

Article

# Effectiveness of Artificial Neural Networks for Solving Inverse Problems in Magnetic Field-Based Localization

Ai-ichiro Sasaki 

Department of Electronic Engineering and Computer Science, Kindai University, Higashi-Hiroshima 739-2116, Japan; aisasaki@hiro.kindai.ac.jp; Tel.: +81-82-439-1110 (ext. 620)

**Abstract:** Recently, indoor localization has become an active area of research. Although there are various approaches to indoor localization, methods that utilize artificially generated magnetic fields from a target device are considered to be the best in terms of localization accuracy under non-line-of-sight conditions. In magnetic field-based localization, the target position must be calculated based on the magnetic field information detected by multiple sensors. The calculation process is equivalent to solving a nonlinear inverse problem. Recently, a machine-learning approach has been proposed to solve the inverse problem. Reportedly, adopting the  $k$ -nearest neighbor algorithm ( $k$ -NN) enabled the machine-learning approach to achieve fairly good performance in terms of both localization accuracy and computational speed. Moreover, it has been suggested that the localization accuracy can be further improved by adopting artificial neural networks (ANNs) instead of  $k$ -NN. However, the effectiveness of ANNs has not yet been demonstrated. In this study, we thoroughly investigated the effectiveness of ANNs for solving the inverse problem of magnetic field-based localization in comparison with  $k$ -NN. We demonstrate that despite taking longer to train, ANNs are superior to  $k$ -NN in terms of localization accuracy. The  $k$ -NN is still valid for predicting fairly accurate target positions within limited training times.

**Keywords:** artificial neural networks; indoor localization; inverse problem;  $k$ -nearest neighbor algorithm; magnetic field; optimization; real-time tracking



**Citation:** Sasaki, A.-i. Effectiveness of Artificial Neural Networks for Solving Inverse Problems in Magnetic Field-Based Localization. *Sensors* **2022**, *22*, 2240. <https://doi.org/10.3390/s22062240>

Academic Editors:

Joaquín Torres-Sospedra,  
Ying-Ren Chien, Mu Zhou, Ao Peng  
and Ni Zhu

Received: 8 February 2022

Accepted: 11 March 2022

Published: 14 March 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Location-based services have become indispensable in daily life. This is primarily owing to global positioning systems (GPS), whose performance has reached an unprecedentedly high level [1]. The significance of location-based services will increase in a full-blown era of the Internet of Things (IoT) [2]. Although GPS effectively localizes objects existing outdoors, it is not suitable for indoor localization since radio-wave propagation is disturbed by buildings. Indoor localization techniques have been actively studied to compensate for the limitations of GPS [3–7].

There are two main approaches for realizing indoor localization. One approach uses radio waves [5–7]. Since radio waves can reach far points, it is possible with this approach to obtain a large coverage area. A remarkable feature of radio waves is that they are reflected by walls, floors, and obstacles. On account of this feature, it has been a severe problem to improve localization accuracy with radio waves.

The second approach uses magnetic fields. The magnetic field-based approach is further divided into two different methods. One method utilizes geomagnetic fields, which are static fields [8–11]. The geomagnetic field-based method adopts fingerprinting techniques and is conducive for covering relatively wide areas. To execute the geomagnetic field-based localization, we must gather considerable amounts of data of the field strength at various points of the target area in advance. Moreover, it is required to update the magnetic field data since geomagnetic field patterns gradually change over periods of time.

Since it is a tedious task to frequently gather the numerous data, establishing accurate localization systems using geomagnetic fields is very tough.

The second magnetic field-based localization method utilizes artificially generated magnetic fields, which are not static but time-varying fields [12–15]. Generally, the frequencies used for this method are within low-frequency (LF) or high-frequency (HF) bands. A remarkable feature of artificially generated magnetic fields is that the field patterns can be easily calculated with reasonable precision. Owing to this feature, the tedious task involved in the geomagnetic field-based method is no longer needed with the artificially generated fields. The other feature of magnetic fields is that the interaction between the fields and materials is much weaker than that between radio waves and materials, and this feature is not only limited to artificially generated fields but also to geomagnetic fields. Owing to these two features, the method involving artificially generated magnetic fields is best in terms of localization accuracy in non-line-of-sight (NLOS) environments [4].

Localization accuracy is considered the most important aspect for location-based services. Therefore, in this study, we treat localization techniques based on artificially generated magnetic fields. The only weakness of these techniques is that their coverage distance is typically limited to less than 10 m, which is much shorter than that of radio waves. It is due to the fact that artificially generated magnetic fields attenuate more steeply than radio waves. Localization techniques that combine radio waves with a magnetic field-based approach have been proposed to compensate for the weak point of artificially generated fields [5,6]. Moreover, techniques for amplifying magnetic fields without increasing the power consumption have also been investigated [16,17].

Typical localization systems that use artificially generated magnetic fields are composed of multiple sensors placed at various positions within the target area. Magnetic fields generated by a target device are detected by these sensors. The position of the target device must be calculated using the magnetic field data detected by the multiple sensors. However, this calculation is not trivial since it involves solving nonlinear inverse problems. Three different methods are known for solving inverse problems in localization with artificially generated magnetic fields.

The simplest method entails directly calculating the position of the target using closed-form formulae [12]. The main advantage of this method is its short computational time. It becomes possible with this method to establish real time location tracking of persons and objects. However, the method is applicable only to systems having limited sensor configurations in terms of numbers, positions, and their angles. Therefore, it is difficult to flexibly design localization systems with the simplest method.

The second method solves optimization problems that are reduced from inverse problems. The conventional optimization method can be employed to predict the position of the target from the detected signals by minimizing an appropriately constructed objective function [3,13,14]. An advantage of the optimization method is that the method can be utilized for any sensor configurations. Therefore, the optimization method surpasses the closed-form formulae method from a viewpoint of flexibility in the system design. However, it is difficult to implement real-time tracking of moving targets with the optimization method since it requires a much longer computational time. Moreover, since the optimization method requires certain skills, it is not easy for beginners to obtain appropriate solutions using the method. In fact, experienced engineers are often bothered about the problem of local minima.

The third method involves the application of machine learning to solve the inverse problems [18,19]. We have demonstrated that the computational time required for calculating the target position is significantly reduced by adopting machine learning. Moreover, it has been shown that the prediction accuracy of the target position is improved by using machine learning instead of the conventional optimization method. Therefore, it is expected that real-time tracking of moving targets is possible with high precision using machine learning. Additionally, machine learning can be applied to any sensor configuration, which is a significant advantage. Furthermore, even beginners can easily obtain

appropriate solutions for inverse problems as many useful machine-learning tools are becoming increasingly available.

Since there are many algorithms for machine learning, it is critical to select an algorithm suitable for magnetic field-based localization. It has been demonstrated that we can predict the target positions in reasonable accuracy within a short computational time by adopting the  $k$ -nearest neighbor algorithm ( $k$ -NN). Furthermore, it is well known that artificial neural networks (ANNs) exhibit excellent performance in various machine-learning applications. It was also suggested that better prediction accuracy may be obtained with ANNs in comparison with  $k$ -NN [20]. However, a performance comparison between these two algorithms has not yet been reported.

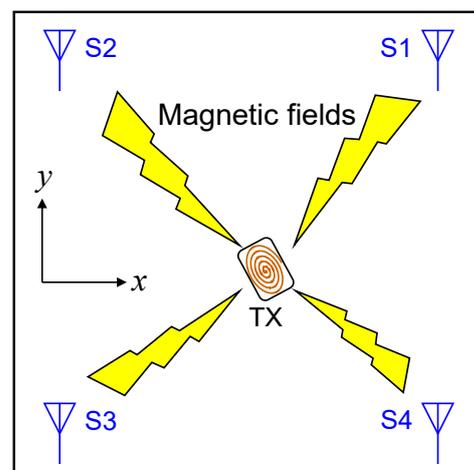
In this study, we thoroughly investigated the effectiveness of ANNs for magnetic field-based localization in comparison with  $k$ -NN. In Section 2, we formulate the inverse problems to be solved for realizing magnetic field-based localization. Moreover, the machine-learning approach for solving inverse problems is also explained. In Section 3, the performances obtained with the two different machine-learning algorithms, ANNs and  $k$ -NN, are compared from various viewpoints. It was demonstrated that ANNs exhibit better prediction accuracy, although they require a much longer training time in comparison with  $k$ -NN.

## 2. Formulation of Inverse Problems

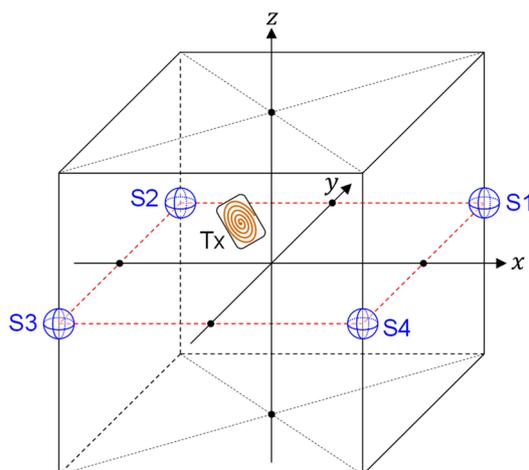
In this section, we describe the basic of calculation techniques required for magnetic field-based localization and formulate the inverse problems to be solved by machine learning [18,19].

### 2.1. Calculation of Artificially Generated Magnetic Fields

Figures 1 and 2 show a typical localization system that utilizes artificially generated magnetic fields. Suppose that the target device having a transmitter (TX) is freely moving inside the cubic space. Magnetic fields are generated from a single coil connected to the TX and they are detected by magnetic field sensors (S1–S4) placed at four corners. Although the four sensors are depicted in Figures 1 and 2 as an example, the number of sensors is not limited to four. Furthermore, locations of the sensors are not limited to the corners. Our machine-learning approach has no limitation regarding the number and location of the sensors. In this study, however, we assume that four sensors are located at four corners of the cubic space.



**Figure 1.** A conceptual image of a localization system with artificially generated magnetic fields (Top view). Magnetic fields generated by a TX are detected by multiple sensors (S1–S4) located at various points of the target space. The position of the TX is calculated from the information of the magnetic fields detected by the sensors.



**Figure 2.** A three-dimensional (3D) drawing of a localization system with artificially generated magnetic fields.

We also consider that the sensors can detect all components of the magnetic field vectors  $\mathbf{B} = (B_x, B_y, B_z)$ . In other words, each sensor is equipped with three receiving coils, that are perpendicular to each other. Our objective is to calculate the position of the TX based on the magnetic field information detected by the four sensors.

Magnetic field patterns generated inside the cubic space depend on both the position and angles of the TX coil. Therefore, magnetic-field vector components detected by the  $k$ -th sensor can be formally written as follows:

$$\mathbf{B}^{(k)} = \begin{pmatrix} B_x^{(k)} \\ B_y^{(k)} \\ B_z^{(k)} \end{pmatrix} = \begin{pmatrix} B_x^{(k)}(x^{(t)}, y^{(t)}, z^{(t)}, \theta^{(t)}, \varphi^{(t)}) \\ B_y^{(k)}(x^{(t)}, y^{(t)}, z^{(t)}, \theta^{(t)}, \varphi^{(t)}) \\ B_z^{(k)}(x^{(t)}, y^{(t)}, z^{(t)}, \theta^{(t)}, \varphi^{(t)}) \end{pmatrix}, \quad (1)$$

where  $(x^{(t)}, y^{(t)}, z^{(t)})$  denote the position of the TX coil and  $(\theta^{(t)}, \varphi^{(t)})$  represent the direction of a vector normal to the TX coil in spherical coordinates. The superscript “(t)” indicates that the symbols having “(t)” represent the physical quantities of the TX. Hereinafter,  $(x^{(t)}, y^{(t)}, z^{(t)}, \theta^{(t)}, \varphi^{(t)})$  are referred to as “TX-state parameters” since they represent the physical state of the TX.

The magnetic fields created at an arbitrary point  $(x, y, z)$  by the TX coil placed at a coordinate origin can be calculated using the following equation [18,19]:

$$\begin{pmatrix} B_x(x, y, z, \theta^{(t)}, \varphi^{(t)}) \\ B_y(x, y, z, \theta^{(t)}, \varphi^{(t)}) \\ B_z(x, y, z, \theta^{(t)}, \varphi^{(t)}) \end{pmatrix} = \frac{\mu_0}{4\pi} \frac{m}{(x^2 + y^2 + z^2)^{5/2}} \times \left[ \cos \theta^{(t)} \begin{pmatrix} 3xz \\ 3yz \\ 2z^2 - x^2 - y^2 \end{pmatrix} + \sin \theta^{(t)} \begin{pmatrix} 3xy \sin \varphi^{(t)} + (2x^2 - y^2 - z^2) \cos \varphi^{(t)} \\ 3xy \cos \varphi^{(t)} + (2x^2 - y^2 - z^2) \sin \varphi^{(t)} \\ 3z(x \cos \varphi^{(t)} + y \sin \varphi^{(t)}) \end{pmatrix} \right], \quad (2)$$

where  $\mu_0$  and  $m$  denote the vacuum permeability and dipole moment associated with the TX coil, respectively. The dipole moment can be expressed by the parameters of an  $N$ -turn coil attached to the TX as follows:

$$m = NIS, \quad (3)$$

where  $I$  and  $S$  denote the current and area size of the TX coil, respectively.

Note that the magnetic-field vector components detected by the  $k$ -th sensor can be written in terms of the left-hand side of Equation (2) as follows:

$$\begin{pmatrix} B_x^{(k)}(x^{(t)}, y^{(t)}, z^{(t)}, \theta^{(t)}, \varphi^{(t)}) \\ B_y^{(k)}(x^{(t)}, y^{(t)}, z^{(t)}, \theta^{(t)}, \varphi^{(t)}) \\ B_z^{(k)}(x^{(t)}, y^{(t)}, z^{(t)}, \theta^{(t)}, \varphi^{(t)}) \end{pmatrix} = \begin{pmatrix} B_x(x^{(k)} - x^{(t)}, y^{(k)} - y^{(t)}, z^{(k)} - z^{(t)}, \theta^{(t)}, \varphi^{(t)}) \\ B_y(x^{(k)} - x^{(t)}, y^{(k)} - y^{(t)}, z^{(k)} - z^{(t)}, \theta^{(t)}, \varphi^{(t)}) \\ B_z(x^{(k)} - x^{(t)}, y^{(k)} - y^{(t)}, z^{(k)} - z^{(t)}, \theta^{(t)}, \varphi^{(t)}) \end{pmatrix}, \quad (4)$$

where  $(x^{(k)}, y^{(k)}, z^{(k)})$  denote the positions of the  $k$ -th sensor. Substituting Equation (2) into the right-hand side of Equation (4), we can derive a useful formula to calculate the magnetic-field vector components detected by the  $k$ -th sensor for any TX-state parameters as shown below.

$$\begin{pmatrix} B_x^{(k)}(x^{(t)}, y^{(t)}, z^{(t)}, \theta^{(t)}, \varphi^{(t)}) \\ B_y^{(k)}(x^{(t)}, y^{(t)}, z^{(t)}, \theta^{(t)}, \varphi^{(t)}) \\ B_z^{(k)}(x^{(t)}, y^{(t)}, z^{(t)}, \theta^{(t)}, \varphi^{(t)}) \end{pmatrix} = \frac{\mu_0}{4\pi} \frac{m}{\{(x^{(k)} - x^{(t)})^2 + (y^{(k)} - y^{(t)})^2 + (z^{(k)} - z^{(t)})^2\}^{5/2}} \times \left[ \cos \theta^{(t)} \begin{pmatrix} 3(x^{(k)} - x^{(t)})(z^{(k)} - z^{(t)}) \\ 3(y^{(k)} - y^{(t)})(z^{(k)} - z^{(t)}) \\ 2(z^{(k)} - z^{(t)})^2 - (x^{(k)} - x^{(t)})^2 - (y^{(k)} - y^{(t)})^2 \end{pmatrix} + \sin \theta^{(t)} \begin{pmatrix} 3(x^{(k)} - x^{(t)})(y^{(k)} - y^{(t)}) \sin \varphi^{(t)} + \{2(x^{(k)} - x^{(t)})^2 - (y^{(k)} - y^{(t)})^2 - (z^{(k)} - z^{(t)})^2\} \cos \varphi^{(t)} \\ 3(x^{(k)} - x^{(t)})(y^{(k)} - y^{(t)}) \cos \varphi^{(t)} + \{2(x^{(k)} - x^{(t)})^2 - (y^{(k)} - y^{(t)})^2 - (z^{(k)} - z^{(t)})^2\} \sin \varphi^{(t)} \\ 3(z^{(k)} - z^{(t)}) \{(x^{(k)} - x^{(t)}) \cos \varphi^{(t)} + (y^{(k)} - y^{(t)}) \sin \varphi^{(t)}\} \end{pmatrix} \right] \quad (5)$$

### 2.2. Solving Inverse Problems Using Machine Learning

Since the above formula has been obtained, it is easy to calculate  $(B_x^{(k)}, B_y^{(k)}, B_z^{(k)})$  from the arbitrary TX-state parameters  $(x^{(t)}, y^{(t)}, z^{(t)}, \theta^{(t)}, \varphi^{(t)})$ . It is a straightforward problem. However, magnetic field-based localization requires the calculation of  $(x^{(t)}, y^{(t)}, z^{(t)})$  from  $(B_x^{(k)}, B_y^{(k)}, B_z^{(k)})$ , and the calculation is not easy since it is a nonlinear inverse problem. Solving the inverse problem is equivalent to finding a function  $F$  that satisfies

$$\mathbf{x}^{(t)} = \mathbf{F}(\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \mathbf{B}^{(3)}, \mathbf{B}^{(4)}), \quad (6)$$

where  $\mathbf{x}^{(t)} = (x^{(t)}, y^{(t)}, z^{(t)})$ . After observing Equation (5), it is hopeless to obtain the exact form of  $F$ . Therefore, the next step involves constructing a function that can “predict” the approximate values of the TX position based on the magnetic field information detected by the four sensors. In equation form, the predictor function  $P$  can be expressed as follows:

$$\mathbf{x}^{(p)} = \mathbf{P}(\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \mathbf{B}^{(3)}, \mathbf{B}^{(4)}), \quad (7)$$

where  $\mathbf{x}^{(p)} = (x^{(p)}, y^{(p)}, z^{(p)})$  denotes predicted TX-position vector. It is emphasized with the superscript (p) that  $\mathbf{x}^{(p)}$  is the predicted quantity and is not identical to  $\mathbf{x}^{(t)}$ , which is a true TX-position vector. Although  $\mathbf{x}^{(p)}$  is considered an approximation of  $\mathbf{x}^{(t)}$ , we should

note that  $\mathbf{x}^{(p)}$  is not completely equal to  $\mathbf{x}^{(t)}$ . In the component representation, Equation (7) can be written as follows:

$$\begin{pmatrix} x^{(p)} \\ y^{(p)} \\ z^{(p)} \end{pmatrix} = \begin{pmatrix} P_x \left( B_x^{(1)}, B_y^{(1)}, B_z^{(1)}, B_x^{(2)}, B_y^{(2)}, B_z^{(2)}, B_x^{(3)}, B_y^{(3)}, B_z^{(3)}, B_x^{(4)}, B_y^{(4)}, B_z^{(4)} \right) \\ P_y \left( B_x^{(1)}, B_y^{(1)}, B_z^{(1)}, B_x^{(2)}, B_y^{(2)}, B_z^{(2)}, B_x^{(3)}, B_y^{(3)}, B_z^{(3)}, B_x^{(4)}, B_y^{(4)}, B_z^{(4)} \right) \\ P_z \left( B_x^{(1)}, B_y^{(1)}, B_z^{(1)}, B_x^{(2)}, B_y^{(2)}, B_z^{(2)}, B_x^{(3)}, B_y^{(3)}, B_z^{(3)}, B_x^{(4)}, B_y^{(4)}, B_z^{(4)} \right) \end{pmatrix} \quad (8)$$

Generally, regression analysis is an effective method for constructing  $\mathbf{P}$ . It is possible to construct  $\mathbf{P}$  with reasonable accuracy through regression analysis by gathering sufficient volume of training data. In equation form, the training data are expressed as follows:

$$\left( \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \mathbf{B}^{(3)}, \mathbf{B}^{(4)} \right) \mapsto \mathbf{x}^{(t)}, \quad (9)$$

where the left- and right-hand sides denote the input and desired outputs of  $\mathbf{P}$ , respectively. In the component representation, Equation (9) can be written as follows:

$$\left( B_x^{(1)}, B_y^{(1)}, B_z^{(1)}, B_x^{(2)}, B_y^{(2)}, B_z^{(2)}, B_x^{(3)}, B_y^{(3)}, B_z^{(3)}, B_x^{(4)}, B_y^{(4)}, B_z^{(4)} \right) \mapsto \left( x^{(t)}, y^{(t)}, z^{(t)} \right) \quad (10)$$

Hereinafter, the training-data representation used in Equations (9) and (10) is referred to as “linear representation”.

Generally speaking, it is often required to gather training data from a real world for the regression analysis. However, for our purpose, measurements in the real world are not required since the magnetic-field patterns formed by the TX coil can be calculated by Equation (2) in reasonable accuracy. Actually, existing localization systems that use artificially generated magnetic fields are based on the fact that the fields are governed by Equation (2) [12–14].

Therefore, for our purpose, sufficient quantity of training data can be easily generated by applying Equation (5) to various TX-state parameters. This is a straightforward procedure. Presently, it is well known that machine learning is effective for regression analysis. When machine learning works properly with a sufficient number of training data, we can obtain a reasonable predictor function  $\mathbf{P}$  that satisfies Equations (7) and (8).

Given that  $\mathbf{B}^{(k)}$  depends on  $\mathbf{x}^{(t)}$ ,  $\theta^{(t)}$ , and  $\varphi^{(t)}$ , the predictor function can also be viewed as a function of these variables. Subsequently, we introduce a new predictor function,  $\tilde{\mathbf{P}}$  defined as

$$\mathbf{x}^{(p)} = \mathbf{P} \left( \mathbf{B}^{(1)} \left( \mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)} \right), \mathbf{B}^{(2)} \left( \mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)} \right), \mathbf{B}^{(3)} \left( \mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)} \right), \mathbf{B}^{(4)} \left( \mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)} \right) \right) \quad (11)$$

$$\triangleq \tilde{\mathbf{P}} \left( \mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)} \right).$$

It is reasonable to appraise the performance of the predictor function by the difference between  $\mathbf{x}^{(t)}$  and  $\mathbf{x}^{(p)}$ . Hence, we define an error distance function (EDF) as follows:

$$d \left( \mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)} \right) \triangleq \left\| \mathbf{x}^{(t)} - \tilde{\mathbf{P}} \left( \mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)} \right) \right\|_2, \quad (12)$$

where the subscript “2” in the right-hand side of Equation (12) denotes that the EDF is defined by the Euclidean distance between  $\mathbf{x}^{(t)}$  and  $\mathbf{x}^{(p)}$ . Since the EDF represents the error of the predictor function, the values of  $d$  decrease with improvements in the predictor functions.

### 2.3. Method for Generating Better Predictor Functions

When  $\theta^{(t)} = 0$ , Equation (2) can be simplified by using a spherical coordinate.

$$\begin{pmatrix} B_r(r, \theta, \varphi) \\ B_\theta(r, \theta, \varphi) \\ B_\varphi(r, \theta, \varphi) \end{pmatrix} = \frac{\mu_0 m}{4\pi r^3} \begin{pmatrix} 2 \cos \theta \\ \sin \theta \\ 0 \end{pmatrix} \quad (13)$$

Here  $(r, \theta, \varphi)$  denote variables of the spherical coordinate system. Note that  $(\theta, \varphi)$  are different from TX-state parameters  $(\theta^{(t)}, \varphi^{(t)})$ . It is understood from Equation (13) that the amplitude of fields generated by a magnetic source is proportional to  $r^{-3}$ . This is a remarkable property of near fields generated by a magnetic dipole, which is nothing but a current loop. In contrast, it is well known that the amplitude of radio waves is proportional to  $r^{-1}$ . More specifically, the spatial attenuation ratio of magnetic fields (60 dB/decade) is much larger than that of radio waves (20 dB/decade).

In this study, we treat the magnetic near fields, whose attenuation ratio is 60 dB/decade. Performances of the localization systems that use artificially generated magnetic fields are governed mainly by the large attenuation ratio. When the TX moves proximity of a sensor, the sensor detects a considerably larger signal level. Moreover, the variations in the detected signal level also become very large. On the other hand, when the TX are existing in the regions distant from a sensor, both the signal level and its variations become extremely small. It was suggested that the prediction accuracy obtained with machine learning is limited by these differences in the behavior of signal levels [18].

To solve this problem, an effective method for preprocessing training data has been proposed [19]. The core idea of the method is to logarithmically transform the magnetic field values in the training data. In the equation form, the transformation is written as follows:

$$\hat{B}_i^{(k)} \triangleq \text{sign}(B_i^{(k)}) \log \left( \frac{|B_i^{(k)}|}{\max_{(\text{all training data})} \{|B_i^{(k)}|\}} \right), \quad (14)$$

where  $i$  can be  $x$ ,  $y$ , or  $z$ . Note that the positive/negative information of  $B_i^{(k)}$  is encoded in  $\hat{B}_i^{(k)}$ . Moreover, a one-to-one correspondence is satisfied between  $B_i^{(k)}$  and  $\hat{B}_i^{(k)}$  [19]. The preprocessed training data are then written as follows:

$$\left( \hat{B}_x^{(1)}, \hat{B}_y^{(1)}, \hat{B}_z^{(1)}, \hat{B}_x^{(2)}, \hat{B}_y^{(2)}, \hat{B}_z^{(2)}, \hat{B}_x^{(3)}, \hat{B}_y^{(3)}, \hat{B}_z^{(3)}, \hat{B}_x^{(4)}, \hat{B}_y^{(4)}, \hat{B}_z^{(4)} \right) \mapsto \left( x^{(t)}, y^{(t)}, z^{(t)} \right) \quad (15)$$

As with Equation (9), we can express Equation (15) in a simple form using vector notation.

$$\left( \hat{\mathbf{B}}^{(1)}, \hat{\mathbf{B}}^{(2)}, \hat{\mathbf{B}}^{(3)}, \hat{\mathbf{B}}^{(4)} \right) \mapsto \mathbf{x}^{(t)} \quad (16)$$

Hereinafter, the training-data representation used in Equations (15) and (16) is referred to as “normalized-signed log (NSL) representation.” Accordingly, we denote the predictor function generated by the training data in the NSL representation by  $\mathbf{Q}$ . Subsequently, the equations corresponding to Equations (7), (8), (11), and (12) can be written as follows:

$$\mathbf{x}^{(p)} = \mathbf{Q} \left( \hat{\mathbf{B}}^{(1)}, \hat{\mathbf{B}}^{(2)}, \hat{\mathbf{B}}^{(3)}, \hat{\mathbf{B}}^{(4)} \right) \quad (17)$$

$$\begin{pmatrix} x^{(p)} \\ y^{(p)} \\ z^{(p)} \end{pmatrix} = \begin{pmatrix} Q_x \left( \hat{B}_x^{(1)}, \hat{B}_y^{(1)}, \hat{B}_z^{(1)}, \hat{B}_x^{(2)}, \hat{B}_y^{(2)}, \hat{B}_z^{(2)}, \hat{B}_x^{(3)}, \hat{B}_y^{(3)}, \hat{B}_z^{(3)}, \hat{B}_x^{(4)}, \hat{B}_y^{(4)}, \hat{B}_z^{(4)} \right) \\ Q_y \left( \hat{B}_x^{(1)}, \hat{B}_y^{(1)}, \hat{B}_z^{(1)}, \hat{B}_x^{(2)}, \hat{B}_y^{(2)}, \hat{B}_z^{(2)}, \hat{B}_x^{(3)}, \hat{B}_y^{(3)}, \hat{B}_z^{(3)}, \hat{B}_x^{(4)}, \hat{B}_y^{(4)}, \hat{B}_z^{(4)} \right) \\ Q_z \left( \hat{B}_x^{(1)}, \hat{B}_y^{(1)}, \hat{B}_z^{(1)}, \hat{B}_x^{(2)}, \hat{B}_y^{(2)}, \hat{B}_z^{(2)}, \hat{B}_x^{(3)}, \hat{B}_y^{(3)}, \hat{B}_z^{(3)}, \hat{B}_x^{(4)}, \hat{B}_y^{(4)}, \hat{B}_z^{(4)} \right) \end{pmatrix} \quad (18)$$

$$\mathbf{x}^{(p)} = \mathbf{Q} \left( \hat{\mathbf{B}}^{(1)}(\mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)}), \hat{\mathbf{B}}^{(2)}(\mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)}), \hat{\mathbf{B}}^{(3)}(\mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)}), \hat{\mathbf{B}}^{(4)}(\mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)}) \right) \quad (19)$$

$$\triangleq \tilde{\mathbf{Q}}(\mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)})$$

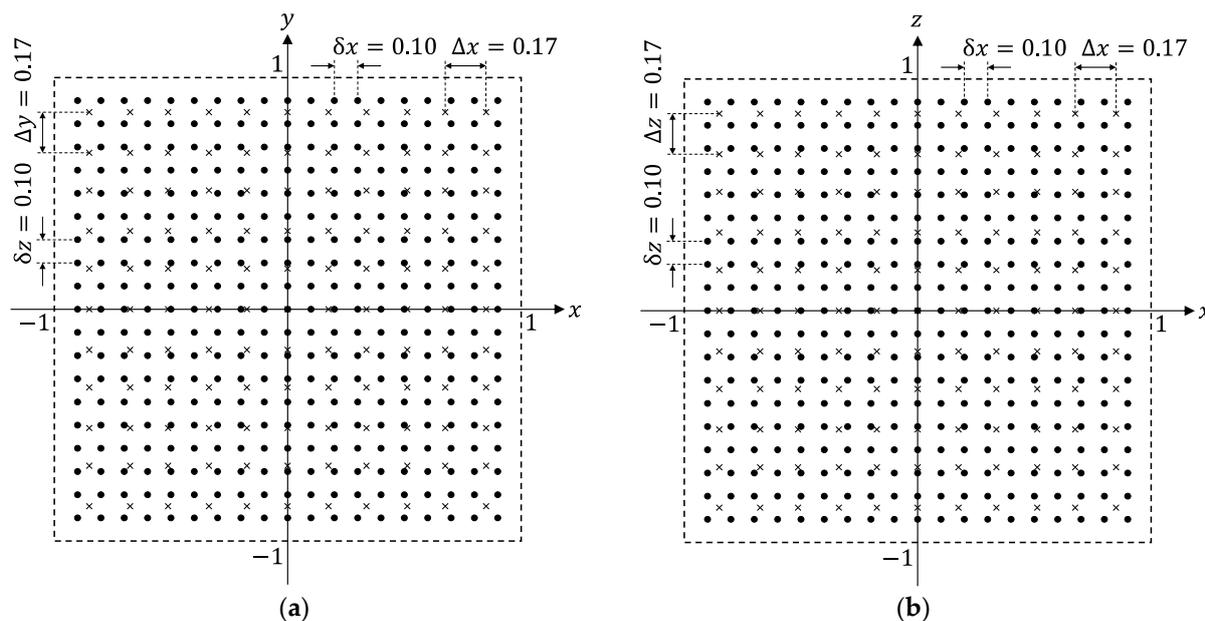
$$d(\mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)}) \triangleq \|\mathbf{x}^{(t)} - \tilde{\mathbf{Q}}(\mathbf{x}^{(t)}, \theta^{(t)}, \varphi^{(t)})\|_2 \quad (20)$$

### 3. Performance Evaluation of Machine Learning in Solving the Inverse Problems

In this section, the performance of predictor functions generated by machine learning is thoroughly investigated. In this study, we adopted ANNs and  $k$ -NN as methods for machine learning.

#### 3.1. Conditions of Numerical Calculation

The scenarios and conditions of our investigation are shown in Figures 1–3. In this study, we assume that the target (TX) exists inside a cubic space of length (2 m)  $\times$  height (2 m)  $\times$  width (2 m). However, the dimensions of the cubic space are not fixed since the scaling law holds in localization with artificially generated magnetic fields. Therefore, the results obtained with the 2 m  $\times$  2 m  $\times$  2 m cubic space can be easily extended to cubic spaces with arbitrary dimensions. For example, it is possible to obtain a predictor function valid for a cubic space of length (4 m)  $\times$  height (4 m)  $\times$  width (4 m) by using the same model without increasing training data. Details on this topic are discussed in [18].



**Figure 3.** (a)  $x$ - $y$  plane and (b)  $x$ - $z$  plane sliced in the cubic space shown in Figure 2. The numerical values are described in meters. In this case, the length of one side of the cubic space is supposed to be 2 m. TX positions used to calculate training data are indicated with black dots. The distance between the two nearest dots is 0.10 m. TX positions used to calculate EDF are indicated using crosses. The distance between the two nearest crosses is considered to be 0.17 m. The positions of the dots and crosses do not coincide except for the coordinate origin.

What we should do first is to generate sufficient numbers of training samples in the forms of Equations (10) or (15). Fortunately, we do not have to gather training samples by cumbersome measurement in real systems. It is possible to obtain numbers of training samples just by calculating  $\mathbf{B}^{(k)}$  for many different TX-state parameters using Equation (5). The TX-positions chosen to calculate the training samples are indicated by black dots in Figure 3a,b, depicting the  $x$ - $y$  and  $x$ - $z$  planes of the cubic space, respectively. The numerical values in Figure 3 are in meter. Hence, the distance between the two adjacent dots is

$\delta x = \delta y = \delta z = 0.10$  m. In this condition, the amount of TX positions for the training samples reaches  $19^3 = 6859$ . It should be noted that  $\mathbf{B}^{(k)}$  is dependent on both positions and angles of the TX. Hence, we must calculate  $\mathbf{B}^{(k)}$  for various TX-angle sets  $(\theta^{(t)}, \varphi^{(t)})$  for each TX position. Here, we adopt  $\delta\theta^{(t)} = \delta\varphi^{(t)} = 11.25^\circ$  to calculate the training samples for different TX-angle sets. Therefore, the number of angle sets was 482. Hence, the total number of TX states used for training data reached  $N_{\text{training}} = 6859 \times 482 = 3,306,038$ .

In this study, Wolfram Mathematica 12.0 was used to execute machine learning [18,19]. Mathematica includes highly automated functions related to machine learning. We used the “Predict” command to generate predictor functions  $\mathbf{P}$  and  $\mathbf{Q}$ . The command can be executed to automatically generate the predictor functions by inputting training data. We also selected “Quality” as an option command of Mathematica for setting a performance goal. Additionally, it is possible to select several algorithms for machine learning using Mathematica. In this study, we selected “Neural Network” and “Nearest Neighbors” for the algorithms and used a standard computer with Intel Xeon W-2223 and 64-GB RAM.

To evaluate generalization performances of predictor functions obtained by machine learning, we must calculate the EDF values ( $d$ ) for various TX-state parameters that were not used for calculating training samples. The TX positions used for evaluating the generalization performances are indicated by crosses in Figure 3. It can be confirmed that the positions of the crosses do not coincide with those of dots, which are used for training.

### 3.2. Performance Evaluation of Predictor Functions

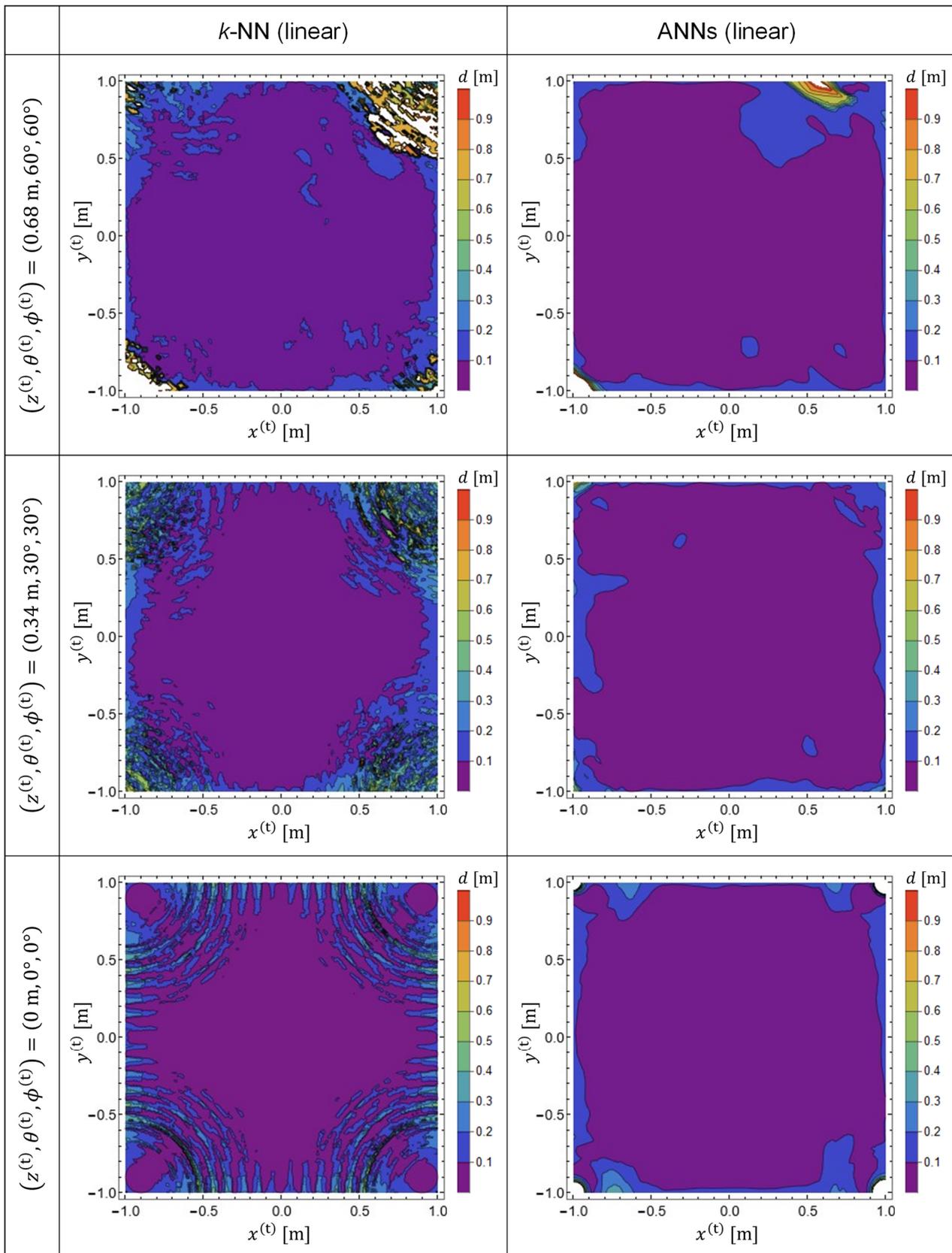
We calculated the EDF values and plotted them within  $x$ - $y$  planes for three different TX-state parameters to compare the performances of the predictor functions generated by  $k$ -NN and ANNs, as shown below.

$$(z^{(t)}, \theta^{(t)}, \varphi^{(t)}) = (0 \text{ m}, 0^\circ, 0^\circ), (0.34 \text{ m}, 30^\circ, 30^\circ), (0.68 \text{ m}, 60^\circ, 60^\circ) \quad (21)$$

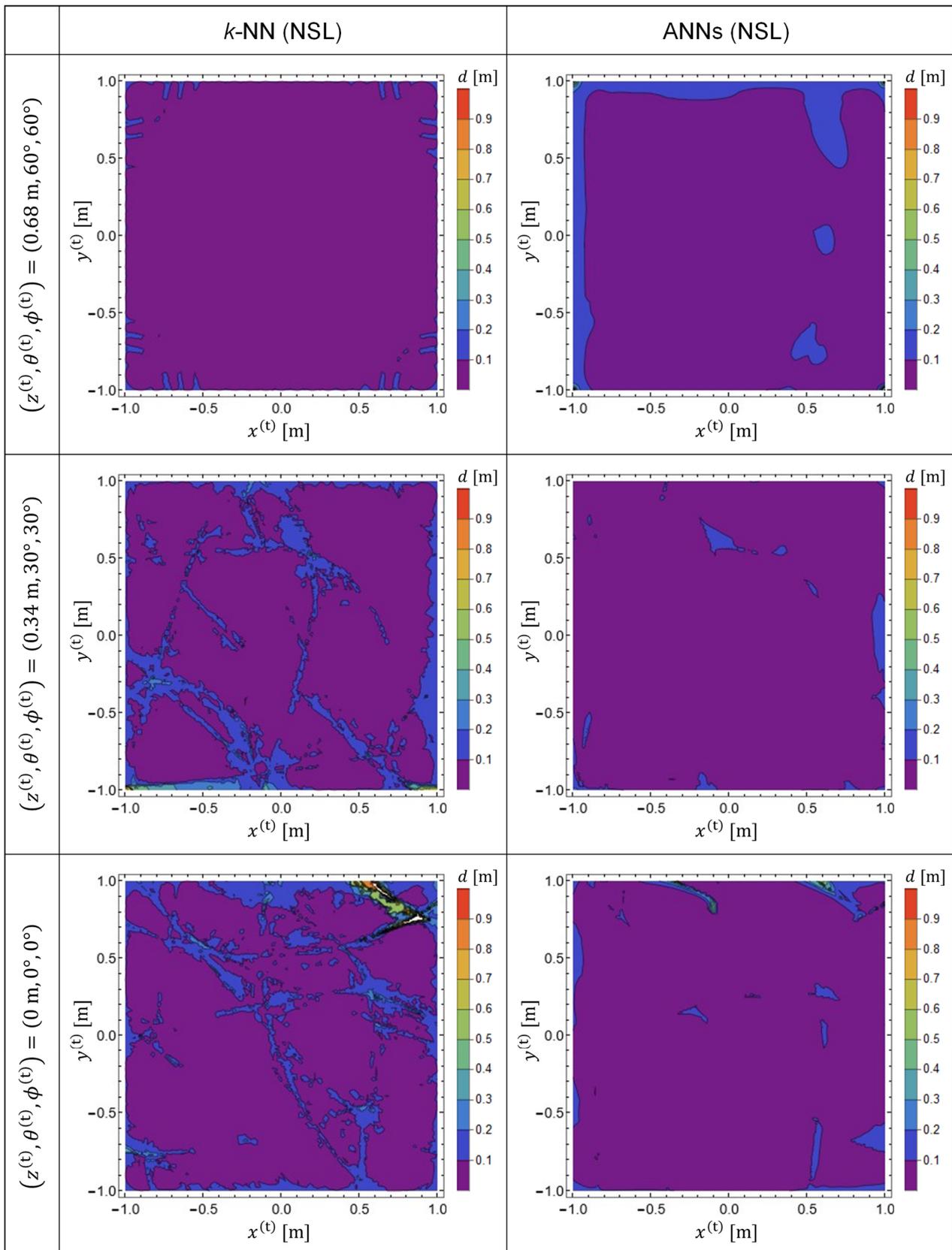
The results are presented in Figure 4. The predictor functions used for plotting Figure 4 are  $\mathbf{P}$ , which were generated from the training data in a linear representation. Note that the TX-state parameters in Equation (21) were not used for the calculation of the training data. Therefore, the results plotted in Figure 4 exhibit the generalization performance of the predictor functions generated by machine learning. The left and right columns indicate the EDF values obtained using  $k$ -NN and ANNs, respectively. It was observed that, with  $k$ -NN, the prediction accuracy is reduced in the vicinity of the four corners of the target space [18,19]. Meanwhile, it was confirmed that the prediction accuracy was significantly improved by adopting ANNs.

Figure 5 shows the EDF patterns obtained with the predictor functions  $\mathbf{Q}$ , which were generated from training data in the NSL representation. By comparing Figures 4 and 5, it is evident that the prediction accuracy is improved for both ANNs and for  $k$ -NN by using training data in NSL representation. However, it was observed that ANNs showed better performance for both representations of the training data.

Although it has been demonstrated in Figures 4 and 5 that ANNs and NSL representation show better performances, quantitative assessments of the predictor functions have not yet been carried out. Hence, we quantitatively evaluated the performances of the predictor functions via a statistical approach. To do this, we calculated the EDF values at TX positions indicated by crosses in Figure 3 and analyzed the statistical distribution of the values for predictor functions generated from different combinations of the learning algorithms and training-data representations. As shown in Figure 3, the distance between the two adjacent crosses was 0.17 m ( $\Delta x = \Delta y = \Delta z = 0.17$  m). The EDF values were calculated for various angles ( $\Delta\theta^{(t)} = \Delta\varphi^{(t)} = 30^\circ$ ) for each cross point. As a result, the number of TX states used for the quantitative evaluation of the predictor functions reached 82,522. Since the TX states used to generate the training samples were not included in the 82,522 states, the statistical distribution of the EDF values reflects generalization performances of the predictor functions.

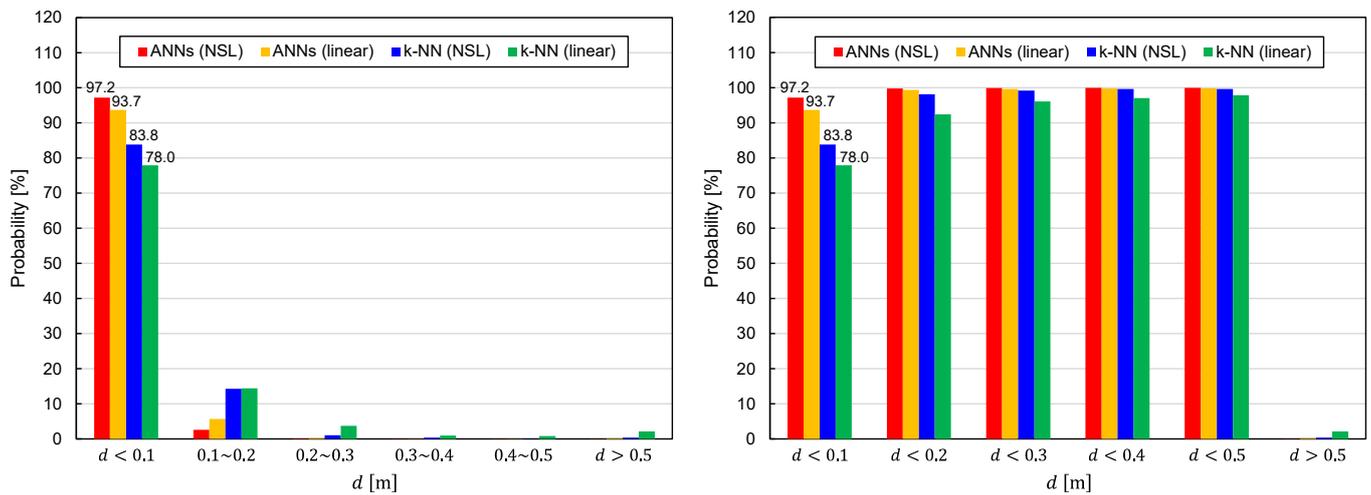


**Figure 4.** EDF patterns plotted in several cross sections parallel to the  $x$ - $y$  plane. The left and right columns demonstrate results obtained with predictor functions generated by  $k$ -NN and ANNs, respectively. Both predictor functions ( $\mathbf{P}$ ) were generated from training data in linear representation.

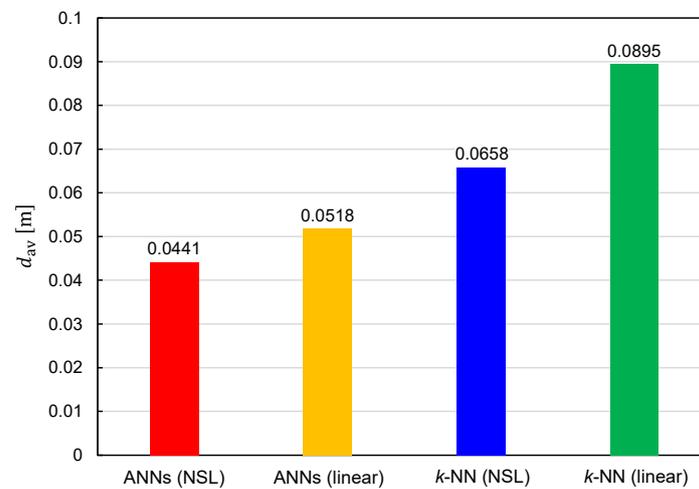


**Figure 5.** EDF patterns plotted in several cross sections parallel to the  $x$ - $y$  plane. The left and right columns demonstrate results obtained with predictor functions generated by  $k$ -NN and ANNs, respectively. Both predictor functions (Q) were generated from training data in NSL representation.

The statistical distributions of the EDF values are shown in Figure 6. It can be seen that the predictor functions generated by ANNs exhibit better performance for both representations of the training data (NSL and linear). With regard to the training-data representations, the NSL representation is superior to the linear representation, regardless of the learning algorithms. Figure 7 illustrates the EDF values averaged over 82,522 TX states. The average EDF value is denoted by  $d_{av}$ . The benefits of using ANNs and NSL representation are again confirmed by Figure 7.

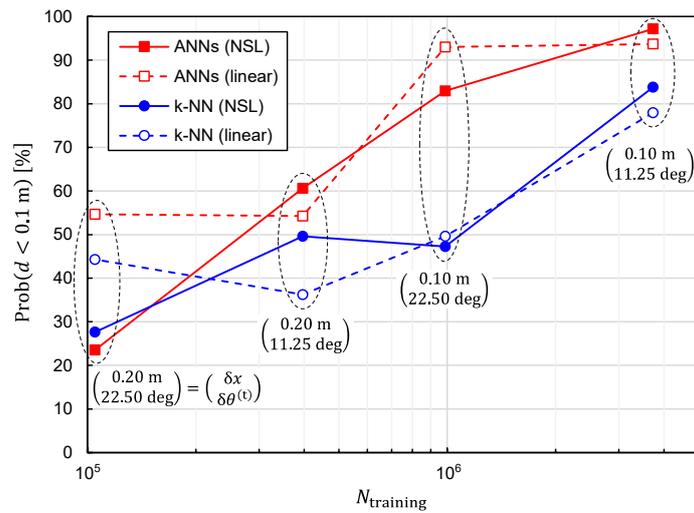


**Figure 6.** Statistical distributions of EDF values obtained using four different combinations of the learning algorithms (ANNs/ $k$ -NN) and training-data representations (NSL/linear).

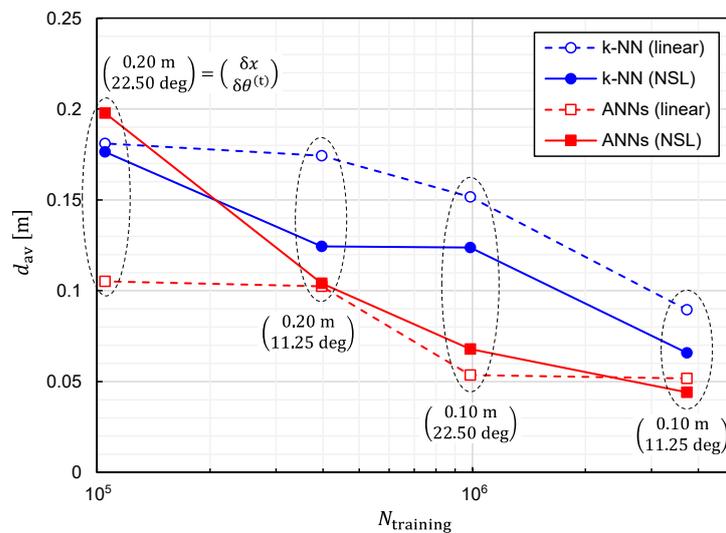


**Figure 7.** Average EDF values obtained using four different combinations of the learning algorithms (ANNs/ $k$ -NN) and training-data representations (NSL/linear).

It is also critical to investigate the relationship between the prediction accuracy and the number of training samples,  $N_{\text{training}}$ . We plotted  $\text{Prob}(d < 0.1 \text{ m})$  and  $d_{av}$  as functions of  $N_{\text{training}}$  in Figures 8 and 9, respectively, to investigate the relationship. Since reducing  $N_{\text{training}}$  is equivalent to reducing  $\delta x (= \delta y = \delta z)$  and  $\delta \theta^{(t)} (= \delta \varphi^{(t)})$ , the simulations have been executed for different combinations of  $(\delta x, \delta \theta^{(t)})$ . It can be observed from these figures that the prediction accuracy was improved by increasing  $N_{\text{training}}$ . Moreover, it is expected that the prediction accuracy will be further improved by increasing  $N_{\text{training}}$  except for the combinatorial use of ANNs and linear representation. In other words, the situation of overfitting has not yet occurred for  $N_{\text{training}} = 3,306,038$ .



**Figure 8.** Relationship between  $\text{Prob}(d < 0.1 \text{ m})$  and the number of training samples plotted for predictor functions generated from four different combinations of the learning algorithms (ANNs/ $k$ -NN) and training-data representations (NSL/linear).



**Figure 9.** Relationship between average EDF values and the number of training samples plotted for predictor functions generated from four different combinations of the learning algorithms (ANNs/ $k$ -NN) and training-data representations (NSL/linear).

In real-world applications, signals detected by sensors are affected by noise and irregularities of fabricated coils. Therefore, it is important to estimate the influences of these factors on prediction accuracy. Here we first discuss the influences of noise.

Since the amplitude of received signals depends on TX-state parameters, it is valid to define the reference magnetic-field amplitude for quantitatively discussing the influences of noise. Thus, we introduce the reference magnetic-field amplitude defined by

$$B_{\text{ref}} \triangleq \left| B_z^{(k)}(0, 0, 0, 0) \right|. \quad (22)$$

It is understood that  $B_{\text{ref}}$  means the magnetic-field amplitude generated at the sensor positions by the TX directed toward  $z$ -axis at a coordinate origin. Note that  $B_{\text{ref}}$  is independent

of  $k$  when the sensors are located at the corners of a cubic space as shown in Figure 2. We also define the signal-to-noise ratio (SNR) of sensors as

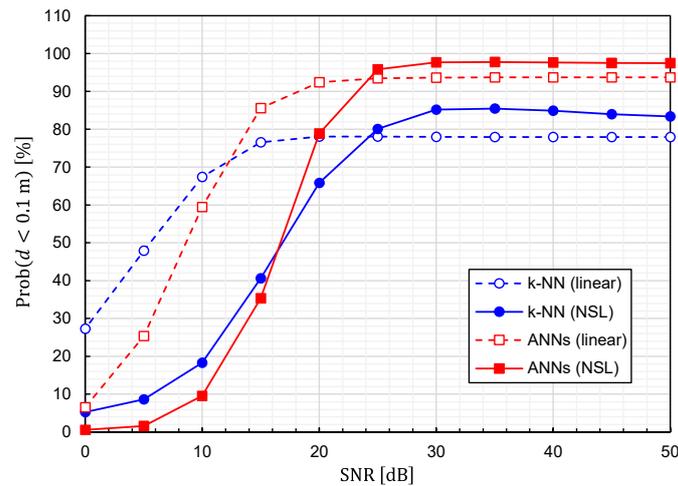
$$\text{SNR [dB]} \triangleq 20 \log_{10} \left( \frac{B_{\text{ref}}}{B_{\text{ein}}} \right), \quad (23)$$

where  $B_{\text{ein}}$  denotes amplitude of equivalent-input noise associated with each coil of the sensor. By considering the noise, the signal amplitude detected by the sensor can be written as

$$B_i^{(k),\text{real}} = B_i^{(k)} \sqrt{1 + \left( \frac{B_{\text{ein}}}{B_i^{(k)}} \right)^2} = B_i^{(k)} \sqrt{1 + 10^{-\frac{\text{SNR [dB]}}{10}} \cdot \left( \frac{B_{\text{ref}}}{B_i^{(k)}} \right)^2}, \quad (24)$$

where  $B_i^{(k),\text{real}}$  and  $B_i^{(k)}$  denote the amplitudes of detected signals with and without noise, respectively.

Using Equation (24), we calculated the prediction accuracy as functions of the SNR. The results for  $\text{Prob}(d < 0.1 \text{ m})$  and  $d_{\text{av}}$  are plotted in Figures 10 and 11, respectively. It is observed that noise immunity depends on training-data representations. For NSL representation, the prediction accuracy is almost constant for  $\text{SNR} > 25 \text{ dB}$ . On the other hand, for linear representation, it is kept constant for  $\text{SNR} > 15 \text{ dB}$ . It is interesting to see that linear representation is superior to NSL representation in terms of noise immunity.

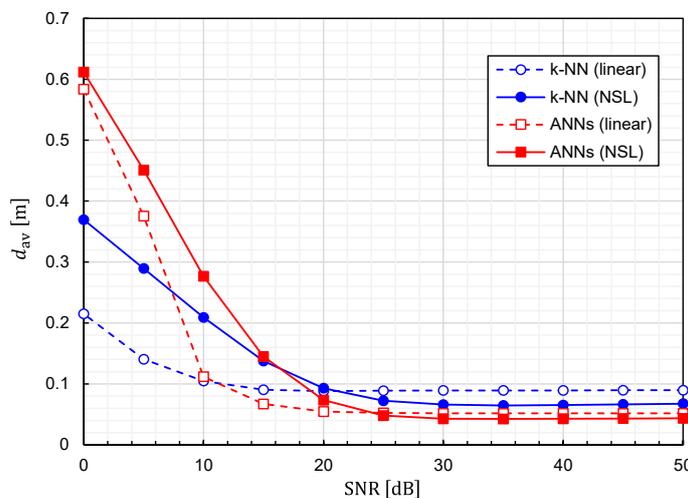


**Figure 10.** Relationship between  $\text{Prob}(d < 0.1 \text{ m})$  and SNR for predictor functions generated from four different combinations of the learning algorithms (ANNs/ $k$ -NN) and training-data representations (NSL/linear).

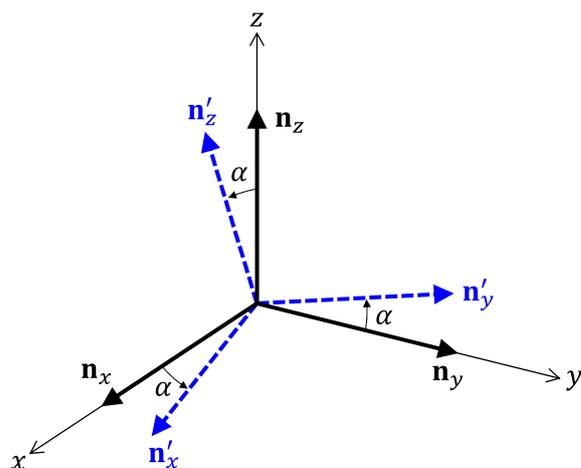
Next, we consider the influences of misalignment of receiving coils. In this study, we suppose that each sensor is equipped with the three coils. Let us denote unit vectors perpendicular to the three coils by  $\mathbf{n}_x$ ,  $\mathbf{n}_y$ , and  $\mathbf{n}_z$ . It is obvious that  $\mathbf{n}_x$ ,  $\mathbf{n}_y$ , and  $\mathbf{n}_z$  must be parallel to  $x$ ,  $y$ , and  $z$ -axes fixed to a target space, respectively. However, it is difficult to align the coils so that  $\mathbf{n}_x$ ,  $\mathbf{n}_y$ , and  $\mathbf{n}_z$  become completely parallel to the coordinate axes. As shown in Figure 12, we consider the situation of the misalignment, where the wrongly directing unit vectors are denoted by  $\mathbf{n}'_x$ ,  $\mathbf{n}'_y$ , and  $\mathbf{n}'_z$ . For simplicity, let the error angle  $\alpha$  be common for all unit vectors. Moreover, we assume that  $\mathbf{n}'_x$ ,  $\mathbf{n}'_y$ , and  $\mathbf{n}'_z$  are parallel to  $x$ - $y$ ,  $y$ - $z$ , and  $z$ - $x$  planes, respectively. In this situation, the influences of the misalignment can be written as

$$\begin{pmatrix} B_x^{(k),\text{ma}} \\ B_y^{(k),\text{ma}} \\ B_z^{(k),\text{ma}} \end{pmatrix} = \begin{pmatrix} \cos \alpha & \sin \alpha & 0 \\ 0 & \cos \alpha & \sin \alpha \\ \sin \alpha & 0 & \cos \alpha \end{pmatrix} \begin{pmatrix} B_x^{(k)} \\ B_y^{(k)} \\ B_z^{(k)} \end{pmatrix}, \quad (25)$$

where  $B_i^{(k),ma}$  and  $B_i^{(k)}$  denote amplitudes of the detected signals with and without the misalignment, respectively.

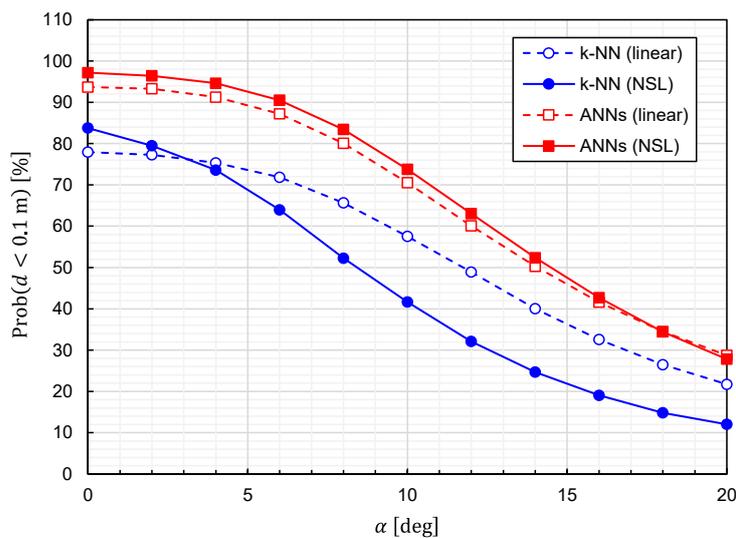


**Figure 11.** Relationship between average EDF values and SNR for predictor functions generated from four different combinations of the learning algorithms (ANNs/*k*-NN) and training-data representations (NSL/linear).

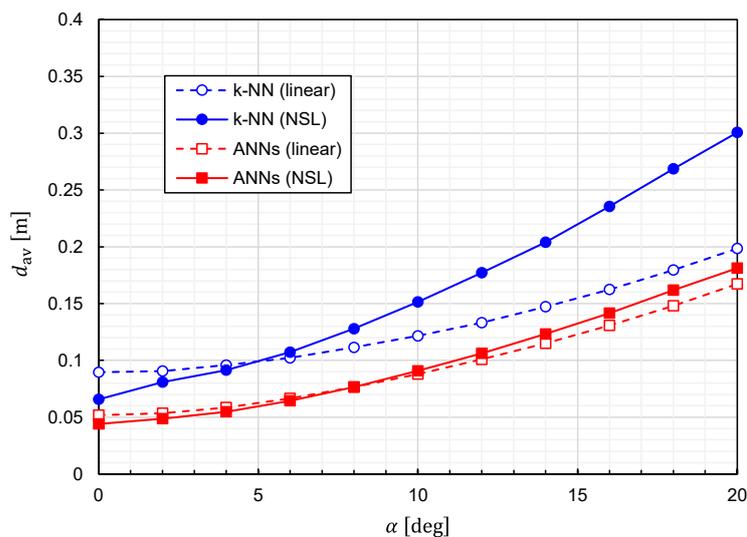


**Figure 12.** Relationship between coordinate axes and unit vectors perpendicular to receiving coils. Ideal vectors are denoted by  $\mathbf{n}_x$ ,  $\mathbf{n}_y$  and  $\mathbf{n}_z$ . On the other hand,  $\mathbf{n}'_x$ ,  $\mathbf{n}'_y$ , and  $\mathbf{n}'_z$  represent the unit vectors that include angle errors.

Using Equation (25), we calculated the prediction accuracy as functions of  $\alpha$ . The results for  $\text{Prob}(d < 0.1 \text{ m})$  and  $d_{av}$  are plotted in Figures 13 and 14, respectively. It is observed that the predictor function generated from the combination of *k*-NN and NSL is the most sensitive to  $\alpha$ . The other three predictor functions show similar characteristics against  $\alpha$ . We can say that the influences of the misalignment are limited for  $\alpha < 5 \text{ deg}$ .

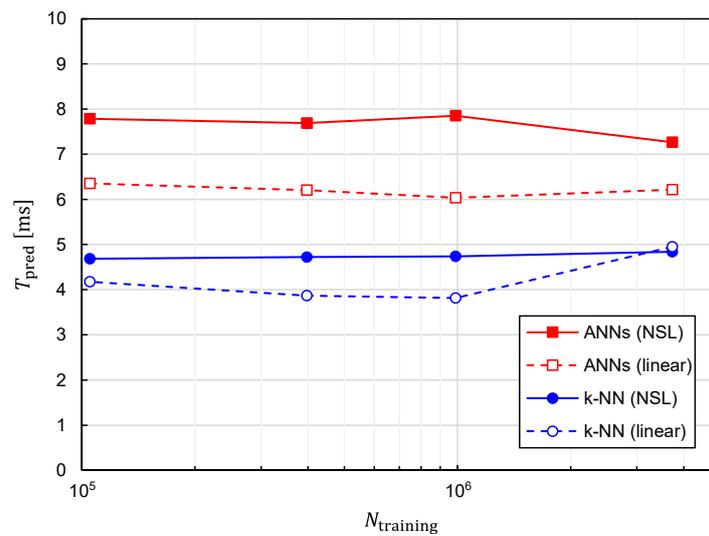


**Figure 13.** Relationship between  $\text{Prob}(d < 0.1 \text{ m})$  and error angle for predictor functions generated from four different combinations of the learning algorithms (ANNs/*k*-NN) and training-data representations (NSL/linear).



**Figure 14.** Relationship between average EDF values and error angle for predictor functions generated from four different combinations of the learning algorithms (ANNs/*k*-NN) and training-data representations (NSL/linear).

In addition to prediction accuracy, the computational speed is another critical aspect that needs to be evaluated. Since one of the advantages of machine learning to the conventional optimization method is the speed of predicting target positions, it is imperative to evaluate the time required for the prediction with machine learning, which we denote as  $T_{\text{pred}}$ . The relationship between  $T_{\text{pred}}$  and  $N_{\text{training}}$  is plotted in Figure 15. The following three features can be observed in Figure 15.

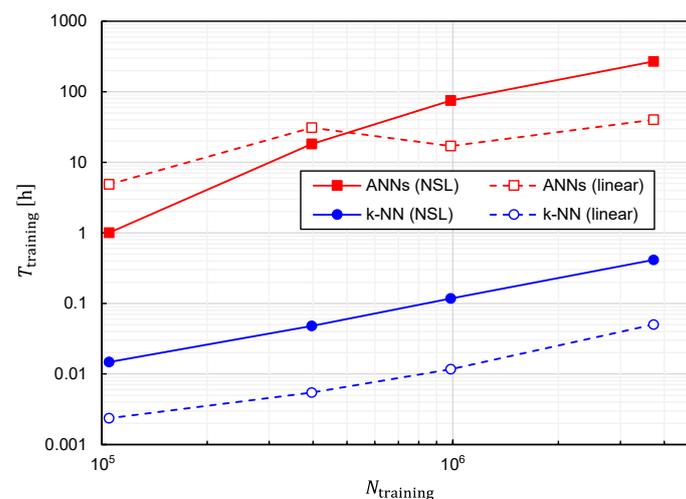


**Figure 15.** Relationship between the time required for predicting a TX position and the number of training samples plotted for predictor functions generated from four different combinations of the learning algorithms (ANNs/ $k$ -NN) and training-data representations (NSL/linear).

1.  $T_{\text{pred}}$  is almost independent of  $N_{\text{training}}$ ;
2.  $T_{\text{pred}}$  is increased by approximately 1.5 times with ANNs in comparison with  $k$ -NN;
3.  $T_{\text{pred}}$  is increased by approximately 1.2 times with training data of NSL representation in comparison with those of linear representation.

In terms of prediction accuracy, the best combination of the algorithm and training-data representation is that of the ANNs and NSL. Fortunately,  $T_{\text{pred}}$  considered with the best combination is kept less than 10 ms and is not drastically increased in comparison with the other combinations. Therefore, real-time tracking is possible with the best combination.

Regarding the computational speed, another critical index is the time required to generate a predictor function from training data, which we denote as  $T_{\text{training}}$ . Therefore, we also investigated the relationship between  $T_{\text{training}}$  and  $N_{\text{training}}$ , which is plotted in Figure 16. The following three features can be observed in Figure 16.



**Figure 16.** Relationship between the time required for training and the number of training samples plotted for predictor functions generated from four different combinations of the learning algorithms (ANNs/ $k$ -NN) and training-data representations (NSL/linear).

1.  $T_{\text{training}}$  is an almost monotonically increasing function of  $N_{\text{training}}$ ;

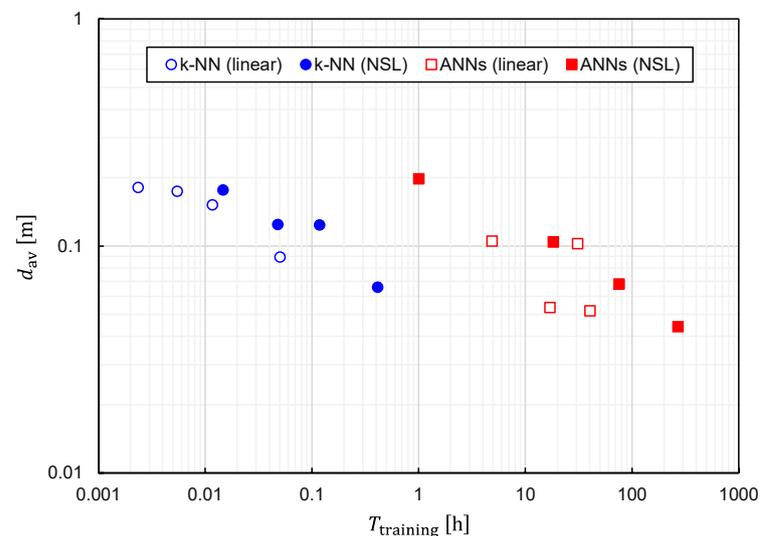
2. For large  $N_{\text{training}}$ ,  $T_{\text{training}}$  is increased by approximately 500 to 1000 times with ANNs in comparison with  $k$ -NN;
3. For large  $N_{\text{training}}$ ,  $T_{\text{training}}$  is increased by approximately 5 to 10 times with training data of NSL representation in comparison with those of linear representation.

From a practical viewpoint, the only drawback of the best combination is that  $T_{\text{training}}$  becomes considerably longer. In fact, it takes 269 h for  $N_{\text{training}} = 3,306,038$  with the computer used in this study (Intel Xeon W-2223 and 64-GB RAM). The considerably large  $T_{\text{training}}$  is primarily due to the adoption of ANNs. However, as shown in Figures 8 and 9, a better prediction accuracy is obtained with ANNs. Therefore, there is a trade-off relationship between  $T_{\text{training}}$  and prediction accuracy. The performances obtained thus far are summarized in Table 1.

**Table 1.** Summary of performances of predictor functions generated from four different combinations of the learning algorithms (ANNs/ $k$ -NN) and training-data representations (NSL/linear).

	ANN (NSL)	ANN (Linear)	$k$ -NN (NSL)	$k$ -NN (Linear)
Prob( $d < 0.1$ m) [%]	97.2	93.7	83.8	78.0
$d_{\text{av}}$ [mm]	44.1	51.8	65.8	89.5
$T_{\text{pred}}$ [ms]	7.27	6.21	4.84	4.95
$T_{\text{training}}$ [h]	269	40.3	0.414	0.0503

We plotted the relationship between  $d_{\text{av}}$  and  $T_{\text{training}}$  for several different values of  $N_{\text{training}}$  in Figure 17 to clearly establish the trade-off relationship. As expected, the trade-off relationship can be seen as a whole. However, it was also observed that  $k$ -NN and ANNs formed different clusters. The right cluster composed of squares suggests that ANNs are suitable for obtaining the best prediction accuracy, although  $T_{\text{training}}$  becomes considerably long. Meanwhile, the left cluster composed of circles implies that a fairly good prediction accuracy is obtainable with  $k$ -NN while keeping  $T_{\text{training}}$  fairly short.



**Figure 17.** Relationship between the average EDF values and time required for training plotted for several different numbers of training samples.

#### 4. Conclusions

Localization systems that utilize artificially generated magnetic fields are required to calculate target positions based on the information of magnetic fields detected by multiple sensors. This calculation is not easy since it involves solving nonlinear inverse problems. In this study, we demonstrated that machine learning is suitable for solving inverse problems.

Our emphasis was on comparing performances obtained with  $k$ -NN and ANNs that were adopted for machine learning. We numerically evaluated the accuracy of the target positions predicted by  $k$ -NN and ANNs by considering the  $2\text{ m} \times 2\text{ m} \times 2\text{ m}$  cubic space.  $\text{Prob}(d < 0.1\text{ m})$  and  $d_{\text{av}}$  obtained with  $k$ -NN were 84% and 66 mm, respectively, and those obtained with ANNs were 97% and 44 mm, respectively. It was demonstrated that ANNs are superior to  $k$ -NN in terms of prediction accuracy.

Furthermore, we also evaluated the computational times taken with the  $k$ -NN and ANNs. Accordingly,  $T_{\text{pred}}$  and  $T_{\text{training}}$  obtained with  $k$ -NN were 4.8 ms and 0.4 h and those obtained with ANNs were 6.2 ms and 269 h, respectively. It was demonstrated that  $T_{\text{pred}}$  obtained with both  $k$ -NN and ANNs was short enough to execute real-time tracking. However, it was revealed that ANNs require much longer  $T_{\text{training}}$  than  $k$ -NN. Therefore,  $k$ -NN remains a valid method for generating fairly good predictor functions within limited training times.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Conflicts of Interest:** The author declares no conflict of interest.

## References

1. Moore, S.K. Superaccurate GPS chips coming to smartphones in 2018. *IEEE Spectr.* **2018**, *54*, 10–11. [[CrossRef](#)]
2. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Niyato, D.; Dobre, O.; Poor, H.V. Internet of things: A comprehensive survey. *IEEE Internet Things J.* **2022**, *9*, 359–383. [[CrossRef](#)]
3. Mautz, R. Indoor Positioning Technologies. Habilitation Thesis, ETH-Zürich, Zürich, Switzerland, February 2012. [[CrossRef](#)]
4. Pasku, V.; Angelis, A.D.; Angelis, G.D.; Arumugam, D.D.; Dionigi, M.; Carbone, P.; Moschitta, A.; Ricketts, D.S. Magnetic field-based positioning system. *IEEE Commun. Surv. Tuts.* **2017**, *19*, 2003–2017. [[CrossRef](#)]
5. Davidson, P.; Piché, R. A survey of selected indoor positioning methods for smartphones. *IEEE Commun. Surv. Tuts.* **2017**, *19*, 1347–1370. [[CrossRef](#)]
6. Guo, X.; Ansari, N.; Hu, F.; Shao, Y.; Elikplim, N.R.; Li, L. A survey on fusion-based indoor positioning. *IEEE Commun. Surv. Tuts.* **2020**, *22*, 566–594. [[CrossRef](#)]
7. Nessa, A.; Adhikari, B.; Hussain, F.; Fernando, X.N. A survey of machine learning for indoor positioning. *IEEE Access* **2020**, *8*, 214945–214965. [[CrossRef](#)]
8. Ashraf, I.; Hur, S.; Park, Y. Enhancing performance of magnetic field based indoor localization using magnetic patterns from multiple smartphones. *Sensors* **2020**, *20*, 2704. [[CrossRef](#)] [[PubMed](#)]
9. Fernandes, L.; Santos, S.; Barandas, M.; Folgado, D.; Loenardo, R.; Santos, R.; Carreiro, A.; Gamboa, H. An infrastructure-free magnetic-based indoor positioning system with deep learning. *Sensors* **2020**, *20*, 6664. [[CrossRef](#)] [[PubMed](#)]
10. Bhattarai, B.; Yadav, R.K.; Gang, H.-S.; Pyun, J.-Y. Geomagnetic field based indoor landmark classification using deep learning. *IEEE Access* **2019**, *7*, 33943–33956. [[CrossRef](#)]
11. Chen, Y.; Zhou, M.; Zheng, Z. Learning sequence-based fingerprint for magnetic indoor positioning system. *IEEE Access* **2019**, *7*, 163231–163244. [[CrossRef](#)]
12. Sheinker, A.; Ginzburg, B.; Salomonski, N.; Frumkis, L.; Kaplan, B.-Z. Localization in 3-D using beacons of low frequency magnetic field. *IEEE Trans. Instrum. Meas.* **2013**, *62*, 3194–3201. [[CrossRef](#)]
13. Arumugam, D.D.; Griffin, J.D.; Stancil, D.D.; Ricketts, D.S. Three-dimensional position and orientation measurements using magneto-quasistatic fields and complex image theory. *IEEE Antennas Propag. Mag.* **2014**, *56*, 160–173. [[CrossRef](#)]
14. Pasku, V.; Angelis, A.D.; Angelis, G.D.; Moschitta, A.; Carbone, P. Magnetic field analysis for 3-D positioning applications. *IEEE Trans. Instrum. Meas.* **2017**, *66*, 935–943. [[CrossRef](#)]
15. Qi, X.; Chen, L.; An, K.; Wang, J.; Ma, C.; Zhang, B.; Xu, K.; Li, H.; Ye, D.; Huangfu, J.; et al. Wireless indoor positioning with vertically uniform alternating magnetic fields. *IEEE Trans. Instrum. Meas.* **2018**, *67*, 2733–2735. [[CrossRef](#)]
16. Sasaki, A.; Ouellette, O.; Beaudry-Marchand, M.; Hirata, A.; Morimura, H. Analysis and experimental study of magnetic-field amplification by a double coil. *IEEE Trans. Ind. Electron.* **2017**, *64*, 3216–3226. [[CrossRef](#)]
17. Sasaki, A. Double-solenoid ferrite coils for magnetic-field amplification in LF and MF bands. *IEEE Trans. Magn.* **2019**, *55*, 1–9. [[CrossRef](#)]
18. Sasaki, A.; Ohta, E. Magnetic-field-based position sensing using machine learning. *IEEE Sens. J.* **2020**, *20*, 7292–7302. [[CrossRef](#)]

19. Sasaki, A.; Fukushima, K. Machine-learning approach for solving inverse problems in magnetic-field-based positioning. *IEICE Trans. Fundam.* 2022; *in press*. [[CrossRef](#)]
20. Ohta, E.; Sasaki, A. Magnetic-field-based positioning of mobile devices using neural networks. In Proceedings of the 2020 IEICE Society Conference, Online, 15–18 September 2020; p. B15-15. (In Japanese)