbf{T}_1, \mathbf{T}_2,$ and $\mathbf{T}_3$ deployed separately, Scheme $\mathbf{S}_2$: $\mathbf{T}_1$ and $\mathbf{T}_2$ are shared and $\mathbf{T}_3$ is separate, Scheme $\mathbf{S}_3$: $\mathbf{T}_1$ is separate and $\mathbf{T}_2$ and $\mathbf{T}_3$ are shared, Scheme $\mathbf{S}_4$; $\mathbf{T}_1$ and $\mathbf{T}_3$ are shared and $\mathbf{T}_2$ is separate, and Scheme $\mathbf{S}_5$: all three tasks are integrated (ADLP system). Figure 4.4 illustrates different schemes considered in this

work. These schemes aim to decide the level of sharing of different tasks depending upon the requirement of performance, memory, and inference time. A shared representation denotes the performance of tasks that profoundly affect each other. Thus, a shared representation preserves the accuracy and requires lower inference time due to task dependency.

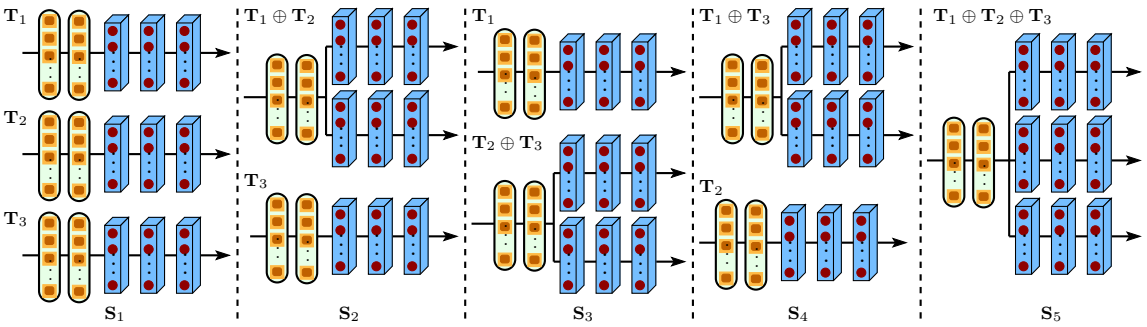


Figure 4.4: Illustration of different deployment schemes for label prediction ($\mathbf{T}_1$), location prediction ($\mathbf{T}_2$), and starting time prediction ($\mathbf{T}_3$), where symbol $\oplus$ denotes sharing (integration) of models.

4.3.2 Datasets

We carry out experiments on collected and existing datasets to evaluate the system's performance. We use publicly available real datasets [44] collected in CASAS smart home testbeds at Washington State University.

4.3.2.1 Data collection

In this work, we have collected sensory data of various environmental and wearable sensors. To facilitate the easier and effective collection of sensory data, we develop a prototype setup. The prototype is deployed in an apartment of IIT (BHU) Varanasi. We have used multiple sensors, including the Inertial Measurement Unit (IMU), temperature and humidity, magnetic, sound, and Passive InfraRed (PIR) motion sensors attached with a NodeMCU. Motion sensors are deployed on the ceiling to detect motion

Table 4.1: Illustration of used sensors with their description.

Sensor	Description
Magnetic	detect door and cabinet openings and closings.
IMU	provide information about the relative body movement.
DHT-11	measure room temperature and humidity.
Sound	set to high if sound is detected inside the room.
PIR motion	set to high if movement is detected in range.

and magnetic sensors on doors and cabinets to detect their openings and closings. Temperature and humidity sensors are deployed in the kitchen and bathroom. Additionally, the sensory values from the IMU sensor deployed in the smart band and shoes of the user are also recorded for better inference of activity along with location and time of occurrence. The data collection involves one volunteer living in the apartment. The volunteer was instructed to wear the smart band and shoes with IMU sensors. The sensory values of environmental and wearable sensors are wirelessly transmitted to the PC inside the apartment.

The data were recorded for 90 days by the volunteer. The description of sensors considered during the prototype implementation is given in Table 4.1. The deployed sensors generate events when the inhabitant performs activities within the sensing range of the sensors. These sensory events are recorded on the PC inside the apartment. We assume that an activity recognition algorithm is available to label each sensor event with its corresponding activity label, location, and start time. We use this information to train the activity predictor. During the recognition, the unlabelled sensory values from different sensors are transferred to the PC (Dell system inside the room) via NodeMCU, acting as a communication module. The sensors are directly connected (wired connection) to the NodeMCU that wirelessly transmits sensory data to the PC. Further, a Python script is developed for performing entire operations, including data collection from sensors, communication to PC, and recognition using a built classifier on PC. To successfully run the python script, we install Anaconda and Keras on PC.

The two major components in the collected dataset are as follows.

Smart home layout: Figure 4.5 exhibits a sample layout and sensor placement for the considered scenario during data collection in this work. The data collection area has a bedroom, kitchen, dining area, and bathroom. The inhabitant (volunteer) in the apartment performed normal unscripted activities of daily living. Four environmental sensors are used during data collection: Sound, magnetic, temperature and humidity, and PIR motion sensors denoted by r_1, r_2, r_3, and r_4, respectively. Additionally, we have used the sensory values of IMU sensors (denoted as r_5 and r_6) deployed in smart band and shoes.

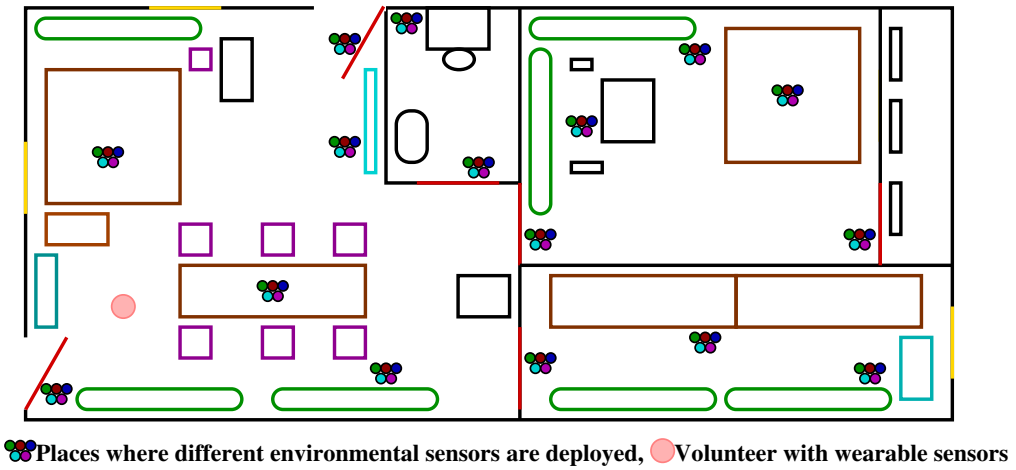


Figure 4.5: Smart home floor plan and sensor layout for collected dataset.

Embedded sensors deployment in the apartment: In this work, to deploy embedded environmental sensors, we have adopted the same strategy proposed by CASAS [41, 44]. Motion sensors are deployed on the ceiling to detect motion, and magnetic sensors on doors and cabinets to detect their openings and closings. The location of the deployment of sensors in a given smart home mainly depends on the ADL we want to monitor and the minimum number of sensors required depends on the number of different ADL we want to monitor. For example, to detect any movement

in a whole room, a motion sensor needs to be placed on the ceiling in a location that allows monitoring of the entire room but not motion outside of the room. To monitor the "sleeping in bed" activity, one motion sensor is needed to be mounted on the ceiling above the bed, and the position of the sensor should be over the participants' chest when they are sleeping. To monitor "eating" activity, one motion sensor should be placed on the ceiling above the dining table. The volunteer was asked to make entries of activities while performing ADL. These entries were later used to manually annotate sensor events with corresponding labels to produce accurate ground truth labels to inference activities.

4.3.2.2 Existing datasets

The experimental analysis on the proposed work is also performed on publicly available real datasets collected in the CASAS smart home testbeds at Washington State University [44]. There is a prior assumption that the different types of sensors are placed at a distinct location inside the apartment to monitor the inhabitants' physical surroundings. We have used five datasets collected from five CASAS smart home testbeds, each housing one older adult with no known cognitive impairments. We denote these datasets as $\mathbf{D}_2$ - $\mathbf{D}_6$. Details about CASAS datasets are available at [41]. We denote the collected dataset as $\mathbf{D}_1$.

4.3.2.3 Activities of daily living

In the experimental analysis, we have considered eleven ADL for activity prediction in set $\mathcal{A}$ ($\mathcal{A} \in$\{Bathing, Bed toilet transition, Eating, Enter home, Housekeeping, Leave home, Meal preparation, Personal hygiene, Resting on couch, Sleeping in bed, Taking medicine\}), which are denoted as $\{a_1, \cdots, a_{11}\}$ [70]. The dataset includes the date, time, sensor ID, sensor message linked with each sensor event, and annotated activity label.

4.3.3 Implementation details

This section discusses settings of various parameters that are used in the implementation of the proposed system. We use the 10-fold cross-validation approach for the training-testing split and average our results over these ten combinations. We use an Adam optimizer with a mini-batch size of 64. The learning rate is initialized at 0.001. The proposed model converged within about 100 epochs. We empirically select the window size $N = 2$ to $N = 11$ for feature extraction of the activities. The experiments are performed on a PC with Intel Core $i7$-CPU, 2.83 GHz clock speed, 6 GB RAM, and Ubuntu 18.04 operating system. We use Python-based libraries, Keras and Scikit-learn, for implementation.

4.3.4 Performance metrics

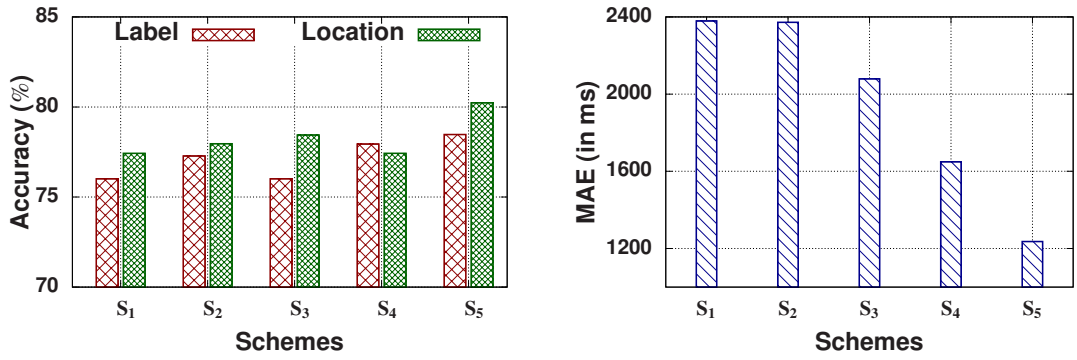
For evaluation, we compute accuracy for the classification tasks (activity label and location prediction). For the accuracy metric, the higher value indicates better prediction performance. We compute Mean Absolute Error (MAE) for the regression task (activity starting time prediction). The lower the value of MAE the better is the prediction performance. Well-known error metrics for regression tasks are MAE and Root Mean Square Error (RMSE). Since the RMSE squares each term, it brings a disadvantage in effectively weighting large errors more heavily than small ones. We evaluate the predictions using MAE, as RMSE would be very sensitive to errors on outlier data points, where the time between two activities is very large [39].

4.3.5 Results and discussion

This section carries out the experimental evaluation to validate the performance of the proposed multitask model. The main objective of these experiments is to analyze how well our framework can predict the labels, locations and starting times of future activities.

4.3.5.1 Prediction accuracy

This work carried out experiments on six datasets discussed in Section 4.3.2 to estimate the performance (averaged over all datasets) of different schemes ($\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$, $\mathbf{S}_4$, and $\mathbf{S}_5$) on activity, location, and starting time predictions. The disjoint task prediction in Scheme $\mathbf{S}_1$ suffers from performance degradation ($\approx 4\%$) in contrast with Scheme $\mathbf{S}_5$ when all three prediction tasks are in sharing mode. It is because the joint-model can learn effectively from the shared representations as all three tasks are correlated. A trend of prediction performance improvement in activity label and activity location is observed in part (a) of Figure 4.6. Part (b) depicts that the Scheme $\mathbf{S}_5$ tends to reduce generalization error for predicting the starting time of activity. The starting time prediction of different activities is best for Scheme $\mathbf{S}_5$. The separate deployment of all task prediction models in the case of Scheme $\mathbf{S}_1$ increases the generalization error that results in reduced starting time prediction performance on different datasets.



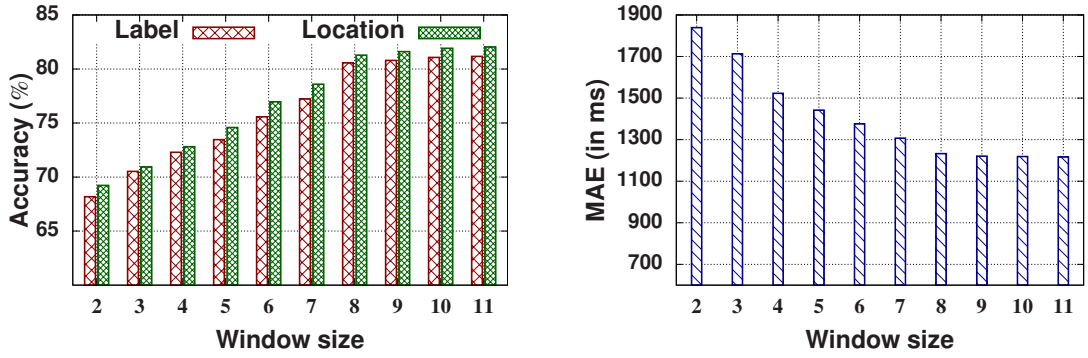
(a) Label and location prediction

(b) Starting time prediction

Figure 4.6: Performance (averaged over all datasets) under different schemes (MAE=Mean Absolute Error).

4.3.5.2 Window size

The performance of the ADLP system varies based on the window size, as illustrated in Figure 4.7. To determine the optimal window size, we change the window size from 2 to 11 and observe its effects on the prediction performance of activity label,



(a) Label and location

(b) Start time

Figure 4.7: Impact of window size on performance of ADLP system.

location, and start time. Part (a) of Figure 4.7 illustrates the impact of the window size on the performance of label and location predictions (aggregated across all datasets). It shows that the performance increases as window size increases until performance plateaus (after window size 8). Based on these results, we consider a window size of 8 for the experimental results. Increasing the window size further may give a small benefit; however, the complexity rises as the window size increases. Similarly, results are obtained for activity starting time prediction, as illustrated in part (b) of Figure 4.7.

Table 4.2: Comparison of proposed approach ($\mathbf{S}_5$) with existing approaches for predictions of activity, location, and starting time.

Classifiers	Label prediction accuracy (%)						Location prediction accuracy (%)						Staring time prediction (MAE in ms)					
	$\mathbf{D}_1$	$\mathbf{D}_2$	$\mathbf{D}_3$	$\mathbf{D}_4$	$\mathbf{D}_5$	$\mathbf{D}_6$	$\mathbf{D}_1$	$\mathbf{D}_2$	$\mathbf{D}_3$	$\mathbf{D}_4$	$\mathbf{D}_5$	$\mathbf{D}_6$	$\mathbf{D}_1$	$\mathbf{D}_2$	$\mathbf{D}_3$	$\mathbf{D}_4$	$\mathbf{D}_5$	$\mathbf{D}_6$
KNN	51.15	39.67	57.64	40.09	47.66	41.35	55.66	50.21	63.20	46.28	55.59	54.46	1789	1616	2086	2265	2227	2984
SVM	57.14	49.68	62.19	44.76	47.56	42.15	62.51	56.11	65.57	50.32	57.52	55.91	2102	1600	2685	2262	2480	2574
Decision Tree	71.86	61.31	65.44	56.74	66.59	57.72	75.02	64.44	70.16	61.08	65.38	67.26	1343	1303	2031	1466	1862	1791
Random Forest	78.76	69.40	74.09	60.55	71.50	59.58	81.30	72.17	77.35	62.54	69.80	73.95	1062	1055	1461	1198	1589	1436
Gradient Boost	78.47	69.29	74.10	66.39	75.06	65.78	82.02	73.47	77.27	68.82	73.45	76.29	1190	1176	1648	1287	1612	1538
LSTM	80.16	74.32	72.68	69.02	77.29	69.18	82.46	75.89	77.45	73.60	75.27	71.94	1065	1085	1430	1261	1475	1351
Proposed	**84.24**	**75.93**	**74.28**	**72.54**	**78.86**	**73.13**	**85.48**	**77.08**	**77.49**	**74.17**	**81.74**	**76.46**	**1039**	**1052**	**1395**	**1215**	**1439**	**1275**

4.3.6 Comparison with other approaches

Finally, in Table 4.2, we compare the ADLP system (Scheme $\mathbf{S}_5$) with the existing classification approaches including, KNN, SVM, Decision Tree, Random Forest, and Gradient Boost, and LSTM. We perform a comparison for activity label prediction,

activity location prediction, and starting time prediction. The detailed descriptions are as follows.

- *Activity label prediction:* It helps to determine how well the proposed work can predict the label of future activities compared to existing approaches. Table 4.2 lists the performance achieved by different machine learning classifiers for activity label prediction on six datasets. The obtained results illustrate that the proposed approach outperforms the existing approaches for future activity label predictions.
- *Activity location prediction:* The activity location prediction plays a decisive role in the correct identification of activity location. The proposed approach predicts the activity's location with the accuracy highest among the existing approaches, as shown in Table 4.2 for all six datasets. The results indicate that the location prediction accuracy for the dataset $\mathbf{D}_1$ is more than 85% with a performance gain of at least 3%.
- *Activity starting time prediction:* Table 4.2 illustrates that the deep learning-based approaches for predicting the starting time of activity are more reliable than that of machine learning approaches and can gain significantly higher performance (in terms of Mean Absolute Error (MAE)) for dataset $\mathbf{D}_1$. The results illustrate that Gradient Boost has achieved comparable performance to that of deep learning approaches. We also observe the proposed approach's performance gain for starting time prediction similar to that of activity label and location predictions.

4.4 Conclusion

This chapter has presented the design, implementation, and evaluation of an ADL prediction system to jointly predict label, location, and starting time of future activities. Existing conventional machine-learning and deep learning-based systems do not address the multi-tasks predictions, where three separate models reduce overall prediction accu-

racy. The proposed system uses cross-entropy and mean absolute error loss functions for capturing the discrepancy between the predicted output labels and actual labels. The experimental results illustrate that the multi-task system comparatively gives higher accuracy than the single-task system.