# ANOMALY DETECTION IN THE HOME WITH SEISMIC SENSORS

by

Siraphat Boonchan

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Data Science and Artificial Intelligenc

| | |
|---|---|
| Examination Committee: | Prof. Matthew N. Dailey (Chairperson)<br>Dr. Mongkol Ekpanyapong<br>Dr. Attaphongse Taparugssanagorn |
| Nationality: | Thai |
| Previous Degree: | Bachelor of Engineering in Electrical Engineering<br>Kasetsart University<br>Thailand |
| Scholarship Donor: | The Royal Thai Government |

Asian Institute of Technology
School of Engineering and Technology
Thailand
August 2021

# ABSTRACT

Falls is a global public public health problem. Falls happen to people of all ages, especially on the elderly. Throughout the last decade, we have seen improvements in fall detection system due to technology development and the revolution of deep learning. However, using vibration signal analysis can compensate the weakness and also overcomes the drawbacks associated with the traditional system, and this is a novel idea that needs to be studied further. This thesis studies the embedded system and design space for unsupervised anomaly detection model using modern deep learning best practices. The performance and effectiveness of this system to immediately send alert message to user via LINE apllication when abnormal events occur.

**Keywords:** falling down, anomaly detection, deep learning, Transformer model.

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 Background of the Study

Globally, injuries after a fall is a significant public health problem. Each year, approximately 37 million falls requires medical attention, and approximately 684,000 individuals die from falls. Falls are the second leading cause of unexpected injury death, after road traffic injuries. According to the World Health Organization (WHO, 2018), the highest death rate from falls in all regions around the world was faced among adults who are over the age of 60 years. The frequency of falling down increases with age and weakness level. In the future, injuries caused by falls will affect more civilians as the population ages, and fall deaths are expected to double by 2030. According to Fuller (2013), The elderly, who represent 12 percent of the population, account for 75 percent of those who die from falls.

In addition, the Ministry of Public Health in Thailand (ThaiNCD.com, 2019) says that one-third or greater than 3 millions of Thailand's people fall in their homes every year. Approximately 66% of the cases involve slippery floors, stumblings and missing a step on the same ground level. They report an average of 140 calls to local ambulances per day, and on average, 2 people die each day. More than 55% of falls occur inside the home environment (Pynoos, Steinman, & Nguyen, 2010), most frequently in the bathroom, kitchen and dining room. Therefore, when victims fall, if nobody knows about the accident, and nobody takes care of the victim immediately, it can result in more serious injury, long term impairments, and even death.

From the statistics mentioned above, developing any technology able to help decrease or mitigate false will be useful. I am specifically interested in artificial intelligence approaches to detection of fall events that can also immediately alert caretakers or assistants.

## 1.2 Statement of the Problem

There has been great deal of research on fall detection. Researchers try to find the best methods to detect and mitigate falls. Each approach has pros and cons, depending on the situation and the environment as following:

1. User-activated fall alert with a pendant: Although manually-activated fall alarms are simple and low cost, they are only successful when a user who has fallen activates the alarm button by himself or herself manually. This system is ineffective if the person is not wearing the pendant because he or she refuses to press the emergency button, forgets it, or cannot press it. Elders may hesitate to push an emergent button for several reasons such as concern about bothering others and privacy.
2. Automatic Wearable Devices (Degen, Jaeckel, Rufer, & Wyss, 2003; Yang & Hsu, 2010; Rihana & Mondalak, 2016): This solution is popular because it is uncomplicated and provides high accuracy. Devices in this group are based on inertial measurement units (IMU)s, which contain an accelerometer and gyrometer. A significant disadvantage of this solution is that the user has to wear the device all the time, which can lead to discomfort, and if the device cannot be wore in the shower, the device will miss the period in which individuals have the highest probability of falling. Moreover, a wearable may even cause injury when people fall down.
3. Cameras (Tsai & Hsu, 2019; Ramirez et al., 2021; Taufeeque, Koita, Spicher, & Deserno, 2021): Many researchers have developed camera-based systems to detect falls, since cameras can track residents, and falls can be detected based on image processing algorithms trained to identify abnormal activity. However, the drawbacks of cameras are that residents may feel uncomfortable and concerned about privacy, even if the images are not leaked. Moreover, when a victim falls in a place out of view of the camera, e.g. an aed occluded by furniture, the method cannot alert caretakers. Also, cameras cannot be installed in the toilet or bathroom, again missing some of the highest risk periods of time.
4. Vibration analysis (Alwan et al., 2006; Liu, Jiang, Su, Benzoni, & Maxwell, 2019; Clemente, Li, Valero, & Song, 2020): This approach has not been explored as much as the others. Vibration has several limitations in terms of data collection:
    - Vibration sensors: The general sensors popular in the commercial market have low sensitivity. When the floor is concrete, it is quite difficult to detect vibrations with a general sensor. Traditional high sensitivity vibration sensor requires embedding in the ground, making it difficult to install. In addition, when the area is large, more sensors are required, which increases cost and complexity of the system.
    - Sample fall data: White falls can be simulated to get data for IMU or camera sensors, vibration data from a fall have specific characteristics

depending on the type of floor, the weight of the subject, and the distance of the sensor to the locus of the event. Realistically, real falls on concrete and other hard surfaces are too dangerous to simulated.

Despite these limitations, the benefits of the vibration signals for fall detection does overcome the drawbacks associated with all previous methods. As vibration signals have been analyzed further to include human activity and peoples' heart rates (Jia, Howard, Zhang, & Zhang, 2017), using vibrational signals to detect falls may significantly advance the technology available in this area, and it help mitigate the elderly fall problem.

Research on vibration data has thus far used supervised classification models including k-nearest-neighbors (Shao et al., 2020), support vector machines (S. Wang, Chen, Zhou, Sun, & Dong, 2015; Kasturi & Jo, 2017; Liu et al., 2019), and neural networks (Sultana, Deb, Dhar, & Koshiba, 2021). Others have used unsupervised learning methods such as k-means (Shao et al., 2020) and simple amplitude thresholds to classify fall events (Alwan et al., 2006; Charlon, Bourennane, Bettahar, & Campo, 2013; Britto Filho & Lubaszewski, 2020). Classification with supervised data requires collecting real fall data, which, as mentioned above, is dangerous, because faking a fall can lead to serious injury if we make a mistake while doing an experiment. Liu et al., (2019) solve this problem using dummy humans, but realistic dummies are expensive.

As falls occur infrequently and diversely, and there also are several types of falls such as forward falls, backward falls and lateral falls (El-Bendary, Tan, C. Pivot, & Lam, 2013), any attempt to exhaustively train a supervised classifier can lead to a lack of sufficient data for training. Although, falling events occurring during different activities such as walking, standing, sleeping, or sitting share some characteristics in common, they also have significant differences (X. Wang, Ellul, & Azzopardi, 2020). It is difficult to anticipate all possible patterns in advance. Furthermore, as fall events rarely occur in daily life, if we train a model with an imbalanced dataset, it can result in bias.

Anomaly detection methods may be the key to addressing all of these issues. I will apply anomaly detection methods to detect adverse event such as falls indirectly. The main advantage of anomaly detection beside addressing the diversity of fall is that anomaly detection will not only detect falls but also detect other abnormal activities such as fighting and any other activities the model is not trained on.

### 1.3 Research Questions

The purpose of this paper is to develop a robust automated anomaly detection system capable of detecting falls and other anomalies by combining knowledge from signal processing, embedded systems, machine learning, and edge devices. The study aims to answer the following questions:

1. Can a seismic sensor and an embedded system detect human activity on the surface of a typical concrete floor in the home?
2. What are the best methods for detecting anomaly events such as falls using seismic sensors?
3. Can a system be designed and implemented that identifies falls in daily human activities in real time?
4. Can the system thus designed be deployed in real home environments?

### 1.4 Objectives of the Study

The main objective of this study is to alert caretakers immediately when an anomalous event such as a fall occurs in the home. To fulfill this main objective, I will take the following specific steps:

1. Design and build a filter, amplifier, and embedded system to digitize and analyze signals from seismic sensors characterizing human activities.
2. Collect data on daily human activities by many subjects.
3. Build an anomaly detection and alerting system for detecting anomaly patterns.
4. Deploy the model in the dining room in my home.
5. Evaluate the deployed model in terms of its accuracy in identifying unusual events.

### 1.5 Scope and Limitations

The scope and limitations of this study are as follows :

1. The study will focus on concrete floor because most household floors in Thailand are concrete material covered with tile.
2. I assume the home has only a single elderly person.
3. Accuracy may suffer if multiple people are present and active at the same time.

### 1.6 Contributions

Transformer model for anomaly detection in time series.

# CHAPTER 2

# Literature Review

Nowadays, Falls are concernable problem around the world. Fall detection is an interested topic that researchers prefer to receive the best accuracy. Several methods have tried to overcome this problem, but they have suffered with a lot of constrains. Nonetheless, using vibration signal to detect fall actions may highly modernize in order to mitigate senile fall problem.

There are several knowledge related fields which start from vibration untill artificial intelligence model, and every section of this system as software and hardware are equally important. Thus, we have to explore and deeply understand in each branch in order to build the best system.

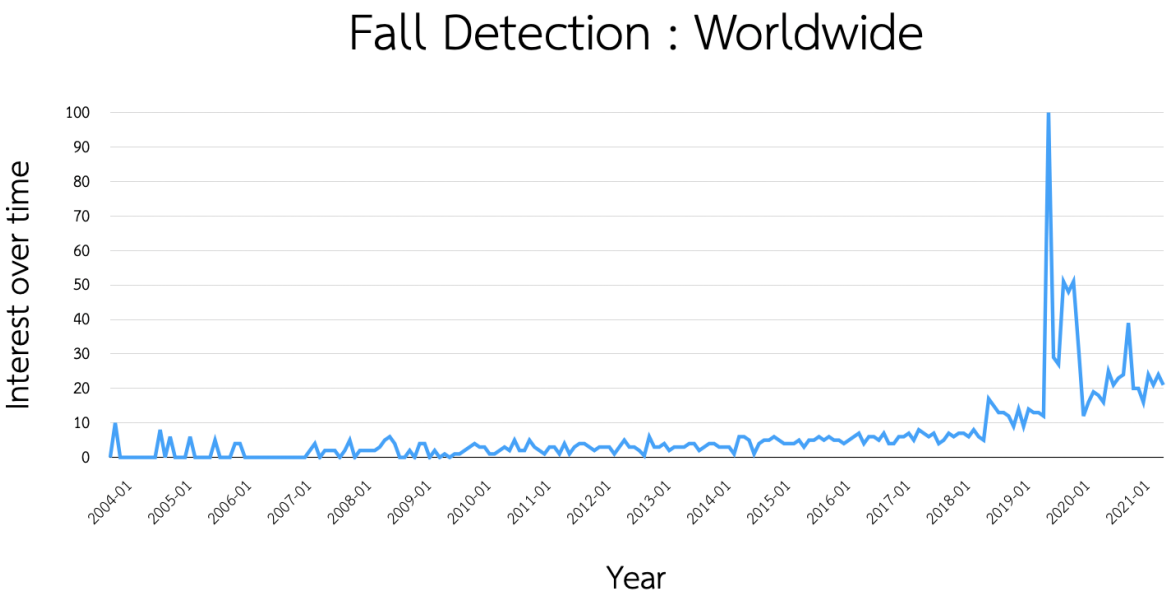
## 2.1 Fall

Falls happen to people of all ages, but older people have a high probability of being harmed and are more likely to fall, especially if they have an abnormal health conditions or balance problems. Falls are a common but often disregarded cause of injury. According to NHS (2019) one in three adults over 65 and half of the people over 80 have at least one fall per year. Most falls do not result in serious injury, but there is always a risk that a fall could lead to broken bones, and it can cause the person to have paralysis. In addition, the level of injury depends on the timeliness of the assistance. Unintentional falls can cause severe injuries and even death, especially if no immediate assistance is given.

### *2.1.1 Fall Detection*

Global trends in fall detection are illustrated in Figure 2.1. The data are downloaded from Google Trends with the search topic “Fall Detection”. Fall detection has gotten increasingly more attention over time and significantly increased in 2019. The values are indexed to be 100, where 100 is the maximum search interest for that period of time with specific location. Researchers have developed systems using a variety of different sensors and methods depending on their proposes and technological industry. Consequently, we can conclude that this topic is of interest and is becoming increasingly popular.

**Figure 2.1**
Interest in “Fall Detection” over time from 2004 to present according to Google Trends.




### *2.1.2 Fall detection by using vibration sensors*

Table 2.1 shows the evolution of fall detection from floor vibration. Most researches use classifier models to detect fall events with training performed on simulated fall data that were not real falls. Furthermore, none of these researchers have deployed their system in real environments, so the real world performance of the models is not convincing. To overcome these weaknesses, I will apply anomaly detection to compensate for the rarity of events such as real falls and other or unseen patterns, and send alerts to the caretaker or an assistance who can take care of a victim who is alone as soon as possible.

**Table 2.1**
*Summary of literature review for fall detection from floor vibration.*

| Authors | Data Collection | Sensors | Algorithms | Alarm |
|---|---|---|---|---|
| Alwan et al. (2003) | Simulated by people. | N/A | Threshold | N/A |
| Alwan et al. (2006) | Simulated by people and dummies. | Piezoelectric | Threshold | Send messages to a pager |
| Litvak et al. (2008) | Simulated by people and dummies. | Microphone Accelerometer | Gaussian model Sequential forward floating selection (SFFS) | N/A |
| Davis et al. (2011) | Simulated by people. | N/A | Threshold | N/A |
| Yazar et al. (2014) | N/A | Pyroelectric infeared (PIR) Vibration sensor | Support vector machine (SVM) | N/A |
| Shao et al. (2020) | Simulated by 3d-printed skeleton | Accelerometer on smartphone | K-nearest-neighbor (KNN) | N/A |
| Liu et al. (2019) | Simulated by people and dummies. | Seismic | A multi-features semi-supervised support vector machines (MFSS - SVM) | N/A |
| Clemente et al. (2020) | N/A | Seismic | One-class SVM | N/A |
| Mukherjee and Zhang (2020) | N/A | Motion sensor Heat sensor Vibration sensor | Threshold | N/A |

## 2.2 Human Activity

To create training data for anomaly detection in the home, it is important to cover all of the typical activities that people are expected to engage in in the home.

Schrader et al. (2020) say there is no common definition or description of human activities because human activity is highly diverse. Nonetheless, the most fundamental activity in home is clearly walking, since a resident needs to move several inside the home to perform any other activities (Oukrich, 2019). There also are other general activities that every person does. I summarized activity catagories proposed in the literature on human activity in home in Table 2.2. Each paper in the table includes experiments on different activities of interest, and there are some common activities across most of the studies such as sitting, walking,

standing and lying.

**Table 2.2**
Summary of literature review on human activity.

| Authors | Objective of study | Related Sensors | Identified Activities |
|---|---|---|---|
| Roggen et al. (2010) | Collect complex activity datasets in home | Microphone<br>Accelerometers<br>Gyroscope<br>Magnetometer<br>Inertial sensor | Sitting<br>Walking<br>Standing<br>Lying |
| (Chen & Xue, 2015) | Classify human activity by single accelerometer | Accelerometer | Walking<br>Standing<br>Lying<br>Running<br>Rope jump<br>Vacuum cleaning<br>Downstairs<br>Upstairs |
| (Reiss & Stricker, 2012) | Published a new public dataset for physical activity | Gyroscope<br>Magnetometer | Sitting<br>Step walking<br>Walking quickly<br>Falling<br>Jumping<br>Running<br>Downstairs<br>Upstairs |
| Ugolotti et al. (2011) | Detect and classify human activities | Camera<br>Accelerometer | Sitting<br>Walking<br>Standing<br>Lying<br>Get up<br>Fall<br>Rise |
| (Abbate et al., 2012) | Detect fall events | Accelerometer on smartphone | Sitting<br>Walking<br>Lying<br>Running<br>Jumping<br>Hitting the sensor |

## 2.3 Floor Vibrations

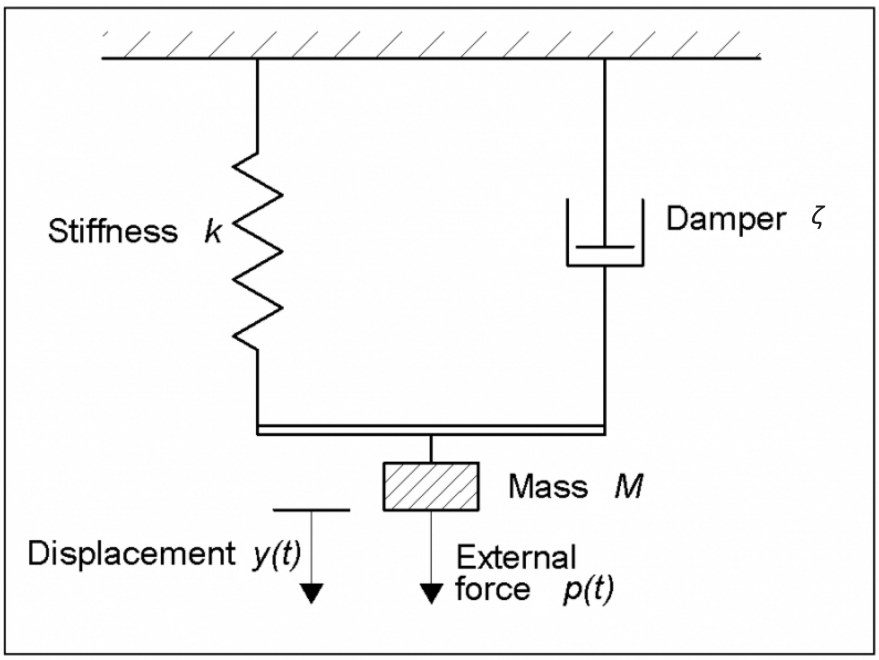
Movement in a building by residents during their normal activities causes floor vibration. This vibration is normally vertical (SteelConstruction, 2016). Floor vibrations are generated by dynamic loads caused directly by people (e.g. walking, dancing, jumping) or machinery,

or they may be generated indirectly by the external environment (e.g. traffic). Theoretically, vibrations are cyclic motions with two significant attributes, frequency and amplitude. In practice, floor vibrations are quite complex dynamic systems with unlimited vibrational modes. Ljunggren (2006) summarises the parameter that influence the dynamic system of a floor:

- Stiffness ($k$): Stiffness controls the springiness of the floor. Higher stiffness can decreases the vibrational amplitude occurring due to a force.
- Damping ($\zeta$): This factor depends on the material making up the surface. It is extremely difficult to obtain an exact damping value.
- Mass ($M$): Higher mass surfaces have reduced vibrational amplitudes. Lower mass is desirable if we want to observe vibrations. However, when the mass is too little, the resulting strong vibrations may disturb residents.
- Fundamental frequency: Floor vibrations are assumed to be occur mainly at a natural frequency, which depends on the stiffness and the mass. Higher frequencies are usually less annoying to residents than lower frequencies.

The complexities of the dynamic system can be modeled as a series of simple mass and spring models with a single degree of freedom (Gavin, 2015). The characteristics of a vibration model are illustrated in Figure 2.2.

**Figure 2.2**
Single degree of freedom system mass-spring model for floor vibration.
Reprinted from SteelConstruction (2016).




## 2.4 Time Series

A time series is a sequence of measurements of a particular random variable at specific sequence of discrete points in time. Generally, the data should be sampled at a constant interval expresssed in as seconds, minutes, hours, days, months, and/or years. In time series analysis, we would generally like to predict a traget variable at particular time lags given a window of previous measurements. This is unusual in that in ordinary supervised classification or regression, the target at time $t$ is not used as a feature at a later time, but in time series analysis, this is often the case.

There are many diverse techniques for analyzing sequential data. The simplest techniques are a special case of regression analysis in which we want to capture four different elements as following (Dash, 2020):

- Seasonal variations: Repeating shape or appearance occuring during a specific period such as daily, weekly, monthly, or seasonally.
- Trend: Possible trends are upwards, downwards, or constantard can be linear or nonlinear.
- Cyclical variations: Movement that follows a specific cyclic period such as business cycles. Cyclical variations are similar to seasonal variations but have different

underlying cases specific to the particular problem.

- Random variations: The variation remaining after the first three types of predictable variation are accounted for.

### *2.4.1 Autoregressive (AR)*

Autoregressive models, the simplest time series models, which are used for predict or forecasting proposes, operate under the assumption that each new value depends on some or all of the the past values. The generative model for a linear of autoregressive process is shown below:

$$Y_t = \varphi_1 Y_{t-1} + \cdots + \varphi_p Y_{t-p} + \varepsilon_t$$

where $\varepsilon \sim N(0, \sigma^2)$. p is the order of the model, which we write as AR(p). For example, AR(1) means the observation at time $t$ depends only on the observation at time $t-1$ plus noise, whereare. AR(2) means $y_t$ depends on the previous two values as well as a noise sample.

### *2.4.2 Time series classification*

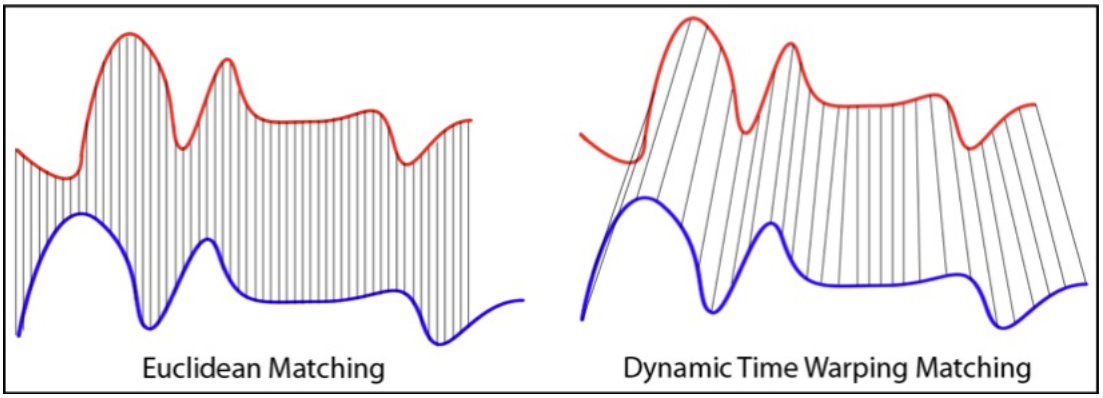
Over the last two decades, one of the most challenging problems in data mining is a classification of time series (Ismail Fawaz, Forestier, Weber, Idoumghar, & Muller, 2019). Several methods have emerged for time series classification. The naive algorithm is Euclidean maching, which is not normally effective without some modification. On the other hand, dynamic time warping (DTW) is an outstanding baseline, and the current state of the art would in the most cases be represented by deep learning classifiers. Dynamic time warping is based on an alignment cost computed between two data sequences that can be stretched or shrunk to accommodate variations along the time axis (Müller, 2007; Toyoda & Sakurai, 2012). Consider two sequences, $X = (x_1, x_2, ..., x_n)$ of length $n$ and $Y = (y_1, y_2, ..., y_m)$ of length $m$. The DTW distance $D(X,Y)$ is defined as:

$$D(X,Y) = D(m,n)$$

$$D(i,j) = (x_i - y_j)^2 + \min \begin{cases} \text{D(i-1, j)} \\ \text{D(i-1, j-1)} \\ \text{D(i, j-1)} \end{cases}$$

where $D(0,0) = 0$, $D(i,0) = D(j,0) = \infty$, $i = (1,2,...,n)$ and $j = (1,2,...,m)$.

**Figure 2.3**
Euclidean maching versus DTW matching.
Reprinted from Dynamic time warping (Wikipedia, 2021).



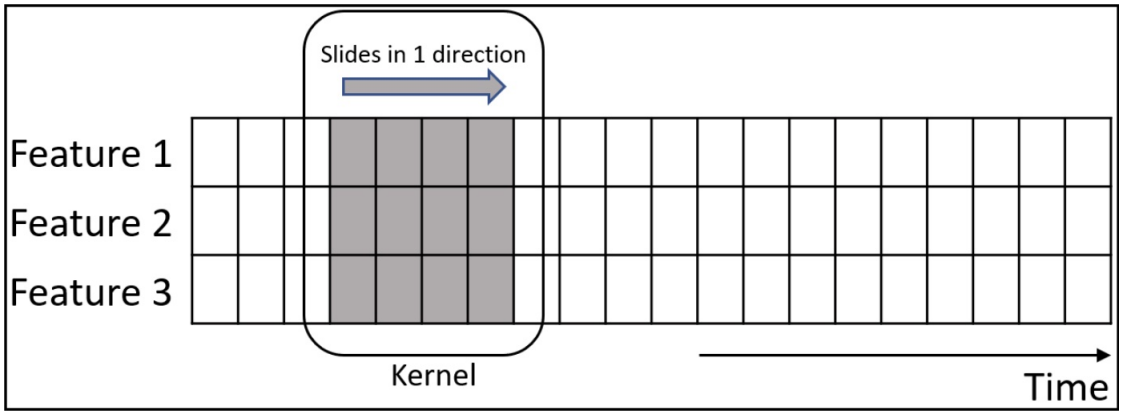

DTW can be used not only for pattern matching or classification, but also for anomaly detection. If the distance between a new signal and each signal in a gallery of historical signals is higher than a set threshold, we can conclude the new signal is an anomaly. The main weaknesses of dynamic time warping is its long processing time. Some more effective learning-based approaches are explained in the following sections.

### *2.4.3 Convolutional Neural Network (CNN)*

A Convolutional neural network is a deep learning model whose input can be an image, video, spatial data, or any multidimensional tensor with locality. One-dimensional CNNs can be used on general data types including text tokens and other types of time series data. CNNs capture spatial and temporal dependencies in a dataset through convolutional filters. A convolution kernel is local linear filter that is slid over the imput tensor along one or more dimensions to obtain a feature map as shown in Figure 2.4. The general method for temporal CNN layer with a nonlinear activation function is

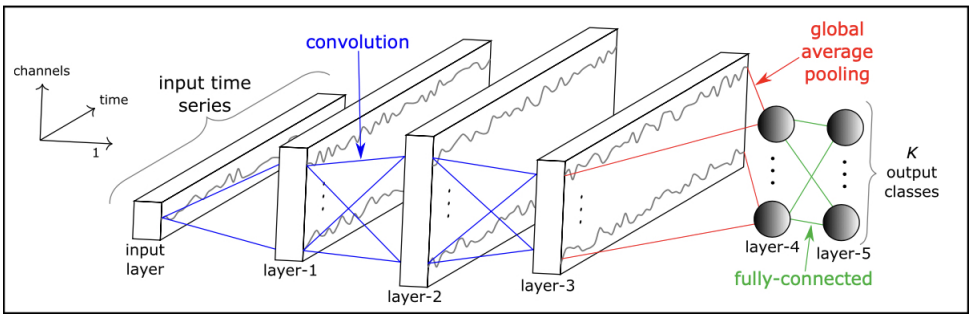
$$C_t = f(W \cdot X_{t-l/2 \to t+l/2} + b) | \forall t \in [1, T],$$

**Figure 2.4**
Convolving on univariate input time series.
Reprinted from Ismail Fawaz et al. (2019).




where $C_t$ is the result of the convolution operation at time $t$ on time series $X$ of length $T$ with a filter $W$ of length $l$, a bias parameter $b$, and a final non-linear function $f$. It can be noticed that the same filter values $W$ and bias $b$ are used at every timestep, a very significant and useful property called weight sharing. When a series of convolutions are completed, the resulting feature maps would typically be fed though fully-connected layer as in the simple neural network architecture shown in Figure 2.5.

**Figure 2.5**
Typical temporal convolutional neural network architecture.
Reprinted from Ismail Fawaz et al. (2019).

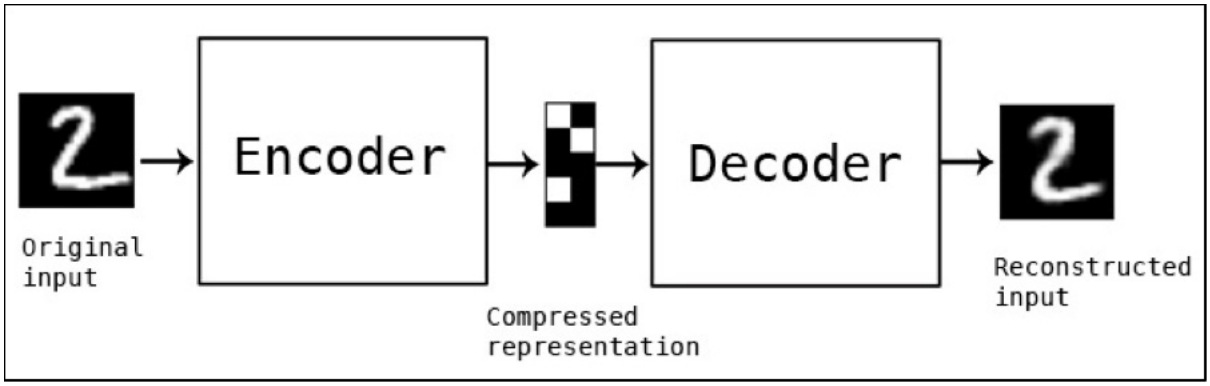



## 2.5 Autoencoders

An autoencoder is a neural network able to compress data similar to what it was trained on. Autoencoders do not require labeled data for training since they utilize unsupervised learning. We just feed the raw input into the model. Figure 2.6 illustrates the intuition of how an autoencoder works.

Besides compression, an autoencoder can be used for denoising by training the autoencoder to reproduce an original noiseless input given a noisy input. This allows the autoencoder to be flexible in the presence of white noise capturing only useful patterns in the data (Vincent, Larochelle, Lajoie, Bengio, & Manzagol, 2010).

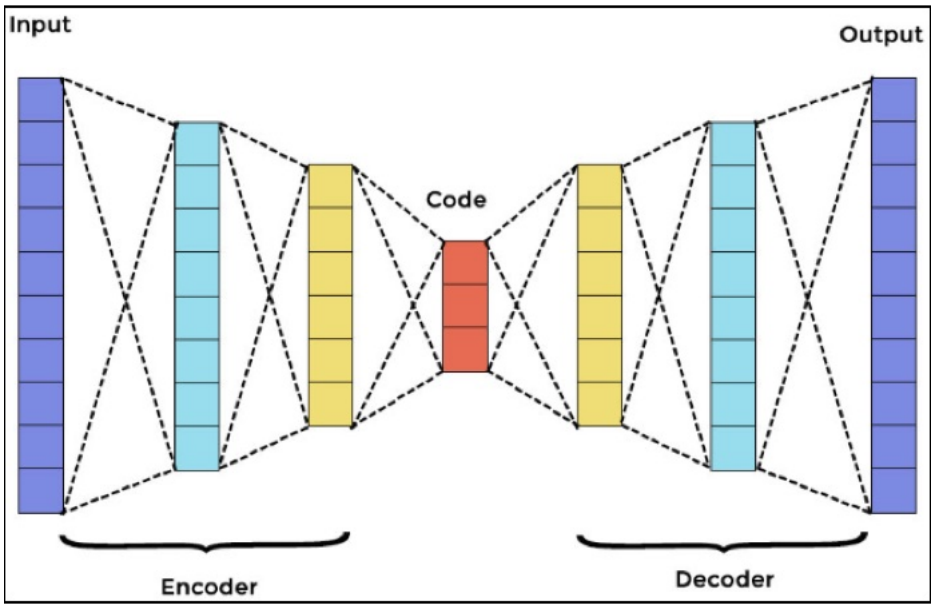
**Figure 2.6**
An autocoder workflow.
Reprinted from Chollet (2016).




An autoencoder has three main components (Badr, 2019): the encoder, the code or bottleneck, and the decoder, as shown in Figure 2.7.

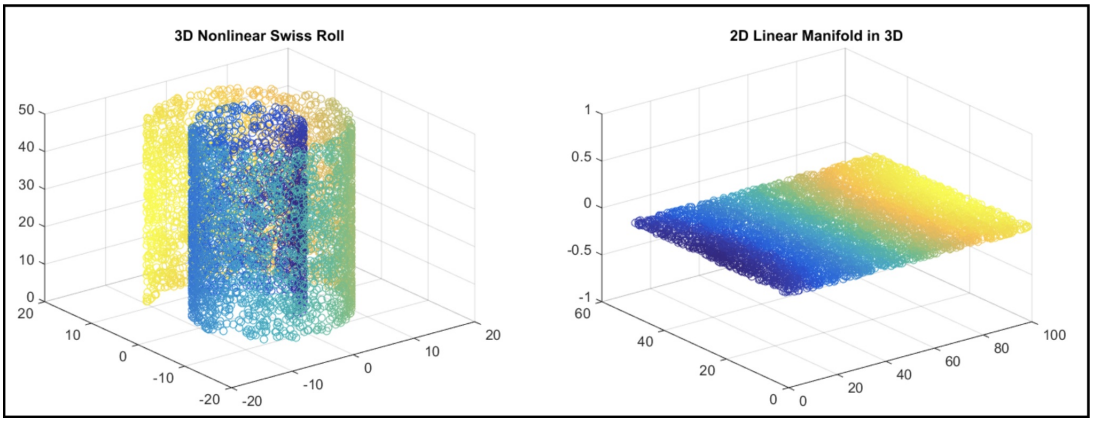
- Encoder: Learns how to reduce input dimensionality, compressing the input data into an encoded representation.
- Bottleneck: The layer that contains the compressed representation of the input data. This code space is also called the latent space.
- Decoder: Learns how to reconstruct as closely as possible the input pattern from the encoded representation.

**Figure 2.7**
The autoencoder architecture.
Reprinted from Pedamkar (2019).




The indirect benefits of this model is that it can be used for dimensionality reduction (Rajan, 2021). The bottleneck has the fewest units of any layer. An example of the kinds of compression an autoencoder can achieve is shown in Figure 2.8. This model reduces the three dimensional input to two dimensions.

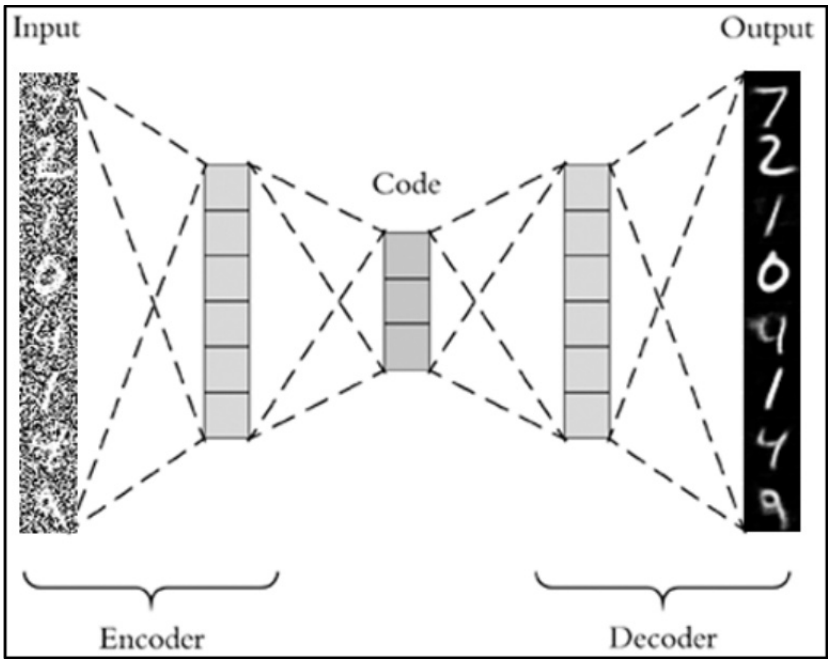
**Figure 2.8**
Visualization of dimensionality reduction using autoencoders.
Reprinted from Johns Hopkins University (2015).




Besides compression, an autoencoder can be used for denoising by training the autoencoder to reproduce an original noiseless input given a noisy input, as shown in Figure 2.9. This is because the optimizer encodes the inputs it was trained on as much as possible.

Vincent, Larochelle, Bengio, and Manzagol (2008) found that the robustness of the code at the bottleneck was improved by adding noise to the original input. This allows the autoencoder to be flexible in the presence of white noise capturing only useful patterns in the data (Vincent et al., 2010).

**Figure 2.9**
An autoencoder trained on "clean" images can correct noisy input.
Reprinted from Rosebrock (2020).

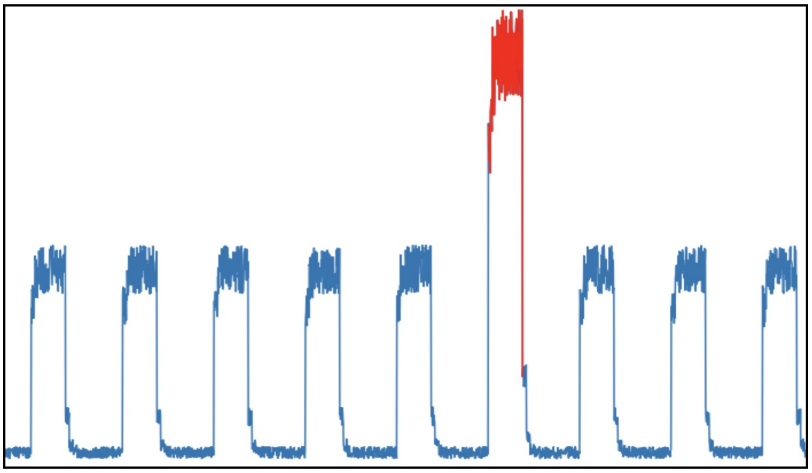



Beside the simple feedforward layers described previously, autoencoder can be combined with long short term memory networks (LSTMs) or convolutional neural networks (CNNs) depending on the type of input. In my thesis, the input, vibration from human activities, is a sequential time series. Therefore, an combination of autoencoder with LSTM networks may be suitable for my purpose.

### *2.5.1 Autoencoders for Anomaly Detection*

Autoencoders are extremely useful as methods of typicality. Consider a person who does the same things every day. Suppose that one day, an unusual event occurs. An autoencoder trained on the usual daily activities will map the new situation to something similar in the training, as shown in Figure 2.10. The reconstruction error in abnormal causes should be high. A model trained on one type of data (the normal activities) will fail when facing abnormal data it has never seen before. The simple autoencoder-based anomaly detection algorithm is shown in Algorithm 1.

**Figure 2.10**
An autoencoder capable of detecting anomalous events in time series.
Reprinted from pavithrasv (2020).



**Algorithm 1** Autoencoder-based anomaly detection

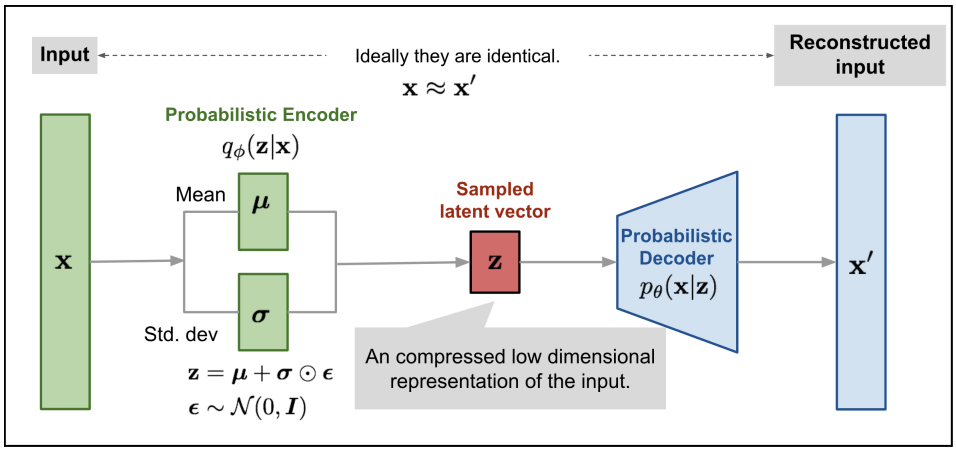
**Input:** Normal dataset: $X^{(i)}(i = 1,...,m)$, abnormal dataset: $x^{(j)}$ $(j = 1,...,n)$, threshold: $\alpha$
**Output:** Reconstruct data: $\hat{X}$
Reconstruction error: $\|X - \hat{X}\|$
Train an autoencoder using the normal dataset X $\rightarrow L^* = \underset{L}{\operatorname{argmin}} \sum_{i=1}^{m} \|X^{(i)} - \hat{X}^{(i)}\|^2$
Testing an autoencoder:
**for** j = 1 to $n$ **do**:
    **if** reconstruction error $< \alpha$:
        $x^{(j)}$ is a nomal.
    **else**:
        $x^{(j)}$ is an anomaly.

### *2.5.2 Variational Autoencoder*

The variational autoencoder (VAE) is a generative model like the ordinary autoencoder, it encodes and decodes data in the training set, but it also attempts to model the probability density over the input space of the examples emitted by the data source, by transforming, e.g., Gaussian distributied latent vectors to elements of the input space. For example, if model is trained with traffic images, the decoder, when passed a sample from the code space, would have a high probability of emitting vehicle images object related to traffic. Other data would have a low probability of being emitted. By sampling in the latent space reconstructing, the variational autoencoder can also generate new examples that look similar to those from the original dataset (Roger, 2021). In the other words, a variational autoencoder is an encoder which is trained to be regularized at the bottomneck in order to guarantee that latent space is a good source for the generative process (Rocca, 2020). The architecture of a variational

autoencoder is illustrated in Figure 2.11. The latent space of a variational autoencoder is easy to sample from.

**Figure 2.11**
Architecture of a variational autoencoder.
Reprinted from Weng (2018).




The principles mentioned above do not mean that a variational autoencoder always has better performance than general autoencoders in anomaly detection tasks (Agmon, 2021), since the objective of the variational autoencoder is as a generative model for new data.

## 2.6 Generative Adversarial Networks (GANs)

The generative adversarial networks was first introduced in the paper "Generative Adversarial Nets" (Goodfellow et al., 2014). GANs consist of two networks, the generator and discriminator. The generators try to fool the discrimitor by creating virtual image. The discriminator try to distinguish different between real images and virtual images that were generated by the generator. Figure 2.12 illustrates the intuition of GAN. The both networks always fight together what is called adversarial training. The important idea is that we want the two networks learn from each other. During training, the generator can improve its performance to create images that look similar to the real images as well as possible. The discriminator also impove to distinguish. When training process reaches equilibrium, the discriminator is no longer able to distinguish the real from the fake, as shown in Figure 2.13. Thus, if the discriminator is well trained and the generator can produce realistic-looking images that fool the discriminator, we have a good generator creating an image that looks like a training set.

**Figure 2.12**
Representation of GAN as a generator and a discriminator.
Reprinted from InnovationHub (2020).

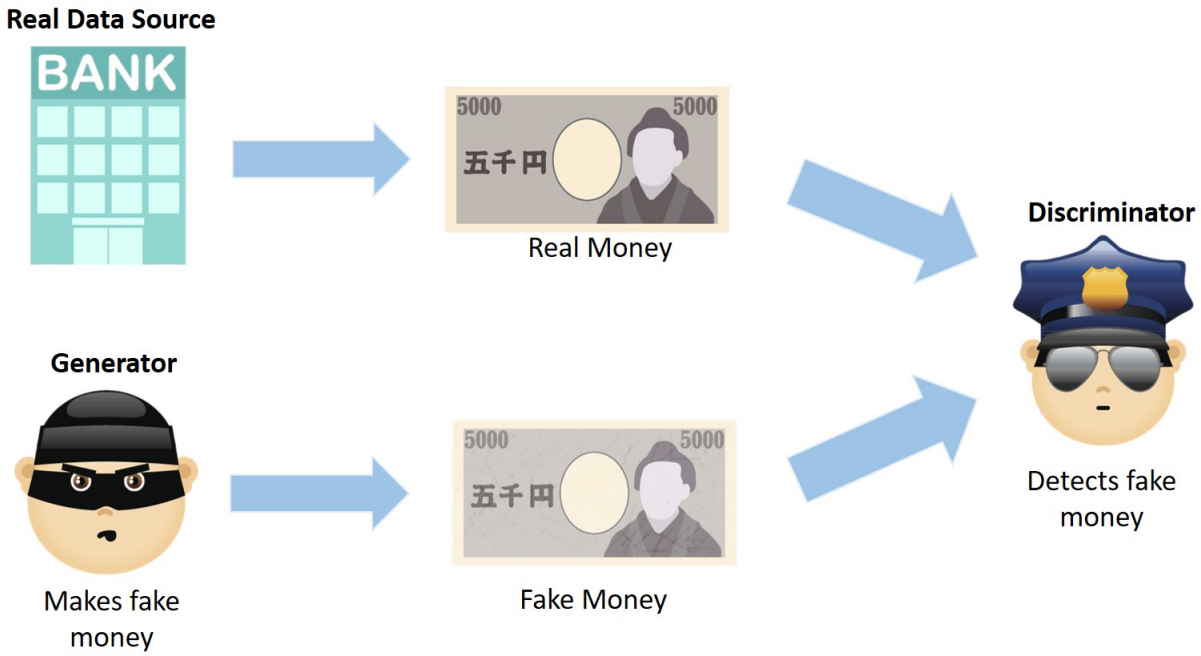
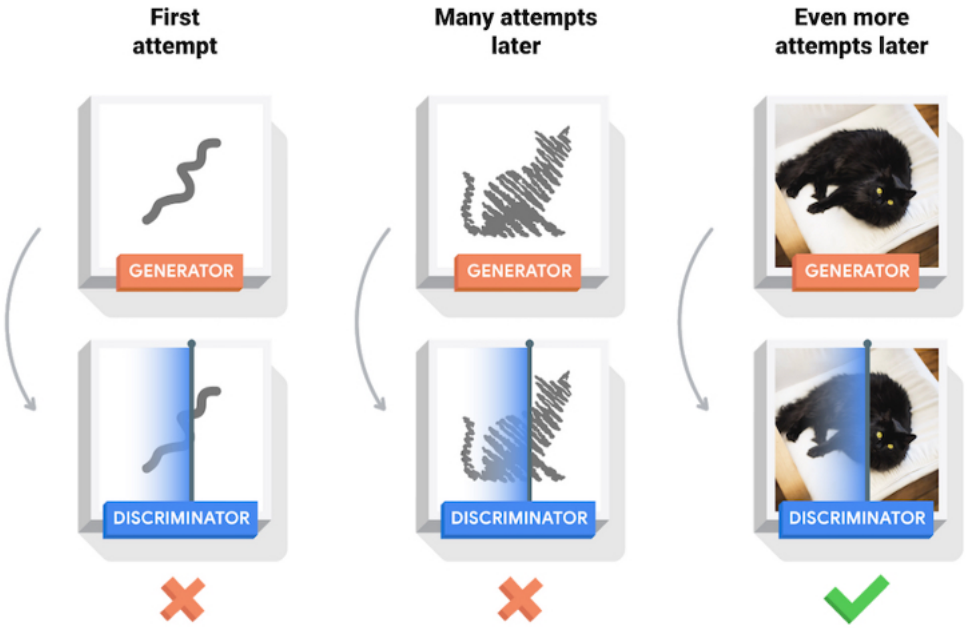


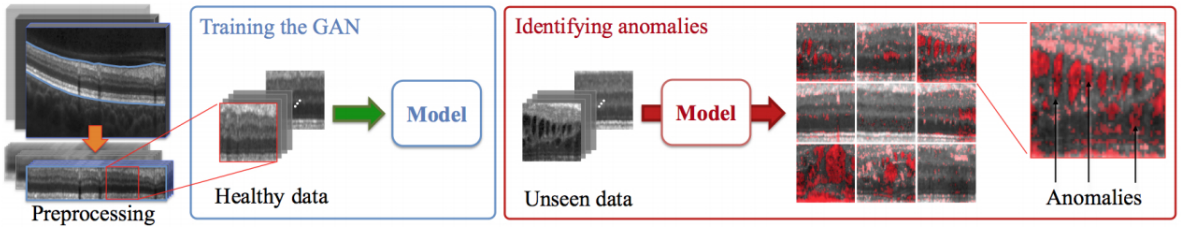

**Figure 2.13**
Training of the generator and discriminator.
Reprinted from Tensorflow (2021).



### *2.6.1 Generative Adversarial Networks for Anomaly Detection*

Anomaly detection by using GANs is possible. Schlegl et al. (2017) (2017) firstly proposed AnoGAN, a deep convolutional generative adversarial network, to detect abnormal image. AnoGAN's architecture is based on standard GAN. AnoGAN is trained only on normal samples, which can lead the generator learning to generate normal samples. When an anomalous or unseen image is fed though the model, the difference between the input and the reconstructed images will be highlight the anomaly area, as shown in Figure 2.14.

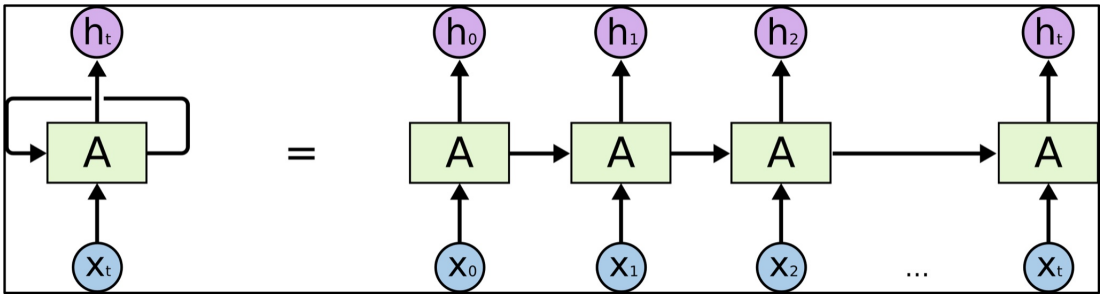
**Figure 2.14**
Anomaly detection using AnoGAN.
Reprinted from Schlegl et al. (2017).




## 2.7 Recurrent Neural Networks (RNNs)

The main idea of recurrent neural networks (RNNs) is to process sequential data such as videos (sequences of images) or text (sequences of words). For example, when we read a book, we observe a sequence of tokens (words) as we read from left to right. To understand what the sentence is about, we observe each token in the sequence, and mix it with the meaning of the words we previously read. A RNN uses the same principle, by modifying simple feedforward neural networks so that the previous state can be combined with new inputs in a sequential time series (Donges, 2019). A key attribute of RNNs is their ability to retain the results of processing at one time, or cell state, for use later in the process. The important elements of a RNN are its input, its hidden state, its output, as well as how all these are connected to each other.

**Figure 2.15**
Recurrent neural network architecture.
Reprinted from Olah (2015).




where $X_t$ is input data at time $t$, $A$ is Hidden layer, and $h_t$ is an output from RNN at time $t$ shown in Figure 2.15. The main benefit of this loop is to bring back the previous hidden state, or simply say that RNN is a Neural Network with more memory to store the previously calculated hidden state.

The main problem in RNNs is called the "vanishing gradient" problem. To update the weights in a neural networks, we use backpropagation of loss using chain rules for the gradient (Arnx, 2019). We calculate the gradient of the loss function ($E$) with respect to the parameters, as shown in Figure 2.16. RNNs are more complicated, because the output $h_t$ depends not only on timestep $t$, but also on timesteps $t-1, t-2, ..., 1$. Therefore, the backpropagation has to include all calculations from the first timestep to the last. When the gradient magnitude is less than one, a series continuous multiplications will cause the gradient magnitude to decrease as the sequence gets longer. Specifically, the simple RNNs still have difficulties with data with sequential dependencies over a long period of time.

**Figure 2.16**
The concept of optimization in a feed-forward neural network.
Reprinted from Donges. (2019)

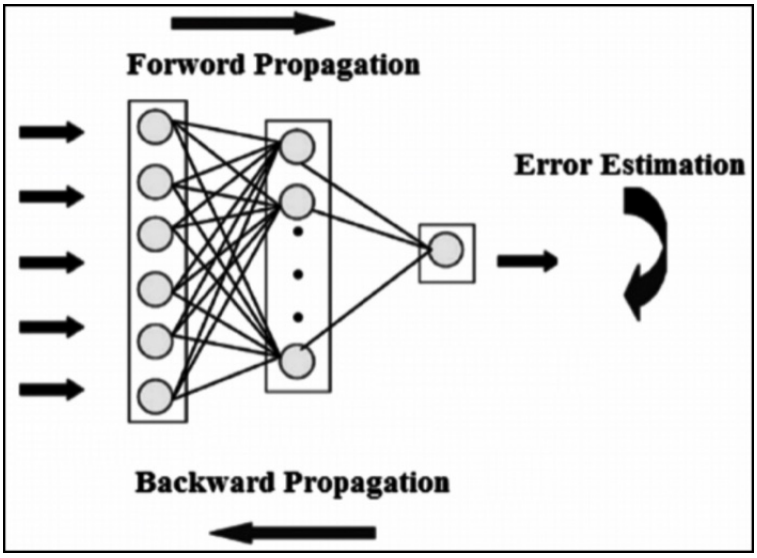
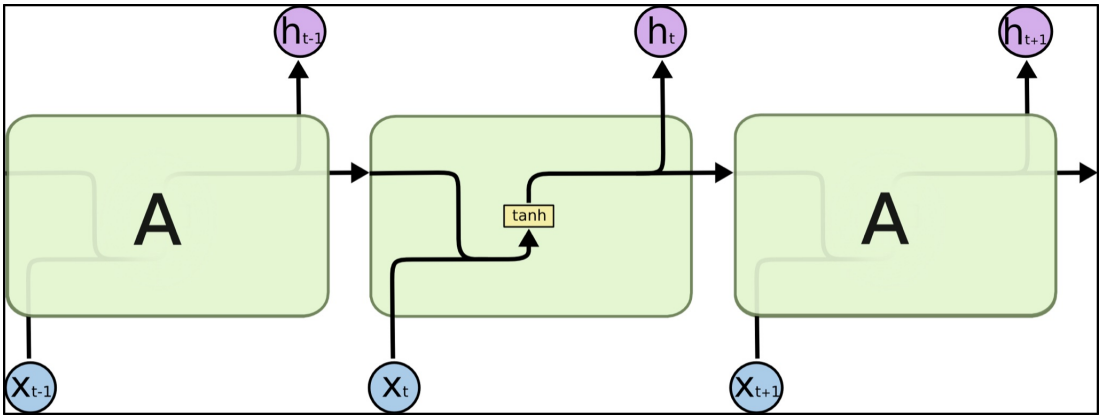


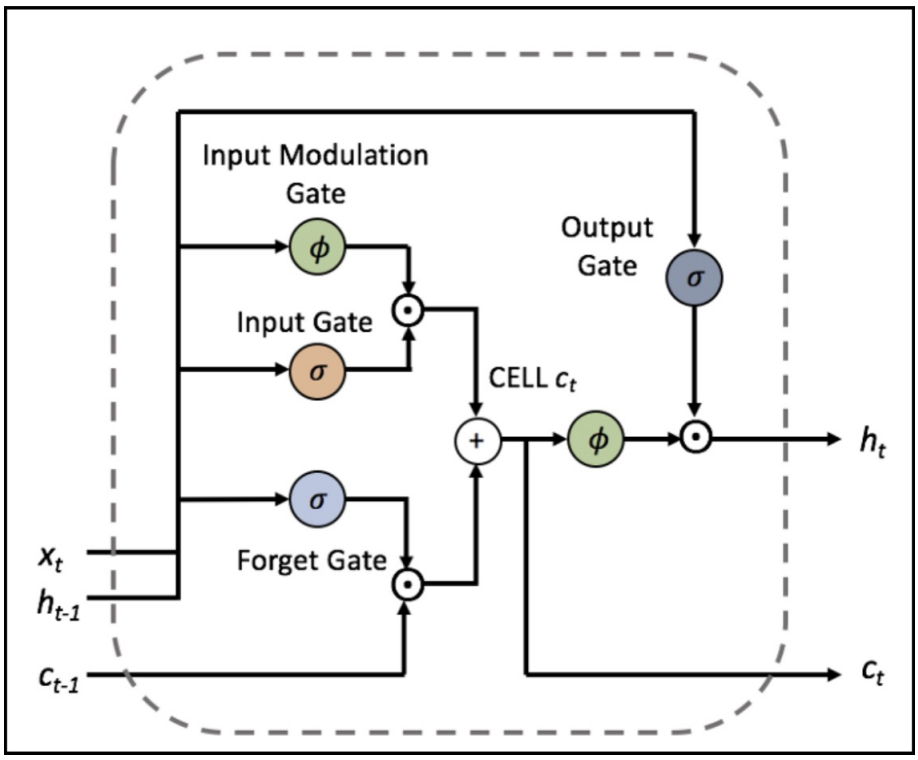

**Figure 2.17**
The repeating module in a standard RNN contains a single layer.
Reprinted from Olah (2015).




## 2.8 Long Short-Term Memory (LSTM)

Long short-term memory networks are an extension of simple RNNs that attempt to mitigate the problem of vanishing gradient under long term dependencies. LSTMs are well suited to learn from important events with long time lags in between (Donges, 2019; Olah, 2015). This is accomplished by giving the memory (hidden state) gates enabling forget (delete), write, and read, as shown in Figure 2.18.

**Figure 2.18**
LSTM structure.
Reprinted from Tangruamsub (2017).




Before specifying the LSTM's details, some variables are needed (Tangruamsub, 2017):

- Cell state: The LSTM's memory.
- Gate: A module controling the flow of data, e.g., allowing a write, a read, or a forget operation.

The three operations in a LSTM are as follows.

**Forget**: Forgeting involves (partly or wholly) clearing an old cell state in preparation for new input. The forget gate has the responsibility of deciding whether to clear cell state. If the forget gate outputs zero, the previous cell state is cleared completely. If the forget gate outputs a one, the model retains the cell state completely. In between 0 and 1, the state is attenuated proportionally. The forget gate integrates the incoming input data and the previous hidden state (according to the RNN formula) for making decisions. A sigmoid activation is to limit the gate range to 0 - 1. The gate output $f_t$ is defined as

$$f_t = \sigma(\mathbf{W_{x^t}} x_t + \mathbf{W_{h^t}} h_{t-1} + b_f),$$

where $\mathbf{W_{xf}}$ is the linear weight vector applied to the model's input at time $t$, $\mathbf{W_{hf}}$ is the linear weight vector for the hidden state propagated from time $t-1$, $x_t$ is the incomming input, $h_{t-1}$ is the previous hidden state, $b_f$ is the forget gate bias, and $\sigma$ is the sigmoid function.

**Write**: When new input is fed to the model, it first must decide whether to update its cell state. This action is controlled by an input gate also using a sigmoid activation. This computation is

$$i_t = \sigma(\mathbf{W_{xi}} x_t + \mathbf{W_{hi}} h_{t-1} + b_i),$$

where $i_t$ is output of the input gate, $\mathbf{W_{xi}}$ is the linear weight vector for the input, $\mathbf{W_{hi}}$ is the weight vector for the hidden state, $x_t$ is the input, $h_{t-1}$ is the previous hidden state, $b_i$ is the input gate bias, and $\sigma$ is the sigmoid function. Secondly, if the model decides to do an update ($i_t$ is large), what value should it update with? This question is answered by the "Input modulation gate". The input modulation gate uses a tanh function instead of a sigmoid. The gate value is

$$g_t = \tanh(\mathbf{W_{xc}} x_t + \mathbf{W_{hc}} h_{t-1} + b_c),$$

where $g_t$ is the input modulation gate's output, $\mathbf{W_{xc}}$ is the linear weight vector for the input, $\mathbf{W_{hc}}$ is the linear weight vector for the hidden state, $x_t$ is the input, $h_{t-1}$ is the previous hidden state, $b_c$ is the input modulation gate bias, and tanh is the hyperbolic tangent function.

**Update cell state**: Once the forget gate, input gate, and input modulation gate values are calculated, the cell state is updated as

$$c_t = f_t \circ c_{t-1} + i_t \circ g_t,$$

where $c_t$ is the current cell state and $c_{t-1}$ is the previous cell state. Clearly, the forget gate optionally deletes the old cell state when $f_t$ is zero. When $f_t$ is a one, the model can propagates $c_{t-1}$. When $i_t$ is a one, the input modulation gate $g_t$ provides the update, based on the current input and previous hidden state. If $i_t$ is close to zero, $g_t$ is overlooked.

**Read**: In sample RNNs, the model produces a hidden state $h_t$ on each time $t$. At time $t+1$, a LSTM similarly takes the previous $h_t$. The term "read" for a LSTM means to allow a downstream elements to read $h_t$ or to block the $h_t$ value from being propagated. The output gate is calculated by

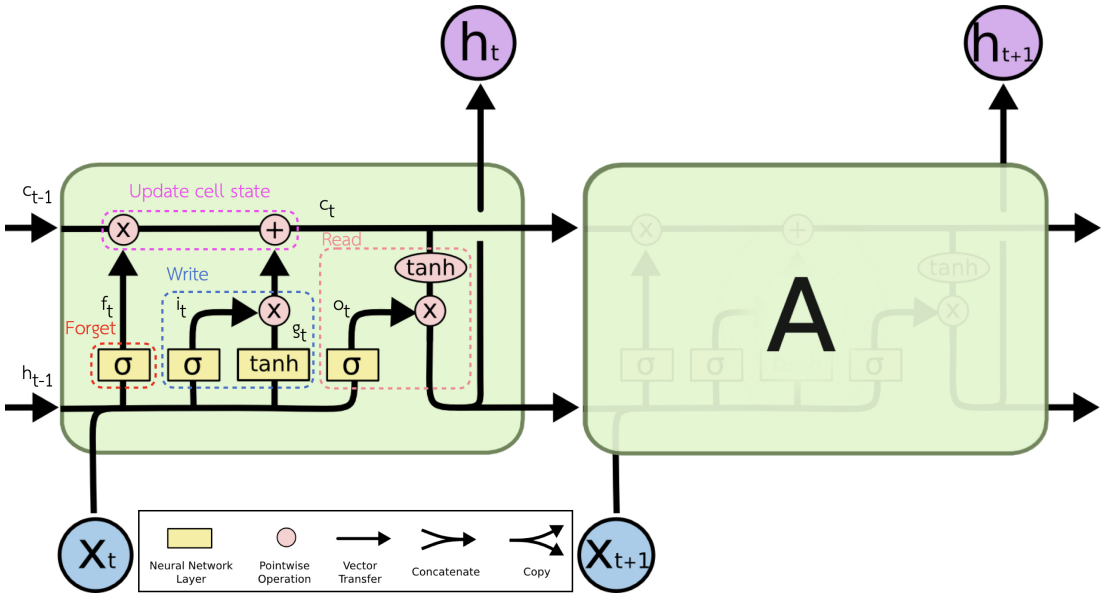
$$o_t = \sigma(\mathbf{W}_{\mathbf{x}^o} x_t + \mathbf{W}_{\mathbf{h}^o} h_{t-1} + b_o),$$

where $\mathbf{W}_{\mathbf{x}^o}$ is the linear weight vector of the input, $\mathbf{W}_{\mathbf{h}^o}$ is the linear weight vector of the hidden state, $x_t$ is the input, $h_{t-1}$ is the previous hidden state, and $b_o$ is the output gate bias. The output $o_t$ is propagated to the next timestep in order to control read operation, i.e.,

$$h_t = o_t \circ \tanh(c_t).$$

If the output gate $o_t$ is zero, $h_t$ is attenuated to zero meaning nothing is sent. On the other hand, if $o_t$ is one, the model propagates $h_t$ as an output and propagates.

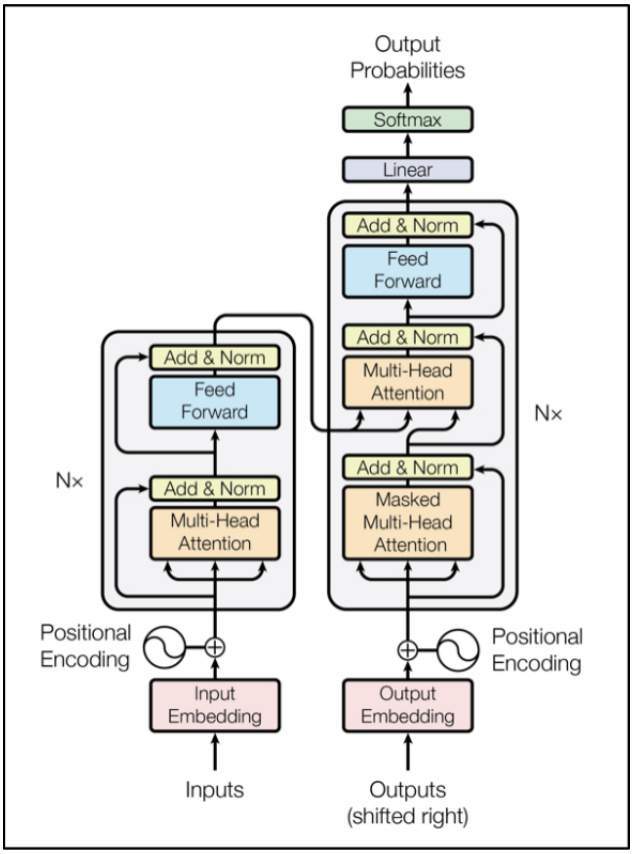
**Figure 2.19**
The repeating module in a LSTM contains four interacting layer.
Reprinted from Olah (2015).




## 2.9 Transformer

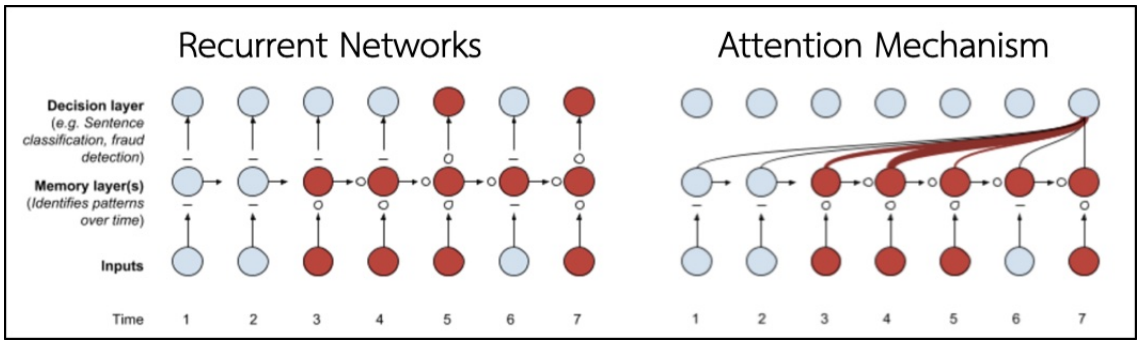
The transformer was proposed in the paper "Attention is All You Need" by Google (Vaswani et al., 2017). This paper proposes a new architecture that replaces RNNs time-locked processing with an attention mechanism called the transformer, as shown in Figure 2.20. The transformer architecture has recently beat benchmarks in many domains. In particular, it has revolutionized the Natural Language Processing (NLP) field, particularly in the machine translation task. This model contains two significant parts, an encoder and a decoder, which work as similar as an autoencoder. In principle, transformers can be used for anomaly detection purposes as well (Mishra, Verk, Fornasier, Piciarelli, & Foresti, 2021).

**Figure 2.20**
The transformer architecture.
Reprinted from Vaswani et al. (2017).




To compare how RNNs and attention deals with the time dimension of the input, RNNs include every information needed about sequential data into the final hidden state of the network. The decision layer can only access the memory accumulated at that timestep. At every timestep, the RNN focuses on a different positions in the input. On the other hand, an attention mechanism can focus on any of the input from several timesteps, and setting weights on each input indicating what should be focused on in order to make a prediction. Figure 2.21 provides the intuition behind both methods.

**Figure 2.21**
Comparison RNNs and Attention.




### *2.9.1 Attention*

According to James (1890), in psychology, attention is a concentration of the mind on a single object or thought, one preferentially selected from among many stimuli using possibly complex process, with a view to limiting or clarifying receptivity by narrowing the range of stimuli. Similarly, attention in neural networks was specifically designed to focus on only the most important subsets of long sequences related to completing a given task (Alammar, 2018, 2019; Klingenbrunn, 2021). The process consists of three main steps, as follows.

1. Create the query (representing the current position vector in the input sequence), key (representing the relative importance of the inputs in the sequence), and value (representing the priority of each position) vectors, they are quite useful for calculating and thinking about attention, for each path and each input token by multiplying by weight matrices as $W^Q$, $W^K$ and $W^V$ as shown in Figure 2.22.
2. For each input token, use the query vector to get a score against all the other key vectors by multiplying the current query vector with all the key vectors as shown in Figure 2.23.
3. Sum up the value vectors after multiplying them by their associated scores. More transparent blocks in Figure 2.24 are those with lower values.

If the model performs the same operation for each input token, the result is a vector representing the context of each token, as shown in Figure 2.25. These vectors are passed to the next sub layer in the transformer block.

**Figure 2.22**
Creating the query, key and value vector in a self-attention module.
Reprinted from Alammar (2018).

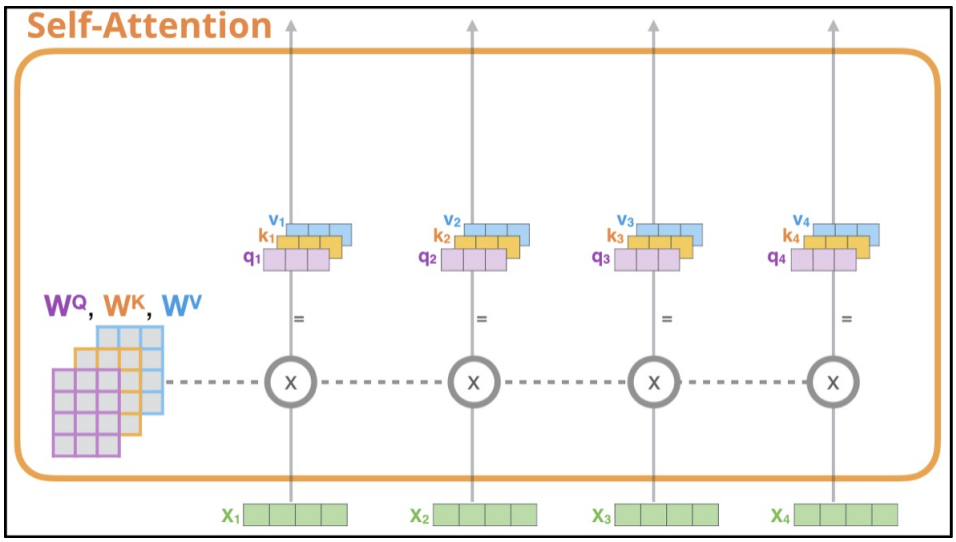



**Figure 2.23**
Getting a score of how each key matches the query in a self-attention module.
Reprinted from Alammar (2018).

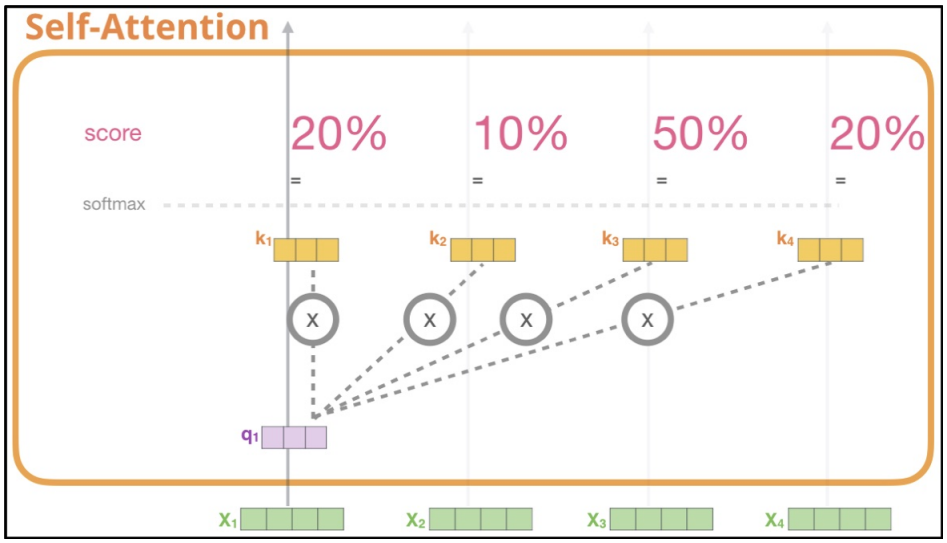


**Figure 2.24**
Summing up the value vectors in a self-attention module.
Reprinted from Alammar (2018).

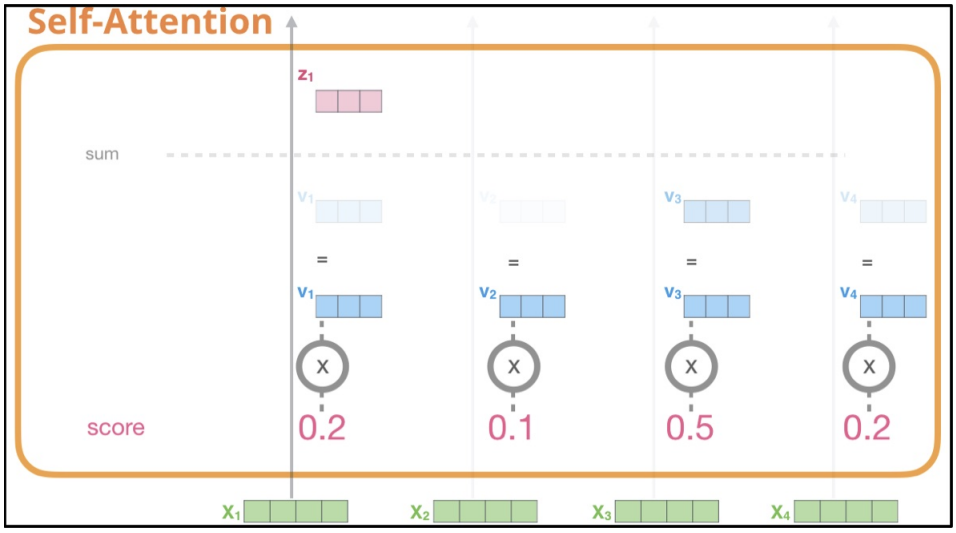
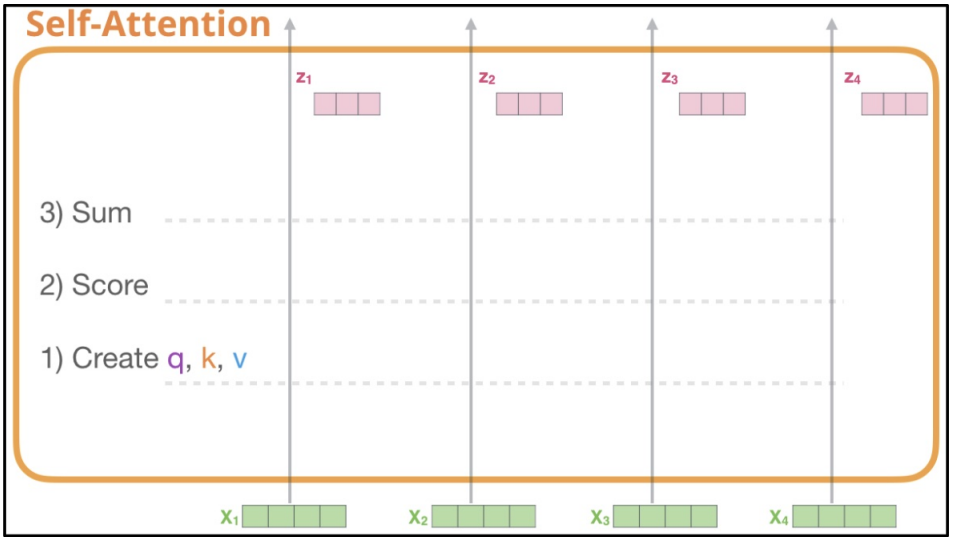
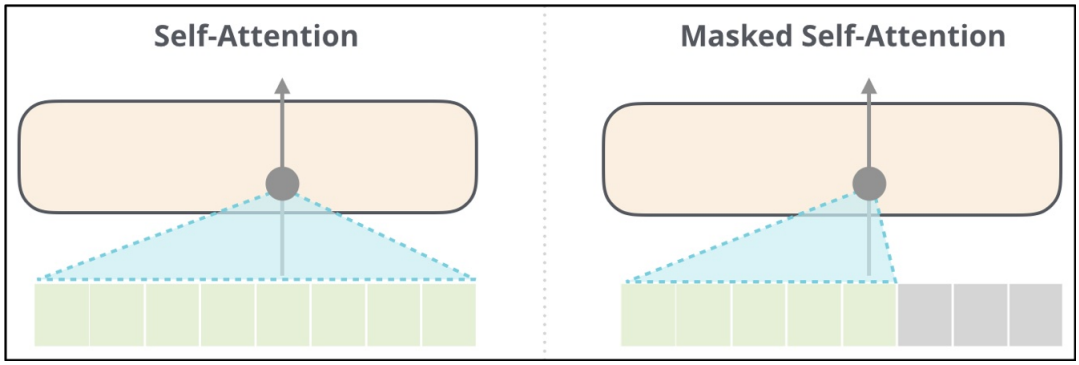



**Figure 2.25**
The outcome of the self-ttention process.
Reprinted from Alammar (2018).




Note that in the decoder, a transformer uses masked self-attention. The difference between self-attention and masked self-attention is shown Figure 2.26. A self-attention allows each position in the output to attend to all positions in the input but Masked Self-Attention only considers previous positions in order to preserve the auto-regressive property.

**Figure 2.26**
Difference between self-attention and masked self-attention.
Reprinted from Alammar (2019).




### *2.9.2 Positional Encoding*

In any sequence to sequence model in which the number of inputs or outputs is variable and order and position are important, if we are not using recurrence or convolution, we need some other principle to make use of the order an input sequence, as shown in Figure 2.27. The formulae for positional encoding are

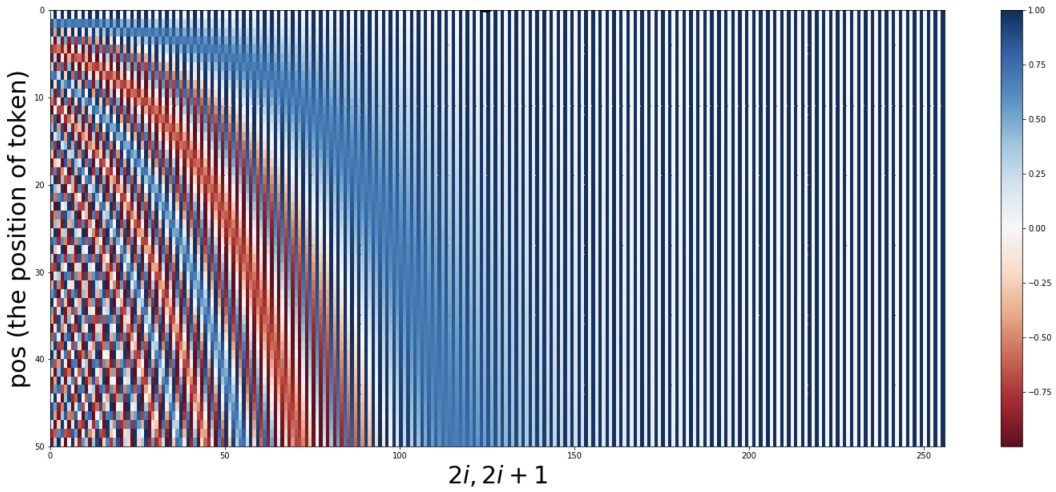
$$PE(pos, 2i) = \sin(\frac{pos}{10000^{2i/d_{model}}}),$$

$$PE(pos, 2i+1) = \cos(\frac{pos}{10000^{2i/d_{model}}}),$$

where $pos$ is the index of a token $\in [0, L-1]$ in the input sequence, $d_{model}$ is the model depth, and $i$ is an along the model depth. The important characteristics of positional encoding are

1. Positional encoding is represent by a matrix with dimension (sequence length $\times$ model depth).
2. Each column of the PE matrix represents the continuous value which varies according to the $pos$ value.
3. Each row of the PE matrix represents the interpolated position of the discrete value.
4. The row vectors are alternating series of sines and cosines, with frequencies that decrease according to a geometric series.

**Figure 2.27**
Positional encoding of a sequence length of length 50 in a model with a model depth of 256.
Reprinted from Tamura (2021).



# CHAPTER 3

# Methodology

The methodology of the proposed study can be separated into four main processes, as shown in Figure 3.1 and as follows.

1. Design and build a filter, amplifier, and embedded system integrating a seismic sensor in order to measure vibrations due to peoples' activities at home.
2. Collect a dataset of normal events such as ordinary human activities in the home.
3. Build anomaly detection models likely to detect fall as anomalies.
4. Deploy the system in a real home environment and evaluate performance.

**Figure 3.1**
Overview of the methodology.

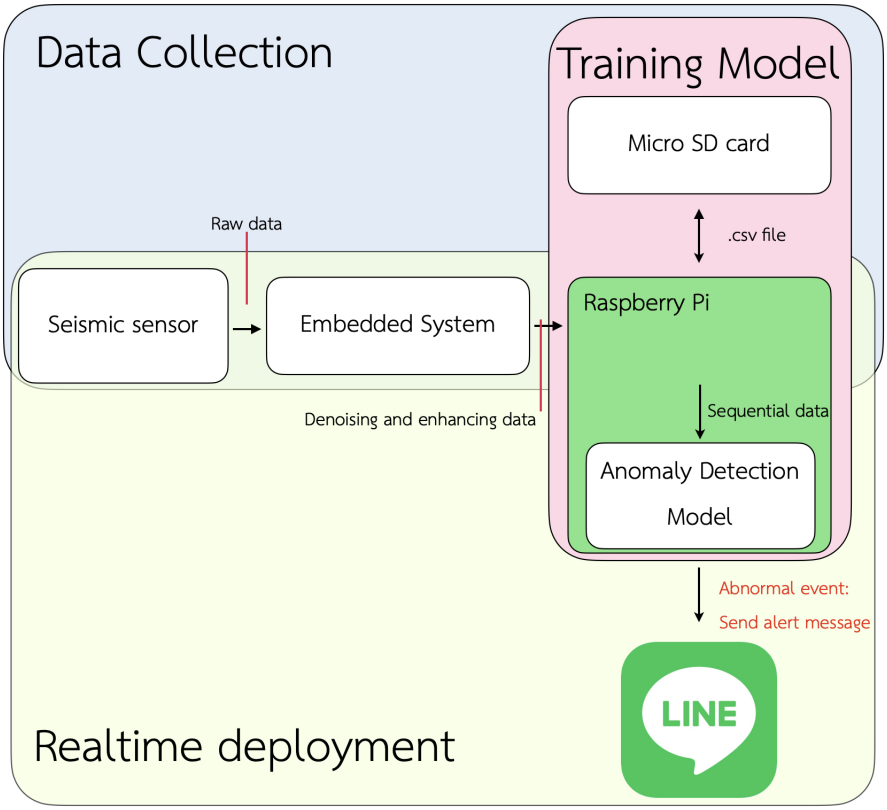


**Figure 3.2**
Data collection - hardware.

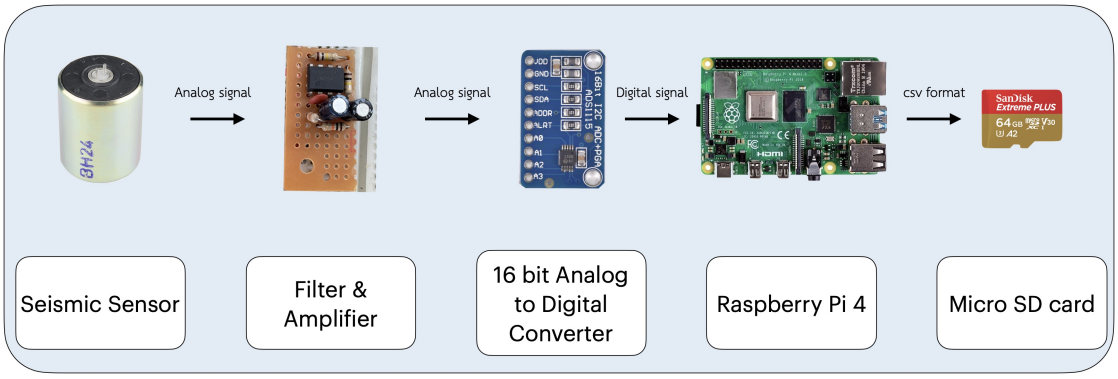



**Figure 3.3**
Training model process.

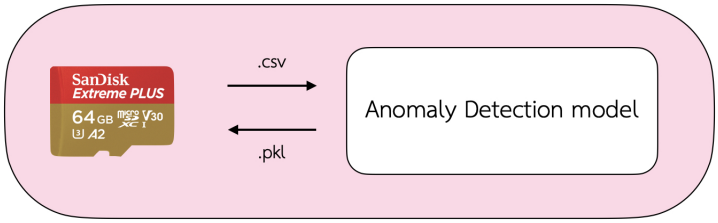
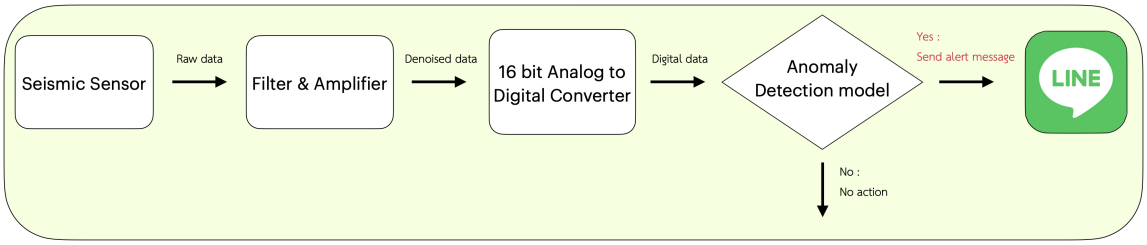



**Figure 3.4**
Realtime deployment system - data flow diagram.



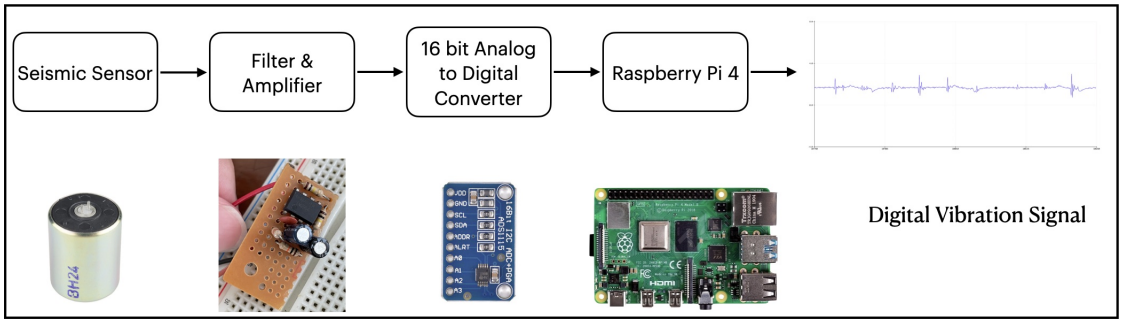

## 3.1 Data Collection

To collect raw data, we need to build a prototype embedded system. To detect human falls, we must simulate falls and other anomalous events along with ordinary activities. The embedded system must also be capable of collecting and recording the vibration signal over a long period of time.

### *3.1.1 Hardware*

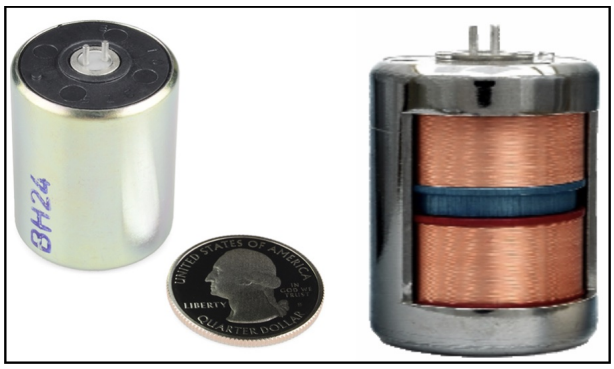
There are four significant hardware components, as shown in Figure 3.2. Each component has a specific propose.

**Figure 3.5**
Hardware required to receive raw vibration signal data.




The first main component is the seismic sensor or geophone shown in Figure 3.6, A geophone is a device that converts ground vibration (velocity) into voltage. Geophones have historically been passive analog devices that typically comprise a spring-mounted wire coil moving within the field of a case-mounted permanent magnet to generate an electrical signal. I decided to use the geophone SM-24 because it has a small size, similar to a coin, and is easy to install by just setting it on the ground. However, the SM-24 cannot connect directly to a microcontroller because it generates voltage up to 28.8 V (corresponding to 1 m/s velocity of the coil). Therefore, we have designed an embedded system to convert the $0-28.8$ V range to the into range $0-5$ V.

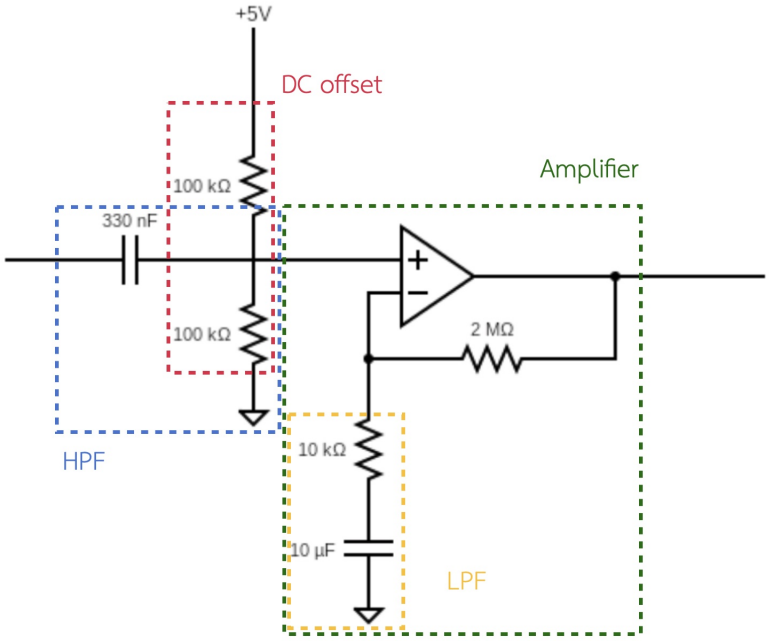
**Figure 3.6**
A geophone SM-24 and its interior elements.
Reprinted from Sparkfun, (2017).



The second main component is the analog circuit of the filter & amplifier, which is shown in Figure 3.7, has four significant logical components, as a DC offset, a high-pass filter (HPF), an amplifier, and a low-pass filter (LPF). The DC offset is used for set the reference signal as 2.5 V. The HPF eliminates any frequencies below the frequency range of interest. It has an ideal cutoff frequency around 100 Hz. The amplifier provides a high voltage gain of around 200 to prepare for sampling at a high resolution at the analog to digital converter

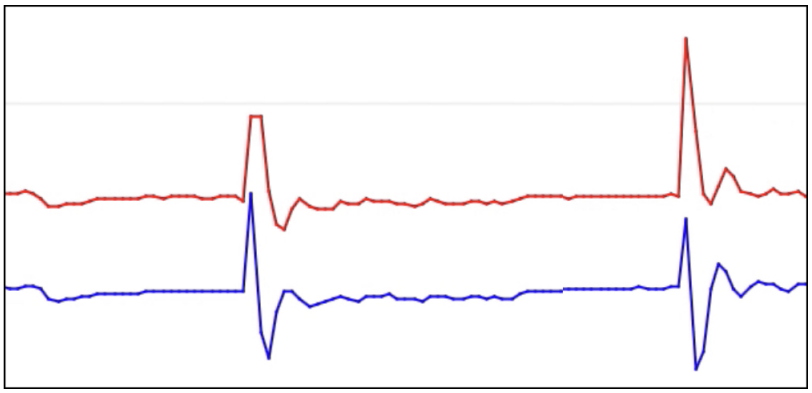
(ADC). Lastly, the low-pass filter has a cut-off frequency at 100 kHz, making the overall frequency range of interest 100 - 100 kHz. However, this is a first protitype, and the goal is 0 - 200 Hz.

**Figure 3.7**
Analog circuit measure vibrations caused by human activity.



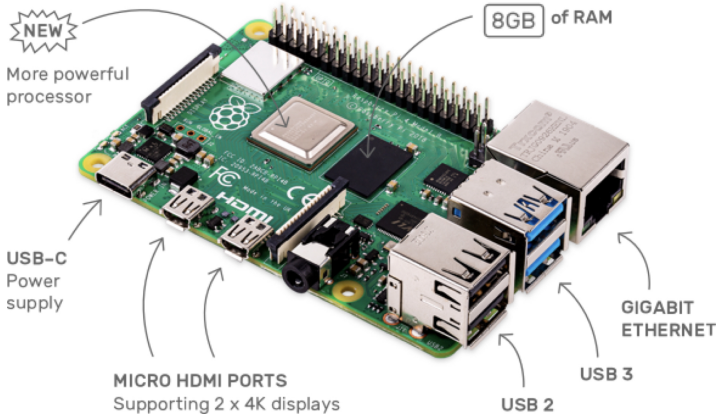

The third main component is analog to digital converter (ADC), Higher bit width means a filter that can represents the data stream more accurately. Figure 3.8 provides the intuition behind the difference between a 10-bit and 16-bit ADC. The red line shows a signal measured at 10 bits in the adc pin of the Arduino Mega, and the blue line shows the same original signal measured with a 16-bit ADC. After the raw analog signal is filtered and amplified, it needs to be sampled at a rate of 500 samples/second and quantized at a resolution of 16 bits/sample in order to prepare it for digital transmission. A 16-bit ADC is capable of distinguishing 65536 ($2^{16}$) different voltage levels within a narrow voltage range from $0-5$ Volts. It means that each level represents approximately 76.3 uV, which is sufficient to capture accurate signals from the seismic sensor.

**Figure 3.8**
Comparison 10-bit (red) and 16-bit (blue) ADC.



The fourth main component, shown in Figure 3.9, is the single-board, a Raspberry Pi 4 model B, which is a small size, low price system with several useful functions, and sufficient power for machine learning inference. The digital signal is fed from ADC to Raspbery Pi via I2C which is a synchronous serial communication interface specification used for short-distance communication. And then these raw signals are saved to the file system os a mircro SD card.

**Figure 3.9**
Raspberry Pi 4 model B.
Reprinted from Raspbery Pi officially




## 3.2 Anomaly Detection Models

I choose two candidate models as autoencoder with LSTM architecture as shown in Figure 3.10 and the Transformer shown in Figure 3.11 , which are both state of the art models for distinguishing abnormal event in sequencial data.

**Figure 3.10**
The autoencoder with LSTM architecture (SEQ2SEQ structure).

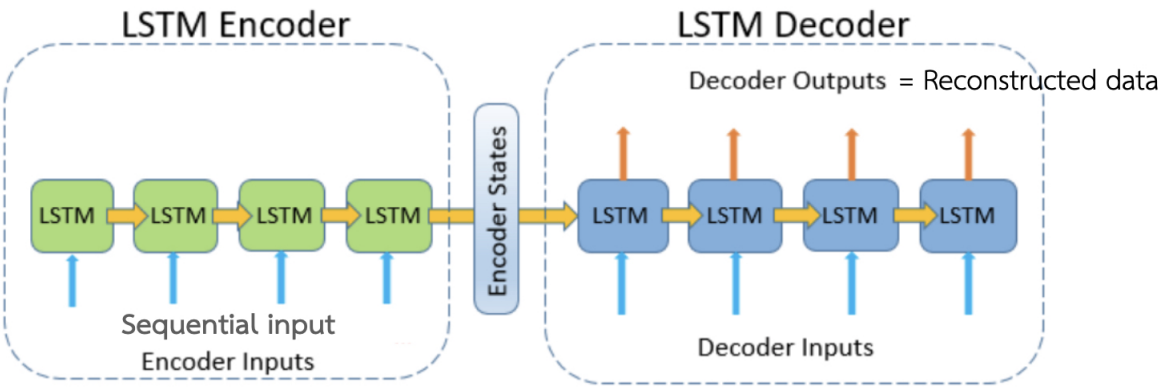
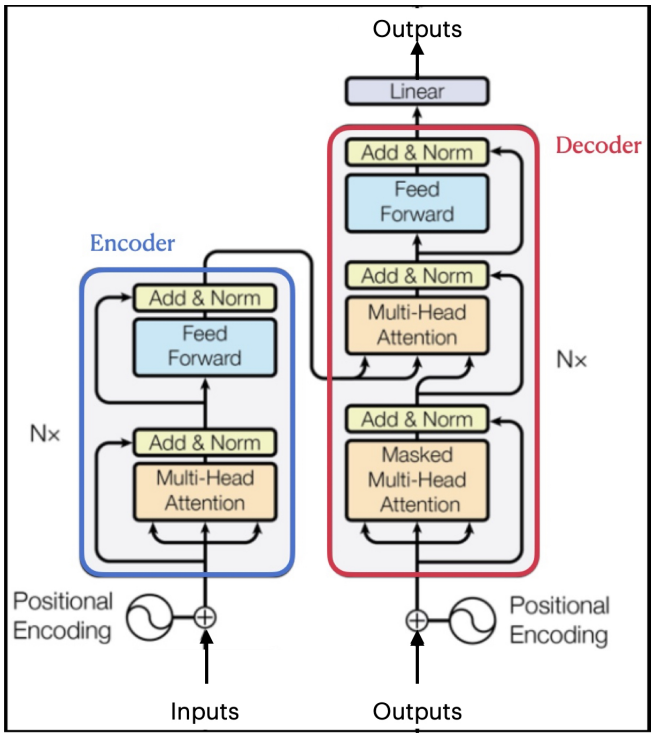



**Figure 3.11**
The transformer architecture.




### 3.3 Evaluation Plan

To assess the performance of the model, the model will be tested on untrained activities such as jumping and dropping the ball at different locus around the installed system. The prefered accuracy should be a 75% hit rate for anomolous events and less than 1 fall positive per day. (Not sure may be the number need to be changed)

# CHAPTER 4

# PRELIMINARY EXPERIMENTS

## 4.1 Experimental Setup

To collect the raw data, experiments are performed in the living room of my house in Bangkok, Thailand, which was built from reinforced concrete with tile, as shown in Figure 4.1. The system can detect vibration in a range around 3 meters, and this room has dimension $3.5 \times 3.5$ m$^2$. The hardware should be installed near the corner in order to be as suitable for the application as possible.

**Figure 4.1**
Living room area used for preliminary experiment.

## 4.2 Ordinary Activiting

There are several activities that occur normally during daily life. I focus on typical activities such as walking, sitting, standing and lying down, as shown in Table 4.1. Under COVID-19, I cannot invite outside volunteers to come indoor for data collection. However, if the COVID situation in Thailand improves, I will invite approximate 3-5 friends to participate in ordinary activities. In the meantime, I plan to collect activities of four subjects, my father, my mother, my older brother, and me. Details of them are shown in Table 4.2.

**Table 4.1**
The detail of each activity and its number of action.

| Human Activity | Number of action |
|---|---|
| Walking | 2,500 |
| Sitting | 400 |
| Standing | 400 |
| Lying | 400 |

**Table 4.2**
Details on each participant.

| Subject | Sex | Age | Weight $(kg)$ |
|---|---|---|---|
| 1 | M | 23 | 58 |
| 2 | M | 25 | 70 |
| 3 | M | 55 | 70 |
| 4 | F | 58 | 75 |

### 4.3 Deployment

The desirable system should be plug & play. Therefore, every component such as seismic sensor, embedded system and Raspberry Pi must be integrated into a small box requiring only a power adapter, as shown in Figure 4.2. When anomaly activities occur, an alert message should be sent via the LINE application to me. I plan to train on data from the living room and test on the dining room in my house, as shown in 4.3.

**Figure 4.2**
The complete system.

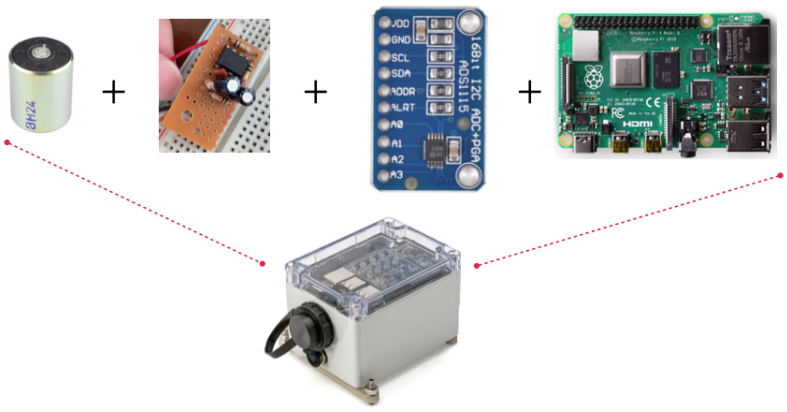


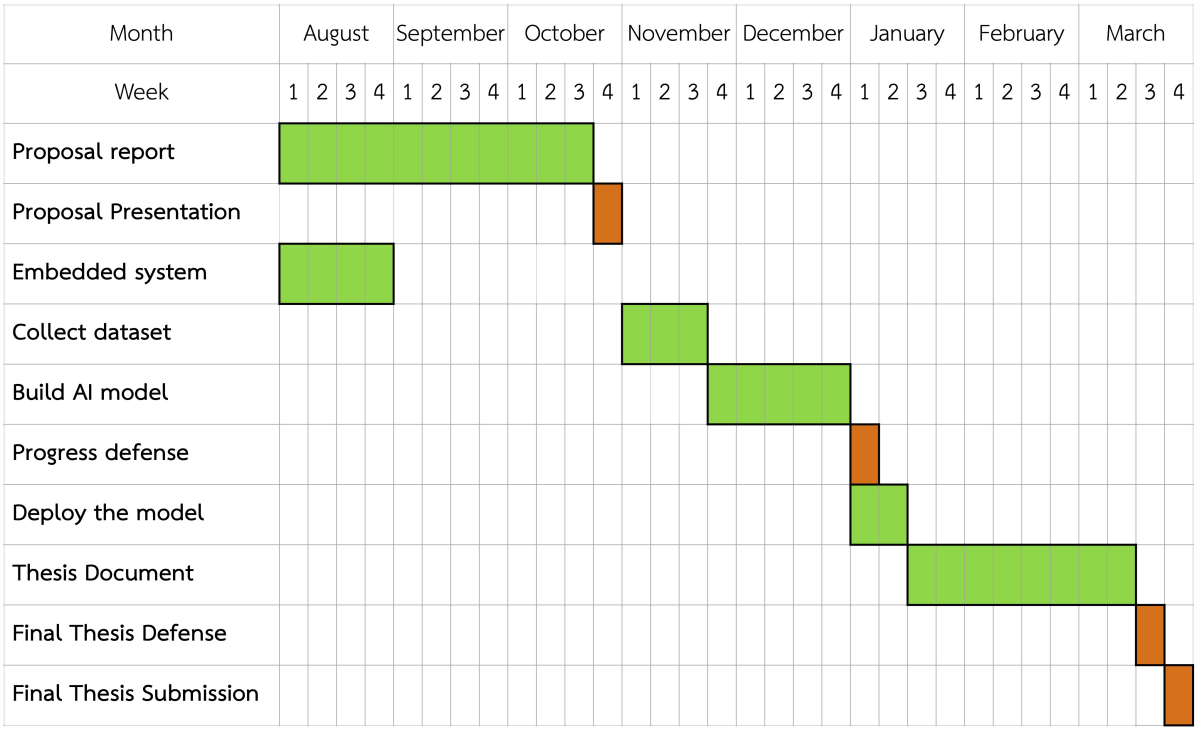
**Figure 4.3**
The dining room in my home.

# CHAPTER 5

# WORK PLAN

Figure 5.1 illustrates the working plan of this study in the year of August 2021 to March 2022.

**Figure 5.1**
The schedule working plan of this study.



# REFERENCES


Abbate, S., Avvenuti, M., Bonatesta, F., Cola, G., Corsini, P., & Vecchio, A. (2012). A smartphone-based fall detection system. *Pervasive and Mobile Computing*, *8*(6), 883–899. doi: 10.1016/j.pmcj.2012.08.003

Agmon, A. (2021). *Hands-on anomaly detection with variational autoencoders.* Retrieved from https://towardsdatascience.com/hands-on-anomaly-detection-with-variational-autoencoders-d4044672acd5

Alammar, J. (2018). *The illustrated transformer.* Retrieved from https://jalammar.github.io/illustrated-transformer/

Alammar, J. (2019). *The illustrated gpt-2 (visualizing transformer language models).* Retrieved from https://jalammar.github.io/illustrated-gpt2/#part-2-illustrated-self-attention

Alwan, M., Dalal, S., Kell, S., & Felder, R. (2003). Derivation of basic human gait characteristics from floor vibrations.

Alwan, M., Rajendran, P., Kell, S., Mack, D., Dalal, S., Wolfe, M., & Felder, R. (2006). *A smart and passive floor-vibration based fall detector for elderly* (Vol. 1). Retrieved from https://ieeexplore.ieee.org/document/1684511 doi: 10.1109/ICTTA.2006.1684511

Arnx, A. (2019). *First neural network for beginners explained (with code).* Towards Data Science. Retrieved from https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf

Badr, W. (2019). *Auto-encoder: What is it? and what is it used for? (part 1).* Retrieved from https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726

Britto Filho, J. C., & Lubaszewski, M. (2020). A highly reliable wearable device for fall detection. *2020 IEEE Latin-American Test Symposium (LATS)*. doi: 10.1109/lats49555.2020.9093673

Charlon, Y., Bourennane, W., Bettahar, F., & Campo, E. (2013). Activity monitoring system for elderly in a context of smart home. *IRBM*, *34*(1), 60–63. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/S1959031812001509 doi: 10.1016/j.irbm.2012.12.014

Chen, Y., & Xue, Y. (2015). A deep learning approach to human activity recognition based on single accelerometer. *2015 IEEE International Conference on Systems, Man, and Cybernetics*. Retrieved from https://ieeexplore.ieee.org/document/7379395/ doi: 10.1109/smc.2015.263

Chollet, F. (2016). *Building autoencoders in keras.* Retrieved from https://blog.keras.io/building-autoencoders-in-keras.html

Clemente, J., Li, F., Valero, M., & Song, W. (2020). Smart seismic sensing for indoor fall detection, location, and notification. *IEEE Journal of Biomedical and Health Informatics*, *24*(2), 524–532. doi: 10.1109/jbhi.2019.2907498

Dash, S. (2020). *An overview of time series forecasting models part 1: Classical time series forecasting models.* Retrieved from https://shaileydash.medium.com/an-overview-of-time-series-forecasting-models-part-1-classical-time-series-forecasting-models-2d877de76e0f

Davis, B. T., Caicedo, J. M., Langevin, S., & Hirth, V. (2011). Use of wireless smart sensors for detecting human falls through structural vibrations. *Civil Engineering Topics, Volume 4*, 383–389. doi: 10.1007/978-1-4419-9316-8_37

Degen, T., Jaeckel, H., Rufer, M., & Wyss, S. (2003). Speedy:a fall detector in a wrist watch. *Seventh IEEE International Symposium on Wearable Computers, 2003. Proceedings.*. doi: 10.1109/iswc.2003.1241410

Donges, N. (2019). *Recurrent neural networks 101: Understanding the basics of rnns and lstm.* Retrieved from https://builtin.com/data-science/recurrent-neural-networks-and-lstm

El-Bendary, N., Tan, Q., C. Pivot, F., & Lam, A. (2013). Fall detection and prevention for the elderly: a review of trends and challenges. *International Journal on Smart Sensing and Intelligent Systems*, *6*(3), 1230–1266. doi: 10.21307/ijssis-2017-588

Fuller, G. F. (2013). Falls in the elderly. *American Family Physician*, *61*(7), 2159. Retrieved from https://www.aafp.org/afp/2000/0401/p2159.html

Gavin, H. (2015). *Vibrations of single degree of freedom systems*. Retrieved from https://people.duke.edu/~hpgavin/cee201/sdof-dyn.pdf

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). *Generative adversarial networks.* Retrieved from https://arxiv.org/abs/1406.2661

InnovationHub, A. R. . (2020). *What is gan and can we use it for our anomaly detection problems?* Retrieved from https://www.macnica.co.jp/business/ai_iot/columns/135130/

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*. doi: 10.1007/s10618-019-00619-1

James, W. (1890). *The principles of psychology, vol i.* Henry Holt and Co. doi: 10.1037/10538-000

Jia, Z., Howard, R. E., Zhang, Y., & Zhang, P. (2017, Apr). Hb-phone: a bed-mounted geophone-based heartbeat monitoring system. *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*. doi: 10.1145/3055031.3055042

Kasturi, S., & Jo, K.-H. (2017). Classification of human fall in top viewed kinect depth images using binary support vector machine. *2017 10th International Conference on Human System Interactions (HSI)*. doi: 10.1109/hsi.2017.8005016

Klingenbrunn, N. (2021). *Transformer implementation for time-series forecasting.* Retrieved from https://medium.com/mlearning-ai/transformer-implementation-for-time-series-forecasting-a9db2db5c820

Litvak, D., Zigel, Y., & Gannot, I. (2008). Fall detection of elderly through floor vibrations and sound. *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. doi: 10.1109/iembs.2008.4650245

Liu, C., Jiang, Z., Su, X., Benzoni, S., & Maxwell, A. (2019). Detection of human fall using floor vibration and multi-features semi-supervised svm. *Sensors*, *19*(17), 3720. doi: 10.3390/s19173720

Ljunggren, F. (2006). *Floor vibration: dynamic properties and subjective perception* (Unpublished doctoral dissertation). Luleå tekniska universitet.

Mishra, P., Verk, R., Fornasier, D., Piciarelli, C., & Foresti, G. L. (2021). *Vt-adl: a vision transformer network for image anomaly detection and localization*. Retrieved from https://arxiv.org/abs/2104.10036

Mukherjee, A., & Zhang, Z. (2020). Multisense: A highly reliable wearable-free human fall detection systems. In *Sensornets* (pp. 29–40).

Müller, M. (2007, 01). Dynamic time warping. *Information Retrieval for Music and Motion*, *2*, 69-84. doi: 10.1007/978-3-540-74048-3_4

NHS. (2019). *Overview - falls.* Retrieved from https://www.nhs.uk/conditions/Falls/

Olah, C. (2015). *Understanding lstm networks – colah's blog.* Retrieved from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Oukrich, N. (2019). *Daily human activity recognition in smart home based on feature selection, neural network and load signature of appliances* (Unpublished doctoral dissertation).

pavithrasv. (2020). *Timeseries anomaly detection using an autoencoder.*

Pedamkar, P. (2019). *Autoencoders — main components and architecture of autoencoder.* Retrieved from https://www.educba.com/autoencoders/

Pynoos, J., Steinman, B. A., & Nguyen, A. Q. (2010). Environmental assessment and modification as fall-prevention strategies for older adults. *Clinics in Geriatric Medicine*, *26*(4), 633–644. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6036911/ doi: 10.1016/j.cger.2010.07.001

Rajan, S. (2021). *Dimensionality reduction using autoencoders in python.* Retrieved from https://www.analyticsvidhya.com/blog/2021/06/dimensionality-reduction-using-autoencoders-in-python/

Ramirez, H., Velastin, S. A., Meza, I., Fabregas, E., Makris, D., & Farias, G. (2021). Fall detection and activity recognition using human skeleton features. *IEEE Access*, *9*, 33532–33542. doi: 10.1109/access.2021.3061626

Reiss, A., & Stricker, D. (2012). Introducing a new benchmarked dataset for activity monitoring. *2012 16th International Symposium on Wearable Computers*. doi: 10.1109/iswc.2012.13

Rihana, S., & Mondalak, J. (2016). *Wearable fall detection system.* Retrieved from https://ieeexplore.ieee.org/abstract/document/7745414/ doi: 10.1109/MECBME.2016.7745414

Rocca, J. (2020). *Understanding variational autoencoders (vaes).* Retrieved from https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Roger. (2021). *Variational autoencoder(vae).* Retrieved from https://medium.com/geekculture/variational-autoencoder-vae-9b8ce5475f68

Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Forster, K., Troster, G., ... Millan, J. d. R. (2010). Collecting complex activity datasets in highly rich networked sensor environments. In (p. 233 - 240). doi: 10.1109/INSS.2010.5573462

Rosebrock, A. (2020). *Denoising autoencoders with keras, tensorflow, and deep learning.* Retrieved from https://www.pyimagesearch.com/2020/02/24/denoising-autoencoders-with-keras-tensorflow-and-deep-learning/

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., & Langs, G. (2017). Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging* (pp. 146–157).

Schrader, L., Vargas Toro, A., Konietzny, S., Rüping, S., Schäpers, B., Steinböck, M., ... Bock, T. (2020). Advanced sensing and human activity recognition in early inter-

vention and rehabilitation of elderly people. *Journal of Population Ageing*, *13*(2), 139–165. doi: 10.1007/s12062-020-09260-z

Shao, Y., Wang, X., Song, W., Ilyas, S., Guo, H., & Chang, W.-S. (2020). Feasibility of using floor vibration to detect human falls. *International Journal of Environmental Research and Public Health*, *18*(1), 200. doi: 10.3390/ijerph18010200

SteelConstruction. (2016). *Floor vibrations.* Retrieved from https://www.steelconstruction.info/Floor_vibrations

Sultana, A., Deb, K., Dhar, P. K., & Koshiba, T. (2021). Classification of indoor human fall events using deep learning. *Entropy*, *23*(3), 328. doi: 10.3390/e23030328

Tamura, Y. (2021). *Positional encoding, residual connections, padding masks: Covering the rest of transformer components – data science blog.* Retrieved from https://data-science-blog.com/blog/2021/04/22/positional-encoding-residual-connections-padding-masks-all-the-details-of-transformer-model/

Tangruamsub, S. (2017). *Long short-term memory (lstm).* Medium. Retrieved from https://medium.com/@sinart.t/long-short-term-memory-lstm-e6cb23b494c6

Taufeeque, M., Koita, S., Spicher, N., & Deserno, T. M. (2021). Multi-camera, multi-person, and real-time fall detection using long short term memory. *Medical Imaging 2021: Imaging Informatics for Healthcare, Research, and Applications*. doi: 10.1117/12.2580700

Tensorflow. (2021). *Deep convolutional generative adversarial network — tensorflow core.* Retrieved from https://www.tensorflow.org/tutorials/generative/dcgan

ThaiNCD.com. (2019). *Old thai people do not fall, no secret tips! for the prevention of falls in tall people.* Retrieved from https://thaincd.com/2016/media-detail.php?id=13551&gid=1-015-009

Toyoda, M., & Sakurai, Y. (2012). *Subsequence matching in data streams — ntt technical review.* Retrieved from https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201301ra1.html

Tsai, T.-H., & Hsu, C.-W. (2019). Implementation of fall detection system based on 3d skeleton for deep learning technique. *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*. doi: 10.1109/gcce46687.2019.9015609

Ugolotti, R., Sassi, F., Mordonini, M., & Cagnoni, S. (2011). Multi-sensor system for detection and classification of human activities. *Journal of Ambient Intelligence and Humanized Computing*, *4*(1), 27–41. doi: 10.1007/s12652-011-0065-z

University, J. H. (2015). *The swiss roll matching example.* Retrieved from http://www.cis.jhu.edu/~cshen/html/PublishSwissRoll.html

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., N. Gomez, A., ... Polosukhin, I. (2017). *Attention is all you need*. Retrieved from https://arxiv.org/abs/1706.03762v5

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning - ICML '08*. doi: 10.1145/1390156.1390294

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, *11*(110), 3371-3408. Retrieved from http://jmlr.org/papers/v11/vincent10a.html

Wang, S., Chen, L., Zhou, Z., Sun, X., & Dong, J. (2015). Human fall detection in surveillance video based on pcanet. *Multimedia Tools and Applications*, *75*(19), 11603–11613. Retrieved from https://link.springer.com/article/10.1007%2Fs11042-015-2698-y doi: 10.1007/s11042-015-2698-y

Wang, X., Ellul, J., & Azzopardi, G. (2020). Elderly fall detection systems: a literature survey. *Frontiers in Robotics and AI*, *7*. doi: 10.3389/frobt.2020.00071

Weng, L. (2018). *From autoencoder to beta-vae.* Retrieved from https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

WHO. (2018). *Falls.* World Health Organization: WHO. Retrieved from https://www.who.int/news-room/fact-sheets/detail/falls

Wikipedia. (2021). *Dynamic time warping.* Retrieved from https://th.wikipedia.org/wiki/Dynamic_time_warping

Yang, C.-C., & Hsu, Y.-L. (2010). A review of accelerometry-based wearable motion detectors for physical activity monitoring. *Sensors*, *10*(8), 7772–7788. doi: 10.3390/s100807772

Yazar, A., Erden, F., & Cetin, A. (2014, 05). Multi-sensor ambient assisted living system for fall detection..