Sensor-Based Human Activity Recognition for Smart Healthcare: A Semi-supervised Machine Learning

Chapter · July 2019

DOI: 10.1007/978-3-030-22971-9_39

Abrar Zahin
Utah State University
4 PUBLICATIONS **9** CITATIONS
SEE PROFILE

Tan Le
Old Dominion University
63 PUBLICATIONS **594** CITATIONS
SEE PROFILE

Rose Qingyang Hu
Utah State University
358 PUBLICATIONS **6,245** CITATIONS
SEE PROFILE

Sensor-based Human Activity Recognition for Smart Healthcare: A Semi-supervised Machine Learning *

Abrar Zahin[0000-0002-0130-9566], Le Thanh Tan[0000-0003-0807-4357], and
Rose Qingyang Hu[0000-0002-1571-3631]

Utah State University, Logan, Utah 84322-4120, USA.
abrarzahin303@gmail.com, tan.le@usu.edu,rose.hu@usu.edu.

Abstract. Human action recognition is an integral part of smart health monitoring, where intelligence behind the services is obtained and improves through sensor information. It poses tremendous challenges due to huge diversities of human actions and also a large variation in how a particular action can be performed. This problem has been intensified more with the emergence of Internet of Things (IoT), which has resulted in larger datasets acquired by a massive number of sensors. The big data based machine learning is the best candidate to deal with this grand challenge. However, one of the biggest challenges in using large datasets in machine learning is to label sufficient data to train a model accurately.Instead of using expensive supervised learning, we propose a semi-supervised classifier for time-series data. The proposed framework is the joint design of variational auto-encoder (VAE) and convolutional neural network (CNN). In particular, the VAE intends to extract the salient characteristics of human activity data and to provide the useful criteria for the compressed sensing reconstruction, while the CNN aims for extracting the discriminative features and for producing the low-dimension latent codes. Given a combination of labeled and raw time-series data, our architecture utilizes compressed samples from the latent vector in a deconvolutional decoder to reconstruct the input time-series. We intend to train the classifier to detect human actions for smart health systems.

Keywords: Action Recognition · Variational Auto-encoder · Convolutional Neural Network · Semi-supervised Learning · Internet of Things · Smart Health System.

1 Introduction

1.1 Motivations

The emerging ubiquitous mobile and sensor-rich devices have led to higher demands for the human action recognition (HAR). Some of the major applications,

* This work is supported in part by National Science Foundation under grants NeTS 1423348 and EARS 1547312, in part by Natural Science Foundation of China under grant 61728104, and in part by Intel Corporation

which are benefited from HAR, are daily lifelogging, healthcare, senior care, personal fittings and etc. [1–3]. The efficacy of an HAR system greatly depends on the salient features extracted from the raw signals such as accelerometer readings [4]. There has been extensive research into HAR, but generally, few practical deployed applications have been proposed to address the following arising issues. Firstly, the rapid spread of smart devices have resulted in the scarcity of label data. This has emerged to be a critical issue for developing the learning model, which is capable to learn the salient features and to further recognize the unseen actions. Furthermore, a massive number of sensors will participate in generating big data in an Internet of Thing (IoT) smart health environment. The big data provides more difficulties to the machine learning systems including: *1) the problem of using compressed sampling to acquire the large datasets, 2) the problem of improving the quality of reconstructed data from large amounts of measurements and 3) the problem of performing data labeling at scale.*

The big data based artificial intelligence (AI) and machine learning are the emerging tools to solve the major problems faced in 5G and beyond networks as well as in industrial IoT systems [6–12]. The innovative These intelligent tools can achieve the goals of acquiring accurate and scalable data in the era of big data. Recent works in compressed sensing (CS) reveal that a signal having a sparse representation in one basis can be recovered from a small number of projections onto a second basis that is incoherent with the first one [5]. Especially, the compressed sampling technology has a capability of receiving the sparse signal at the rates lower than the traditional Nyquist sampling. At receiver side, the signal reconstruction is proposed by solving a convex optimization problem called min-l_l with equality constraints. Some advanced techniques, namely the basic pursuit, the orthogonal matching pursuit, the tree-based orthogonal matching pursuit [5] are also employed to improve the performance of reconstruction.

As a result, many researchers successfully applied the CS technology into various real applications such as cognitive radio networks [6–8] and Smartgrids [9, 10]. In particular, they make efforts to design the CS-based compressive spectrum sensing frameworks [6–8], which operate in distributed manner. Moreover, their proposed models are reliable and appropriate to the practical applications by reducing a hardware complexity. Furthermore, Tan *et al.* [11, 12] developed the deep reinforcement learning based framework to solve the joint communication, edge caching and computing design problem in vehicular networks. In particular, they use the classic AI (i.e. the particle swarm optimization scheme) for mobility-aware reward function at the associated large timescale level, while they employ deep reinforcement learning at the small timescale level of their sophisticated twin-timescale solution [11]. These proposed AI schemes are proved to achieve operational excellence and cost efficiency.

1.2 Our Contributions

In this paper, we propose a semi-supervised classifier model, where we employ the deep generative model [16] for recognizing the human activities in the smart healthcare systems. Specifically, the contributions of this paper can be summarized as follows.

1. In particular, the time-series data acquired by sensors can also be represented as an image [13] and the compressed sensing technology developed for image processing would be applied to our data. Then, we use the variational autoencoder (VAE) [14, 15] as the underlying generative model recovery in our proposed compressed sensing algorithm, which provides novel insights into our proposed inexpensive data sampling and acquisition. The core idea of VAE is to learn the true data distribution of a training set in order to generate new data points with some minor variation. It implies that the VAE has the capability of extracting the salient characteristics of human activity data and reliably providing more useful additional criteria for the CS reconstruction objective. According to this idea, our model learns the hidden parameters or the latent codes, which enable it to draw a more precise decision boundaries for the classification than a classifier based on labeled data alone [16].
2. To produce the most defining latent code, we need to capture the discriminative features of the action signal, which are then used for the classification. Therefore, we propose the convolutional neural network (CNN) in the encoder part, which not only extracts the discriminative features but also produces the latent codes in lower dimensions. For the data reconstruction, we employ the deconvolution network in the decoder, where the latent code is the input. The classifier is trained until the reconstruction loss is less than the predetermined threshold. Then the output is the reconstructed signal, which has a higher dimension than the input (i.e. the latent code).
3. To validate our proposed semi-supervised classifier model, we use the data set, namely Actitracker Dataset [17], which contains 2,980,765 samples with six daily attributes. We present experimental results for illustrating the performances of our proposed algorithm such as the VAE loss for training and testing data, the reconstruction loss and the classification accuracy. We also make a fair comparison with the existing works, where our proposed semi-supervised learning method outperforms the supervised learning method in terms of the classification accuracy for the same number of measurements for the Actitracker Dataset.

The outline of this paper is as follows. In Section 2, we discuss important related works on machine learning. In Section 3, we describe our system model. Section 4 briefly presents the proposed framework of semi-supervised learning for time-series human action. Section 5 presents our performance results followed by our concluding remarks in Section 6.

2 The Recent State of Machine Learning Applications: Challenges and Solutions

Deep learning has emerged to be an well-established technique to discover compact and meaningful representations of raw data without relying on domain knowledge [6–12, 16, 18–20]. In particular, there are four main types of tasks within the field of machine learning, which are referred as the supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. The main difference between supervised learning and unsupervised learning [16, 20] is that supervised learning is done using a ground truth. It implies that we have prior knowledge of what the output values for our samples should be. So given a sample of input data and desired outputs, supervised learning aims for building the best function approximation of the relationship between input and output. In unsupervised learning, we only have input data and do not have corresponding output variables (i.e. the labeled data). Therefore, unsupervised learning aims for inferring the natural structure present within a set of data points. It means that its goal is to model the underlying structure or distribution in the data to learn more about the data. We can readily observe that supervised learning requires a large amount of labeled samples to generalize well but it is an expensive and difficult process. Also in unsupervised learning, there are no correct answers and there is no teacher. To deal with these limitations, semi-supervised learning [16,20] is presented, where it takes a middle ground between supervised learning and unsupervised learning. Here, we have a large amount of input data and only some of the data is labeled. It uses a morsel amount of labeled data bolstering a larger set of unlabeled data. Finally, reinforcement learning [11, 12] trains an algorithm with a reward system. Then, it provides feedback, when an artificial intelligence agent performs the best action in a particular situation. In the following, we discuss how these machine learning mechanisms can be applied to the real applications and what are the main constraints of the existing works.

In reinforcement learning, let us focus on the proposed artificial intelligent framework eligible for the real application, which is the example application in edge caching and computing framework in vehicular networks. As we know that the action space and state space are very large. Especially, the scenario of user mobility causes the increase of these spaces. Therefore, we would determine the effective methods to deal with these concerns. [12] is the first step to apply the deep reinforcement learning to edge caching and computing. Here, the deep reinforcement learning is employed for both the large timescale model and the small timescale model. The large timescale model aims for reducing the possible sets of connecting road site units (RSUs) and vehicles for the tagged vehicle, which are then used in the small timescale model. Therefore, the large timescale model roughly estimate the reward and the reserved sets of RSUs and vehicles for the tagged vehicle. However, this initial proposed method cannot work well for the large scale networks, when the number of nodes increase. It means that the proposed method needs to be improved, when we want to deploy it in the practical application. Moreover, the computational complexity of model is still high, and hence it is very hard to apply it to the real application. Motivated from this, Tan *et al.* [11] develop the particle swarm optimization (PSO) for the large timescale model. In fact, the advanced PSO is used to guarantee the global optimal solutions with the fast convergence and high stability. Because the parameter setting and updating is modified to match well with the dataset in the scenario of edge caching and computing. Moreover, the proposed PSO can avoid the local optimal solutions. Therefore, the solutions in the large timescale model are optimal. Remember that in [12], the large timescale model roughly determine the solutions, i.e. these solutions may not optimal; and the small timescale model needs to correct them. However, in the case that the possible sets of RSUs and vehicles for the tagged vehicle are wrongly chosen, this would cause the wrong decision and/or increase the learning time.

Similarly, many algorithms are proposed to deal with the grand challenges and solutions in supervised learning, unsupervised learning and semi-supervised learning. Let us discuss the action recognition, where one of the most important tasks is the feature extraction. Statistical features such as mean, standard deviation, entropy, correlation coefficients and etc are the most widely used handcrafted features in the action recognition [21]. Moreover, the advanced eigenvalue models is used for blind feature extraction in [23], where the modern random matrix theory, i.e. the spiked population model, can provide an efficient solution to the problem with unknown information. In [6–8], the spectrum data is transformed into a domain, where it is sparse. For example, a wide-band signal is sampled in the Fourier domain and then is transformed to Wavelet space, where it has a sparse representation. Based on that, salient information is extracted and then is used for data recovery. Similarly, Tamura *et al.* [22] used Fourier transform, Wavelet transform and discrete cosine transform to learn the underlying features. The other application of using CS is data processing in Smartgrids [9, 10], where the smart-meter data are projected into Wavelet domain for the further sampling salient extraction. Furthermore, deep neural network [18] and restricted Bozltman machine [19] were proposed as deep learning techniques to extract features. Another deep learning model, namely shift-invariant sparse coding [20] was used to perform unsupervised learning to train an auto-encoder network. In [27], CNN was proposed to recognize human activities using mobile sensors. To exploit the huge unlabeled samples likely to be produced by IoT, semi-supervised learning can be a great alternative. The most recent additions to the conventional semi-supervised algorithms, like transductive support vector machines [20] and semi-supervised support vector machines [16] are belong to the probabilistic model based semi-supervised learning.

In summary, the developed artificial intelligence framework should be eligible for the real application by employing the advanced machine learning mechanism. It can avoid many barriers observed such as the massive devices and big data. It means that the proposed artificial intelligent framework must ensure the optimal solutions with low computational complexity and high stability. Furthermore, there is no machine learning algorithm, which can be efficiently applied to every practical application. Because it totally depends on the characteristics and properties of the observed dataset in every application. To work efficiently with the dataset in the specific application, we must modify the machine learning algorithms and/or combine both the classic and advanced learning methods. The main goal of our paper is to tackle these critical challenges and to develop the algorithms to determine the optimal solutions.

3 System Model

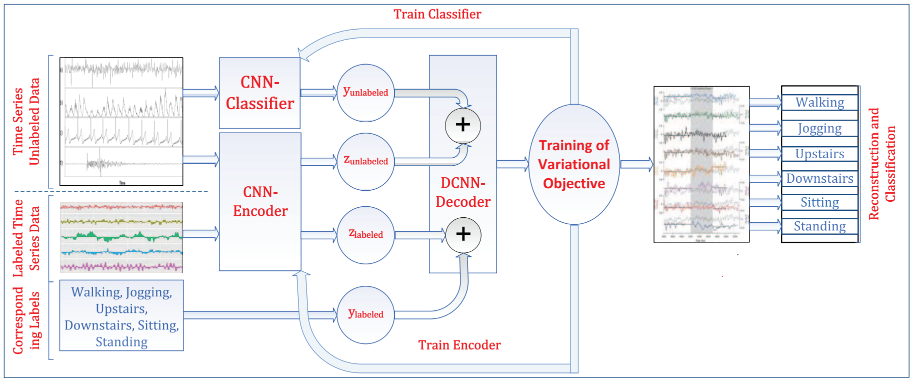

Fig. 1. Semi-supervised Classification for Time-Series Human Actions in the Smart Healthcare System

3.1 Applications and Protocol Design

Our proposed semi-supervised classifier can be implemented in a module for various kind of applications in the Smart Healthcare System, i.e. *1) reporting day to day activities, 2) reporting activities to center for monitoring and 3) life-logging for daily activities.* For simplicity, we consider the six daily activities in our model, namely "Walking","Jogging","Upstairs","Sitting", "Standing" and "Downstairs". Our classifier module can be built with different cascaded classifiers like the action classifiers and the fall-down classifier. The system implementation is presented in Fig. 1, where we consider the classifier models including the wearable sensor devices, the apps (action classifier and other classifier like fall-down detection). Let us consider application 2, i.e. *reporting activities to center for monitoring*, which may be used in the healthcare system at the senior homes. In particular, we consider the reporting activities for fall-down monitoring scenario. For this monitoring, our system model would operate in two modes, i.e. the distributed protocol and the centralized protocol. Let's consider a human activity "Upstairs". Upon receiving the accelerometer data, the sensor will pass it through the action classifier. Then, this app detects the action of "Upstairs", which may cause an accident due to falling down. In the distributed manner, this app sends a command to turn on the camera to monitor the activity of the person going upstairs. It also activates the app of fall-down detection. If the fall classifier detect that a person falls, it then sends a command to the nearest healthcare to take necessary actions. In the centralized mechanism, the cloud is responsible for fall-down detection and there is only app of action classifier at

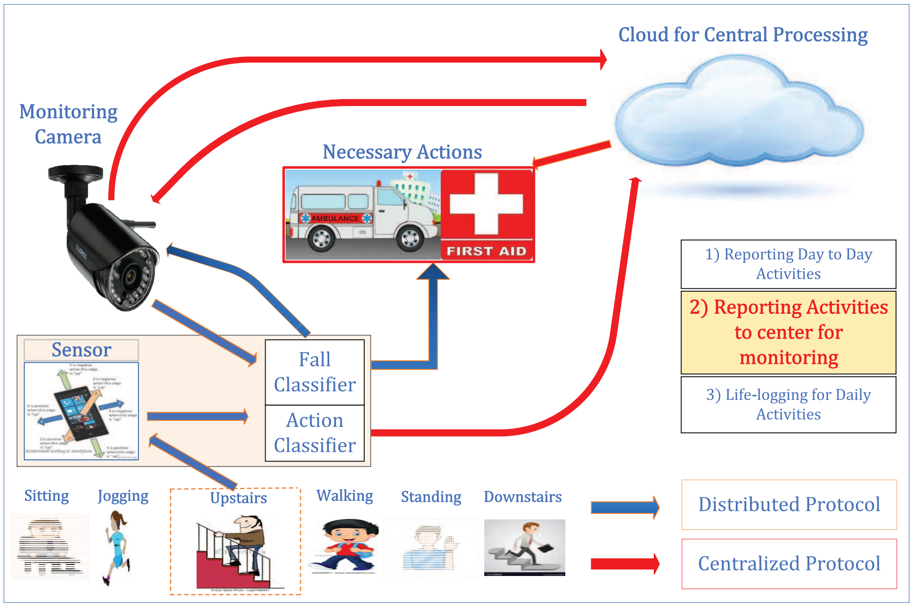

Fig. 2. Real Application in the Healthcare System using Human Activity Classification

the wearable sensor device. In this centralized manner, the action classifier sends the detected action of "Upstairs" to the cloud. The cloud concurrently send the command to activate the camera and start to monitor the person's activity. If the cloud detects the falling, it will call to the healthcare for assistance. Similarly, we have the same procedure with two protocols for the other applications. Detailed operations of the healthcare system using the action classifier is illustrated in Fig. 2

3.2 Sensor-based Human Activity Recognition

We consider the recognition of human activities, where the sensor like the accelerometer is used to record the time-series input. This time-series input of different human actions can be represented as a three dimensional image of the dimension $[sample\,size, window\,length, axis]$.

We employ the latent variable generative model, i.e. VAE, for learning, which is a combination of Variational Inference and Deep Learning. It differs from the conventional auto-encoder in two key ways. Firstly, a deterministic latent representation z is replaced by a posterior distribution $q(z|x)$, which helps a decoder to reconstruct an input by sampling z from this posterior. Secondly, the posterior $q(z|x)$ is regularized with Kullback–Leibler (KL) divergence from a prior distribution $p(z)$, where the prior is conventionally chosen to be a Gaussian with zero mean and unit variance. By doing so, we make sure that the system can sample from any point in latent dimensions and still generate valid and diverse outputs.

Let us present the notable contribution of VAE. Firstly, VAE performs a good generalization on reconstructing the approximated input by imposing some restrictions on the latent space. Specifically, by sampling from $q(z|x)$ instead of taking a deterministic z, it forces the model to map an input to a region of space rather than to a single input. It implies that VAE can acquire the salient features of the data, which support for the data reconstruction. Furthermore, the good reconstruction performance depends on estimating a very sharp probability distribution corresponding to a single input in latent space. The KL term in VAE loss function prevents this behavior by training the model to find a solution for low reconstruction error and predicting a posterior distribution close to the prior. In this process, the decoder part of the VAE is capable of reconstructing the sensor data samples from every point in the latent space.

We employ the semi-supervised classification to train the model, where the time-series data are both labeled and unlabeled data. For the unlabeled data, two networks has been trained. The first one is the encoder network $q_\phi(z|X_U)$, where its output is z; while the other is classifier network $q_\phi(y|X_U)$, where its output is y. For our model, both encoder and classifier networks employ the same traditional CNN network. According to the encoder output z and the label y, the decoder would perform the variational inference over them by $p_\theta(\hat{X_U}|z, y)$ and try to approximate the input as likely as possible. For the labeled data, we only train the encoder network $q_\phi(z|X_L)$. Then, we insert its label to the decoder for decoding data. Again, we perform the variational inference over the encoder output and the label by $p_\theta(\hat{X_L}, \hat{Y_L}|z)$ and obtain the reconstructed data. To evaluate the performance, we use the loss function, given as

$$\mathcal{J}_{VAE} = KL(q(z|x)||p(z)) - \mathbb{E}_{q(z|x)}[\log p(x|z)]. \tag{1}$$

Detailed semi-supervised approach for analyzing this loss function would be described in the next section. After a proper training, we store the configured parameters of the classifier $q_\phi(y|x)$ and use them to test the real time-series input. In particular, we implement the classifier module, which takes the raw time-series input from a wearable of an object and classifies a numerous unseen variation of some specific actions.

4 Semi-supervised Learning for Time-series Human Actions in the Smart Healthcare System

In this section, we provide a comprehensive mathematical formulation for all the core parts of our proposed model, i.e. encoder network, decoder network, classifier network and training model with semi-supervised objective. Before that we present all notations and their descriptions, which are summarized in the Table 1.

Note that we would have the labeled and unlabeled time-series data, which are used to train the model for our proposed semi-supervised classification. For

Table 1. Definition of Parameters

Notation	Description
$\circledast$	Convolution Operator
$\sum$	Concatenation of feature maps
$(F_E)^i$	Encoder Filter for layer i
$(F_D)^i$	Decoder Filter for layer i
$M_1 = 1, 2...m_1$	Total filters in layer 1
$M_2 = 1, 2...m_2$	total filters in layer 2
c	Convolved output
c_p	Convolved output after pooling
vec	Vectorize a concatenation
MLP	Multi Layer Perceptron
FC	Fully Connected Layer
π	Labels in Categorical Distribution
$q_\phi(z\|x)$	Probability of observing z given x
$q_\phi(y\|x)$	Probability of observing y given x
$p_\theta(\hat{x}\,\|z\,)$	Probability of observing x given z
$q_\phi(y\|x)$	Classifier Network
ϕ	Weight vector of Encoder and Classifier
θ	Weight vector of Decoder

the labeled data, we have the inputs, $\mathbf{X}_L = \{x_1, x_2, \ldots, x_L\}$ and their corresponding labels, $\mathbf{Y}_L = y_1, y_2.....y_L$, where L is the length of labeled data; while we only have the inputs, $\mathbf{X}_u = \{x_{l+1}, x_{l+2}, \ldots, x_{l+U}\}$ for the unlabeled data, where U is the length of the unlabeled data. For each data point x_i, there are corresponding latent variables z_i, which are obtained from an encoder structure and are then forwarded to a decoder. At the decoder side, we reconstruct an approximation of the given input of $\hat{\mathbf{X}}_L, \hat{\mathbf{Y}}_L$ and $\hat{\mathbf{X}}_U$ for labeled and unlabeled data, respectively. We train the parameters of ϕ and θ to enhance the overall performance of the model. Recall that ϕ is the weight vector of the encoder and classifier, while θ is the weight vector of the decoder. In the following, we present detailed description of encoder structure, decoder structure, classifier structure and training of the model.

4.1 Encoder Structure

The input data of our proposed encoder model can be recorded from the sensor devices. Usually, human activity data are observed by the tri-axis accelerometer. So the acceleration data are three-dimension (3D) data, which include x, y and z directions. In our encoder convolutional network, there are two kinds of layer, i.e. the convolutional layer and the fully connected layer. To reduce computational complexity, we aim for using the depth-wise separable convolution or 1D convolution [18]. Unlike the traditional CNN, the final layer of encoder in our proposed scheme is replaced by the fully connected layer. In particular, it can output the mean and standard deviation for the distribution of latent code. Furthermore, it can continuously generate the input of the decoder section by sampling the code.

As the observation above, our accelerator input data are the 3D time-series signal, which include x, y and z directions. Therefore, our input data can be categorized as the red-green-blue (RGB) image, which would have been treated successfully by using the conventional CNN. We describe the detailed implementation as follows. WLOG, the dimension of an input time-series signal, D^n is assumed as $(1, T, 3)$, where T is total points of time, the height is 1 and the depth is 3 for x, y and z directions. In the first step, we perform an $1D$ convolution by convolving a single filter m_1 with the input. This operation thus produces a 2D matrix, $\left\{c^{1,n,m_1}\right\}$ of $1 \times T$ dimensions, where $m_1 \in [1, 2, \ldots, M_1]$ and M_1 is the number of filters. Then, M_1 filters produce a concatenation of M_1 parameters of c^{1,n,m_1}, which represent all the feature maps of layer 1. We aim for using multiple filters to learn the great variety of patterns. So for a single filter m_1, c^{1,n,m_1} is derived as

$$c^{(1,n,m_1)} = D^{(n)} \circledast F_E^{1,m_1},\ m_1 \in [1, 2, \ldots, M_1]\,, \tag{2}$$

where F_E^{1,m_1} is the coefficients of the filter m_1. For M_1 filters, the concatenation parameter of $c^{(1,n)}$ is calculated as

$$c^{(1,n)} = \sum_{m_1=1}^{M_1} c^{(1,n,m_1)}. \tag{3}$$

Similarly, we perform another $1D$ convolution in layer 2 and concatenate all the slices. We have

$$c^{(2,n,m_2)} = c^{(1,n)} \circledast F_E^{2,m_2}, \tag{4}$$

$$c^{(2,n)} = \sum_{m_2=1}^{M_2} c^{(2,n,m_2)}, \tag{5}$$

$$C_f = vecc^{(2,n)}, \tag{6}$$

where C_f is the vectorized form of $c^{2,n}$, which has the dimension of $\left[1 \times dim(c^{2,n})\right]$. So C_f is gone through the fully-connected layer, which is the design of multi layer perceptron. Its output is given as $FC_1 = MLP(C_f)$, where MLP represents the multi layer perceptron. These fixed latent dimensions of the fully connected layers are then used as the input at the final layer. The outputs are the vector of mean and the vector of standard deviation, $z_n \sim \Big(\mu_\phi(FC_1), \sigma_\phi(FC_1)\Big)$, which would be used to sample the latent code for the decoder network.

4.2 Decoder Structure

We consider the decoder, which consists of three transposed convolutional layers and three fully connected layers. The implementation of our decoder network is presented as follows. The latent code at a fully connected layer is forwarded to the decoder. The result is the vectorized/flatten form of convolved output at layer 2 in the decoder and is given as

$$vec\ c^{(2,n)} = FC(z_n), \tag{7}$$

$$C_f = vec\ c^{(2,n)}, \tag{8}$$

where FC represents the fully connected layer. This flatten output is gone through the three transposed convolution layers. The outputs at three layers are presented as

$$c^{(2,n)} = C_f \circledast {F_D}^2, \tag{9}$$

$$c^{(3,n)} = c^{(2,n)} \circledast {F_D}^3, \tag{10}$$

$$c^{(4,n)} = c^{(3,n)} \circledast {F_D}^4, \tag{11}$$

where F_D^i represents the coefficient vector of the transposed convolutional filter at layer i, where $i \in \{2, 3, 4\}$. The convolved output are then gone through the other two fully connected layers to reconstruct the approximate version of the input. It implies that the reconstructed data D_{approx} would be given as

$$D_{FC_2} = FC(c^{(4,n)}), \tag{12}$$

$$D_{approx} = FC(D_{FC_2}). \tag{13}$$

4.3 Classifier Structure

We model the classifier structure by using the traditional CNN network, which consists of the convolutional layers, the max-pooling layers, the fully connected layers and the softmax classifier. The last layer, i.e. the softmax classifier, is responsible for providing the probability distribution of the class (or the label). For example, we present our dataset at the numerical results, which has six daily activities, namely "Walking", "Jogging", "Stairs", "Sitting", "Standing" and "Lying Down". We pass the unlabeled time-series data into two 1D convolutional layers (i.e. layers 1 and 2), one 1D max-pooling layer (i.e. layer 3) and three fully connected layers (i.e. layers 4, 5 and 6) and the softmax classifier. In the following, max-pooling operation is down-sampling an input representation to reduce its dimensionality and to allow for assumptions to be made about features contained in the sub-regions. Furthermore, softmax classifier takes a vector of arbitrary real-valued scores form the last layer (c^6 for our case) and squashes it to a vector of values between zero and one, where the summation of all elements equals one. Given unlabeled time-series data x_u, these operations in the classifier network are summarized as

$$
\begin{aligned}
&C^1 = x_u \circledast {F_c}^1, && (14)\\
&C^2 \sim maxpool(C^1), && (15)\\
&C^3 = C^2 \circledast {F_c}^3, && (16)\\
&C^4 = FC(C^3), && (17)\\
&C^5 = FC(C^4), && (18)\\
&C^6 = FC(C^5), && (19)\\
&f_{c_i} = (C^6)_i, && (20)\\
&q_\phi(y|x_u) = -log \frac{e^{f_{c_i}}}{\sum_{i=1}^{dim(C)} e^{f_{c_i}}}, && (21)
\end{aligned}
$$

where f_{c_i} is the i^{th} element of last layer, C^6, $dim(c)$ denotes the total elements of layer C^6, C^i, $i \in \{1, 2, \ldots, 6\}$ are the output of layer i, while $q_\phi(y|x_u)$ is the probability distribution of label y given the unlabeled time-series data x_u.

4.4 Training of Semi-supervised Model for Time-Series Classification

We now present the training steps of the semi-supervised model for both two variational objectives [16], which correspond to the cases of unlabeled data and labeled data.

Variational Objective with Unlabeled Data In the case of unlabeled data, we treat z and y as latent variables. Therefore, they are considered as the inputs of the decoder by using the operation of concatenation. It means that we train for both the encoder and classifier networks.

$$
\begin{aligned}
&y = Classifier(x_u), && (22)\\
&z = Encoder(x_u), && (23)\\
&Concatenation(z, y) \rightarrow Decoder. && (24)
\end{aligned}
$$

Thus, our variational objective for unlabeled data is

$$\min_{\theta,\phi} \quad KL(q_\phi(y, z|x) \mid\mid p_\theta(y, z|x)). \tag{25}$$

Here $q_\phi(y, z|x)$ and $p_\theta(y, z|x)$ denotes the approximate posterior distribution of y, z given x and corresponding true distribution respectively.

Proposition 1 *We have the final lower bound for unlabeled data as* $\log\ p_\theta(x) \geq U(x)$*, where the lower bound,* $U(x)$*, is determined as*

$$U(x) = \mathbb{E}_{q_\phi(y|x)} \left[-\mathcal{M}_{(x,y)} - log\ q_\phi(y|x) \right]. \tag{26}$$

Here, $\mathcal{M}_{(x,y)} = -\mathbb{E}_{q_\phi(z|x)}[\log\ p_\theta(x|y,z)] + K_1 - KL(q_\phi(z|x)\ ||\ p_\theta(z))$ *and* $K_1 = \mathbb{E}_{q_\phi(z|x)} \log\ p_\theta(y)$*.*

Proof. The proof is provided in Appendix 7.1.

Variational Objective with Labeled Data In the case of labeled data, we skip the classifier because we can inject the label from outside. Hence, we use the concatenation of z_L and y_L and then pass the result to the decoder. The procedure is summarized as

$$y_L = \ GivenLabels\ , \tag{27}$$

$$z = Encoder(x_L), \tag{28}$$

$$Concatenation\ (z, y_L) \rightarrow Decoder, \tag{29}$$

where $q_\phi(z,\pi|x,y)$ and $p_\theta(z,\pi|x,y)$ denote by the approximate posterior distribution of (z,π) given (x,y) and its corresponding true distribution, respectively. Thus, the variational objective for labeled data is

$$\min_{\theta,\phi} \quad KL\ q_\phi(z,\pi|x,y)\ ||\ p_\theta(z,\pi|x,y), \tag{30}$$

where $p_\theta(x,y|z,\pi)$ denotes the decoder probability of generating x and y based on the observation z and π, while $q_\phi(z,\pi|x,y)$ represents the probability of correct encoding of z and π given the measurements x and y.

Proposition 2 *We have the final lower bound for labeled data as* $\log p_\theta(x,y) \geq L(x,y)$*, where the lower bound,* $L(x,y)$*, is determined as*

$$L(x,y) = -\mathcal{M}_{(x,y)} - KL\,[\ q_\phi(\pi|x)||p_\theta(\pi|y)]\,. \tag{31}$$

Here, $\mathcal{M}_{(x,y)} = -\mathbb{E}_{q_\phi(z|x)}[\log\ p_\theta(x|y,z)] + K_1 - KL(q_\phi(z|x)\ ||\ p_\theta(z))$ *and* $K_1 = \mathbb{E}_{q_\phi(z|x)} \log\ p_\theta(y)$*, while the Kullback-Leibler divergence function is* $KL\,[\ q_\phi(\pi|x)||p_\theta(\pi|y)] = \mathbb{E}_{q_\phi(\pi|x)}[-\log\ p_\theta(\pi,|y) + \log q_\phi(\pi|x)]$*.*

Proof. The proof is provided in Appendix 7.2.

The total VAE loss is computed by adding both the labeled loss and the unlabeled loss as

$$\mathcal{J} = \sum_{(x,y)\sim\hat{p_l}} L(x,y) + \sum_{(x)\sim\hat{p_u}} U(x). \tag{32}$$

As stated before, the whole point of semi-supervised learning is train encoder and classifier network at the same time. Hence, in spite of having labels (for labeled case), we need to train the classifier with labeled data too, if we want to use this distribution as a classifier. Thus, according to [16], a classification loss has been added with (32).

$$\mathcal{J}^\alpha = \mathcal{J} + \alpha * E_{(x,y)}[-\log q_\phi(y|x)]. \tag{33}$$

In the above function α is an hyperparameter which adjusts the contributions of the classifier in the learning process. During the training process, the stochastic gradient descent of $\mathcal{J}$ is computed at each training step to update the parameters θ and ϕ. We summarize the training procedure in Alg. 1.

Algorithm 1 Semi-supervised Learning for Time-series Classification in the Smart Healthcare System

```
Input: Labeled Data: $X_L, Y_L$, Unlabeled Data: $X_U$, Test Data: $X_T$
Output: Labels for Test Data
Get samples from Labeled & Unlabeled Data and initialize parameters $\phi$ and $\theta$
while Training do
    for Labeled Data: $X_L, Y_L$ do
        $X_L \rightarrow Encoder$
        Compute $z_n$
        Concatenate $z_n$ & $Y_L$
        $z_n, Y_L \rightarrow Decoder$
        Calculate $L(x, y)$
    end for
    for Unlabeled Data: $X_U$ do
        $X_U \rightarrow Encoder$
        Compute $z_n$
        $X_U \rightarrow Classifier$
        Compute $Y_U$
        Concatenate $z_n$ & $Y_U$
        $z_n, Y_U \rightarrow Decoder$
        Calculate $U(x)$
    end for
    Compute $\mathcal{J} = \sum_{(x,y)\sim\hat{p_l}} L(x, y) + \sum_{(x)\sim\hat{p_u}} U(x)$
    Update $\mathcal{J}$ with Adam Optimizer
end while
while Testing do
    Evaluate Model on Test data to calculate VAE loss
    Evaluate Classifier on Test data
    Evaluate Reconstruction loss by $L2norm$
end while
```

5 Experimental Analysis

For the experiment of our semi-supervised algorithm we selected publicly available Actitracker dataset [2], which contains six daily activities, i.e. walking, jogging, upstairs, downstairs, sitting, standing. This dataset is collected in an controlled laboratory environment, where it is recorded from 36 users. The dataset from these users is collected from their cellphone operating at 20Hz sampling rate, which results in 29,000 frames. In this analysis, we present our experimental results of training and testing data, where the evaluated performances are the VAE loss, the reconstruction loss and the classification loss. Our simulation model is based on 6-dimensional hidden variable z. We assume that an amount of labeled data equals that of unlabeled data. The representation of our model is summarized in Table 2.

5.1 VAE loss for training and test data

In Fig. 3, we present the VAE loss of training data for three different learning rates of 0.001, 0.0001 and 0.00001. Recall that loss function of VAE is composed of two terms, i.e. the reconstruction loss and the regularization term. Here, the reconstruction loss is the expected value of negative likelihood of all data points. This term encourages the decoder reconstruct our input data, whereas the regularization term is the measurement of information lost while approximating p with q. The VAE loss for training data is the addition from both the labeled loss and the unlabeled loss, whereas for test case, the input data is passed through the unlabeled structure to measure the loss. It is easily observed that as a hyper parameter, learning rate plays a crucial role in calculating the loss. Similarly, we test our model with the learning rates of 0.001, 0.0001 and 0.00001. In particular, the VAE loss of testing data is illustrated in Fig. 4.

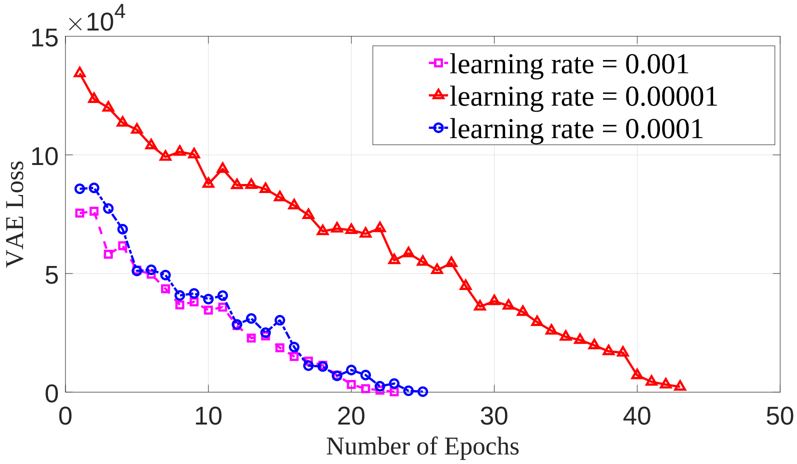

Fig. 3. VAE loss of training data for three different learning rates 0.001, 0.00001 and 0.0001

Table 2. Representation of our model

Encoder Structure	
Layer 1	60 *conv* 1*D filters*
Layer 2	180 *conv* 1*D filters*
Layer 3	*FC with* 120 *units*
Layer 4	*FC with* 90 *units*
Classifier Structure	
Layer 5	*FC with* 6 *units*
Layer 6	60 *conv* 1*D filters*
Layer 6	*max pool size* 20
Layer 7	180 *conv* 1*D filters*
Layer 8	*FC with* 100 *units*
Layer 9	*FC with* 100 *units*
Decoder Structure	
Layer 10	*FC with* 90 *units*
Layer 11	*FC with* 90 *units*
Layer 12	*FC with* 90 *units*
Layer 13	60 *conv* 2*D Transpose filters*
Layer 14	120 *conv* 2*D Transpose filters*
Layer 15	60 *conv* 2*D Transpose filters*
Layer 16	*FC with* 9 *units*

We have the following observations from these experimental results. Firstly, we train our model for both labeled and unlabeled data as well as experiment the efficacy of our model in terms of VAE loss on test data. It is readily seen that the VAE loss reaches the minimum value, when the number of epochs is nearly 50. During training, we test our model on testing data with the trained model parameters of ϕ and θ after every 5 epochs. The VAE losses from both training and testing data for learning rate of 0.00001 shows a gradual and slow decay. However, they ultimately reaches the considerably low losses, i.e. 215.314 and 121.457, respectively. For other two learning rates of 0.001 and 0.0001, the VAE losses start with the comparatively low loss. Then, they quickly reach the negative values before even reaching considerably low values. This is because 1) *the VAE excellently extract the most salient characteristics of human activity data, which generate the useful criteria for the compressed sensing recovery*; 2) *furthermore, our proposed CNN successfully extracts the most discriminative features, which provide the essential low-dimension latent codes.* So our joint design significantly improves training stability and efficiency, because it has capability of tuning the loss multipliers automatically. Additionally, our proposed mechanism makes better use of the training data and achieves a high accuracy even with a limited amount of labeled data available for training. Thank to the discriminative performances and the regularization effects, which enhance our proposed method on real dataset in the semi-supervised setting.

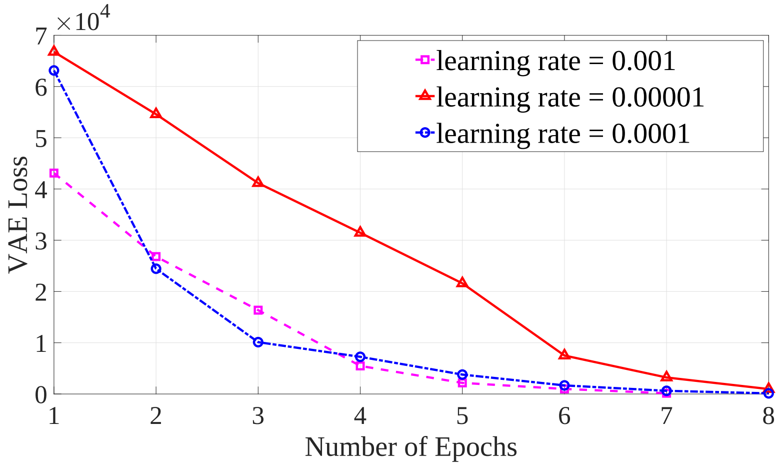

Fig. 4. VAE loss of testing data for three different learning rates 0.001, 0.00001 and 0.0001

5.2 Reconstruction Loss

One of the major objectives in our proposed model is to reconstruct the input time-series by performing semi-supervised learning. We evaluate the reconstruction loss based on *L*2 *norm*. If our true input and approximated input are defined as x_i and x_r respectively, then *L*2 *norm* for computing reconstruction loss can be defined as follows. For N samples, this quantity is calculated as

$$\mathcal{R}_{Loss} = \sum_{n=1}^{N} (x_i - x_r)^2. \tag{34}$$

We experiment the reconstruction loss on our train data. For test case, we consider data points x of both labeled and unlabeled data. Then, we compute the reconstruction loss of the final output by using *L*2 *norm*. Our analysis of reconstruction loss with two different learning rates 0.001 and 0.0001 for 500 epochs is provided in Fig. 5.

5.3 Classification Accuracy:

Finally, we present our comparison of the classification accuracy between our proposed model (termed as *Semi-supervised Learning*) and a benchmark supervised learning model trained by CNN (called as *Supervised Learning*). We train both the supervised learning model and semi-supervised learning model by using 20% and 25% labeled data. For a learning rate of 0.00001 and with 500 iterations our semi-supervised classifier outperforms the supervised one by 5.28%

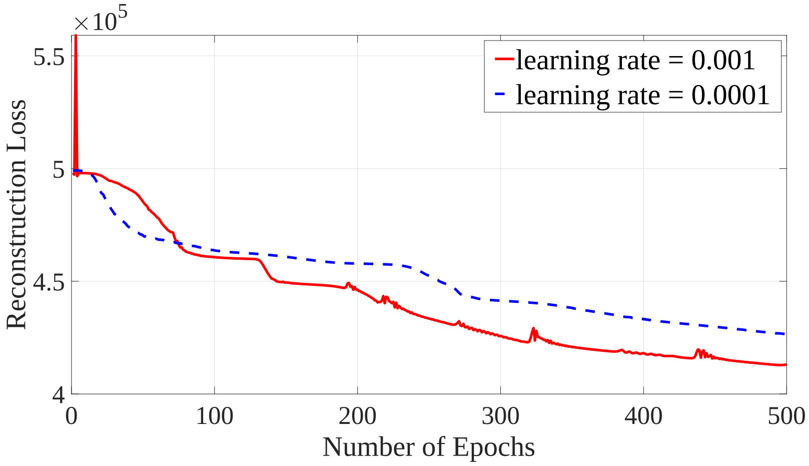

Fig. 5. Reconstruction loss of training data for two different learning rates 0.001 and 0.0001

and 5.15% respectively (Fig. 6 and Fig. 7). Moreover, it is easily observed that our proposed model has a higher convergence rate than the benchmark model. This observation confirms theoretical analysis that our proposed model does not typically require more epochs to reach convergence, because it only requires a small amount of data used for training. We also experimented with 30% labeled data but in that case we found no significant difference in classification accuracy after 500 iterations. Thus, we can rightfully conclude that, increasing number of labeled data will not make our semi-supervised model more constructive than the supervised one. It implies that our model can work efficiently with any scenario of big data applications, while the benchmark supervised learning model only achieves a good performance on small data. The desired experimental results demonstrate that our proposed model is benefited by the joint design of variational auto-encoder and convolutional neural network.

6 Conclusion

In this paper, we proposed a semi-supervised classifier which extracts salient and discriminative features to train a classifier in order to recognize numerous human actions. The experimental result has shown that with a small amount of labeled data our model outperforms the conventional supervised learning. For further study, experiments with larger datasets are needed to study the efficacy of the proposed model. We also aim to use fall action datasets in order to develop a fall classifier for a comprehensive smart home monitoring solution.

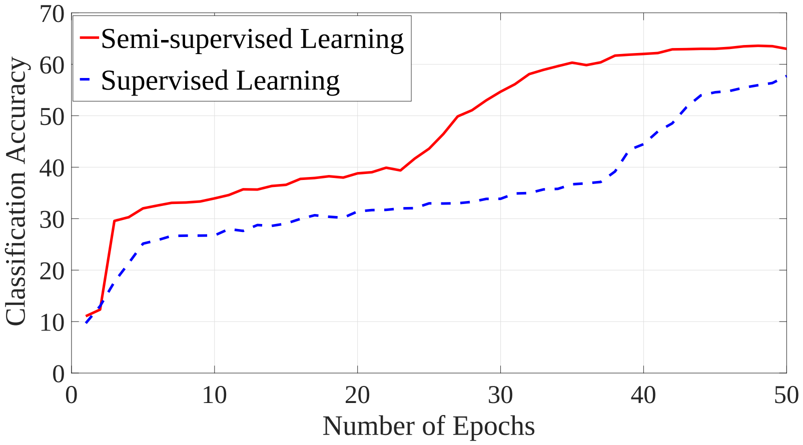

Fig. 6. Classification Comparison for 20% labeled data

7 APPENDIX

7.1 Variational Objective with Unlabeled Data

Recall that in the case of unlabeled data, we treat z and y as latent variables and we train a encoder and classifier network. Thus, our variational objective for unlabeled data is

$$\min_{\theta,\phi} \quad KL(q_\phi(y,z|x) \mid\mid p_\theta(y,z|x)),$$

where KL is the Kullback-Leibler divergence function between the two distributions, which can be obtained as $KL(q||p) = \int_{-\infty}^{+\infty} q_i log(q_i/p_i)$. So we have

$$\begin{aligned} KL(q_\phi(y,z|x) \mid\mid p_\theta(y,z|x)) &= \int_{-\infty}^{+\infty} q_\phi(y,z|x)) \log \frac{q_\phi(y,z|x)}{p_\theta(y,z|x)}\, dy\, dz \\ &= \mathbb{E}_{q_\phi(y,z|x)} \left[\log \frac{q_\phi(y,z|x)}{p_\theta(y,z|x)} \right]. \end{aligned}$$

From Baye's Rule, we know that $p(y,z|x) = \left[\frac{p(x|y,z)p(y)p(z)}{p(x)}\right]$. Thus, we can write

$$\begin{aligned} KL(q_\phi(y,z|x) \mid\mid p_\theta(y,z|x)) = \mathbb{E}_{q_\phi(y,z|x)} & [\log q_\phi(y,z|x) - \log p_\theta(x|y,z) \\ & - \log p_\theta(y) - \log p_\theta(z) + \log p_\theta(x)]. \end{aligned}$$

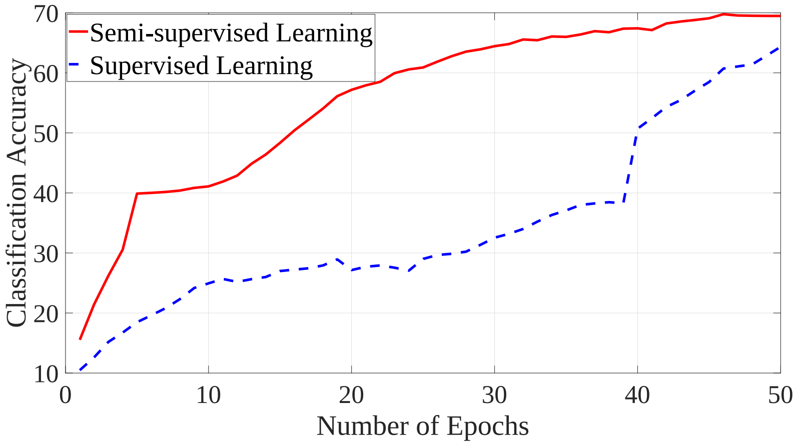

Fig. 7. Classification Comparison for 25% labeled data

So we obtain

$$\log p_\theta(x) = KL(q_\phi(y,z|x)||p_\theta(y,z|x)) \\ +E_{q_\phi(y,z|x)}\left[\log p_\theta(y) + \log p_\theta(x|y,z) - \log q_\phi(y,z|x) + \log p_\theta(z)\right] \quad (35)$$

As KL-divergence describes the similarity between two distributions, it is always a non-negative term. So, we characterize $\log p_\theta(x)$ in (35) as follows:

$$\begin{aligned} log\ p_\theta(x) &\geq \mathbb{E}_{q_\phi(y,z|x)}\left[\log \frac{p_\theta(x|y,z)p_\theta(y)p_\theta(z)}{q_\phi(y,z|x)}\right], \\ &\geq \mathbb{E}_{q_\phi(y,z|x)}\left[\log\ p_\theta(x|y,z) + \log\ p_\theta(y) + \log\ p_\theta(z)\right. \\ &\qquad\qquad \left. -log\ q_\phi(y,z|x)\right], \\ &\geq \mathbb{E}_{q_\phi(y|x)}\left[\mathbb{E}_{q_\phi(z|x)}\left[\log\ p_\theta(x|y,z) + \log\ p_\theta(y)\right.\right. \\ &\qquad \left.\left. +log\ p_\theta(z) - log\ q_\phi(y|x) - log\ q_\phi(z|x)\right]\right]. \end{aligned}$$

Finally, we obtain

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(y|x)}\left[\mathbb{E}_{q_\phi(z|x)}\left[\log\ p_\theta(x|y,z)\right] + K_1 \right. \\ \left. -KL(q_\phi(z|x)\ ||\ p_\theta(z)) - \log\ q_\phi(y|x)\right]. \quad (36)$$

where $K_1 = \mathbb{E}_{q_\phi(z|x)} \log p_\theta(y) = \log p_\theta(y)$. Here, $p_\theta(y)$ is considered as a constant. Because $p(y) = p(y|\pi)p(\pi)$ is a Dirichlet- multinomial distribution and for unlabeled case, we assume that each label y is equally likely.

We further define $\mathcal{M}_{(x,y)}$ as

$$-\mathcal{M}_{(x,y)} = \mathbb{E}_{q_\phi(z|x)} \left[\log\ p_\theta(x|y,z)\right] + K_1 - KL(q_\phi(z|x)\ ||\ p_\theta(z)).$$

Then, we get

$$log\ p_\theta(x) \geq \mathbb{E}_{q_\phi(y|x)} \left[-\mathcal{M}_{(x,y)} - log\ q_\phi(y|x)\right]. \tag{37}$$

So we explicitly write the expectation with respect to y. Hence, we get

$$log\ p_\theta(x) \geq \sum_y q_\phi(y|x) \left[-\mathcal{M}_{(x,y)} - log\ q_\phi(y|x)\right]. \tag{38}$$

Therefore, we have completed the proof of Proposition 1.

7.2 Variational Objective with Labeled Data

For the labeled case, both (x, y) and (z, π) are treated as unknown latent variables. The variational inferences for both z and π use a fully posterior dependent on only x. It means that we can derive $q(z, \pi)$ and $p(y)$ as follows:

$$\begin{aligned} q(z,\pi) &= q(z,\pi|x) = q(z|X) * q(\pi|X), \\ p(y) &= Cat(y|\pi). \end{aligned}$$

Here, "Cat" represents a categorical distribution, which is a discrete probability distribution. It describes the possible results of a random variable that can take one of π possible categories, with the probability of each category separately specified. For our case, $\pi = 6$ as we have six daily activities. Furthermore, α represents a hyperparameter, which controls the weight of how strongly we want to train our classifier. Note that in order to train a classifier for the labeled case, we also aim for choosing π such that it only depends on x. By doing so, this enforces the classifier to work as the conventional classifier. It means that the classifier network $q_\phi(y|x)$ will also classify the x_L and compare its performance with the true labels. So we have

$$\min\ KL\ q_\phi(z,\pi|x,y)\ ||\ p_\theta(z,\pi|x,y).$$

By exploiting Kullback-Leibler divergence function, we obtain

$$KL\ q_\phi(z,\pi|x,y)\ ||\ p_\theta(z,\pi|x,y) = \mathbb{E}_{q_\phi(z,\pi|x,y)} \left[log\ q_\phi(z,\pi|x,y)\ - log\ p_\theta(z,\pi|x,y)\right].$$

According to Baye's Rule, we have $p(z,\pi|x,y) = \left[\dfrac{p(x,y|z,\pi)p(z,\pi)}{p(x,y)}\right]$. Thus, we can write $KL\ q_\phi(z,\pi|x,y)\ ||\ p_\theta(z,\pi|x,y)$ as

$$\begin{aligned} KL\ q_\phi(z,\pi|x,y)\ ||\ p_\theta(z,\pi|x,y) = \mathbb{E}_{q_\phi(z,\pi|x,y)} \big[&log\ q_\phi(z,\pi|x,y)\ - log\ p_\theta(x,y|z,\pi) \\ &- \log\ p_\theta(z,\pi) + \log\ p_\theta(x,y)\big]. \end{aligned}$$

Using the simple manipulations, $\log p_\theta(x,y)$ can be calculated as

$$\begin{aligned}\log\ p_\theta(x,y) = {} & KL(\ q_\phi(z,\pi|x,y)||p_\theta(z,\pi|x,y)) \\ & + \mathbb{E}_{q_\phi(z,\pi|x,y)}\left[\log p_\theta(x,y|z,\pi) + \log p_\theta(z,\pi)\ - \log\ q_\phi(z,\pi|x,y)\right].\end{aligned}$$

Recall that KL-divergence describes the similarity between two distributions. Hence, it is always a positive term. So we characterize $\log\ p_\theta(x,y)$ in (39) as follows:

$$\log p_\theta(x,y) \geq \mathbb{E}_{q_\phi(z,\pi|x,y)}\left[\log\ p_\theta(x,y|z,\pi) + \log\ p_\theta(z,\pi) - \log\ q_\phi(z,\pi|x,y)\right].$$

According to the probability chain rule, we have $P(A,B|C,D) = P(A|B,C,D) * P(B|C,D)$. By employing the simple manipulations, we obtain

$$\begin{aligned}\log p_\theta(x,y) & \geq \mathbb{E}_{q_\phi(z,\pi|x,y)}\left[\log\ p_\theta(x,|y,z,\pi) + \log\ p_\theta(y|z,\pi)\right. \\ & \qquad \left. + \log\ p_\theta(z,\pi) - \log\ q_\phi(z,\pi|x,y)\right], \\ & \geq \mathbb{E}_{q_\phi(z,\pi|x,y)}\left[\log\ p_\theta(x,|y,z,\pi) + \log\ p_\theta(y,z,\pi)\right. \\ & \qquad \left. - \log\ q_\phi(z,\pi|x,y)\right], \\ & \geq \mathbb{E}_{q_\phi(z,\pi|x,y)}\left[\log\ p_\theta(x,|y,z,\pi) + \log\ p_\theta(y,\pi)\right. \\ & \qquad \left. + \log\ p_\theta(z) - \log\ q_\phi(z,\pi|x,y)\right], \\ & \geq \mathbb{E}_{q_\phi(z,\pi|x,y)}\left[\log\ p_\theta(x,|y,z,\pi) + \log\ p_\theta(\pi|y)\right. \\ & \qquad \left. + \log\ p_\theta(y) + \log\ p_\theta(z) - \log\ q_\phi(z,\pi|x,y)\right].\end{aligned}$$

We now separate $\mathbb{E}_{q_\phi(z|x)}$ and $\mathbb{E}_{q_\phi(\pi|x)}$. This is because for labeled case, $z|x$ and $\pi|x$ are two different networks and thus they are independent. We then obtain

$$\begin{aligned}\log p_\theta(x,y) \geq {} & \mathbb{E}_{q_\phi(z|x)}\left[\log\ p_\theta(x|y,z) + \log\ p_\theta(y) + \log\ p_\theta(z)\right. \\ & \left. - \log q_\phi(z|x)\right] + \mathbb{E}_{q_\phi(\pi|x)}\left[\log p_\theta(\pi|y) - \log q_\phi(\pi|x)\right].\end{aligned}$$

From (37), the first expectation at the RHS is exactly equal to $\mathcal{M}_{(x,y)}$. So we can write

$$\log p_\theta(x,y) \geq -\mathcal{M}_{(x,y)} - \mathbb{E}_{q_\phi(\pi|x)}\left[-\log\ p_\theta(\pi,|y) + \log q_\phi(\pi|x)\right].$$

Note that the remaining expectation is the KL divergence form. Finally, the labeled loss $\log p_\theta(x,y)$ has the lower bound as

$$\log p_\theta(x,y) \geq -\mathcal{M}_{(x,y)} - KL(\ q_\phi(\pi|x)\ ||\ p_\theta(\pi|y)). \tag{39}$$

Hence, we have completed the proof of Proposition 2.

Acknowledgment

The research of A. Zahin, L. T. Tan and R. Q. Hu was supported in part by National Science Foundation under grants NeTS 1423348 and EARS 1547312, in part by Natural Science Foundation of China under grant 61728104, and in part by Intel Corporation.

References

1. S. Chennuru, P. -W. Chen, J. Zhu, and J. Y. Zhang, "Mobile lifelogger-recording, indexing, and understanding a mobile users life," In *Proc. ICMCAS, 2010.*
2. K. Forster, D. Roggen, and G. Troster, "Unsupervised classifier self-calibration through repeated context occurrences: is there robustness against sensor displacement to gain?," In *Proc. ISWC, 2009.*
3. P. Wu, J. Zhu, and J. Y. Zhang, "Mobisens: A versatile mobile sensing platform for real-world applications," *Mobile Networks and Applications*, vol 18, no. 1, pp. 60-80, Feb. 2013
4. M. Zhang and A. A. Sawchuk, "A feature selection-based framework for human activity recognition using wearable multimodal sensors," In *Proc. ICST, 2011.*
5. E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Trans. Info. Theo.*, vol. 52, no. 12, pp. 5406-5425, Dec. 2006.
6. T. L. Thanh, K. H. Yun and B. V. N. Quoc, "Projected Barzilai-Borwein methods applied to distributed compressive spectrum sensing," in *Proc. IEEE DySPAN 2010.*
7. L. T. Tan and H. Y. Kong, "A novel and efficient mixed-signal compressed sensing for wide-band cognitive radio," in *Proc. IFOST 2010.*
8. T. T. Le and H. Y. Kong, "Using subspace pursuit algorithm to improve performance of the distributed compressive wide-band spectrum sensing," *J Electromagn Eng Sci*, vol. 11, no. 4, pp. 250-256, Dec. 2011.
9. L. T. Tan and L. B. Le, "Compressed sensing based data processing and MAC protocol design for smartgrids," in *Proc. IEEE WCNC 2015.*
10. L. T. Tan, and L. B. Le, "Joint data compression and MAC protocol design for smartgrids with renewable energy," *Wirel. Commun. Mob. Com.*, vol.16, no. 16, pp. 2590-2604, July 2016.
11. L. T. Tan, R. Q. Hu and L. Hanzo, "Twin-timescale artificial intelligence aided mobility-aware edge caching and computing in vehicular networks," *IEEE Trans. Veh. Tech.*, to appear, 2019.
12. L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing framework in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Tech.*, vol. 67, no. 11, pp. 10190-10203, Nov. 2018
13. Z. Wang, and T. Oates, "Encoding time-series as images for visual inspection and classification using tiled convolutional neural networks," In *Proc. AAAI 2011.*
14. M. Dhar, A. Grover and S. Ermon, "Modeling sparse deviations for compressed sensing using generative models," *Proc. ICML 2018.*
15. M. Mardani et al., "Deep generative adversarial neural networks for compressive sensing MRI," in *IEEE Trans. Medical Imaging*, vol. 38, no. 1, pp. 167-179, Jan. 2019.

16. D. P. Kingma, S. Mohamed, D. J. Rezende, M. Welling, "Semi-supervised learning with deep generative models," *Proc. Adv. Neural Inf. Process. Syst 2014*.
17. J. W. Lockhart, G. M. Weiss, J. C. Xue, S. T. Gallagher, A. B. Grosner, and T. T. Pulickal, "Design considerations for the wisdm smart phone-based sensor mining architecture," In *Proc. Fifth IWK DSD ACM 2011*.
18. T. Plotz, N. Y. Hammerla, and P. Olivier, "Feature learning for activity recognition in ubiquitous computing," In *Proc. Twenty-Second IJCAI 2011*.
19. G. Hinton, S. Osindero, and Y. -W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol 18, no. 7, pp. 1527-1554, Jul. 2006.
20. T. Joachims, "Transductive inference for text classification using support vector machines," In *Proc. ICML 1999*.
21. D. Figo, P. C. Diniz, D. R. Ferreira, and J. M. Cardoso, "Preprocessing techniques for context recognition from accelerometer data," *Personal and Ubiquitous Computing* , vol. 4, no. 7, pp. 645-662, Oct. 2010.
22. T. Tamura, M. Sekine, M. Ogawa, T. Togawa, and Y. Fukui. "Classification of acceleration waveforms during walking by wavelet transform," *Methods of information in medicine*, vol. 36, no. 4-5, pp. 356-359, Oct. 1997
23. T. T. Le and H. Y. Kong, "Performance of spiked population models for spectrum sensing," *J Electromagn Eng Sci*, vol. 12, no. 3, pp. 203-209, Sept. 2012.
24. C. Vollmer, H. -M. Gross, and J. P. Eggert. "Learning features for activity recognition with shift-invariant sparse coding," In *Proc. ICANN 2013*.
25. A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," In *Proc. ICML 2001*, pp. 19-26.
26. M. Cote and H. Larochelle, "An Infinite Restricted Boltzmann Machine,"*Neural computation*, vol. 28, no. 7, pp. 1265-1288, Jul. 2016.
27. M. Zeng et al., "Convolutional neural networks for human activity recognition using mobile sensors," In *Proc. MobiCASE 2014*.