# SUPERVISED AND SEMI-SUPERVISED DEEP LEARNING-BASED MODELS FOR INDOOR LOCATION PREDICTION AND RECOGNITION

*Weizhu Qian, Fabrice Lauri, Franck Gechter*

CIAD, Univ. Bourgogne Franche-Comté
UTBM, F-90010, Belfort, France



## ABSTRACT

Predicting smartphone users location with WiFi fingerprints has been a popular research topic recently. In this work, we propose two novel deep learning-based models, the convolutional mixture density recurrent neural network and the VAE-based semi-supervised learning model. The convolutional mixture density recurrent neural network is designed for path prediction, in which the advantages of convolutional neural networks, recurrent neural networks and mixture density networks are combined. Further, since most of real-world datasets are not labeled, we devise the VAE-based model for the semi-supervised learning tasks. In order to test the proposed models, we conduct the validation experiments on the real-world datasets. The final results verify the effectiveness of our approaches and show the superiority over other existing methods.

***Index Terms***— Mixture density network, variational autoencoder, semi-supervised learning, WiFi fingerprint, indoor positioning

## 1. INTRODUCTION

Location based services (LBS) are essential for applications like location-based advertising, outdoor/indoor navigation and social networking, etc. With the help of significant advancement of the smartphone technology in recent decades, smartphone devices are integrated with various built-in sensors, such as GPS modules, WiFi modules, cellular modules, etc. Acquiring the data from such kinds of sensors enables researchers to study human activities.

There are several types of data can be utilised for such research purpose. For instance, GPS equipment can provide relative accurate position information with latitudes an longitudes directly when the smartphone users stay outdoors, the GPS-based methods are favored by many researchers [1], [2]. However, this type of methods are not suitable for indoor positioning tasks because GPS coordinates are no longer available if the users stay indoors.

On such occasion, we have to utilize other indirect data to interpret the location information. Since WiFi connections are widely used nowadays, one frequently used approach is to make use of the detected WiFi fingerprints of smartphone devices. In this case, the received signal strength indicator (RSSI) values of WiFi access points (WAPs) scanned by the mobile phones, instead of latitudes and longitudes, are used to compute the locations of the users. Compared to the GPS-based methods, the WiFi fingerprints-based localization method not only can function indoors but also are less energy-consuming.

In this work, we attempt to interpret WiFi fingerprints (RSSI values) into accurate users location (coordinates). However, this problem is not easy to solve. Typically, the RSSI vaule data are vectors whose elements represent the unique WAP IDs. To provide the good quality of WiFi connection, modern public buildings are equipped with a relatively large number of WiFi access points. As a result, this leads to the high dimensionality problem. Another issue is that the RSSI values are not always stable due to the signal-fading effect and the multi-path effect [3]. According to our investigation, we find that an ordinary deep-learning regressor with euclidean distance-based loss is not powerful enough to overcome such difficulties.

In order to solve these problems, we propose two deep learning-based models in this work. The first proposed model is called the convolutional mixture density recurrent neural network (CMDRNN). CMDRNN is designed to predict the user paths. In the CMDRNN model, to address the high dimensionlity issue, we deploy an one-dimensional convolutional neural network as a sub-structure to detect the feature of the input. In CMDRNN, in order to overcome the the instability problem of the data, a mixture density network substructure is incorporated into our model for calculating the final output. Meanwhile, since our task is a time-series prediction, we model the state transition via a recurrent neural network sub-structure. With such unique design, the CMDRNN model is able to predict user location with WiFi fingerprints.

As we know, labelling data usually is time-consuming and labor-consuming, thus most of real-word data in fact is unlabeled. However, even so, we still want to make as much use of accessible data as possible. For doing this, we proposed the second deep learning-based model, the VAE-based semi-supervised learning model. In this approach, we assume that

Email: weizhu.qian@utbm.fr

the input (RSSI values) and the target (user location) are governed by the same latent distribution. therefore, in the unsupervised learning process, we use the variation auto-encoder model to learn the latent distribution of the input whose information is relatively abundant. Then, in the supervised learning process, the labeled data is used to train the predictor. In this way, we can exploit more information of the dataset than other supervised learning methods.

The main contributions of our work are summarized as follows.

- We devise an novel hybrid deep-learning model (CM-DRNN) which allow us to predict the accurate position of the smartphone users based on detected WiFi fingerprints.
- We devised the VAE-based deep-learning models to perform semi-supervised learning for accurate indoor positioning.
- We conduct the evaluation experiments on the real-world datasets and compare our methods with other deep learning methods.

The reminder of the paper is organized as follows. Section 2 surveys the related work. Section 3 addresses the problem we will solve in this paper. In Section 4, the proposed method is introduced. Section 5 presents the validation experiments and the results with the real user data. Finally, we draw the conclusions and discuss about the potential future work in Section 6.

## 2. RELATED WORK

In literature, researchers have explored various types of machine learning techniques, both conventional machine learning and deep learning methods, on location recognition and prediction with WiFi fingerprints data.

### 2.1. Conventional machine learning methods

In the work of [4], the researchers compared many traditional machine learning methods, decision trees (DT), K-nearest neighbors (KNN), naive Bayes (NB), neural networks (NN), etc., for classifying buildings, floors and regions. In [5], the authors clustered the 3D coordinates data by K-means and clustered the RSSI data by the affinity clustering algorithm, respectively. Specially, in [6], the researchers compared $51$ distance metrics to investigate the most suitable distance functions for accurate WiFi-based indoor localization. Some researchers used Gaussian processes (GPs) [7] to model the relationship between WiFi signal strengths and the indoor locations [8], [9], [10]. Whereas GPs are not scalable to large datasets due to the expensive computational cost.

### 2.2. Deep learning methods

Deep learning methods, such as convolutional neural networks (CNNs) [11], auto-encoders (AEs) [12] and recurrent neural networks (RNNs) also have been utilized in WiFi-based positioning tasks. [13] used the CNN model for time-series analysis. Generally, a buildings has many different WiFi access points, thus the RSSI data in many situations, can be very high dimensional. For this reason, it is reasonable to reduce the data dimension before carrying out a regression or a classification task. Some deep learning-based dimension-reduction methods like auto-encoders can be an appropriate choice [14], [15], [16]. For example, in the research of [15], the authors used an auto-encoder network to reduce the data dimension, then used a CNN to proceed accurate user positioning. In [14], [16], the researchers used auto-encoders to reduce the input dimension before using a multi-layer perceptron (MLP) to classify the buildings and floors.

For time series predictions, there are two typed of applicable deep architectures, CNNs and RNNs. In [3], the authors compared different types Recurrent Neural Networks including vanilla RNN, long short-term memory (LSTM), gated recurrent units (GRU) and bidirectional LSTM for accurate RSSI indoor localization. And they emplyed a weighted filter for both input and output to enhance the sequential modeling accuracy.

### 2.3. Limitation of convectional neural networks

Traditional neural networks (NNs) can be regarded as deterministic models, which can be described as follow.

$$y = \mathcal{F}(x; w) \tag{1}$$

where, $x$ and $y$ are the input and output of the NN, respectively. $F$ represents the neural network structure and $w$ is the wight of the NN.

Accordingly, the training loss (for instance, typically, mean squared errors) of NNs can be described as follow.

$$loss = \frac{1}{N}\sum_{n=1}^{N}(\hat{y_n} - y_n)^2 \tag{2}$$

where, $N$ is the total number of the input and $\hat{y}$ is the model target.

In many situations, a NN model is powerful enough to obtain satisfying results. However, in some cases, for instance, a high non-Gaussian inverse problem, the traditional neural networks will lead to very poor modeling results[17]. A good solution to this issue is to seek a framework that can calculate conditional probability distributions.

Mixture density networks (MDNs) solve this problem by using maximum likelihood estimation (MLE) [18]. In MDNs, the final output is sampled by a mixture distribution rather than computed directly. One advantage of MDNs is that it

can be applied to an estimation situation in which a large variety lies. For instance, we can incorporate more mixture of Gaussians to a MDN to enhance its estimating capacity for more complex distributions. However, MLE also has obvious disadvantages. First, it needs to set some hyper-parameters properly (i.e., the mixture number of a MDN), otherwise, it may not provide the desirable result. Moreover, MLE can be biased when samples are small so MDNs are not suitable for semi-supervised learning. In practice, we also find that MDNs suffer from computational instability when the mixture number is large.

To alleviate the disadvantages of MDNs, the researchers introduced Bayesian neural networks (BNNs) [19], which applies Bayesian inference. BNNs follow the scheme of maximum a posterior (MAP) estimation, in which the prior knowledge of models and the likelihood are combined. MAP has the regularizing effect which can prevent overfitting. However, in practice, we find that BNNs are not flexible enough for very complex distribution like our case. We conjecture that this could be caused by the simple choice of the prior.

To solve this problem, we deploy a variational auto-encoder [20], which is a deep latent generative model, in the proposed model to introduce a richer prior information. Meanwhile, since variational auto-encoders (VAEs) are unsupervised learning methods, we can use it to learn knowledge from the unlabeled data and to devise the semi-supervised learning model.

## 3. PROBLEM DESCRIPTION

In this work, our purpose is to interpret the smartphone user location with the corresponding WiFi fingerprints. In contrast to some previous work, we aim at obtaining the accurate user location, namely, the coordinates (either in meters or longitudes and latitudes) rather than treat this subject a classification task (identifying buildings and floors). The first task of our work is a time-series prediction task whose purpose is to predict the next time-point user location using current time-point WiFi fingerprints. The second task of our work is a smei-supervised learning location recognition task. We attempt to make use of not only the labeled data but also the unlabeled data to improve the location recognition accuracy.

## 4. PROPOSED METHODS

We will introduce two proposed deep-learning models. The first proposed model is called the cconvolutional mixture density recurrent neural network (CMDRNN), which is designed to prediction path with WiFi fingerprints. The second proposed model is a VAE-based model, which is designed to proceed semi-supervised learning for WiFi-based positioning.

### 4.1. Convolutional mixture density recurrent neural network

#### *4.1.1. 1D convolutional neural network*

In our first task, the input features are composed of the RSSI values of all the WiFi access points (WAPs) in the buildings, thus the input can be very high dimensional. In practice, we discover that the adjacent input features (Wifi access point values) are more likely to have the similar numerical values than the remote input feature. For this reason, to deal with the high dimensionality problem, we resort to the technique of convolutional neural network (CNN) [11]. CNN is a powerful tool for detecting feature and it is widely used for tasks like imagine processing, natural language processing and sensor signal processing. In particular, since the input of our model is the RSSI value vector, we adopt the 1D convoultional neural network to extract the proprieties of high dimensional input.

#### *4.1.2. Recurrent neural network*

Recurrent neural networks (RNNs) [21] are widely used for natural language process (NLP), computer vision and other time series prediction task.

The state transition of RNNs can be expressed as follow.

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \tag{3}$$

where, $x_t$ is the input, $h_t$ is the hidden state, $\sigma_h$ is the activation function, $W_h$ is the hidden weight for the input, $U_h$ is the hidden weight for the hidden state and $b_h$ is the hidden bias.

The output of a conventional RNN can be expressed as follow.

$$y_t = \sigma_y(W_y h_t + b_y) \tag{4}$$

where, $y_t$ is the output, $\sigma_y$ is the activation function, $W_y$ is the hidden weight for the input and $b_y$ is the output bias

However, in practice, RNNs may suffer the long-term dependency problem during learning process. To address this problem, the researchers proposed long short-term memory networks (LSTMs) [22], which is a variant of RNNs. In our model, we employ the LSTM network to predict user location. More recently, the researchers proposed a variant of RNN, gated recurrent units (GRUs) [23], which have the similar accuracy as LSTMs but less computing cost. We will deploy these three RNN architectures into the proposed model as comparisons.

#### *4.1.3. Mixture density network*

A traditional neural network with a loss function, for instance, mean square errors, is optimized by a gradient-descent based method. Generally, such model can perform well on the problems that can be described by a deterministic function $f(x)$, i.e., each input's corresponding output can only be one single specific value. However, for some stochastic problems, one

input may be corespondent to more than one possible values. Generally, this kind of problems are better to be described as a conditional distribution $p(y|x)$ than a deterministic function $y = f(x)$. In such case, traditional neural networks may not work as expected.

To tackle with this kind of problems, intuitively, we can replace the original loss function with a conditional function, for a regression task, the Gaussian distribution can be the proper choice. Moreover, to utilize the mixed Gaussian distributions can improve the representation capacity of the model.

The researchers proposed the mixture density networks (MDNs) model [18]. In contrast with traditional neural network, the output of MDN is the parameters a set of mixed Gaussian distributions and the loss function become the conditional probabilities. Therefore, the optimization process is to minimize the negatived log probability. Hence, the loss function can be described as follow:

$$P(y|x) = \sum_{k=1}^{K} \pi_k P(y|x; \theta_k) \tag{5}$$

where, $x$ is the input, $\pi_k$ is the assignment probability for each model, with $\sum_{k=1}^{K} \pi_k = 1, (0 < \pi_k < 1)$, and $\theta_k$ is the internal parameters of the base distribution. For Gaussians, $\theta_k = \{\mu_k, \sigma_k\}$, $\mu_k$ is the means and $\sigma_k$ is the variances.

Accordingly, in the proposed model, the original output layer of the RNN, Eq. (4) ,is rewritten as this:

$$\theta_t = \sigma_\theta(W_\theta h_t + b_\theta) \tag{6}$$

where, $\theta_t$ is the output of the RNN sub-model and also the input of the MDN sub-model, $\sigma_\theta$ is the activation function, $W_\theta$ is the hidden weight for the input and $b_\theta$ is the output bias.

After the training process, we can use the neural network along with the mixed Gaussian distributions to illustrate the target distributions.

#### *4.1.4. Proposed model*

Combined with the merits of three aforementioned neural networks, we devised a novel deep neural network architecture, called the convolutional mixture density recurrent neural network (CMDRNN). In the CMDRNN model, an 1D CNN is used to capture the features of the high dimensional inputs, then the state transitions of the time series data is modeled by a LSTM-RNN model, and the output layer composed of mix Gaussian densities to enhance the pediction accuracy. With such an structure, we believe that our model is able to illustrate complicated high dimensional time series data. Fig 1 shows the whole structure of the CMDRNN model and Algorithm. 1 demonstrates the learning process of the CMDRNN model.

The uniqueness of our method is that, compare other existing models in literature, our model adopt a sequential density estimation approach. Thus, the learning target of the proposed become a conditional distribution of the data rather than a common regressor. Thanks to this, our model can conquer such complicate modeling tasks.

**Algorithm 1** Algorithm: CMDRNN model

**Input:** $X$ (RSSI Values)
**Output:** $Y$ (Coordinates)

**while** e $<$ max epoch **do**
  **while** i $<$ batch num **do**
    $h_0 \leftarrow Conv1d(X)$ ▷ convolutional operation
    $h_1 \leftarrow$ max pool $h_0$
    $f \leftarrow$ flatten $h_1$
    $h_t \leftarrow \sigma_h(W_h * f_t + U_h * y_{t-1} + b_h)$ ▷ update hidden states
    $\theta \leftarrow \sigma_y(W_y * h_t + b_y)$ ▷ compute network output
    $\theta_k \leftarrow \theta$ ▷ assign mixture density parameters
    minimize loss function: $-p(y_t|x_t; \theta)$
  **end while**
**end while**

$Y \sim p(y|x; \theta)$ ▷ final output

**return** $Y$

### 4.2. VAE-based semi-supervised learning

We assume that $x$ (RSSI values) and $y$ (corrdiantes) are governed by the same latent variable $z$ (building structures). However, in many real cases, the available datasets have more information about input $x$ and less information about target $y$, thus it is more reasonable to infer the distribution of $z$ via $p(z|x)$ rather than via $p(z|y)$. This procedure can be described as:

$$p(z|x) \rightarrow q(z) \tag{7}$$

where $q(z)$ represents the prior distribution of $z$.

Afterwards, if we apply the chain rule and assume the conditional generative scheme as follow.

$$p(y|x) = p(y|z)p(z|x) \tag{8}$$

Accordingly, the predicting model (either deterministic or probabilistic) can be described as:

$$y \sim p(y|z) \tag{9}$$

We can implement Eq. (7) and Eq. (9) by an unsupervised learning process an a supervised process, respectively. Therefore, our method consists of two steps:

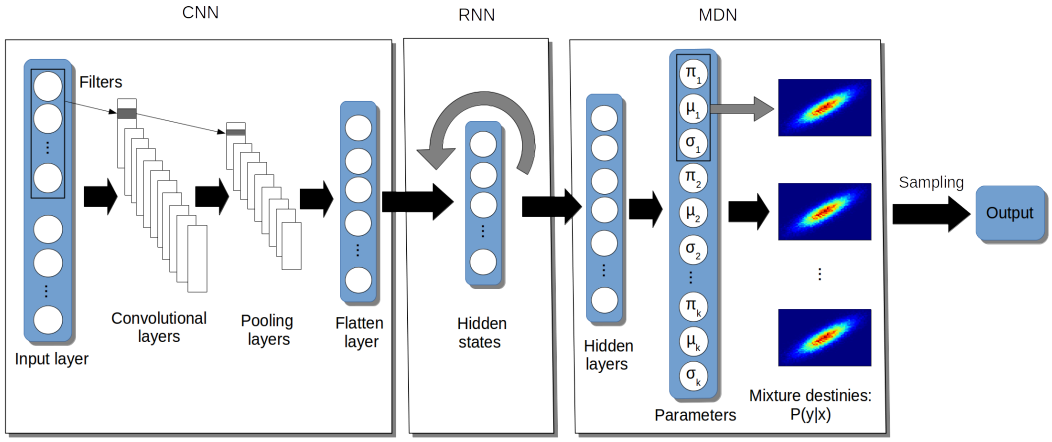



**Fig. 1**: Convolutional mixture density recurrent neural network.

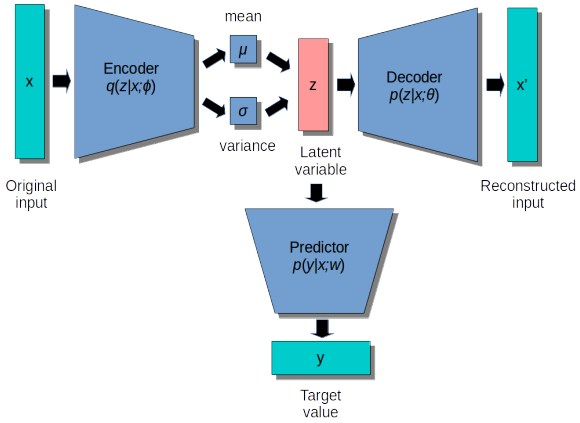



**Fig. 2**: VAE-based semi-supervised learning model.

- The first step (unsupervised learning): we employ a deep generative model to obtain the latent distribution $p(z|x)$.
- The second step (supervised learning): we employ a MLP model to obtain the target value $y$.

The structure of the VAE-based semi-supervised learning model is illustrated as Fig. 2

#### *4.2.1. Unsupervised learning procedure*

For the unsupervised learning process, we adopt a variational autoencoder as the generative model to learn the latent distribution. Variational autoencoders (VAEs) [20] are deep latent generative models.

In VAEs, the prior of the latent variable $z$ yields to a standard Gaussian distribution.

$$z \sim \mathcal{N}(0, \mathbb{I}) \tag{10}$$

In order to obtain the VAE encoder $p(z|x;\phi)$ via a MLP, $z$ is reparameterized by the reparameterization trick:

$$z = \mu_z + \sigma_z \odot \epsilon,\ \epsilon \sim \mathcal{N}(0; \mathbb{I}) \tag{11}$$

where, $\mu_z$ is the mean of $z$ and $\sigma_z$ is the variance of $z$.

The VAE decoder can be described as:

$$x' \sim p(x|z;\theta) \tag{12}$$

where $x'$ is the reconstructed input.

The evidence lower bound (ELBO) of VAEs can be written as:

$$\mathcal{L}_{\theta,\phi}(x) = \log p(x;\theta) - \mathrm{D}_{\mathrm{KL}}(q(z|x;\theta)||p(z;\phi)) \tag{13}$$

Once $\mathcal{L}_{\theta,\phi}(x)$ is maximized, we have the approximate posterior $p(z|x;\phi)$.

#### *4.2.2. Deterministic predictor (M1 model)*

After supervised training, we have the latent distribution $z$. For the supervised learning, to obtain the target $y$, we devise two predicting models, one is deterministic, the other is probabilistic.

As a nave approach, we can build a deterministic predictor which consists of two predicting steps:

Step 1: to obtain the means of latent variables

$$\mu_z = F_\mu(x;\phi) \tag{14}$$

where, $F_\mu(x;\phi)$ can be regarded as the encoder of the VAE.

Step 2: to obtain the final prediction based on the output of Step 1.

$$y' = F_y(\mu_z; w) \tag{15}$$

where, $F_y(\mu; w)$ is a deterministic multilayer perceptron model.

Consequently, the loss function is:

$$\mathcal{L}(D; w) = \frac{1}{N}\sum_{n=1}^{N}(y'_n - y_n)^2 \tag{16}$$

Note that the direct input of this predictor is the latent variable $z$, which introduces the information of the distribution to the predictor. Hence, this predictor does not suffer the problem as conventional neural networks.

**Algorithm 2** Algorithm: M1 model

**Input:** $X_a$ (all inputs),$X_l$ (labeled inputs),$Y_l$ (labels)
**Output:** $Y'$ (predictions)

**while** unsupervised learning **do**
  $\mu_z, \sigma_z \leftarrow E_\phi(X_a)$ ▷ $E_\phi(X_a)$: VAE encoder
  $z \sim \mathcal{N}(\mu_z, \sigma_z)$ ▷ Sample latent codes
  $X'_a \leftarrow G_\theta(z)$ ▷ Reconstruct inputs
  minimize loss function $\mathcal{L}_{\theta,\phi}(x)$ ▷ Eq. (13)
**end while**

**while** supervised learning **do**
  $\mu'_z \leftarrow E_\phi(X_l)$ ▷ get latent codes
  $Y' \leftarrow F_y(\mu'_z; w)$ ▷ get predictions
  minimize loss function $\mathcal{L}(D; w)$ ▷ Eq. (16)
**end while**
**return** $Y'$

The scheme of M1 model is summarized in Algorithm. 2.

#### *4.2.3. Probabilistic predictor (M2 model)*

The proposed M2 model is more robust to the noise.

We apply the chain rule to obtain the factorized conditional distribution:

$$p(y|x, z; w, \phi) = p(y|z; w)p(z|x; \phi) \tag{17}$$

$$\log p(y|x; w) = \log p(y|z; w) + \log p(z|x; \phi) \tag{18}$$

Since Eq. (18) cannot be solved explicitly, we use Monte Carlo method to draw samples $z$, $y$. First, we draw the latent variables $z$ from the VAE encoder:

$$z \sim p(z|x; \phi) \tag{19}$$

Then, we draw the predicted values $y$ based on:

$$y \sim p(y|z; w) \tag{20}$$

Then, the loss function of the predictor can be written as:

$$\mathcal{L}(D; w) \approx -\frac{1}{N}\sum_{n=1}^{N}[\log p(y|z; w) + \log p(z|x; \phi)] \tag{21}$$

Since $\phi$ is already trained, here we only care about optimizing $w$. Thus, the loss function becomes:

$$\mathcal{L}(D; w) \approx -\frac{1}{N}\sum_{n=1}^{N}\log p(y|z; w) \tag{22}$$

where, $N$ is the mini batch size.

Assume that the likelihood function $p(y|z; w)$ is a Gaussian distribution with noise $\sigma_y$.

**Algorithm 3** Algorithm: M2 Model

**Input:** $X_a$ (all inputs),$X_l$ (labeled inputs),$Y_l$ (labels)
**Output:** $Y'$ (predictions)

**while** unsupervised learning **do**
  $\mu_z, \sigma_z \leftarrow E_\phi(X_a)$ ▷ $E_\phi(X_a)$: VAE encoder
  $z \sim \mathcal{N}(\mu_z, \sigma_z)$ ▷ Sample latent codes
  $X'_a \leftarrow G_\theta(z)$ ▷ Reconstruct inputs
  minimize loss function $\mathcal{L}_{\theta,\phi}(x)$ ▷ Eq. (13)
**end while**

**while** supervised learning **do**
  $z' \sim \mathcal{N}(\mu_z, \sigma_z)$ ▷ Sample latent codes
  $Y' \sim p_y(z'; w)$ ▷ Sample predictions
  minimize loss function $\mathcal{L}(D; w)$ ▷ Eq. (22)
**end while**
**return** $Y'$

The scheme of M2 model is summarized in Algorithm. 3.

## 5. EXPERIMENTS AND RESULTS

### 5.1. Sequential Prediction

#### *5.1.1. Dataset Description*

For the validation dataset, we select two WiFi RSSI-Coordinate paths from the Tampere dataset [24]. As shown in Fig. 3 The input dimension of the Tampere dataset is $489$. The RSSI values of the detected WAPs range from $-110$ dB to $0$ dB and the RSSI values of undetected WAPs are set to be $100$.

#### *5.1.2. CMDRNN architecture overview*

The implementation details of our model are illustrated in Table 1. The CNN sub-network consists three layers, a convolutional layer, a max pooling layer and a flatten layer. The RNN sub-network includes a hidden layer with $200$ neurons. The MDN sub-network has a hidden layer and output layer. The mixed Gaussians number of the MDN outputlayer is $30$, and

| | Input | | | | | | | Output | |
|---|---|---|---|---|---|---|---|---|---|
| Timestamp | WAP001 | WAP002 | ··· | WAP076 | WAP077 | ··· | WAP489 | X | Y |
| 0 | 100 | 100 | ··· | -98 | -89 | ··· | 100 | 179.43 | 71.716 |
| 1 | 100 | 100 | ··· | -93 | -87 | ··· | 100 | 180.43 | 72.716 |
| 2 | 100 | 100 | ··· | -91 | -76 | ··· | 100 | 181.43 | 72.716 |
| | ⋮ | | | ⋮ | | | | ⋮ | |
| N | -77 | -75 | ··· | 100 | 100 | ··· | 100 | 185.43 | 74.716 |

**Fig. 3**: Some data samples.

each mixture has 5 parameters, two dimensional means, diagonal variances and correspondent portion. For the optimizer, we choose RMSprop [25].

### *5.1.3. Comparison with other methods*

In order to prove the effectiveness of our method, we conduct a series of experiments to thoroughly compare our CMDRNN model to other deep learning approaches. The purposes of experiments are indicated as follows.

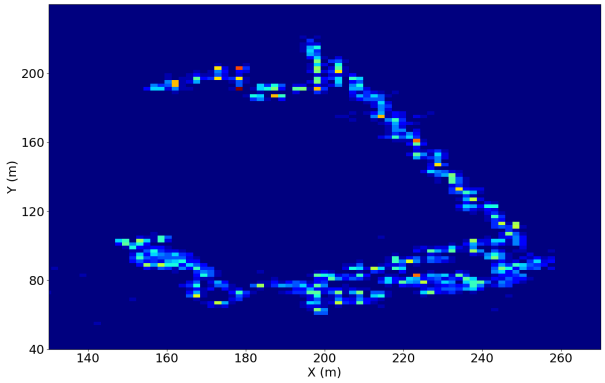
- Comparing optimizers: Adam v.s. RMSProp
- Comparing feature detectors: RNN, RNN+MDN, AE + RNN + MDN, CMDRNN
- Comparing regressors: RNN, CNN+RNN and CMDRNN
- Comparing RNN variants: CMDRNN, CMDLSTM and CMDGRU

The overall results are demonstrated in Table 2.

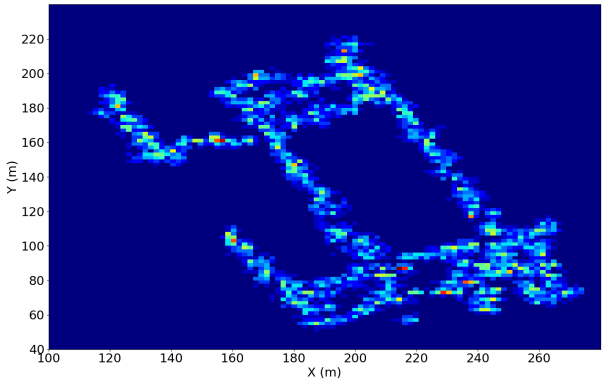
### *5.1.4. Optimizers comparison*

In [26], it reports that RMSProp [25] may have better performance on very non-stationary tasks than the Adam optimizer [27]. To verify that, we train our algorithm with RMSProp and Adam, respectively. As it is shown in Fig. 5, the proposed model can converge to a lower negative log-likelihood via RMSProp than Adam. Thus, we choose RMSProp as the optimizer for our model.

Since the input is high dimensional, the sagacious way to deal with this is to incorporate a sub-network into the model for dimension reduction or feature detection. Many previous research adopted auto-encoders to reduce dimension, while we argue that the more appropriate choice for the task in our work is using an one-dimensional CNN. In order to prove that, we test three different models, one without a feature-detecting structure, one using using an auto-encoder and one using 1D CNN (the proposed model). The auto-encoder model with structure {hidden neurons: 256; hidden neurons: 128; code size: 64 ; hidden neurons: 128; hidden neurons: 256}.




(a) Path 1 Prediction Results.




(b) Path 2 Prediction Results.

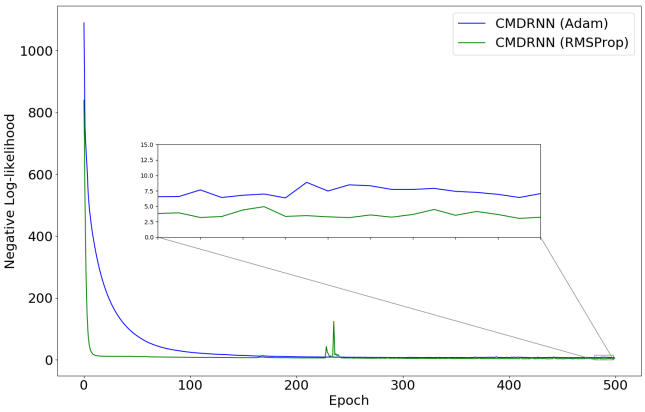



**Fig. 5**: Training loss via different optimizers.

The results in Fig. 6 and Table. 2 illustrate that the proposed model with 1D CNN feature detector can reach lower negative log-likelihood during the training process and has the smallest RMSE on the test data, respectively.

**Table 1**: CMDRNN Implementation Details

| Sub-network | Layer | Hyperparameter | Activation Function |
|---|---|---|---|
| CNN | convolutional layer | filter number: 100; stride: 2 | sigmoid |
| CNN | max pooling layer | neuron number: 100 | relu |
| CNN | flatten layer | neuron number: 100 | sigmoid |
| RNN | hidden layer | memory length: 5; neuron number: 200 | sigmoid |
| MDN | hidden layer | neuron number: 200 | leaky relu |
| MDN | output layer | 5*mixed Gaussians number (5*30) | - |
| Optimizer: RMSProp; learning rate: 1e-3 | | | |

**Table 2**: Path Prediction Results (root mean squared errors)

| Path | RNN | CNN+RNN | RNN+MDN | AE+RNN+MDN | CMDRNN | CMDLSTM | CMDGRU |
|---|---|---|---|---|---|---|---|
| Path 1 | $29.36 \pm 1.61$ | $34.26 \pm 3.04$ | $23.86 \pm 5.50$ | $11.24 \pm 0.86$ | $8.26 \pm 1.31$ | $7.38 \pm 0.89$ | $6.25 \pm 0.80$ |
| Path 2 | $31.61 \pm 0.74$ | $36.75 \pm 6.17$ | $23.58 \pm 2.29$ | $12.01 \pm 1.68$ | $10.17 \pm 0.72$ | $9.26 \pm 0.31$ | $8.67 \pm 0.23$ |

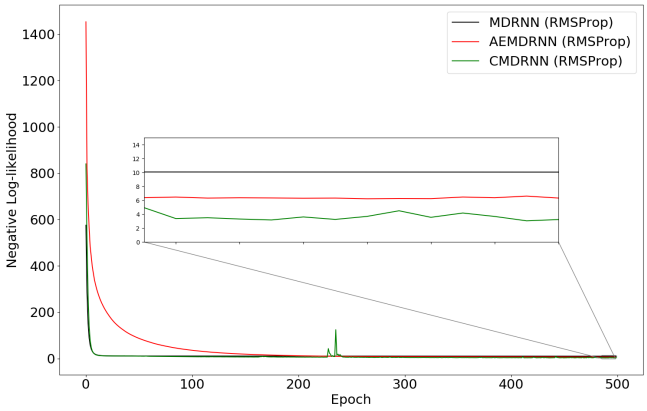



**Fig. 6**: Training loss via different feature detectors.

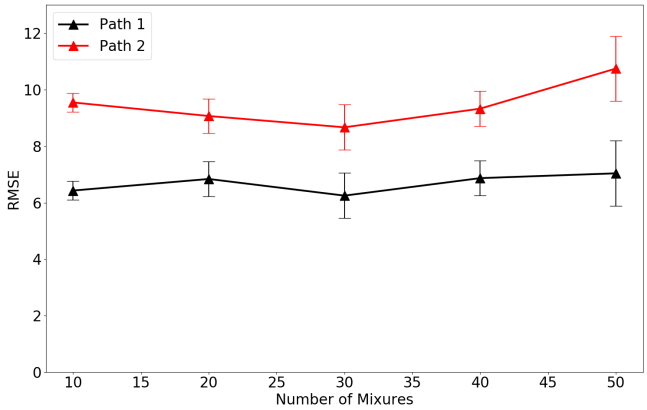



**Fig. 7**: Prediction results by varying mixture numbers in the MDN (bars represent the standard deviations).

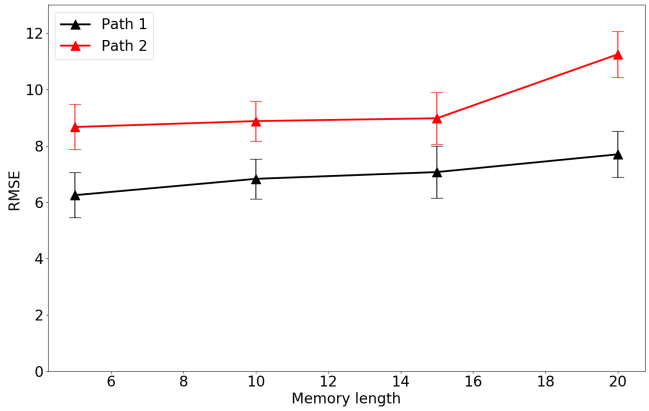



**Fig. 8**: Prediction results by varying memory lengths in the RNN (bars represent the standard deviations).

### 5.2. Semi-supervised learning for location recognition

#### *5.2.1. Dataset description*

For the validation dataset, we use the UJIIndoorLoc dataset [28], which is similar to the Tampere dataset. The input dimension of the UJIIndoorLoc dataset is 520. The RSSI values of the detected WAPs range from $-100$ dB to 0 dB and the RSSI values of undetected WAPs are set to be 100. The coordinates are in longitudes and latitudes and we use the scaled values for the experiments. The total instances number for the experiments is about 20000.

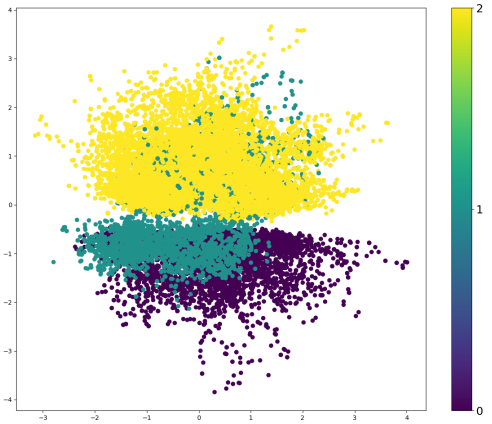
#### *5.2.2. Experimental results*

Different from other supervised We make use of All the Fig. 9 demonstrates the distribution of latent variable $z$.

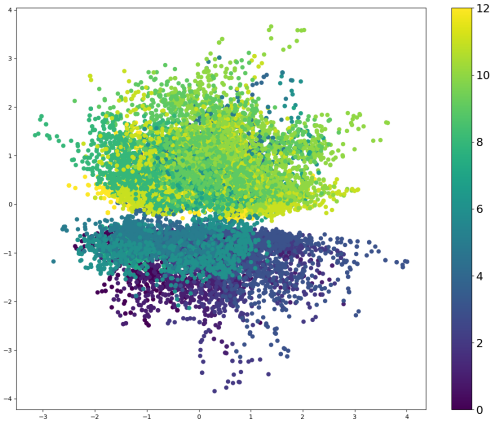
For the experimental set up, we use different portions of labeled data, ranging from 2% to 80%. We use k-NN, GP, MDN with 2 mixtures, noted as MDN(2), MDN with 5 mix-

**Table 3**: Test results with different portions of labeled data (root mean squared errors)

| Labeled data | k-NN | GP | MDN (2) | MDN(5) | BNN | M1 | M2 |
|---|---|---|---|---|---|---|---|
| 2% | $0.208 \pm$ 4e-2 | $0.561 \pm$ 5e-2 | $0.167 \pm$ 3e-3 | $0.168 \pm$ 3e-3 | $1.050 \pm$ 1e-2 | $0.190 \pm$ 4e-3 | $0.202 \pm$ 6e-3 |
| 5% | $0.171 \pm$ 8e-3 | $0.341 \pm$ 1e-2 | $0.141 \pm$ 4e-3 | $0.138 \pm$ 6e-3 | $1.033 \pm$ 6e-3 | $0.130 \pm$ 3e-3 | $0.131 \pm$ 3e-3 |
| 10% | $0.138 \pm$ 6e-3 | $0.295 \pm$ 8e-3 | $0.134 \pm$ 5e-3 | $0.124 \pm$ 2e-3 | $1.038 \pm$ 5e-3 | $0.116 \pm$ 3e-3 | $0.115 \pm$ 3e-3 |
| 20% | $0.121 \pm$ 2e-3 | $0.266 \pm$ 2e-3 | $0.118 \pm$ 5e-4 | $0.113 \pm$ 1e-3 | $1.032 \pm$ 2e-3 | $0.103 \pm$ 2e-3 | $0.102 \pm$ 2e-3 |
| 30% | $0.112 \pm$ 5e-3 | $0.259 \pm$ 3e-3 | $0.117 \pm$ 3e-4 | $0.108 \pm$ 5e-3 | $1.058 \pm$ 7e-3 | $0.101 \pm$ 2e-3 | $0.097 \pm$ 4e-3 |
| 50% | $0.096 \pm$ 2e-3 | $0.256 \pm$ 6e-4 | $0.103 \pm$ 9e-3 | $0.100 \pm$ 2e-3 | $1.043 \pm$ 3e-3 | $0.092 \pm$ 1e-3 | $0.090 \pm$ 2e-3 |
| 80% | $0.092 \pm$ 2e-3 | $0.252 \pm$ 3e-3 | $0.099 \pm$ 3e-4 | $0.103 \pm$ 3e-3 | $1.033 \pm$ 4e-3 | $0.088 \pm$ 1e-3 | $0.087 \pm$ 5e-3 |

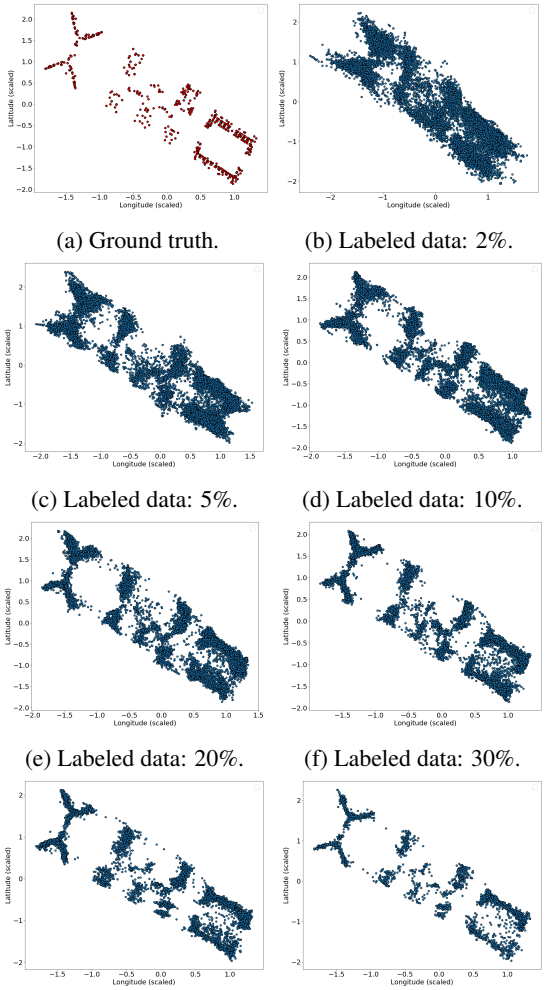


(a) Latent variables labeled with the building IDs.



(b) Latent variables labeled with the floor IDs.

**Fig. 9**: Latent variables with dimension of 5, here shows the 2D projection.



(a) Ground truth. (b) Labeled data: 2%.

(c) Labeled data: 5%. (d) Labeled data: 10%.

(e) Labeled data: 20%. (f) Labeled data: 30%.

(g) Labeled data: 50%. (h) Labeled data: 80%.

**Fig. 10**: Test results for M2 model.

tures, noted as MDN(5) and BNN, as comparisons.

From the results, we can see that the proposed models, M1 and M2, can provide satisfying results even when the labeled data is scarce. The predicting accuracy is improved when the labeled data increases. In contrast with other methods, the proposed models have better performance than other methods. Through the experiments, we also fin that the proposed models, compared to other methods, have such advantages:

- Compared to GPs, the proposed models are less computationally expensive.
- Compared to MDNs, the proposed models are more computationally stable.
- Compared to BNNs, the proposed models are more flexible to complex distributions.

## 6. CONCLUSIONS AND PERSPECTIVES

In this paper, we attempt to tackle the WiFi fingerprint-based user positioning problem. The first task is to predict the next user location with the current WiFi fingerprints. In contrast with existing approaches, our solution is to devise a hybrid deep learning model. The proposed model is composed of three deep neural network, a CNN, a RNN and a MDN. This unique deep architecture combines all the strengths of three deep learning models, which the enables us to recognize and predict user location with high accuracy.

The second task is a semi-supervised learning problem for accurate user location recognition. To tackle this, we propose a VAE-based semi-supervised learning model. This model employs a VAE model to proceed unsupervised learning procedure. Meanwhile, in order to interpret WiFi RSSI values into coordinates, we devise two different predictors, one is a deterministic model, the other is a probabilistic model.

Finally, we test our models on the real-world datasets. For both of the tasks, the results verify the effectiveness of our approaches and show the superiority of our methods compared other deep learning based methods as well.

For the future work, we will explore other deep generative model for the potential applications for this research topic. For instance, normalising flows can be the potential approach to improve the performance.

## 7. REFERENCES


[1] S.-B. Cho, "Exploiting machine learning techniques for location recognition and prediction with smartphone logs," *Neurocomputing*, vol. 176, pp. 98–106, 2016.

[2] C. Yu, Y. Liu, D. Yao, L. T. Yang, H. Jin, H. Chen, and Q. Ding, "Modeling user activity patterns for next-place prediction," *IEEE Systems Journal*, vol. 11, no. 2, pp. 1060–1071, 2017.

[3] M. T. Hoang, B. Yuen, X. Dong, T. Lu, R. Westendorp, and K. Reddy, "Recurrent neural networks for accurate rssi indoor localization," *arXiv preprint arXiv:1903.11703*, 2019.

[4] S. Bozkurt, G. Elibol, S. Gunal, and U. Yayan, "A comparative study on machine learning algorithms for indoor positioning," in *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*. IEEE, 2015, pp. 1–8.

[5] A. Cramariuc, H. Huttunen, and E. S. Lohan, "Clustering benefits in mobile-centric wifi positioning in multi-floor buildings," in *2016 International Conference on Localization and GNSS (ICL-GNSS)*. IEEE, 2016, pp. 1–6.

[6] J. Torres-Sospedra, R. Montoliu, S. Trilles, Ó. Belmonte, and J. Huerta, "Comprehensive analysis of distance and similarity measures for wi-fi fingerprinting indoor positioning systems," *Expert Systems with Applications*, vol. 42, no. 23, pp. 9263–9278, 2015.

[7] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[8] B. Ferris, D. Fox, and N. D. Lawrence, "Wifi-slam using gaussian process latent variable models." in *IJCAI*, vol. 7, no. 1, 2007, pp. 2480–2485.

[9] B. F. D. Hähnel and D. Fox, "Gaussian processes for signal strength-based location estimation," in *Proceeding of robotics: science and systems*, 2006.

[10] S. Yiu and K. Yang, "Gaussian process assisted fingerprinting localization," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 683–690, 2015.

[11] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[13] M. Ibrahim, M. Torki, and M. ElNainay, "Cnn based indoor localization using rss time-series," in *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2018, pp. 01 044–01 049.

[14] M. Nowicki and J. Wietrzykowski, "Low-effort place recognition with wifi fingerprints using deep learning," in *International Conference Automation*. Springer, 2017, pp. 575–584.

[15] X. Song, X. Fan, C. Xiang, Q. Ye, L. Liu, Z. Wang, X. He, N. Yang, and G. Fang, "A novel convolutional neural network based indoor localization framework with wifi fingerprinting," *IEEE Access*, vol. 7, pp. 110 698–110 709, 2019.

[16] K. S. Kim, S. Lee, and K. Huang, "A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on wi-fi fingerprinting," *Big Data Analytics*, vol. 3, no. 1, p. 4, 2018.

[17] C. M. Bishop, *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[18] Bishop, Christopher M, "Mixture density networks," 1994.

[19] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *International Conference on Machine Learning*, 2015, pp. 1861–1869.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[21] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[22] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[23] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[24] E.-S. Lohan, J. Torres-Sospedra, P. Richter, H. Leppkoski, J. Huerta, and A. Cramariuc, "Crowdsourced wifi database and benchmark software for indoor positioning," *Data set], Zenodo. doi*, vol. 10, 2017.

[25] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[26] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[28] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, J. P. Avariento, T. J. Arnau, M. Benedito-Bordonau, and J. Huerta, "Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems," in *2014 international conference on indoor positioning and indoor navigation (IPIN)*. IEEE, 2014, pp. 261–270.