# Indoor Wi-Fi Localisation with Noisy Dataset

*Li Wang*


Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2023

# Abstract

This project is primarily concerned with applying deep learning techniques to the problem of indoor wifi localisation. The application of indoor Wi-Fi localisation typically requires collecting accurate training dataset for the building of interest. However, this can be expensive or infeasible in a commercial setting. An alternative would be to collect the dataset through crowd-sourcing, which saves effort but will result in a noisy dataset with inaccurate labels. In this project, we investigate various methods to reduce the effect of noisy training dataset, including data augmentation, validation-guided training, ensemble method, and self-supervised learning. We achieve a mean error of 1.93m on the test set, a 48% improvement compare to models directly trained on the noisy dataset.

# Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Li Wang*)

# Table of Contents

# Chapter 1

# Introduction [1]

There is a growing commercial demand in the modern society for indoor localisation system, which offers numerous applications including indoor navigation, emergency response, and targeted advertising. Traditional Global Positioning System (GPS) technologies struggle to provide accurate localisation in indoor environments due to signal degradation caused by obstructions and multipath effects [1]. Consequently, various alternative methods, including Wi-Fi-based localisation and Radio Frequency (RF) based localisation [2], have emerged as potential solutions to overcome these limitations. In particular, the use of Received Signal Strength (RSS) from Wi-Fi access points has proven to be a promising approach for indoor localisation, due to the widespread adoption of wireless LANs (WLANs) in almost every building. However, several challenges need to be addressed to achieve the desired accuracy and robustness in localisation performance.

To use RSS signals for localisation, there are usually two steps involved. First, one needs to collect RSS signals at a set of known locations. The collected RSS signals at a given location is known as the *Wi-Fi fingerprint* of that location. The collection step establish the association between fingerprints and their corresponding spatial locations. Then, at inference time, a location prediction is made for an unknown fingerprint based on the collected fingerprints. One of the primary challenges in Wi-Fi-based localisation is the collection of an accurate and comprehensive dataset, which is often limited by time and resource constraints. Crowd-sourcing [3] has emerged as a potential solution to this problem, allowing the collection of large amounts of data at a reduced cost. However, the data collected through crowd-sourcing are often noisy, which can significantly impact the performance of localisation algorithms.

[1] Note that some portions of this section may come from the IPP project.

This dissertation aims to tackle the issue of using noisy and uncertain datasets in indoor Wi-Fi localisation and make accurate predictions given *only* noisy dataset with the help of deep learning techniques. We propose various methods to mitigate the challenges associated with the use of crowd-sourced data, such as data augmentation and validation-guided training. These approaches are designed to improve the reliability and robustness of the localisation system, allowing it to better adapt to the inherent uncertainties in RSS signals and the training data.

## 1.1 Problem Statement

Despite the significant advancements in indoor Wi-Fi localisation, there remains a pressing need to address the uncertainties and inaccuracies associated with crowd-sourced data. Crowd-sourced data are collected by people walking in the building with their smartphone recording the Wi-Fi signals and IMU data. The location associated with each fingerprint in the crowd-sourced data is generated algorithmically using the IMU data collected along with the Wi-Fi fingerprints and the building floor plan to construct a *plausible* route taken by the user. As can be imagined, the generated location might differ by a large amount to the actual location due to inaccurate sensor readings and errors from the prediction algorithm. Consequently, the Wi-Fi localisation system based on such noisy dataset will naturally produce noisy predictions.

The current positioning system typically involves three components as illustrated in fig. 1.1. The first component, which we call the Wi-Fi encoder, predicts a location, i.e., $(x_t^{enc}, y_t^{enc})$, given a Wi-Fi fingerprint. The second component uses IMU information to estimate the number of steps taken by the user, the direction taken, etc. Ultimately, this component estimates a location in its local reference frame, i.e., $(\Delta x_t, \Delta y_t)$. This component will be called the IMU encoder. Finally, a filter is used to integrate the two predictions from encoders, and the past trajectory of the user to produce a final estimate of the current location $(x_t, y_t)$. This filter can be any traditional filter, such as particle filters, or deep learning based filter.

Although the filter takes into account the information from both the IMU encoder and the Wi-Fi encoder, if the Wi-Fi encoder produces very noisy predictions, it will significantly affect the prediction of the filter, causing the final prediction to 'drift'. The challenge is thus to develop a more robust and reliable localisation system that can handle the inherent noise and variability in crowd-sourced datasets.

For a Wi-Fi encoder, which will be the focus of this dissertation, the input is a Wi-Fi

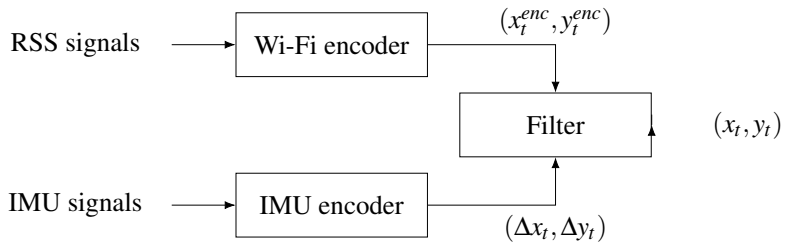


Figure 1.1: The components in an existing indoor localisation system.

fingerprint, which consists of a set of Access Point (AP) names and their corresponding RSS signals received by a phone at a specific location at a specific time. The output is a location prediction, which is a vector of $(x, y)$ coordinates. The problem can thus be formulated as a regression problem, where a mapping is learned from the fingerprint feature space to the coordinate space, and thus various machine learning techniques can be applied to solve this problem.

## 1.2 Research Hypothesis and Objectives

The overarching aim of this project is to enhance the accuracy and reliability of indoor Wi-Fi localisation systems by addressing the challenges associated with noisy and uncertain crowd-sourced datasets. In this dissertation, various methods will be experimented to attempt to increase the accuracy of the above mentioned system. In this research, we will focus on improving the accuracy of the Wi-Fi encoder. We investigate methods that allows training the Wi-Fi encoder on a noisy dataset with the ability to generalise to a clean and unseen test dataset. We hypothesis that with the proposed methods, the localisation system will be able to discover the underlying patterns in a noisy dataset and make accurate predictions on the clean dataset.

## 1.3 Outline

In sec. 2, we overview the background of Wi-Fi localisation problem, and relevant researches on utilising noisy datasets for deep learning. Then, we introduce our proposed model and experimental details in sec. 3. The results of experiments are shown in sec. 4, and are analysed in sec. 5, along with suggestions for future research.

# Chapter 2

# Background and Related Work [1]

## 2.1 Wi-Fi Localisation

Wi-Fi localisation has emerged as a promising solution to the indoor localisation problem, as it leverages the ubiquitous Wi-Fi networks in most modern buildings. Within the domain of Wi-Fi localisation, different types of signals, such as Received Signal Strength (RSS) and Channel State Information (CSI), have been studied intensively over the past few decades [1], [4]. In particular, RSS-based localisation has been widely adopted in the literature due to its simplicity and compatibility with the existing wireless infrastructures without requiring extra hardware. Thus, this paper will primarily focus on RSS-based localisation techniques.

Various techniques have been proposed to exploit the Received Signal Strength (RSS) signals from Wi-Fi access points for localisation purposes, including fingerprinting [5] and trilateration [6]. Wi-Fi fingerprinting, which is the focus of this dissertation, has been particularly popular due to its simplicity. Fingerprinting techniques typically involves two stages. The offline stage comprises the creation of a radio map of the environment by collecting RSS readings at known locations. During the online phase, the RSS readings at an unknown location is compared to the collected fingerprints, and a prediction is made given the similarity of the fingerprints.

Although fingerprinting methods have been shown to achieve high localisation accuracy, they suffer from two main limitations. First, the offline phase is time-consuming and labour-intensive. At each site, the RSS readings need to be collected at multiple locations that cover the entire site, and at each location, multiple readings need to be collected to account for the instability of RSS readings. The amount of time required to

[1] Note that some portions of this section might come from the IPP project.

collect the fingerprints can thus be prohibitive, especially for larger sites. In addition, the method does not scale to large-scale deployments at multiple sites. The second problem is the inherent instability of RSS readings, which can be affected by environmental factors such as temperature, humidity, and the presence of people, which can significantly affect the accuracy of the localisation system.

To address the first problem, crowd-sourcing has been proposed as a solution to reduce the amount of time and effort required to collect fingerprints. In [7]–[9], various approaches have been proposed to record user motion and collect fingerprints simultaneously, and infer the locations associated with the fingerprints using complex algorithms, thus eliminating the need for human intervention and site survey. However, crowd-sourcing introduces a new problem: the locations associated with the fingerprints can be highly inaccurate as the locations are inferred algorithmically, and thus may not reflect the actual location of the user. This problem is further exacerbated by the fact that the crowd-sourced data is often collected by users with different devices, which may have different hardware specifications and thus produce different RSS readings.

## 2.2 Noisy Dataset

Note that a dataset collected with crowd-sourcing has two types of noises: the noise in the RSS readings, i.e., fingerprint noise, and the noise in the location labels, i.e., label noise. Various approaches have been proposed to address these two types of noises.

### 2.2.1 Fingerprint noise

Methods dealing with noise in RSS readings can be broadly classified into two categories, probabilistic approaches and deterministic approaches. Probabilistic approaches make use of the probabilistic framework to handle the noise in the fingerprints, while deterministic approaches use various deterministic methods to robustly extract features from the fingerprints.

#### 2.2.1.1 Probabilistic methods

Probabilistic methods typically model the probabilistic relationship between the RSS readings and the locations. Consider a building with a total of $P$ access points, the fingerprint received at any given time can be represented as a $P$-dimensional vector

$\mathbf{x} = [x_1, x_2, \ldots, x_P]^\top$. Denote the coordinates in the building as $\mathbf{c}$, we have

$$p(\mathbf{c} \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathbf{c})\, p(\mathbf{c})}{p(\mathbf{x})}.$$

For a given fingerprint $\mathbf{x}$,

$$p(\mathbf{c} \mid \mathbf{x}) \propto p(\mathbf{x} \mid \mathbf{c})\, p(\mathbf{c}).$$

If we assume that users are equally likely to be at any location in the building, the equation can be further simplified to

$$p(\mathbf{c} \mid \mathbf{x}) \propto p(\mathbf{x} \mid \mathbf{c}).$$

Probabilistic modelling can thus be reduced to estimating the conditional probability $p(\mathbf{x} \mid \mathbf{c})$. Typically, only a finite amount of coordinates are considered to reduce the modelling complexity, i.e., $\mathbf{c} \in \mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_C\}$, where $C$ is the total number of coordinates considered. The calculated probabilities for each locations are then used as weights to compute the weighted average of the coordinates as the predicted location,

$$\hat{\mathbf{c}} = \sum_{i=1}^{C} \frac{p(\mathbf{x} \mid \mathbf{c}_i)}{\sum_{j=1}^{C} p\left(\mathbf{x} \mid \mathbf{c}_j\right)} \mathbf{c}_i, \tag{2.1}$$

which may not be in $\mathcal{C}$. The weighted average does not really have a probabilistic interpretation, but it is a common practice in the literature [5], [10].

Various approaches have been proposed to model this conditional probability. In [11], the authors further assumed that the RSS readings are independent of each other, and model the conditional probability as

$$p(\mathbf{x} \mid \mathbf{c}) = \prod_{i=1}^{P} p(x_i \mid \mathbf{c})$$

where $p(x_i \mid \mathbf{c})$ are modelled as Gaussian distributions. Similarly, in [10], the independence is assumed and the conditional probabilities are modelled with Gaussian processes.

Another branch of probabilistic methods attempt to model the latent representations of the RSS readings with generative modelling techniques, and use the latent representations for later regression or classification tasks. In [12], a Variational AutoEncoder (VAE) was used as a deep probabilistic model to learn the latent representations of the RSS readings. This class of methods are not actively investigated in the literature, likely due to the large amount of training data required to train a deep probabilistic model.

In summary, although probability provides a natural framework to model the noise in the fingerprints, probabilistic approaches often make unrealistic assumptions about the

data, such as independence between the RSS readings, and the distribution of the RSS readings. In addition, probabilistic models often require more training data compared to deterministic methods. On the upside, probabilistic models can provide uncertainty estimates, which can be integrated into later stages of the localisation pipeline in fig. 1.1 to improve the localisation performance.

#### 2.2.1.2 Deterministic methods

The most basic and most widely used deterministic method is the K-Nearest Neighbours (KNN) algorithm [2]. KNN-based methods define a distance metric between the fingerprints, and use the distances to find the $K$ nearest neighbours of the test fingerprint. The algorithm is somewhat noise tolerant by making predictions using the average of the locations of the $K$ nearest neighbours. The distances can also be used as a weight to compute the weighted average of the locations of the $K$ nearest neighbours, resulting in a Weighted KNN (WKNN). KNN-based methods are simple and easy to implement, but the performance is highly dependent on the distance metric and the choice of $K$. In addition, the inference time scales linearly with the size of the training dataset, which can be a problem for large datasets. Various attempts have been made to improve the performance of KNN-based methods. Some researches focus on improving upon the most commonly used distance metric, Euclidean distance, by using other distance metrics such as cosine similarity [13] and Spearman distance[14]. Some other researches focus on improving the choice of $K$ by using adaptive $K$ [15]. Other researches attempt to reduce the inference time by classifying the fingerprint into clusters before performing KNN [16].

Concurrently in [17], [18], the authors proposed to use a Stacked Denoising AutoEncoder (SDAE) to robustly extract features from the RSS readings. SDAE [19] is a variant of AutoEncoder (AE), which is a neural network that learns to reconstruct its input through a information bottleneck. The SDAEs are trained to reconstruct the input from a corrupted version of the input, thus forcing the network to learn robust features that are invariant to noise. In [17], a SDAE is trained for each reference location, and the reconstruction error is converted between $[0,1]$ to be interpreted as conditional probability, and are used as weights to compute the weighted average as in eq. 2.1. Note that although 'conditional probability' is used here, it is not a probabilistic method, as

$$\int p(\mathbf{x} \mid \mathbf{c})\, \mathrm{d}\mathbf{x} \neq 1.$$

However, as we need as many SDAEs as the number of reference locations, the com-

putation cost can be prohibitive. In [18], the authors proposed to train a single SDAE for feature extraction, and then use a linear layer with a softmax activation to predict the probability of been at each reference location. As only one SDAE is needed for all reference locations, the computation cost is significantly reduced. A similar approach will be investigated in this project.

With the extracted robust feature, different models were used to make predictions. In [20], the authors proposed to use a convolutional neural network (CNN) model to learn the relationship between extracted features from RSS readings and the location. In [21], [22], RNNs and RNN variants were used to process a time series of RSS readings.

Note that in all of the methods mentioned above, the RSS readings are processed as a fix-size vector. To the best of our knowledge, there are no previous work consider processing fingerprints in a sequential manner as in this project.

In summary, Wi-Fi localisation techniques, particularly those based on RSS signals, have gained significant attention due to their simplicity and compatibility with existing wireless infrastructures. Fingerprinting techniques, both deterministic and probabilistic, have shown promising results in various applications. In particular, we will investigate the use of deep learning techniques to extract robust features from the RSS readings, and to process the RSS readings in a sequential manner, which have the potential of giving robust prediction with limited amount of training data.

## 2.3 Noise in the labels

Another type of noise in a crowd-sourced dataset is label noise, which poses a significant challenge to the training of machine learning models. In this section, we review the literature on training with noisy labels. Although the majority of the literature on this topic focuses on classification tasks, we can still draw insights from them for our regression task.

Research on training with noisy labels can be broadly categorised into noise-tolerant algorithms that enable the learning algorithm to be robust to the presence of noisy data, and sample selection methods that attempt to identify the noisy samples and process them differently.

### 2.3.1 Noise-tolerant algorithms

We note that the KNN-based method mentioned in sec. 2.2.1 is inherently noise-tolerant, as long as there are enough clean samples in the neighbourhood of the test sample. It is an example of how simple algorithm can still be as effective as other complex methods.

A class of methods investigate the use of loss functions that are robust to the presence of noisy data. In [23], the author investigate the use of Tukey's biweight loss functions to mitigate the impact of noisy data on model training. The loss function is designed to be less sensitive to outliers and can lead to improved generalization performance in the presence of noisy data.

Noise-tolerant learning algorithms, as reviewed by Frenay and Verleysen [24], are designed to be robust to the presence of noisy data by explicitly modelling the noise process or incorporating noise-aware techniques into the learning process. Examples of such algorithms include noise-tolerant support vector machines and denoising autoencoders.

A subclass of methods in this category assume there is a small set of clean samples available as validation set, and utilise this validation set to provide guidance for training. One notable example is [25], which use the change in validation set performance after training on a batch of data as a signal to assign weights to the samples in a batch. The obtained weight is then used to reweigh the loss during training later. We call this method *validation-guided training*, which will be referred to later on.

### 2.3.2 Sample selection methods

The majority of the sample selection methods used with deep learning models utilise the *memorisation effect* [26] of neural networks, where the model learns the simple patterns in the dataset (usually the clean labels) first, before overfitting to the noisy labels. The idea is thus to use the losses of the model on the training set in the early stages to separate clean and noisy labels. We review the research in this area in chronological order.

In , an extra network (MentorNet) is trained, and select clean samples in the training set for student network to learn from. This method however, is prone to the problem of error accumulation. Some initial error in the Mentor network may cause the model to select noisy samples, which in turn cause the Mentor network to learn more errors and select more noisy samples. [27] addressed this problem by training two network simultaneously and each model select samples for the other model to learn from. The

error will thus flow between the network, and the two network will gradually correct each other's error. This method is called *co-teaching*. The DivideMix method proposed in [28] further improve upon the co-teaching method by utilising the samples with noisy labels. Similar to the co-teaching method, DivideMix train two networks in parallel, and the loss of the networks on the training set is used to select samples for the other network to learn from. The noisy samples however are not simply discarded, but are treated as unlabelled samples. Semi-supervised learning techniques MixMatch [29] are then used to generate pseudo-labels for the noisy samples, which are then used to train the network. The DivideMix method achieved state-of-the-art performance on the various datasets, and will be investigated in this project.

### 2.3.3 Self-supervised learning

Another way to deal with label noise is by removing the labels altogether. Instead of training the model using labels as supervision signals, we can train the model with the input data alone, and discover useful representations from the data. As no label is used during the training process, the obtained representation of the model is thus free from label noise. Although the label noise is not directly addressed, the obtained noise-free representation can be very valuable. [30] showed that a pretrained model is much more robust to label noise compared to a randomly initialised model.

Training model with only input data is known as *self-supervised learning*. The idea is to design a pretext task on input data, such that a model trained to solve the pretext task will learn useful representations of the data. Self-supervised learning can be categorised into two types: *contrastive learning* and *generative learning*. Here we briefly review families of methods in each category. For a more comprehensive review, please refer to [31].

#### 2.3.3.1 Contrastive learning

For contrastive learning, the pretext is a discriminative task, where the model is trained to distinguish between positive and negative samples. In order for the model to solve the pretext task, the representations of the positive samples are pulled closer together, while the representations of the negative samples are pushed further apart, thus leading to useful representations of the data.

The contrastive learning can be subdivided into *instance-instance contrast* and *instance-context contrast*. In SimCLR [32], the positive samples are generated by

applying two different data augmentation to the same data point, while the negative samples are other data points in the same batch. The contrast is between instances of data points, thus the name *instance-instance contrast*. For *instance-context contrast*, the contrast is between a fragment of the data point and its context. In [33], the positive samples are a patch of the image and a summary vector of the whole image, while the negative samples are summary vectors of other images. By performing instance-context contrast, the model maximise the mutual information between the patch of an image and the whole image, thus learning useful representations. We will investigate the SimCLR method in this project due to its simplicity.

#### 2.3.3.2 Generative learning

Generative learning, on the other hand, train the model to generate or reconstruct the input data.

Autoencoders (AEs) [34] learn to reconstruct the input through a information bottleneck where the dimension of the representation at the bottleneck layer is smaller then the input dimension. In order to faithfully reconstruct the input, the bottleneck layer needs to contain as much information about the input as possible in a lower dimension, thus forcing the model to learn a good representation of the input.

Denoising AutoEncoders (DAEs) are very similar to AEs, except that the input to the model is corrupted by noise. The model is then trained to reconstruct the original input. The learned representation is thus robust to noise. Note a special case of the DAE is Masked Language Modelling (MLM) used by BERT [35], where some token in the input sentence is replaced by a `<mask>` token, and the model is trained to predict the masked tokens. We will investigate the DAE method in this project due to its ability to handle both label noise and fingerprint noise.

For generative modelling, we directly model the probabilistic distribution of the input data. The model is trained by maximising the likelihood of the data.

## 2.4 Data augmentation

Another prominent technique used in Wi-Fi localisation is data augmentation. As the Wi-Fi localisation is bounded by the amount of fingerprints collected in off-line stage, many attempts were made to generate more fingerprints from a smaller dataset in order to reduce the effort required for site survey and increase the localisation accuracy. In

[3], [36], the author proposed to use Gaussian Processes (GPs) to estimate fingerprints at unknown locations. Although not explicitly designed to be used with a noisy dataset, the probabilistic framework offered by Gaussian Process provides a robust framework to counteract noise. The downside of the method is that a GP model needs to be trained for each access point, which can be computationally expensive considering there are in total 562 APs in one of our dataset. Other deep learning based methods were also proposed to generate fingerprints, including [37], which use Generative Adversarial Networks (GANs) to generate fingerprints, and [38] convert fingerprints into images and use super-resolution techniques to generate more fingerprints. Although deep learning based methods have potential to generate more fingerprints, they are often computationally expensive and require a large amount of training data, which is not available to us in the first place.

# Chapter 3

# Programme and Methodology

We hereby detail the experimental setup and the methodology used in this project. We first describe the dataset used in this project, followed by our proposed model. We then describe the training process, and finally, the evaluation metrics used in this project.

## 3.1 Dataset

The Wi-Fi fingerprints are collected with a Huawei smartphone in various buildings on campus[1]. The location associated with the fingerprints are computed with an in-house algorithm from Huawei utilising the IMU data collected on the phone along with the fingerprints. By aligning the estimated path with the building floor plan, we obtain our noisy dataset called *radio map*. For our training, we randomly split the samples in the radio map into 80%/20% split, which will be used as training set and validation set respectively. Note that *both* our training set and validation set are noisy, which better simulate the real world scenario in which we do not have access to high quality datasets.

To evaluate the performance of the trained algorithm, we need a noiseless dataset as our test set. To construct the test set, we collect fingerprints in the same building, along with a LiDar sensor. The location associated with the fingerprints are collected using the LiDar sensor and constructed with SLAM algorithm, which is considered to be the *ground truth location*. Note that the ground truth location is not used in training, and is only used for evaluation.

Each sample in the dataset is composed of a dictionary of access point name and signal strength, and the 2D coordinates associated with the sample, as illustrated in tbl. 3.1.

[1] With permission.

Table 3.1: An illustration of the dataset used in this project.

| sample_id | Fingerprint | Coordinates |
|---|---|---|
| 1 | {AP1: -50, AP5: -80,..., APK: -100} | (3.2, 5.4) |
| 2 | {AP3: -30, AP4: -50,..., APN: -20} | (4.2, 4.4) |
| ... | | |

We have in total four datasets, which we refer to as `bayesf1`, `bayesf3`, `forumf0`, `forumf1`, corresponding to different floors in different buildings. As `bayesf3` is the closest to the real deployment environment, some of the experiments will focus extensively on `bayesf3` dataset.

## 3.2 Models

### 3.2.1 Transformer encoder

First, we propose a new method for processing Wi-Fi fingerprint data. Wi-Fi fingerprints received by a phone is of the form of a dictionary of access point name and signal strength

$$\{\text{AP1: -45, AP2: -50, } \dots \text{APL: -57}\},$$

which might have different number $L$ of received APs at different times. Traditionally, a fingerprint is converted to be a fix-sized vector which has the same number of elements as the number of access points in a building. Motivated by [39], we consider transformer [40] to be able to process dictionary typed data, and propose to use transformer to process the fingerprints. First, we encode the AP name (more specifically, the mac address of the AP) to be a fixed sized vector, and the RSS value is appended to the vector. This gives a vector $\mathbf{v}$ for each element in the dictionary, and the dictionary is converted to a sequence of vectors $\mathbf{v}^{1:L}$. The mac address can be encoded in multiple ways, they can be seen as a discrete element and converted to a embedding, like in NLP, or simply be converted to the bit-wise representation of the mac address. We will mainly investigate bit-wise representation in this dissertation.

After the preprocessing, the fingerprints are in the sequential format suitable to be processed by the transformer architecture [40][2]. We use a transformer encoder to

[2]For the detail implementation of transformer architecture, please refer to the original paper.

process the fingerprints, and denote the transformer encoder as $f(\cdot)$. The transformer encoder produces a sequence of hidden states

$$\mathbf{h}^{1:L} = f(\mathbf{v}^{1:L}).$$

In order for the model to make a prediction, we need to aggregate the hidden states $\mathbf{h}^{1:L}$ into a single vector. Various methods are used throughout the project as the complexity of the task increases. Here we list all the methods tested.

### 3.2.2 Pooling layer

The simplest method is to use a mean pooling layer to aggregate the hidden states. The mean pooling layer perform element-wise mean on the hidden states

$$\mathbf{h}_{pool} = \frac{1}{L}\sum_{i=1}^{L}\mathbf{h}^i.$$

A slightly more complex method use attention pooling to aggregate the hidden states. A single head attention pooling layer use a learned query vector $\mathbf{q}$ to compute the attention weights over the hidden states, and then perform a weighted sum

$$\mathbf{h}_{att} = \sum_{i=1}^{L}\alpha_i\mathbf{h}^i,$$

where the attention weights $\alpha_i$ is computed as

$$\alpha_i = \frac{\exp\left(\mathbf{q}^\top\mathbf{h}^i\right)}{\sum_{j=1}^{L}\exp\left(\mathbf{q}_j^\top\mathbf{h}^j\right)}.$$

A multi-head attention pooling layer learns multiple query vectors $\mathbf{q}_1,\ldots,\mathbf{q}_Q$, and the final output is the concatenation of $Q$ single-head attention pooling layer's output.

The most complex method is to use a transformer decoder to process the hidden states and produce a fix-sized vector.

The Wi-Fi encoder as a whole is then composed of a transformer encoder to process the fingerprints, with a pooling layer to aggregate the sequence, followed by a linear layer that predict the coordinates $\hat{\mathbf{c}} = (x^{enc}, y^{enc})$.

## 3.3 Training methods

We will start with training the encoder proposed in sec. 3.2.1 with standard deep learning approach. Assuming a noisy training set $\left\{\left(\mathbf{v}_i^{1:L_i}, \mathbf{c}_i\right), 1 \le i \le N\right\}$, where $\mathbf{v}_i^{1:L_i}$ is a Wi-Fi fingerprint preprocessed as described in sec. 3.2.1, $\mathbf{c}_i = (x_i, y_i)$ is the geographical

coordinates associated with the fingerprint. With the encoder model $\hat{\mathbf{c}} = \Phi(\mathbf{v}^{1:L}, \theta)$, we train the model by minimising the mean error on the training set

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} C(\mathbf{c}_i, \Phi(\mathbf{v}_i^{1:L_i}))$$

where the loss function for a single sample is the distance between the predicted location and label location

$$C(\mathbf{c}_i, \hat{\mathbf{c}}_i) = \sqrt{(x_i^{enc} - x_i) + (y_i^{enc} - y_i)^2}. \tag{3.1}$$

We use the AdamW optimiser proposed in [41] to train the model.

### 3.3.1 Details of validation-guided training

In later section, we investigate the validation-guided training proposed in [25] to mitigate the impact of noisy training samples. The validation-guided training assumes a small clean and unbiased validation set $\left\{ \left( {}^{v}\mathbf{v}_i^{1:L_i}, {}^{v}\mathbf{c}_i \right), 1 \leq i \leq M \right\}$ in addition to the noisy training set ($M \ll N$). For a loss function $C(\mathbf{c}_i, \hat{\mathbf{c}}_i)$, we aim to learn a weight $w_i$ for each sample, such that minimising the reweighed loss

$$\theta^*(\mathbf{w}) = \arg\min_{\theta} \sum_{i=1}^{N} w_i C(\mathbf{c}_i, \Phi(\mathbf{v}_i^{1:L_i}, \theta)),$$

gives the best validation set performance

$$\mathbf{w}^* = \arg\min_{w, w \geq 0} \frac{1}{M} \sum_{i=1}^{M} C({}^{v}\mathbf{c}_i, \Phi({}^{v}\mathbf{v}_i^{1:L_i}, \theta^*(\mathbf{w})))$$

Finding the optimal $\mathbf{w}^*$ requires a nested two level optimisation, which is very expensive when the inner loop is optimising a neural network. Thus, we approximate $w_i$ for every sample in a batch at each training step through online approximation. For a mini-batch sampled from training set $\left\{ \left( \mathbf{v}_i^{1:L_i}, \mathbf{c}_i \right), 1 \leq i \leq n \right\}$, we could find the optimal $\varepsilon_t$ that minimise the validation loss

$$\varepsilon_t^* = \arg\min_{\varepsilon} \frac{1}{M} \sum_{i=1}^{M} C({}^{v}\mathbf{c}, \Phi({}^{v}\mathbf{v}_i^{1:L_i}, \theta_{t+1}^*(\varepsilon))).$$

However, this is still too computationally expensive. We thus approximate $\varepsilon_t^*$ by sampling a mini-batch of size $m$ from the validation set and taking a single gradient descent step on validation loss with respect to $\varepsilon_t$

$$u_{i,t} = -\eta \frac{\partial}{\partial \varepsilon_{i,t}} \frac{1}{m} \sum_{j=1}^{m} C({}^{v}\mathbf{c}, \Phi({}^{v}\mathbf{v}_i^{1:L_i}, \theta_{t+1}^*(\varepsilon))) \Big|_{\varepsilon_{i,t}=0}.$$

The gradient is then rectified and standardised to give the weight approximation at step $t$ $w_{i,t}$

$$\tilde{w}_{i,t} = \max(u_{i,t}, 0)$$
$$w_{i,t} = \frac{\tilde{w}_{i,t}}{\left(\sum_j \tilde{w}_{j,t}\right) + \delta\left(\sum_j \tilde{w}_{j,t}\right)}.$$

The loss is then recalculated based on the obtained estimation $w_{i,t}$, and model parameters are updated based on the reweighed loss.

The reviewed method is general enough to be used as a plug-and-play method that can be added to any gradient based method. In our case, although we do not have a clean validation set, we found that with the validation-guided training, the noisy validation set is able to provide some regularisation in addition to other regular regularisation techniques.

### 3.3.2 Self-supervised learning

We also investigate the self-supervised learning as reviewed in sec. 2.3.3 to mitigate the impact of noisy labels. More specifically, we experiment with both reconstruction method and contrastive method.

For reconstruction method, we train the model with the denoisning autoencoder approach where noises are injected into the input Wi-Fi fingerprint $\mathbf{v}^{1:L}$. We randomly dropout some of the Wi-Fi APs in the input fingerprint and add Gaussian noise to the RSS readings, similar to the data augmentation we used. A problem with the proposed model is that as the fingerprints are of variable length, it would be challenging to reconstruct the sequence directly. Thus, instead of reconstructing whole sequence, we train the model to reconstruct a fix-size vector of RSS readings, with the dimension equals the total number of APs in the buildings, similar to how most Wi-Fi fingerprinting methods work represent the Wi-Fi fingerprint. The loss function is then the mean squared error between target and reconstructed vector. Moreover, to create an information bottleneck, the hidden states $\mathbf{h}^{1:L}$ produced by the transformer are summarised into a fix-sized vector with various methods proposed in sec. 3.2.2. To reconstruct the RSS vectors, a MLP is used to map the summarised hidden states to the target RSS vector.

For contrastive method, we train the model using the SimCLR approach. For a batch of $B$ fingerprints $\{\mathbf{v}_i, 1 \leq i \leq B\}$ (we dropped the sequence length for simplicity), we produce two views of the same data by applying different dropout and different variance of Gaussian noise, and obtain $2B$ samples. We thus have $2B$ classification task, where

the model needs to predict the positive pair for each sample. If we denote the model with transformer encoder and the pooling layer as $g(\cdot)$, the loss can then be given as

$$\mathcal{L} = -\frac{1}{2B}\sum_{i=1}^{2B}\log\left(\frac{e^{g(\mathbf{v}_i)^T g(\mathbf{v}^+)}}{e^{g(\mathbf{v}_i)^T g(\mathbf{v}^+)} + \sum_{k=1}^{2B-2} e^{g(\mathbf{v}_i)^T g(\mathbf{v}^k)}}\right).$$

where $\mathbf{v}^+$ is the positive pair of $\mathbf{v}_i$, i.e., the different view of the same fingerprint, and $\mathbf{v}^k$ are the negative samples, i.e., other fingerprints in the batch.

## 3.4 Evaluation

The Wi-Fi encoder will be evaluated individually on the ground truth dataset collected, as detailed in sec. 3.1. The encoder will be evaluated based on its predictive ability which can be simply calculated as the average prediction error on the ground truth dataset (mean error), same as the objective for training in eq. 3.1.

To evaluate the representation obtained through self-supervised learning, we train a linear regressor on top of the pretrained model and evaluate its mean error on the validation and test set. Similarly, we also use the representation obtained from the pretrained model to train a k-nearest neighbours regressor and evaluate its mean error on the validation and test set.

# Chapter 4

# Experiments

## 4.1 Initial prototyping

We first evaluate the model proposed in sec. 3.2.1 on smaller datasets to verify its feasibility. The proposed model is composed of a transformer encoder layer, a mean pooling layer, and a linear layer that map the hidden states to $(x, y)$ coordinates. We use as baselines a multi-layer perceptron(MLP) and a weighted k-nearest neighbours. For MLP model, we choose hidden state size so that the total number of parameters of the model roughly matches that of the proposed model. All the training processes are terminated if no performance increase is observed on validation set for 30 epochs, i.e., early stopping with patience set to 30. The model with the best validation set performance is selected and evaluated on the test set. All experiments are repeated five times if not specified otherwise. The performances are shown in tbl. 4.1.

Table 4.1: The performance of different models on the four datasets. The number is the mean error in meters, and the number after $\pm$ is the standard mean error. For MLP and transformer, the train, validation, and test set performance are shown in the first, second, and third row respectively. For WKNN, the test set performance is shown.

| Method | bayesf1 | bayesf3 | forumf0 | forumf1 |
|---|---|---|---|---|
| MLP | $1.89 \pm 0.16$ | $2.18 \pm 0.22$ | $2.87 \pm 0.24$ | $2.87 \pm 0.24$ |
| | $3.49 \pm 0.05$ | $4.09 \pm 0.03$ | $4.70 \pm 0.03$ | $4.70 \pm 0.07$ |
| | $4.28 \pm 0.09$ | $6.08 \pm 0.17$ | $6.09 \pm 0.17$ | $8.31 \pm 0.22$ |

| Method | bayesf1 | bayesf3 | forumf0 | forumf1 |
|---|---|---|---|---|
| Transformer | $1.90 \pm 0.04$ | $2.38 \pm 0.04$ | $2.55 \pm 0.22$ | $2.16 \pm 0.17$ |
| | $2.60 \pm 0.03$ | $3.43 \pm 0.09$ | $3.20 \pm 0.06$ | $3.17 \pm 0.04$ |
| | $2.54 \pm 0.10$ | $3.70 \pm 0.14$ | $5.97 \pm 0.12$ | $2.84 \pm 0.04$ |
| WKNN | 3.97 | 4.67 | 8.59 | 4.74 |

As can be seen in tbl. 4.1, the new architecture has the best test set performances across all datasets, while having slightly worse training set performance on several datasets. This shows that converting the Wi-Fi RSSI signals to a sequential format better captures the properties of the data, and prevents the model from quickly overfitting to the noise in the data. What is remarkable with the proposed architecture is that sometimes the test set performance is better than the validation set performance! We believe this is due to the fact that the model successfully learned a good mapping between the Wi-Fi fingerprints and its location without being affected by the noise in the dataset. Thus, it is unable to further reduce its loss on the noisy dataset, whilst having a decent performance on the clean test set.

## 4.2 Scaling up performances

### 4.2.1 Attention Pooling

With the initial demonstration, we move on to scale up the performance of the model. First, we experiment with replacing the mean pooling layer with attention pooling layer with different number of attention heads. As shown in fig. 4.1, adding attention heads does not consistently increasing the performance on all datasets. This is likely due to the fact that some datasets are more noisy than others, and the attention pooling layer is more sensitive to the noise. Thus, on a simple dataset, too many attention heads will cause the model to quickly overfit to the noise.

### 4.2.2 Data augmentation

Due to the noisy nature of the Wi-Fi RSSI signals, the fingerprints received at the exact same location might be significantly different, i.e., some signals might be missing and some signal strength might be different from the training set samples. To simulate this, we propose to apply data augmentation at training time to better prepare the model

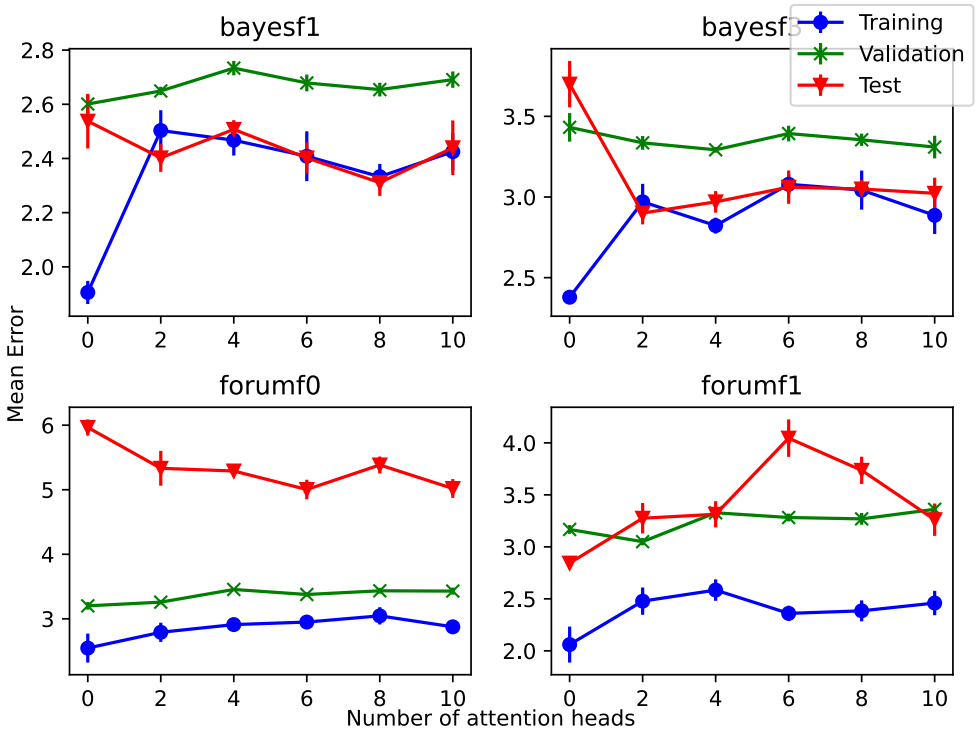


Figure 4.1: The performance of the model with different number of attention heads in the pooling layer on four datasets.

for a different test time distribution. The applied data augmentation includes random dropout of Wi-Fi signals,and addition of Gaussian noises to the RSSI values. To find the best combination of the hyperparameters, we conducted a through hyperparameter grid search, and obtain the graph fig. 4.2 demonstrating the best hyperparameter settings.

As illustrated in fig. 4.2, we found that with a 2 layer model under 0.8 dropout rate and 0.1 Gaussian noise, we obtained the best model performance with a mean error of $2.17 \pm 0.06$ on `bayesf3` dataset.

Future options for data augmentation include mixing Wi-Fi signals from fingerprints that are geographically close to the fingerprints, which might closer simulate the test environment.

## 4.3 Validation-guided training

It seems that at this point we cannot achieve better performance with traditional deep learning techniques alone, and move on to investigate methods that explicitly deal with the noisy in the dataset.

We first note the high correlation between the test set performance and validation

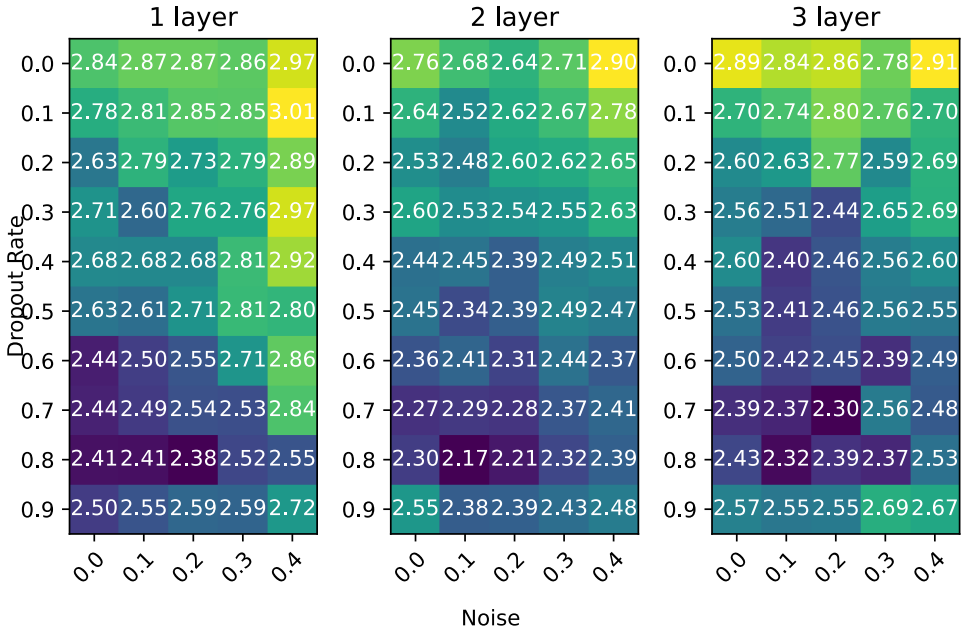


Figure 4.2: Model performance with different number of layers under different level of dropout and noise.

set performance, illustrated in fig. 4.3.

As can be observed, the validation set performance of the model is highly indicative of its performance on the clean test set, despite the fact that the validation set came from the same noisy distribution as the training set. We hypothesis that this is due to the noise in the validation set is slightly different from that of the training set, thus a model overfits to the noise on the training set will have a lower performance on the validation set, and consequently, test set.

Based on this observation, we consider using validation set performance as extra signals for training, using method proposed in [25], as reviewed in sec. 2.3. With the same hyperparameters setting discovered in sec. 4.2.2, we compare the model performance with and without guidance from validation set, illustrated in fig. 4.4. As validation-guided training takes longer to train, we increase the patience and compare the two methods.

In the top figure in fig. 4.4, we select the model with the best validation set performance. At first, it would appear that validation guidance does not consistently improve the model performance, and the improvement is somewhat negligible. However, when comparing the best model obtained via two methods by selecting the model with the

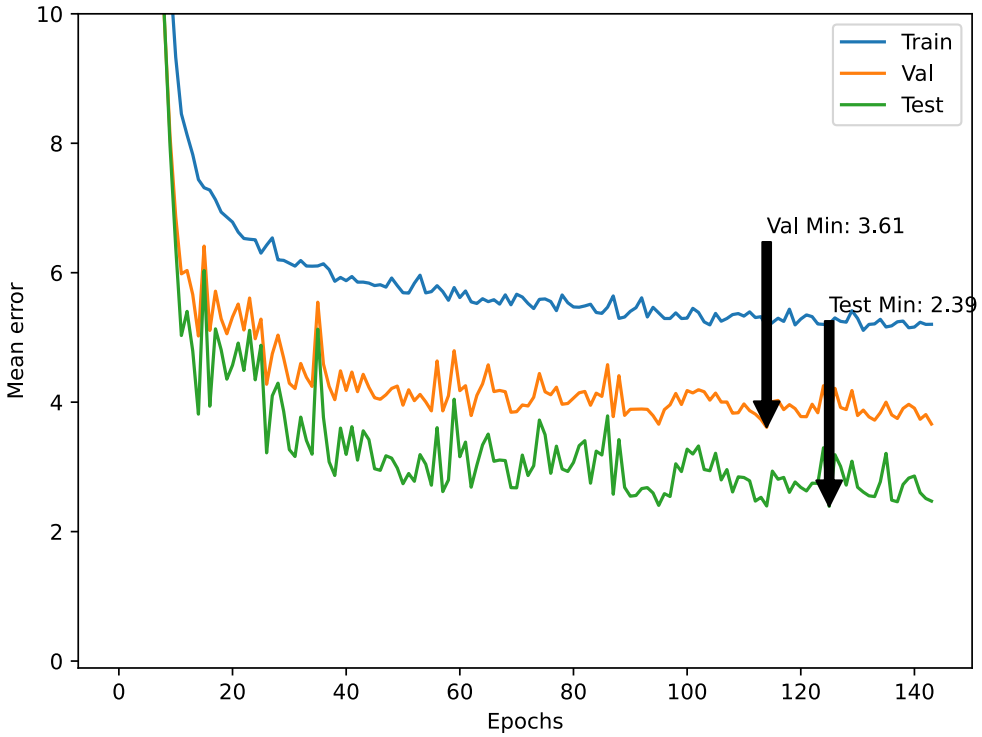


Figure 4.3: The learning curve of a model trained. We can observe the high correlation between the model performance on the validation set and test set.

highest test set performance as in the bottom figure in fig. 4.4, it can be observed that the validation-guided training consistently obtained better model when the number of training epoch increases. The exact reason for the phenomenon is analysed in sec. 5.2.

Although we know that validation guidance allow us to train longer and obtain better models along the training process, we still face the problem of how to select the best model. If we simply select the best performing model on the validation set, we will get the result as the top figure in fig. 4.4, which doesn't consistently demonstrate the advantage that validation-guided training provides. This motivates us to find a better way to select models. Another observation obtained from fig. 5.2 is that due to the shape of the contour, we might be able to obtain better performance by selecting the top-$k$ models with the best validation performance, instead of only selecting top-1 model. Although this still does not give the *best* model, we might be able to further decrease the error on the test set.

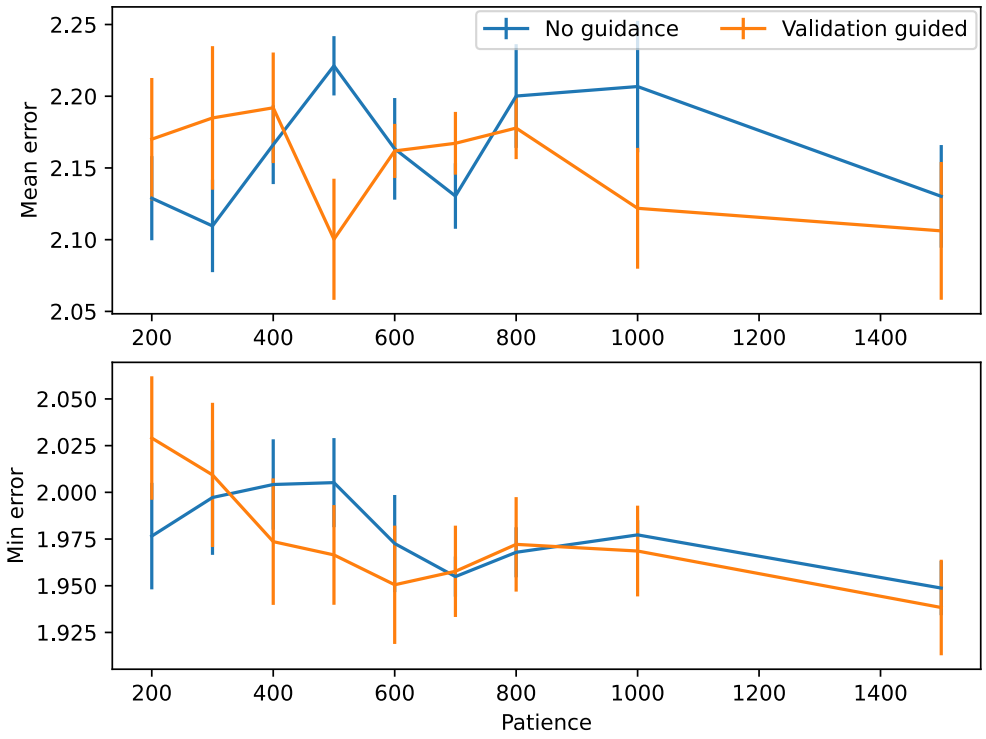


Figure 4.4: Comparison of the test set performance of the model trained with and without guidance from the validation set. The figure on top select the best performing model on the validation set, while in the bottom figure, we select the model with the best test set performance, which is not typically available.

### 4.3.1 Top-$k$ models

One problem that existed throughout the project is to select the best model along the training process. Although the validation performance is *indicative* of its test set performance, the model with the best validation set performance isn't necessarily the model with the best test set performance, as illustrated in fig. 4.3. One possible solution is to select the top-$k$ performing model on the validation set, and predict with $k$ models. We thus compare the performance of the top-$k$ model selected from the training trajectory obtained with and without validation guidance, illustrated in fig. 4.5.

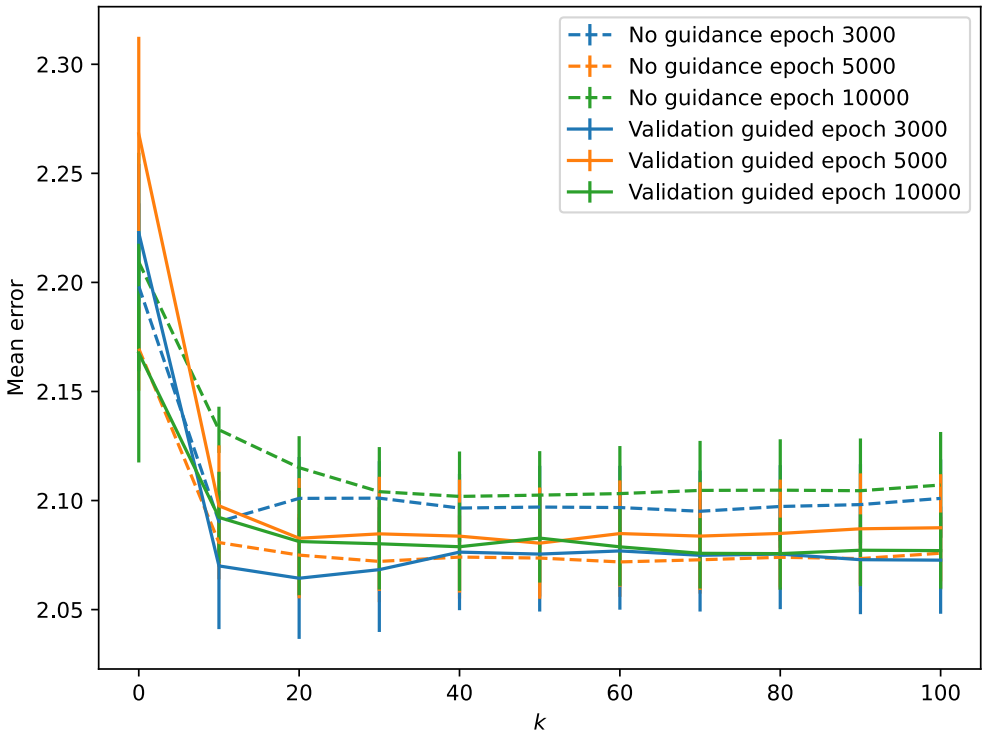


Figure 4.5: The top-$k$ performance of models trained with validation guidance and without after different number of epochs.

We observe that a appropriately selected training epochs along with validation-guided training gives the best test set performance, with a mean error of $2.064 \pm 0.027$.

### 4.3.2 Ensemble

It is natural to think that if we can select the top-$k$ models along one training trajectory, we can also select models across multiple training trajectories and form an ensemble. We provide a comparison between normally trained models and validation guided models, illustrated in fig. 4.6.

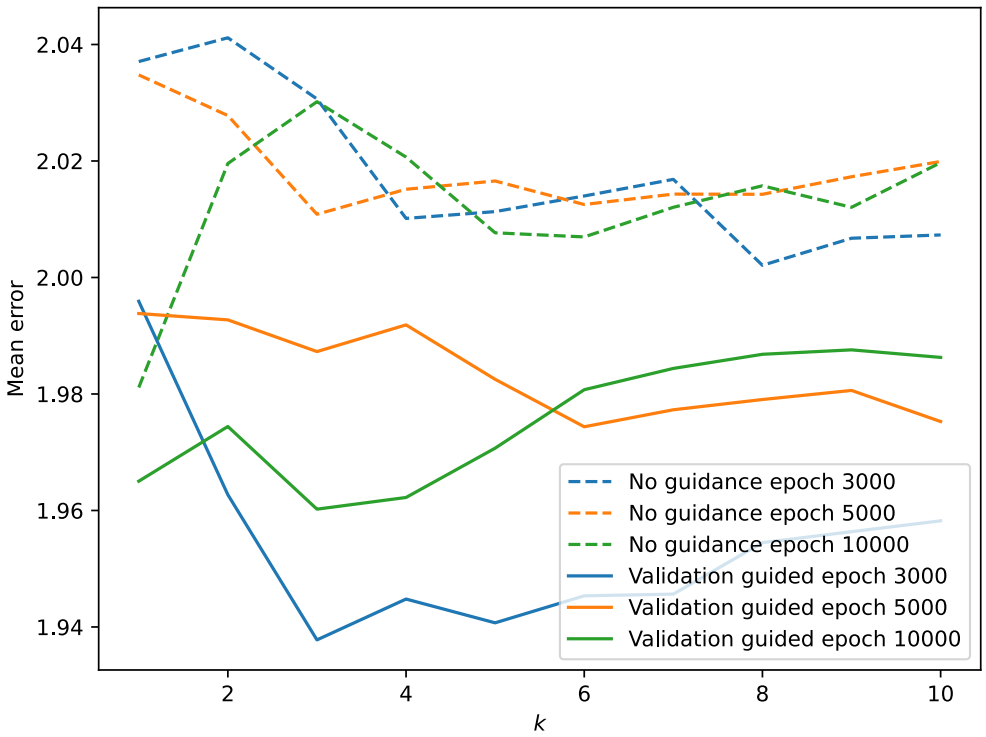


Figure 4.6: The ensemble performance of models trained with validation guidance and without after different number of epochs.

With the ensemble method, we are able to obtain a mean error of 1.93 (note this is not repeated 5 times due to the time constraint).

It is also observed that the validation-guided training works better with the ensemble method, as the ensemble of validation guided models consistently outperforms the ensemble of normally trained models, contrary to selecting models from a single trajectory as in fig. 4.5. This might be because the validation guided training allows us to train longer, and the resulting models are more diverse. The exact reason of this phenomenon is left for future work.

## 4.4 Self-supervised learning[1]

We start with the SimCLR method, using dropout rate randomly sampled between 0 and 0.8, and noise between 0 and 0.1. Initially, we consider the SimCLR method under this setting should work better than the denoising objective, as the denoising objective only requires to recognise a partially corrupted fingerprint as the original

[1]Research in this direction does not yet yield notable result due to time constraint. However, partial results are still provided to facilitate future research.

fingerprint, while the SimCLR objective needs to recognise two differently corrupted fingerprints (when the two views of the same fingerprint are both highly corrupted) as the same, in addition to the denoising objective (where one view is slightly corrupted and the other one strongly corrupted). However, we observe that as the SimCLR loss decreases, the test set performance of the linear regressor remains stable, while the KNN regressor's performance significantly deteriorates, as illustrated in fig. 4.7. This shows that SimCLR loss is not a good objective for fingerprint representation learning, and we thus abandon this direction. Some preliminary explanation are given in sec. 5.4.

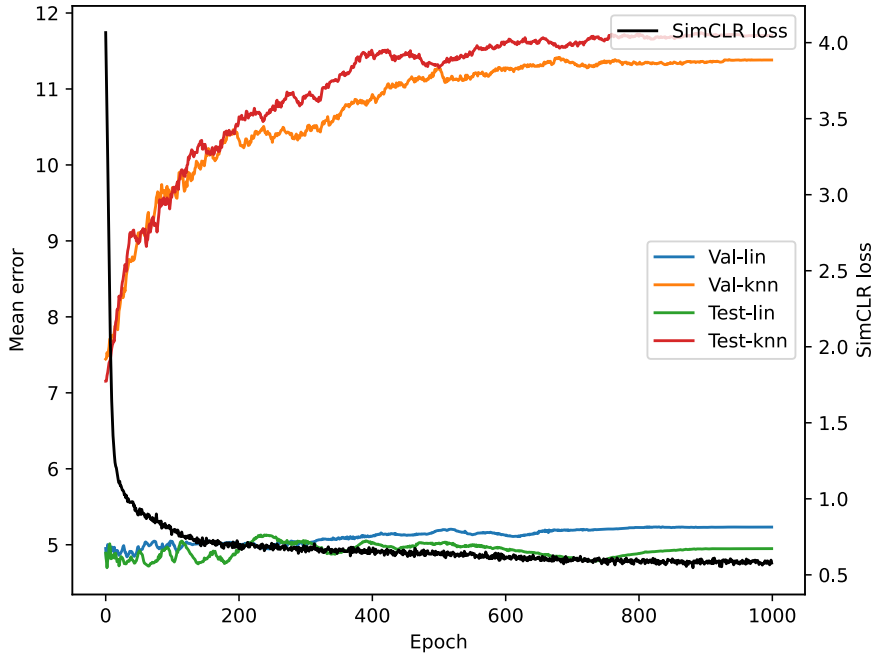


Figure 4.7: The representation quality decreases as we train with the SimCLR loss.

We then move on to train the model with denoising objective, and we observe that the model is able to learn better representation compare to the SimCLR objective. Again we perform the hyperparameter grid search, and display the results in fig. 4.8. We observe that the best model is able to achieve a mean error of $2.31 \pm 0.02$, which is worse than the regression model trained.

Finally, we provide a comparison between all the models above with their cumulative distribution functions of errors on `bayesf3` dataset in fig. 4.9.

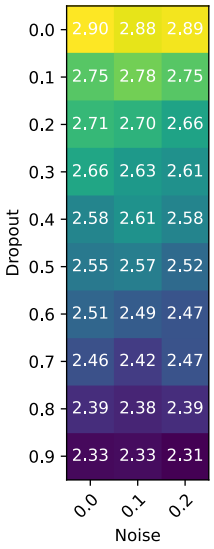


Figure 4.8: The representation quality increases as the dropout increases.

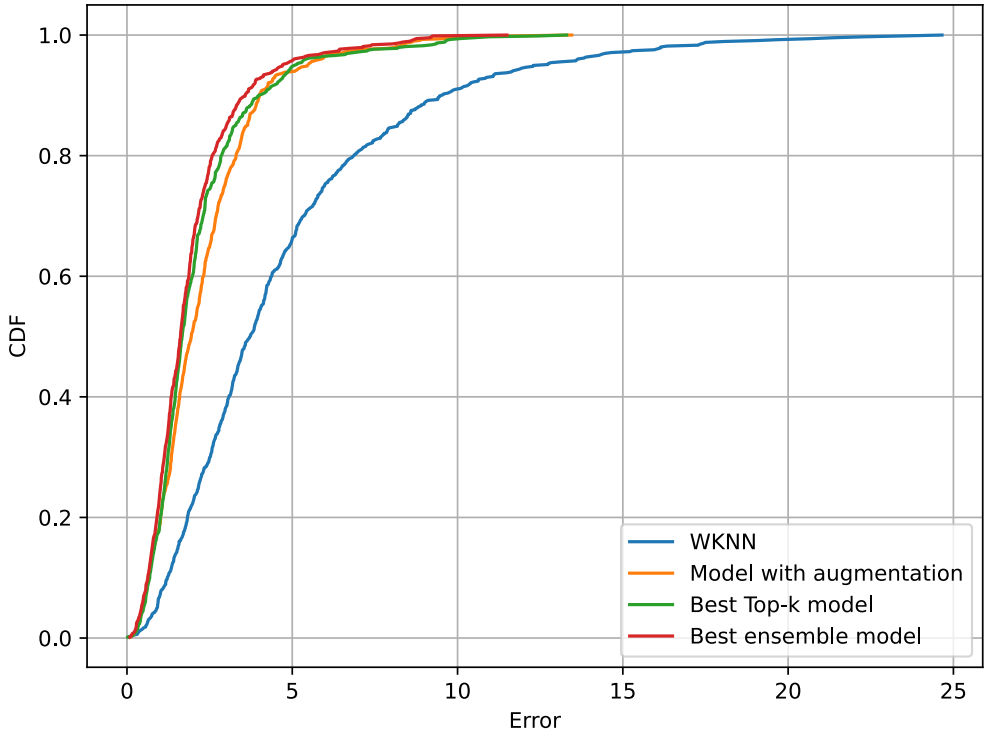


Figure 4.9: The cdf of errors of all the models investigated above. We display the cdf of a randomly selected model among the 5 repeated experiments.

# Chapter 5

# Analysis

## 5.1 Training process

In order to understand how the model is able to train on a such a extremely noisy dataset and achieve good performance on the clean dataset, we start by visualising the output of the model on the training data and test data respectively, in fig. 5.1.

As can be seen in top left figure in fig. 5.1, the model is able to converge to a path within the noisy training set that is close to the ground truth. We hypothesis that this is due to the strong regularisation (0.8 dropout rate) we applied. Wi-Fi fingerprints that are geographically close to each other share a subset of the fingerprint fragments that are similar to each other. The dropout we applied to the Wi-Fi fingerprint will thus generate training samples where the input are those fragments that are similar to each other, while the output can varies due to both the noisy in the labels and the dropout. To minimise the loss, the model will thus output the mean of the noisy labels, sort of similar to implicitly performing k-nearest neighbour on the training set, and thus find a 'path' that is close to the ground truth.

However, in hindsight, the 0.8 dropout rate is perhaps a bit too strong that the model loses some precision and always make predictions that are on the 'path'. A possible future research direction is to combine a lower dropout rate with other regularisation techniques, such as weight decay, to achieve better performance. Another potential direction would be to perform dropout in more complex ways. For example, we might better control the learned model via dropout annealing, i.e., gradually reduce the dropout rate over the training procedure to optimally control fitting to the data. Or, we could tailor the data augmentation scheme by more accurately control the dropout rate. Instead of using one fixed dropout rate, we could use a distribution of dropout rate to generate

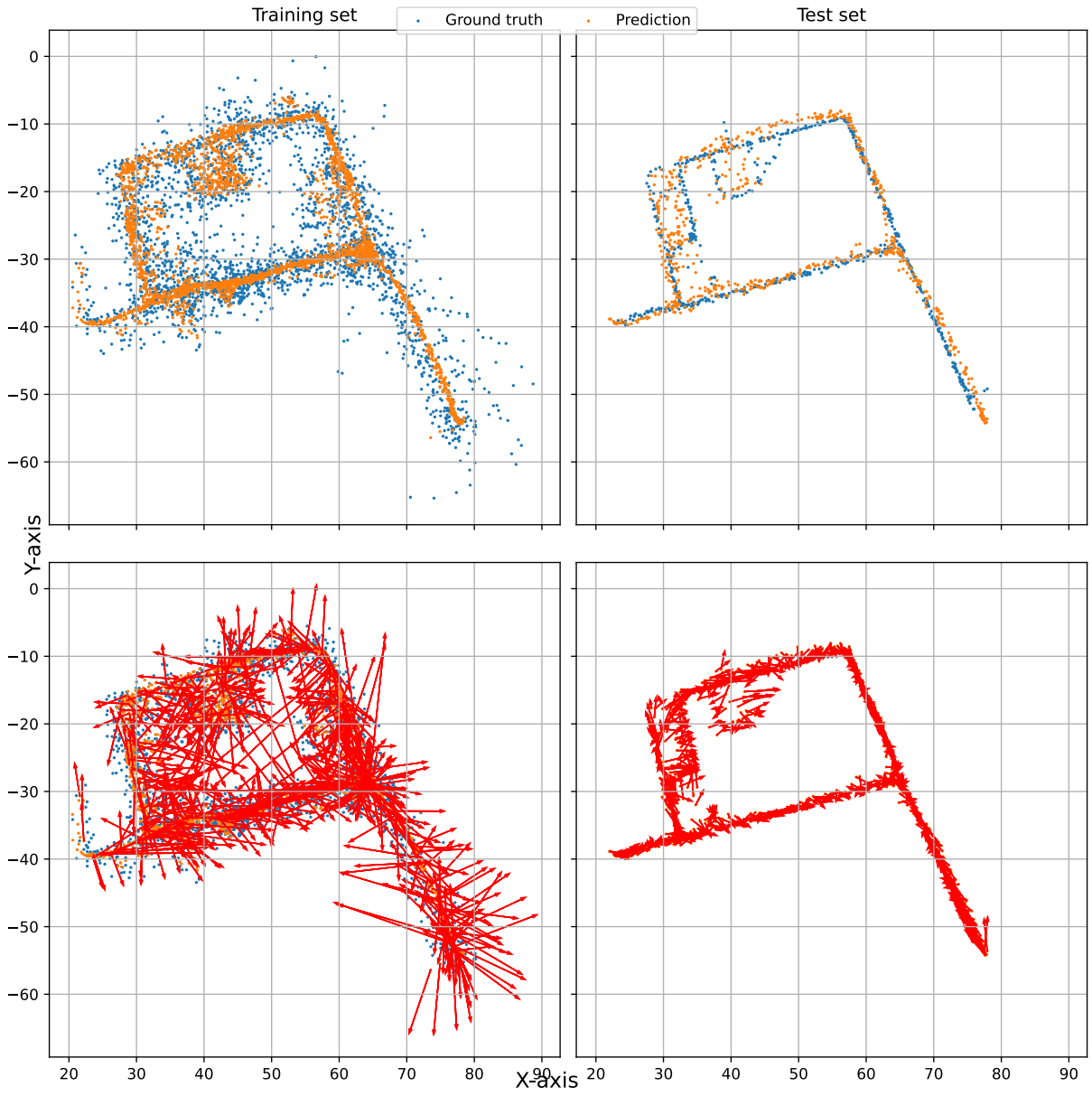


Figure 5.1: The visualisation of model output on training set and test set. The top two figures simply overlay the model prediction and ground truth, while the bottom two figures points an arrow from the model prediction to ground truth. The two figures on the left are model output on the training data, and the two figures on the right are model output on the test data. Note that some arrows are not plotted in the training set to avoid cluttering.

a data distribution that allows the model to learn better representation on the noisy dataset.

In addition, when we perform hyperparameter grid search in sec. 4.2.2, we compare the performance of models by selecting the model with the best validation set performance. However, as we have shown in sec. 5.2, this may not show the hyperparameter setting that allow us to learn the best models. We might be able find a better hyperparameter combination that allows the model to better learn on the training set by selecting the model with the best test set performance instead.

Suggestions for future work would be to separate the model selection process with model training process. During training, the focus should be on which method would allow the model to learn mapping that have the best test set performance, and consider the appropriate selection process later on. Mixing the two problem will cause confusion as to which factor is causing the problem.

## 5.2 Validation-guided training

To better understand how validation-guided training helps with training better models, we trained two models with normal training method and validation-guided training for 15000 epochs, and visualise the relation between validation set performance and test set performance along their training trajectories in fig. 5.2.

As can be seen in fig. 5.2, model trained without validation guidance will gradually overfit to the training data as the training epoch increases, shown as the shift of concentration of dots to the bottom left, and model selected with low validation loss will have higher test loss. In contrast, model trained with validation guidance does not exhibit such a overfitting process, and can thus train longer and discover better ways to fit the data. Although it is possible to train the model normally with early stopping and hope for a decent test set performance, validation guidance provides more guarantees as we can safely train the model for a very long time without overfitting, and consequently the model has more opportunities to discover better ways to fit the data.

The effectiveness of the validation-guided training suggests there exists some internal ‘consistency’ between training samples, i.e., fitting the model to some noisy data will causes the model performance to decrease on some other samples, and validation-guided training utilise this property to provide some regularisation during training. A interesting research direction would be to design a metric to evaluate the consistency between samples, based on which sample selection can be performed.

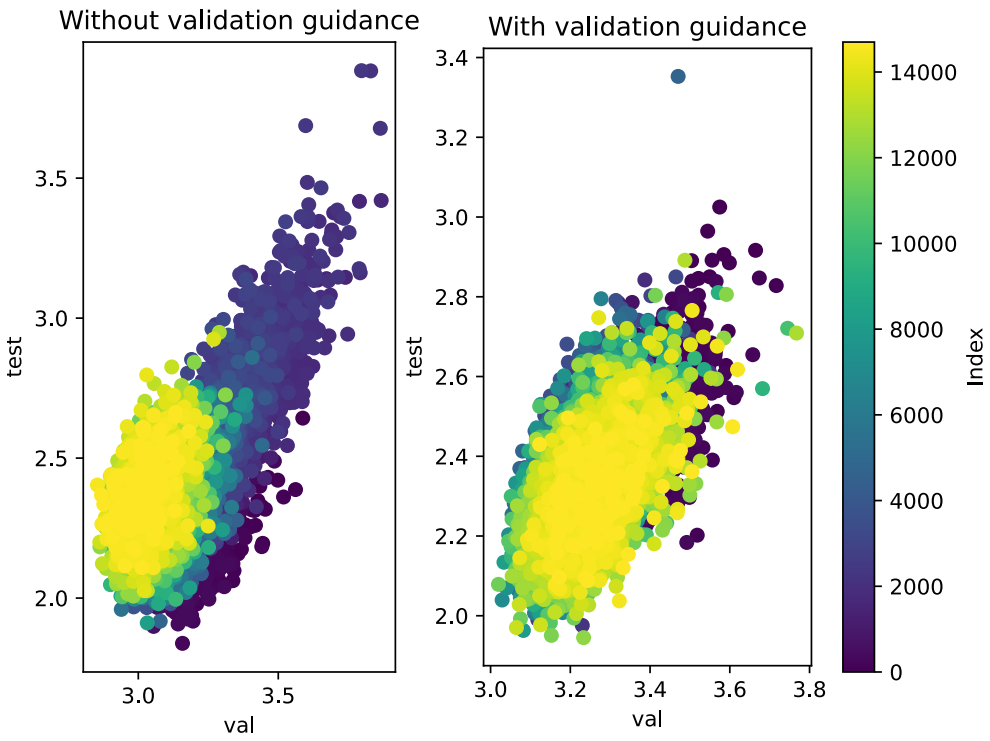


Figure 5.2: Relation between validation set performance and test set performance of the model trained with (right) and without (left) guidance from the validation set.

On the other hand, the current validation set is selected randomly from the same validation set on the training set. With our model trained that perform well on clean test, we can easily select samples that are more likely to be clean in the training samples, via loss modelling [28], etc. A low hanging fruit for future research would be to construct a cleaner validation set to better provide guidance for training. Note that the validation set does not need to be completely clean and noiseless, as guidance that are in the right direction most of the time is sufficient to provide regularisation and prevent overfitting, as demonstrated in fig. 5.2.

In addition, the fact that the validation set and training set come from the same noisy distribution suggests that it might be possible to combine the two sets together and train the model, which gives more training samples model and thus should give better performance. In terms of guidance, instead of sampling a mini-batch from the validation set and calculate the weight for each sample, we could sample the mini-batch from the combined set to provide guidance. In addition, it might be worth investigating the effect of the batch size of the sampled mini-batch. A larger batch size could theoretically provides more accurate guidance during training. This will be left for future research due to the time constraint.

## 5.3 Model selection

In sec. 4.3.1 and sec. 4.3.2, we have shown that we can achieve significant performance improvement by selecting multiple models along one or more training trajectories. However, the selected model is still not the best model we can find (the minimum of the test set loss is usually around 1.89 meters and the ensemble method is only able to achieve 1.93). A future research direction is to find a better way to select models along the training trajectory. One particular method possible is to filter out samples in the validation set that are too noisy. This can be achieved by modelling the distribution of the validation set loss, as in [28]. By creating a cleaner validation set, we can better select models that are more likely to generalise to the test set.

## 5.4 SimCLR

As shown in fig. 4.7, training with SimCLR objective will in fact, deteriorate the learned representation. We hereby hypothesis the potential reasons for this phenomenon.

Consider two fingerprint that are close to each other in the coordinate space. These two fingerprints will share some fragments that are similar, and some fragments that are not. The SimCLR objective will pull the representations of the two dissimilar fragments apart, as they are not the same fingerprint. However, these two fragments should be pull together, as they are close to each other in the coordinate space. This will cause the representations to be less suitable for WKNN.

However, this hypothesis is not verified, as a complicated visualisation tool is required to visualise the latent space of the model. This will be left for future research.

## 5.5 Extra thoughts

A problem that plagued the project is the hardness to understand the dataset. Although we can visualise the geographical property of the dataset and the behaviour of the model as in fig. 5.1, we did not take into account the Wi-Fi fingerprint, as this is difficult to visualise and thus prevents us from understanding the dataset better. It is thus recommended that a visualisation tool be developed in the future to better understand the dataset and the noise inside. In addition, it might be beneficial to visualise the latent space of the model using t-SNE [42] to better understand the model behaviour.

Here I attach some personal experiences gained during the dissertation. During

the project, a lot of time was spent on hyperparameter grid search in order to be 'comprehensive' and 'thorough'. However, this is not a good use of time, as the hyperparameter space is usually very large, and a complete hyperparameter search does not offer any insight into the problem. In addition, by performing statistical analysis on the grid search results, many valuable information contained in the training curve is lost. Instead, a more interactive and iterative process should be used to quickly prototype, test ideas and finding a rough range for hyperparameter where models works well, and the hyperparameter search should only be performed when the model is stable and in a constrained hyperparameter space. An alternative to the manual hyperparameter search approach is to use hyperparameter optimisation library such as Optuna [43] to automate the process.

# Chapter 6

# Conclusion

In this dissertation, we investigate the problem of indoor localisation using noisy dataset. We first propose a novel Wi-Fi encoder that uses transformer architecture to process the Wi-Fi fingerprints, and show that it is able to encode Wi-Fi fingerprints into a latent space that is more suitable for localisation compare to encoding the fingerprints as fix-size vectors. We then find the best data augmentation settings using dropout and Gaussian noise for training the model without overfitting to the noise. The exact mechanism for training on extremely noisy dataset are investigated in sec. 5.1, and some relevant future research direction are discussed.

Next, we investigate the effect of validation-guided training on the performance of the model. We showed that in combination with good model selection techniques, validation-guided training allows us to better train the model without overfitting to the data, and gives significant improvement to the model performance. We show that the validation-guided training provides a principled approach to train models on a noisy dataset without overfitting to the noise, even when a clean validation set is not available. It provides another weapon in the arsenal of machine learning practitioners to deal with noisy data, and can be used in conjunction with other techniques such as dropout to achieve better performance. We also discuss how to improve this method in sec. 5.2.

Combining all the techniques we investigated, including data augmentation, validation-guided training, and ensemble method, we are able to achieve a mean error of 1.93 meters on the test set, 1.77 meter better than training the model directly on the noisy dataset.

# Chapter 7

# References


[1] S. He and S.-H. G. Chan, "Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 466–490, 2016, doi: 10.1109/COMST.2015.2464084.

[2] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, Mar. 2000, vol. 2, pp. 775–784 vol.2, doi: 10.1109/INFCOM.2000.832252.

[3] Y. Dong, G. He, T. Arslan, Y. Yang, and Y. Ma, "Crowdsourced Indoor Positioning with Scalable WiFi Augmentation," *Sensors (Basel)*, vol. 23, no. 8, p. 4095, Apr. 2023, doi: 10.3390/s23084095. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10146501/. [Accessed: Aug. 19, 2023]

[4] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based Fingerprinting for Indoor Localization: A Deep Learning Approach," *IEEE Trans. Veh. Technol.*, pp. 1–1, 2016, doi: 10.1109/TVT.2016.2545523. [Online]. Available: http://ieeexplore.ieee.org/document/7438932/. [Accessed: Apr. 23, 2023]

[5] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, Seattle Washington, Jun. 2005, pp. 205–218, doi: 10.1145/1067170.1067193 [Online]. Available: https://dl.acm.org/doi/10.1145/1067170.1067193. [Accessed: Apr. 23, 2023]

[6] X. Zhu and Y. Feng, "RSSI-based Algorithm for Indoor Localization," *CN*, vol. 5, no. 2, pp. 37–42, 2013, doi: 10.4236/cn.2013.52B007. [Online]. Available: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/cn.2013.52B007. [Accessed: Apr. 23, 2023]

[7] J. Niu, B. Wang, L. Cheng, and J. J. P. C. Rodrigues, "WicLoc: An indoor localization system based on WiFi fingerprints and crowdsourcing," in *2015 IEEE International Conference on Communications (ICC)*, Jun. 2015, pp. 3008–3013, doi: 10.1109/ICC.2015.7248785.

[8] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: zero-effort crowdsourcing for indoor localization," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, Istanbul Turkey, Aug. 2012, pp. 293–304, doi: 10.1145/2348543.2348580 [Online]. Available: https://dl.acm.org/doi/10.1145/2348543.2348580. [Accessed: Aug. 22, 2023]

[9] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, Istanbul Turkey, Aug. 2012, pp. 269–280, doi: 10.1145/2348543.2348578 [Online]. Available: https://dl.acm.org/doi/10.1145/2348543.2348578. [Accessed: Aug. 22, 2023]

[10] B. Ferris, D. Haehnel, and D. Fox, "Gaussian Processes for Signal Strength-Based Location Estimation," in *Robotics: Science and Systems II*, Aug. 2006, doi: 10.15607/RSS.2006.II.039 [Online]. Available: http://www.roboticsproceedings.org/rss02/p39.pdf. [Accessed: Apr. 23, 2023]

[11] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, Seattle Washington, Jun. 2005, pp. 205–218, doi: 10.1145/1067170.1067193 [Online]. Available: https://dl.acm.org/doi/10.1145/1067170.1067193. [Accessed: Apr. 24, 2023]

[12] B. Chidlovskii and L. Antsfeld, "Semi-supervised Variational Autoencoder for WiFi Indoor Localization," in *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Sep. 2019, pp. 1–8, doi: 10.1109/IPIN.2019.8911825.

[13] S. He and S.-H. G. Chan, "Sectjunction: Wi-Fi indoor localization based on junction of signal sectors," in *2014 IEEE International Conference on Communications (ICC)*, Jun. 2014, pp. 2605–2610, doi: 10.1109/ICC.2014.6883716.

[14] Y. Xie, Y. Wang, A. Nallanathan, and L. Wang, "An Improved K-Nearest-Neighbor Indoor Localization Method Based on Spearman Distance," *IEEE Signal Processing Letters*, vol. 23, no. 3, pp. 351–355, Mar. 2016, doi: 10.1109/LSP.2016.2519607.

[15] J. Oh and J. Kim, "AdaptiveK-nearest neighbour algorithm for WiFi fingerprint positioning," *ICT Express*, vol. 4, no. 2, pp. 91–94, Jun. 2018, doi: 10.1016/j.icte.2018.04.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240595951830050X. [Accessed: Aug. 23, 2023]

[16] J. Ren, Y. Wang, C. Niu, W. Song, and S. Huang, "A Novel Clustering Algorithm for Wi-Fi Indoor Positioning," *IEEE Access*, vol. 7, pp. 122428–122434, 2019, doi: 10.1109/ACCESS.2019.2937464. [Online]. Available: https://ieeexplore.ieee.org/document/8812713/. [Accessed: Aug. 23, 2023]

[17] M. Abbas, M. Elhamshary, H. Rizk, M. Torki, and M. Youssef, "WiDeep: WiFi-based Accurate and Robust Indoor Localization System using Deep Learning," in *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, Mar. 2019, pp. 1–10, doi: 10.1109/PERCOM.2019.8767421.

[18] Y. Wang, J. Gao, Z. Li, and L. Zhao, "Robust and Accurate Wi-Fi Fingerprint Location Recognition Method Based on Deep Neural Network," *Applied Sciences*, vol. 10, no. 1, p. 321, Jan. 2020, doi: 10.3390/app10010321. [Online]. Available: https://www.mdpi.com/2076-3417/10/1/321. [Accessed: Aug. 20, 2023]

[19] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion."

[20] X. Song *et al.*, "A Novel Convolutional Neural Network Based Indoor Localization Framework With WiFi Fingerprinting," *IEEE Access*, vol. 7, pp. 110698–110709, 2019, doi: 10.1109/ACCESS.2019.2933921. [Online]. Available: https://ieeexplore.ieee.org/document/8792196/. [Accessed: Aug. 22, 2023]

[21] Z. Chen, H. Zou, J. Yang, H. Jiang, and L. Xie, "WiFi Fingerprinting Indoor Localization Using Local Feature-Based Deep LSTM," *IEEE Systems Journal*, vol. 14, no. 2, pp. 3001–3010, Jun. 2020, doi: 10.1109/JSYST.2019.2918678.

[22] H. Turabieh and A. Sheta, "Cascaded Layered Recurrent Neural Network for Indoor Localization in Wireless Sensor Networks," *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, pp. 1–6, Oct. 2019, doi: 10.1109/ICTCS.2019.8923086. [Online]. Available: https://ieeexplore.ieee.org/document/8923086/. [Accessed: Aug. 22, 2023]

[23] V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab, "Robust Optimization for Deep Regression." arXiv, Sep. 22, 2015 [Online]. Available: http://arxiv.org/abs/1505.06606. [Accessed: Apr. 24, 2023]

[24] B. Frenay and M. Verleysen, "Classification in the Presence of Label Noise: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, May 2014, doi: 10.1109/TNNLS.2013.2292894.

[25] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to Reweight Examples for Robust Deep Learning," Mar. 2018 [Online]. Available: https://www.semanticscholar.org/paper/Learning-to-Reweight-Examples-for-Robust-Deep-Ren-Zeng/c5420ef59d7508d82e53671b0d623027eb58e6ed. [Accessed: Jun. 22, 2023]

[26] D. Arpit *et al.*, "A Closer Look at Memorization in Deep Networks." arXiv, Jul. 01, 2017 [Online]. Available: http://arxiv.org/abs/1706.05394. [Accessed: Aug. 23, 2023]

[27] B. Han *et al.*, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," Apr. 2018.

[28] J. Li, R. Socher, and S. Hoi, "DivideMix: Learning with Noisy Labels as Semi-supervised Learning," *ArXiv*, Feb. 2020.

[29] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "MixMatch: A Holistic Approach to Semi-Supervised Learning," *ArXiv*, May 2019.

[30] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty." arXiv, Oct. 29, 2019 [Online]. Available: http://arxiv.org/abs/1906.12340. [Accessed: Apr. 16, 2023]

[31] X. Liu *et al.*, "Self-supervised Learning: Generative or Contrastive," *IEEE Trans. Knowl. Data Eng.*, pp. 1–1, 2021, doi: 10.1109/TKDE.2021.3090866. [Online]. Available: https://ieeexplore.ieee.org/document/9462394/. [Accessed: Jan. 29, 2023]

[32] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," *ArXiv*, Feb. 2020 [Online]. Available: https://www.semanticscholar.org/paper/A-Simple-Framework-for-Contrastive-Learning-of-Chen-Kornblith/34733eaf66007516347a40ad5d9bbe1cc9dacb6b. [Accessed: Jul. 06, 2023]

[33] R. D. Hjelm *et al.*, "Learning deep representations by mutual information estimation and maximization." arXiv, Feb. 22, 2019 [Online]. Available: http://arxiv.org/abs/1808.06670. [Accessed: Jul. 08, 2023]

[34] D. H. Ballard, "Modular learning in neural networks," in *Proceedings of the sixth National conference on Artificial intelligence - Volume 1*, Seattle, Washington, Jul. 1987, pp. 279–284.

[35] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, May 2019 [Online]. Available: http://arxiv.org/abs/1810.04805. [Accessed: May 04, 2022]

[36] W. Sun, M. Xue, H. Yu, H. Tang, and A. Lin, "Augmentation of Fingerprints for Indoor WiFi Localization Based on Gaussian Process Regression," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10896–10905, Nov. 2018, doi: 10.1109/TVT.2018.2870160. [Online]. Available: https://ieeexplore.ieee.org/document/8464281/. [Accessed: Aug. 20, 2023]

[37] S. Yean, P. Somani, B.-S. Lee, and H. L. Oh, "GAN+: Data Augmentation Method Using Generative Adversarial Networks and Dirichlet for Indoor Localisation," 2021.

[38] X. Wang, Z. Chen, S. Zhang, and J. Zhu, "Super-Resolution Based Fingerprint Augment for Indoor WiFi Localization," *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pp. 1–6, Dec. 2020, doi: 10.1109/GLOBECOM42002.2020.9348146. [Online]. Available: https://ieeexplore.ieee.org/document/9348146/. [Accessed: Aug. 21, 2023]

[39] A. Goyal *et al.*, "Recurrent Independent Mechanisms." arXiv, Nov. 17, 2020 [Online]. Available: http://arxiv.org/abs/1909.10893. [Accessed: Jan. 20, 2023]

[40] A. Vaswani *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, 2017, vol. 30 [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html. [Accessed: May 08, 2022]

[41] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization." arXiv, Jan. 04, 2019 [Online]. Available: http://arxiv.org/abs/1711.05101. [Accessed: Aug. 20, 2023]

[42] L. Maaten and G. E. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, 2008 [Online]. Available: https://www.semanticscholar.org/paper/Visualizing-Data-using-t-SNE-Maaten-Hinton/1c46943103bd7b7a2c7be86859995a4144d1938b. [Accessed: Aug. 21, 2023]

[43] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework." arXiv, Jul. 25, 2019 [Online]. Available: http://arxiv.org/abs/1907.10902. [Accessed: Aug. 22, 2023]