World Electric Vehicle Journal

MDPI

Article

A CNN-Based System for Mobile Robot Navigation in Indoor Environments via Visual Localization with a Small Dataset

Farzin Foroughi, Zonghai Chen and Jikai Wang *

Department of Automation, University of Science and Technology of China, Hefei 230026, China; foroughi@mail.ustc.edu.cn (F.F.); chenzh@ustc.edu.cn (Z.C.)
* Correspondence: wangjk@ustc.edu.cn; Tel.: +86-551-63606104

Abstract: Deep learning has made great advances in the field of image processing, which allows automotive devices to be more widely used in humans' daily lives than ever before. Nowadays, the mobile robot navigation system is among the hottest topics that researchers are trying to develop by adopting deep learning methods. In this paper, we present a system that allows the mobile robot to localize and navigate autonomously in the accessible areas of an indoor environment. The proposed system exploits the Convolutional Neural Network (CNN) model's advantage to extract data feature maps for image classification and visual localization, which attempts to precisely determine the location region of the mobile robot focusing on the topological maps of the real environment. The system attempts to precisely determine the location region of the mobile robot by integrating the CNN model and topological map of the robot workspace. A dataset with small numbers of images is acquired from the MYNT EYE camera. Furthermore, we introduce a new loss function to tackle the bounded generalization capability of the CNN model in small datasets. The proposed loss function not only considers the probability of the input data when it is allocated to its true class but also considers the probability of allocating the input data to other classes rather than its actual class. We investigate the capability of the proposed system by evaluating the empirical studies based on provided datasets. The results illustrate that the proposed system outperforms other state-of-the-art techniques in terms of accuracy and generalization capability.

Keywords: mobile robot; mobile robot localization; convolutional neural network; cross-entropy loss function

check for updates

Citation: Foroughi, F.; Chen, Z.; Wang, J. A CNN-Based System for Mobile Robot Navigation in Indoor Environments via Visual Localization with a Small Dataset. *World Electr. Veh. J.* **2021**, *12*, 134. https://doi.org/10.3390/wevj12030134

Academic Editor: Joeri Van Mierlo

Received: 2 August 2021
Accepted: 22 August 2021
Published: 26 August 2021

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

Copyright: © 2021 by the authors. 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

Nowadays, the growing demand for servant robots has prompted researchers to improve technologies for automating these platforms. One of the most fundamental technologies for automating these platforms is their navigation system [1]. Autonomous mobile robot navigation is an active area of research and has gained the great attention of numerous researchers in the recent past [2,3]. Autonomous Mobile Robots (AMR) are designed and programmed to perform multiple tasks in substitution for humans. Humans are usually unable to perform such tasks due to multiple reasons such as bad environmental conditions or unhealthy and dangerous environments, or even repetitive tasks that can be boring. Due to these conditions and the need for robots in strenuous circumstances, the recent past has seen tremendous research in mobile robots. The intention is to have such devices, which could automatically perform most tasks without human intervention [4]. All the advantages, which can be achieved with AMR usage, also come along with numerous challenges. Localization and navigation are the main challenges in the self-controlling system of these types of robots.

Recent studies have made use of the Global Positioning System (GPS) to address the challenges of location and navigation. A number of studies have been built on the concept of GPS and are very successful in some applications [5]. Although the use of GPS proved to be successful in a number of applications and provides very high accuracy and

precision rates, its usage is mainly in outdoor environments. In indoor environments, the accuracy and precision of GPS fall down rapidly due to certain challenges such as obstacles, walls, and other objects [6]. Many solutions have also been proposed for the problem as mentioned above, such as the use of Ultrasound, Bluetooth, and WiFi in the indoor environment [7]. In ultrasound-based models for localization, the distance between the receiver, which is usually located at a fixed point, and the transmitter, which is a moving target itself or mounted on a moving target, is estimated by ultrasonic waves. This distance estimation is calculated by the time interval between sending the signal and receiving it between the sender and receiver [8]. Bluetooth is energy radiation whose frequency is below the sensitivity of the human eye. Most of the localization methods based on the Bluetooth reflected signal generate the Received Signal Strength Indicator (RSSI), which is designed to indicate how strong a signal is received [9]. Like ultrasound and Bluetooth, the localization models by using WiFi technology are also based on signal strength. The advantage of WiFi over the previous two devices is its greater availability in urban environments and signals with wider coverage area capability. Therefore, the models based on WiFi for localization purposes have better accuracy and cost less than models based on other signal devices [8]. However, all these solutions are based on signal strength, which can suffer from unpredictability and instability of signal propagation and atmospheric interference noises from surrounding objects and components [10,11].

Deep learning comprises a subfield of machine learning, which has proved to be very successful in a number of applications belonging to different domains such as robotics. This success and development in machine learning have led researchers to use deep learning approaches to investigate AMR localization in indoor environments. Most of these approaches are based on vision-based localization, which is usually implemented as image discovery that finds the most related image in the database to the query image by matching features [12,13]. Based on the deep learning approach, many methods have been successfully applied to solve the localization problem; however, they have their limitations due to the limited size of the private datasets. Deep learning-based models have millions of parameters, and training them with small datasets can lead to overfitting. However, despite the fact that this problem can be largely solved using transfer learning and using learned weights from already trained models, the model still has bounded accuracy due to inadequate generalization capabilities.

In this paper, to eliminate the challenges mentioned above, we proposed a deep learning-based approach in an end-to-end fashion for indoor mobile robot localization via image classification with a small dataset. By using the floor plan of the robot workspace, topological map information is derived. The navigable area of the topological map is then extracted, where each of them is classified as a robot workstation. In this manner, the mobile robot has the ability to localize and navigate itself throughout the topological map autonomously. In order to design a navigation and localization system, a novel dataset is extracted from a real apartment environment by using the MYNT EYE camera. The proposed CNN-based classifier model is trained on all of these images and then used for the mobile robot's localization and navigation. The system also uses the CNN-based controller model to detect possible obstacles in the robot's pathway. We proposed an appropriate cost function to tackle the bounded generalization capability of the CNN classifier model that exists due to the limited size of the provided dataset. Unlike most well-known loss functions, the proposed loss not only considers the probability of the input data when it is allocated to its true class but also considers the probability of allocating the input data to other classes rather than its actual class. The results demonstrated the efficiency, effectivity, and generalization capability of the proposed system.

In the following, Section 3 describes the problem statement, along with strategy and data acquisition. Section 4 explains the proposed strategy and details the method implementation. In Section 5, the results of the proposed models are discussed, along with comparisons with other models. Finally, In Sections 6 and 7, we concluded the discussion and outcomes of the current study, respectively.

2. Related Work

In recent years, many models have been proposed that use these layers to perform the classification task. One of the pioneer models is the AlexNet model [14], which proposed Rectified Linear Unit (ReLU) [15] to improve accuracy and performance. The advantage of ReLU is that it rapidly speeds up the convergence process during the training of the network. Later, the VGG model [16] introduced factorized convolutions, which led to improved CNN-based models. VGG uses smaller receptive windows such as kernels or filters in the convolutional layers, which reduces the network's size and enhances performance. However, the parameter size is large, and it can not prove well for embedded systems. Google researchers released a version of their architecture called MobileNet [17]. MobileNet was designed especially for the mobile or embedded architecture-based systems, which have higher hardware constraints. This model reduced the training time and improved the performance. In addition, MobileNet comprised 28 layers in total and achieved an accuracy of 87 percent on the ImageNet dataset.

Junior et al. [18] proposed an online IoT system for mobile robot localization based on machine learning methods. They first collected a dataset by GoPro, and the Omnidirectional camera performed the localization task based on the image classification. They used the advantage of convolutional neural networks (CNN) to extract the features from images and then categorized them by a machine learning classifier algorithm. Although their model achieved good accuracy, it is not performing in an end-to-end fashion. Foroughi et al. [19] introduced a robot localization model in hand-drawn maps with the help of CNN and the Monte Carlo Method (MCL). Their localization model is in two phases. In the first phase, they extracted data through the different locations of the given map and trained these data on the designed CNN network to predict the robot's position. In the second phase, the robot's actual position on the map was then evaluated more accurately using the MCL model. The performance of their proposed model is highly influenced by the quality of the hand-drawn map. Zhang et al. [20] proposed a model named Posenet++ for the robot re-localization problem. Their model is a developed version of the VGG16 network that regresses the camera's position using the input image. Although they used weight from pre-trained models, their model still suffers from the accuracy bounded due to the limited size of the dataset. Radwan et a1. [21] introduced VLocNet++ to estimate the camera's position by utilizing a multitask learning method based on the CNN network. Their model integrated the semantic and geometric information of the surrounded into the pose regression networks. Nevertheless, their model may fail in an environment with similar features due to the limited generalizability. Hou et al. [22] proposed a point cloud map based on the ORB-SLAM algorithm for robot localization and 3D dense mapping in an Indoor Environment. Although their model performed well for the task of 3D mapping, even in a large-scale scene, the effect of the proposed model for the process of robot positioning is poor. Lin et al. [23] introduced a robust loop closure approach to eliminate the cumulative error caused by the long-term localization of mobile robots. Firstly, they use an instance segmentation network to detect objects in the environment and further realize the construction of a semantic map. Then, the environment is represented as a semantic graph with topological information, and the place recognition is realized by efficient graph matching. Finally, the drift error is corrected by object alignment. Although the proposed method has high robustness, it is not suitable for scenes with few objects. Using CNN and Q-Learning, Saravanan et al. [24] make robots autonomously search and reach plants in an indoor environment to monitor the health of the plant. They build an IoT enabled autonomous mobile robot using CNN and reinforcement learning. Ran et al. [25] design a shallow CNN with higher scene classification accuracy and efficiency to process images captured by a monocular camera. In addition, the adaptive weighted control algorithm combined with regular control are used to improve the robot's motion performance. However, its capability of self-learning and continuous decision-making is still not satisfactory.

3. Problem Statement

In this paper, we focused on addressing the coarse localization of mobile robots in an indoor environment. To do this, we made a scenario to deal with the localization and navigation problem of the home robot by using a CNN-based model as place recognition to identify the candidate places where the robot is possibly located.

3.1. Hardware

To do this task, a Pioneer 3DX is equipped with a MYNT EYE camera. The robot, along with its equipment, is shown in Figure 1. Table 1 represents the specifications of the sensor that is used for the proposed approach. Pioneer 3DX is a small lightweight two-wheel two-motor differential drive robot ideal for indoor laboratory or classroom use. The robot comes complete with front SONAR, one battery, wheel encoders, a microcontroller with ARCOS firmware, and the Pioneer SDK advanced mobile robotics software development package. The MYNT EYE camera is equipped with leading hardware solutions such as IR active light, IMU si*x*-axis, hardware-level frame synchronization, global shutter, etc., up to 720 p/60 fps of synchronous image information.

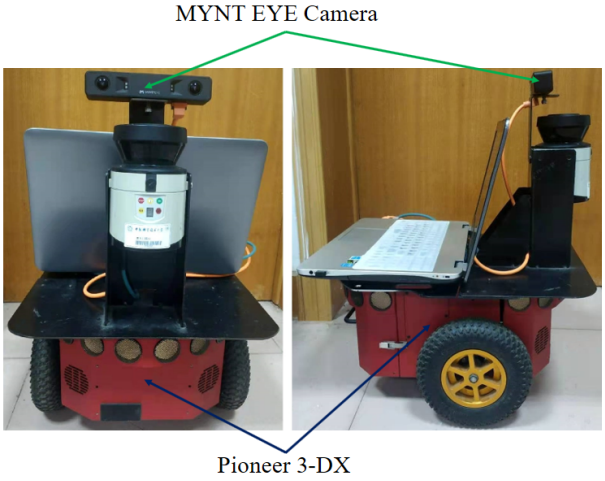


Figure 1. Pioneer 3DX equipped with MYNT EYE camera.

Table 1. Specifications of the MYNT EYE camera.

Sensor	Features
Model	MYNT EYE D-1000-120 FOV
Dimension	3D
View	D:121° H:105° V:58°
Range	0.3 10 m+

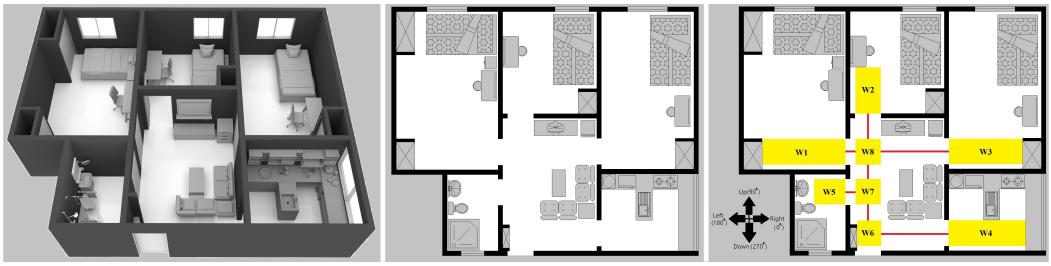
3.2. Environmental Space for Robot Navigation

An apartment is chosen as the environmental space for robot navigation. This type of environment is selected because it has characteristics that contribute to the recognition and navigation of the robot. Figure 2b presents the 2D floor plan of the apartment. The dark space (black and gray) represents obstacles or occupied areas, and the white space represents the free space. The robot cannot move in the dark space and can only take action in the free space. In addition, since the robot has the freedom to move all over the free space of the working space, the application of the controlling theory will be complicated and even unnecessary. Therefore, we limited the robot's movements by allowing the robot to take actions only in the special areas of the apartment. Figure 2c shows the topological map of the apartment that some areas, such as yellow areas, are chosen as the robot workstation.

Each workstation is labeled into the four classes "Right, Up, Left, and Down (0°, 90°, 180°, and 270°)" on the topological map of the apartment. As mentioned, to make control theory easier, we limited the movement of the robot by following rules as follows:

- The robot must have access to all the selected workstations in the apartment, but it does not need to have access to all over the free space of the apartment.
- The robot is only allowed to move between the workstations (W1 to W8) in the straight direction of 0, 90, 180, and 270 degrees. Therefore, these areas must be chosen in a way that there should be no dynamic obstacle between them. As an example, if the robot is placed at any point on the W3, it can move by taking the 180-degree straight path to any point on the W8.
- The robot is allowed to turn to the specific direction only inside the workstations, not anywhere else out of these areas.

As shown in Figure 2c, eight areas are chosen as the robot workstation for this apartment. The robot has access to all the rooms of the apartment such as kitchen, bedrooms, living rooms but not to all the free spaces of the mentioned rooms. There is at least one workstation for each room, but the rules mentioned above might not be enough. For example, W6, W7, and W8 are assigned to the living room since having only one unique workstation in this unit will not fit the requirement, as mentioned earlier. For example, if the robot is placed in the W7 area, it cannot go to the W4 by taking the straight path; therefore, W6 is born. This situation usually happens when a room has a border with several other rooms. As mentioned, each workstation is labeled into four classes; therefore, there are 32 classes for the apartment used in this study, including W1-Right, W1-Up, to W8-Down.



(**a**) (**b**) (**c**)

Figure 2. The apartment that is chosen for this study. (**a**) Perspective view, (**b**) 2D Floor Plan, (**c**) Topological map.

Furthermore, a set of paths is designed as the robot pathway to move between workstations. Some commands for the robot navigation are present in Table 2.

Table 2. Robot navigation command line. Tθ means the robot should turn to the θ-degree ($\theta = 0, 90, 180, 270$). Go means robot should go straight. SWx means the robot must stop at workstation x= $\{1, \ldots, 8\}$.

Route	Origin	Destination	Command
1	W1	W8	T0–Go–SW8
2	W3	W5	T180–Go–SW8–T270–Go–SW5
3	W1	W4	T0–Go–SW8–T270–Go–Go–SW6–T0–Go–SW4
4	W6	W2	T90–Go–Go–Go–SW2
5	W5	W1	T0–Go–SW7–T90–Go–SW8–T180–Go–SW1

3.3. Data Acquisition

By utilizing the MYNT EYE camera, a dataset is created. Using MYNT EYE, we captured RGB images with an angle of 105° height and 58°. Images were captured in positions equivalent to the classes W1-Right to W8-Down. The dataset consisted of 534 Images,

with a different number of images per class (due to the different size of each area) with a resolution of 798 × 448 pixels. Figure 3 and Table 3 present the sample images of the dataset and the number of images for each class, respectively.

Figure 3. Sample images of the dataset.

Table 3. Number of images for each class of dataset.

Workstation	Number of Images			
W1	50	15	15	15
W2	12	12	12	24
W3	12	12	40	16
W4	25	19	32	16
W5	12	9	9	9
W6	19	20	9	9
W7	12	20	17	12
W8	12	14	12	12

4. Methodology

In this section, the proposed approach for mobile robots' localization and navigation is expressed in detail. The proposed system consists of computer vision processing, which incorporates a topological mapping method and CNN models.

In the designed system, the user submits a request to send the robot to the desired destination. The system first creates a route from the robot's current position to the destination. The route is then broken into the commands and runs line by line individually. To execute each command, an image request is first sent to the sensor (camera), which is installed on the robot. The sensor takes an image of the environment in which it is located and sends it to the CNN unit. There are two CNN-based models in the CNN unit. The first model is responsible for estimating the position of the image received from the sensor. In fact, the model that is trained on the dataset beforehand assigns the appropriate class to the image. At this stage, if the location predicted by the first model is the destination desired by the user, the system will send a message to the user stating that the mission has been completed and will wait for the next command. Otherwise, the second model is executed, and, if there is a doorway in the same image, the model detects whether it is closed or open. Closed door means that the robot does not have access to the next destination, in which case an appropriate message will be delivered to the user. However, if there are no closed doors and no obstacles in the pathway, the command is eventually executed, and the robot takes action, which can include turning at a certain angle or moving forward. The flowchart of the proposed approach is presented in Figure 4.

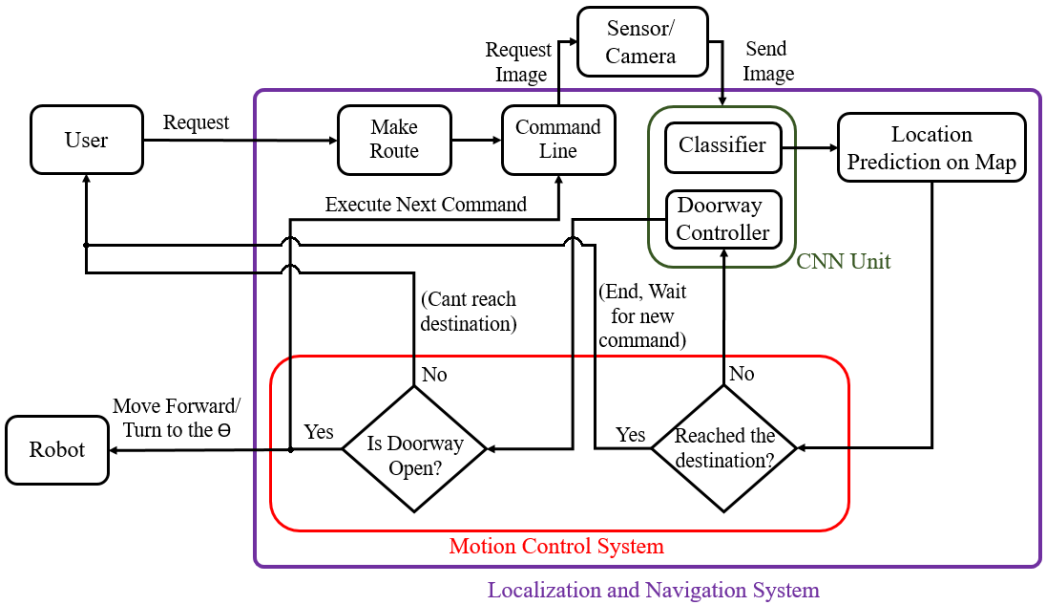


Figure 4. The proposed approach's flowchart.

4.1. CNN Models' Architecture

4.1.1. Classifier Model

The prepared dataset for this study has 32 classes and 534 images. Dealing with such a small dataset with this number of classes for classification requires specific consideration. Deep learning-based models require large amounts of data to show their effectiveness; therefore, the majority of well-known classification models are designed to perform on the dataset with a large amount of data, such as ImageNet [26], which contains millions of images and thousands of classes. Despite the advances made in deep learning in recent years, there are not many standard methods to design a network to fit private and small datasets. Therefore, there are only a few basic points to consider when designing a network to fit the small datasets.

Overfitting is one of the common problems that can happen when the model suffers from a lack of data. Overfitting occurs when a network performs extremely well on the train data but performs poorly on the unseen data. To prevent overfitting on the small datasets, not training the data on the complex model must be considered. The very deep convolutional networks with a huge number of parameters may work well on large datasets, but they are not flexible when exposed to small datasets. Lack of generalization is another major problem, which can happen due to the lack of data. The data augmentation technique is one of the most effective techniques to address this problem. This technique can increase the amount of the data by applying different transformations to the training set, such as flip, rotate, crop, and many others.

Considering the above explanations, we proposed a simple CNN-based model to classify the prepared dataset into one of 32 categories. The classifier model is trained on the prepared datasets for the classification task using the architecture presented in Figure 5. The input of the proposed classifier model is an RGB map image with a fixed size of 256 × 256, and it is consists of four convolution layers, four max-pooling layers, and two fully-connected layers. The convolution layers are equipped with the ReLU activation function. The number of output filters in the first, second, third, and fourth convolution layer is 8, 16, 32, and 64, respectively, and two first convolution layers have kernel sizes of 5 × 5 with stride 5 × 5, and the others have sizes of 3 × 3 with stride 3 × 3. After each convolution layer, there is a max-pooling layer with a pool size of 2 × 2 and stride 2. After that, there are two fully-connected layers, where the first one has 512 channels and the second one has 128 channels and is equipped with the ReLU activation function. Finally, the network ends with the softmax [27] layer. The configuration of the proposed classifier model is outlined in Table 4.

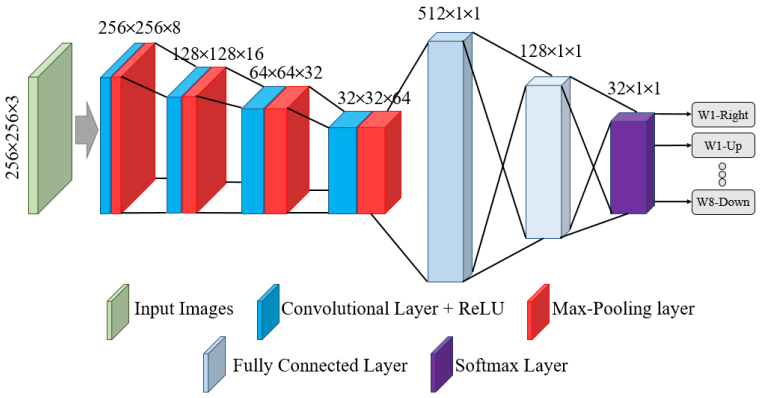


Figure 5. Architecture of the proposed classifier model.

Table 4. Configuration of the proposed classifier model.

Layer	Output Shape	Kernel Size–Stride
Input	256 × 256 × 3	-
Conv2D	256 × 256 × 8	5 × 5–3 × 3
MaxPooling2D	128 × 128 × 8	2 × 2–2
Conv2D	128 × 128 × 16	5 × 5–5 × 5
MaxPooling2D	64 × 64 × 16	2 × 2–2
Conv2D	64 × 64 × 32	3 × 3–3 × 3
MaxPooling2D	32 × 32 × 32	2 × 2–2
Conv2D	32 × 32 × 64	3 × 3–3 × 3
FC1	512	-
FC2	128	-
Softmax	32	-

4.1.2. Controller Model

The proposed controller model is a CNN-based architecture, which detects the presence of possible obstacles in the pathway of the robot by analyzing the image taken from the sensor. These potential obstacles include closed or semi-open doors that prevent the robot from reaching the desired location. The robot can only move from one workstation to another if there is no door between them or be fully open. All the images in the prepared dataset have one of these four modes, including without door, with the fully-open door, with the semi-open door, with the fully closed door (Figure 6). We have not created a new dataset to detect the door's situation in the image and only adjust the prepared datasets according to the needs of the controller model. Therefore, the prepared dataset is classified into two groups as free space and occupied. The free space group contains images with no or fully-open doors, while the occupied group contains images with semi-closed or fully-closed doors. The adjusted dataset for training on the doorway controller model has 534 images, of which 282 images belong to the free space class, and the rest belongs to the occupied class. Considering the above explanations, we proposed a CNN-based network as the controller model, classifying the prepared dataset into one of two categories. The architecture of the proposed controller model is exactly the same as the proposed classifier model (Figure 5), except that the softmax layer has two outputs.

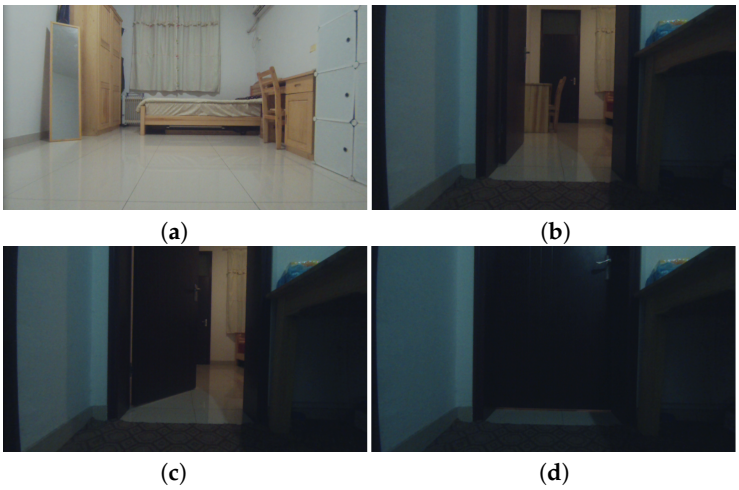


Figure 6. Sample images of the dataset for controller model, (**a**) without the door; (**b**) with fully-open door; (**c**) with the semi-open door; (**d**) with the fully-closed door.

4.2. Data Normalization

In deep learning, the quality of a model is largely affected primarily by training data [28], but there are some techniques to optimize the performance of the model. One of the effective and very popular techniques is normalization. In deep learning-based models, any changes in earlier layers distribution will be enhanced in the next layers; therefore, each layer tries to adjust to the new distribution. This problem, called Internal Covariate Shift, reduces the speed of the training process and causes the network to take more time to reach the global minimum. Normalization techniques are the standard method to overcome this problem. Besides reducing the Internal Covariate Shift, normalization standardizes the inputs, smooths the objective function, stabilizes the learning process, and decreases the training time. Data normalization is performed to improve the training process of the deep neural network and is usually performed as a preprocessing step before the actual execution of the training process. In data normalization, the value of each feature is mapped to the [0,1] range, where 1 represents the maximum value and 0 represents the minimum value for each feature. Most of the systems have features, which have values, where the difference between the maximum and minimum values has a very large scope. Therefore, it is necessary to apply the logarithmic scaling method for scaling to obtain the values in suitable ranges. A logarithmic scale displays numerical data over a very wide range of values in a compact way.

Batch normalization [29] and layer normalization [30] are the two most well-known normalization techniques. In order to investigate the effectiveness of the mentioned methods, we modified and trained both proposed models once by adding a batch normalization layer after each convolution and once by adding layer normalization after each convolution layer. The results are illustrated in Figure 7.

As it can be seen in Figure 7, batch normalization did not improve the performance of the proposed classifier and controller models, which can be because of the size of the dataset. Batch normalization depends on the size of the mini-batch [31] so that the small mini-batch size makes the training process noisy. As the size of the prepared dataset is small, the size of the mini-batch is therefore small as well, and, as a result, batch normalization could not play an effective role. Figure 7 also shows that layer normalization effectively increased the accuracy of both training and validation sets. In addition, the proposed models with layer normalization decreased the noise of the training and validation curves in comparison to the other two models, which makes it more reliable.

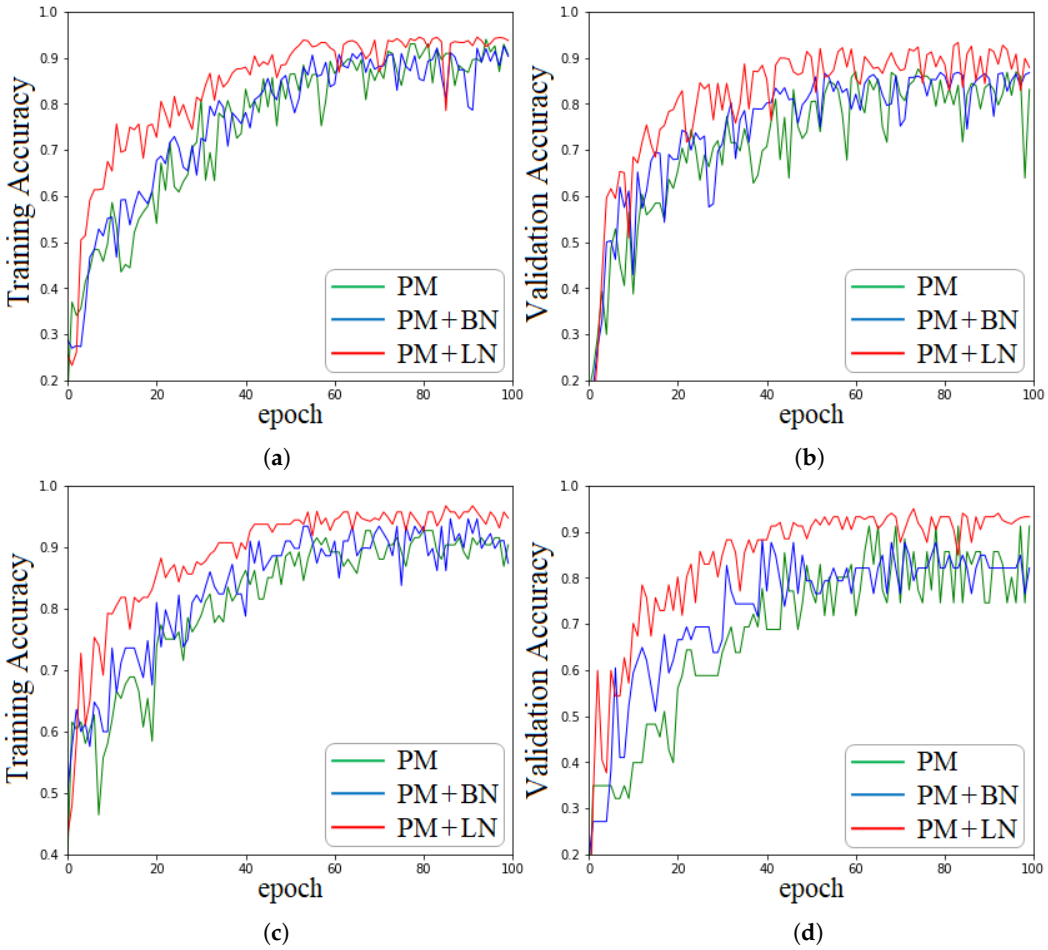


Figure 7. Comparison between results of the Proposed Model (PM), Proposed Model + Batch Normalization (PM+BN), and Proposed Model + Layer Normalization (PM+LN). (**a**) training curves of classifier model; (**b**) validation curves of classifier model; (**c**) training curves of controller model; (**d**) vvalidation curves of the controller model.

4.3. Cost Function

The goal of the neural network during the training process is to reduce the loss function. This means reducing the difference between computed and targeted values. Cross entropy (CE) loss is one of the most well-known and popular loss functions for classification. In a $C-class$ classification task, the training dataset with a batch size of m can be defined as $D = \{(x_i, y_i)|_{i=1}^{m}\}$, where x_i is a sample of the input data ($x_i \in \mathbb{R}^{h \times w \times d}$), and $y_i = \{1, \ldots, C\}$ is the true label of x_i. The network's probability when label x_i is $j = \{1, \ldots, C\}$ can be defined as Equation (1):

$$p_{i,j} = \frac{exp(z_{i,j})}{\sum_{f=1}^{C} exp(z_{i,f})} \tag{1}$$

where $z_{i,f}$ is the network's prediction. The formal definition of the CE loss for $C-class$ can be computed as follows:

$$L_{CE} = -\frac{1}{m \times C} \sum_{i=1}^{m} \sum_{j=1}^{C} 1(y_i = j) \log p_{i,j} \tag{2}$$

When x_i is labeled as y_i ($y_i = j$), then $1(.) = 1$; otherwise, for all $j \neq y_i$, $1(.) = 0$. Based on Equation (2), the CE loss only considers the probability of the input data when it is allocated to its true class. As mentioned earlier, the number of images of the prepared

dataset is quite small, and each class contains a few images. Therefore, in order to increase the performance of the proposed network, we modified the CE loss so that the probability of allocating the input data to other classes rather than its actual class is also considered. Equation (3) represents the loss when the data are allocated to other classes instead of its true class:

$$L_{OC} = \frac{\alpha}{m \times C} \sum_{i=1}^{m} \sum_{j=1}^{C} 1(y_i \neq j) \log p_{i,j} \tag{3}$$

When x_i is not labeled as y_i $(y_i \neq j)$, then $1(.) \neq 0$; otherwise, for all $j = y_i$, the $1(.) = 0$. In addition, α is a variable for controlling the scale of the loss when the data are allocated to other classes instead of its actual class $(\alpha \geq 0)$. By integrating Equations (2) and (3), the proposed loss function (L_{task}) can be obtained, which not only considers the probability when the data are labeled truly but also consider the probability when the data are labeled wrongly:

$$L_{task} = L_{CE} + L_{OC} \tag{4}$$

Besides increasing the probability when the data are labeled truly, at the same time, the proposed loss decreases the probability when the data are allocated to other classes instead of its true class.

4.4. Weight Loss

As presented in Table 3, the prepared dataset for the classifier model involves multiple labels for various locations of the apartment (the same also holds true for the dataset prepared for the controller model). Since the number of images varies for different classes, and, in some classes, the imbalances are higher, which is why the dataset needs to be balanced within each class. As a result, we designed the weight loss plan in order to correct the data imbalances. The weight (ω_C) is defined as follows:

$$\omega_C = \frac{n_{max} - n_c}{\beta \times \sum_{j=1}^{c} n_j} \tag{5}$$

where n_{max} is the number of images of the biggest class in the dataset, n_c represents the current classes' image number, $\sum_{j=1}^{C} n_j$ indicates the total number of images of the dataset, and β is a variable for controlling the scale of the weight loss. Using the presented weight, the classes with smaller data sizes attain the weight with a larger value, and the classes with bigger data sizes attain the weight with a smaller value. By applying Equation (5) into Equation (4), the weighted loss function is defined as follows:

$$L_{task} = \omega_C L_{CE} + \omega_C L_{OC} \tag{6}$$

To evaluate the effectiveness of the proposed loss, we trained both of the proposed classifier and controller models once by using CE loss once by adopting the proposed loss. The results are presented in Figure 8.

According to Figure 8, the proposed loss slightly improves the accuracy of both proposed classifier and controller models. In addition, compared to the CE, the proposed loss decreased the noise of the training and validation curves, made them smoother, and converged faster to maximize accuracy:

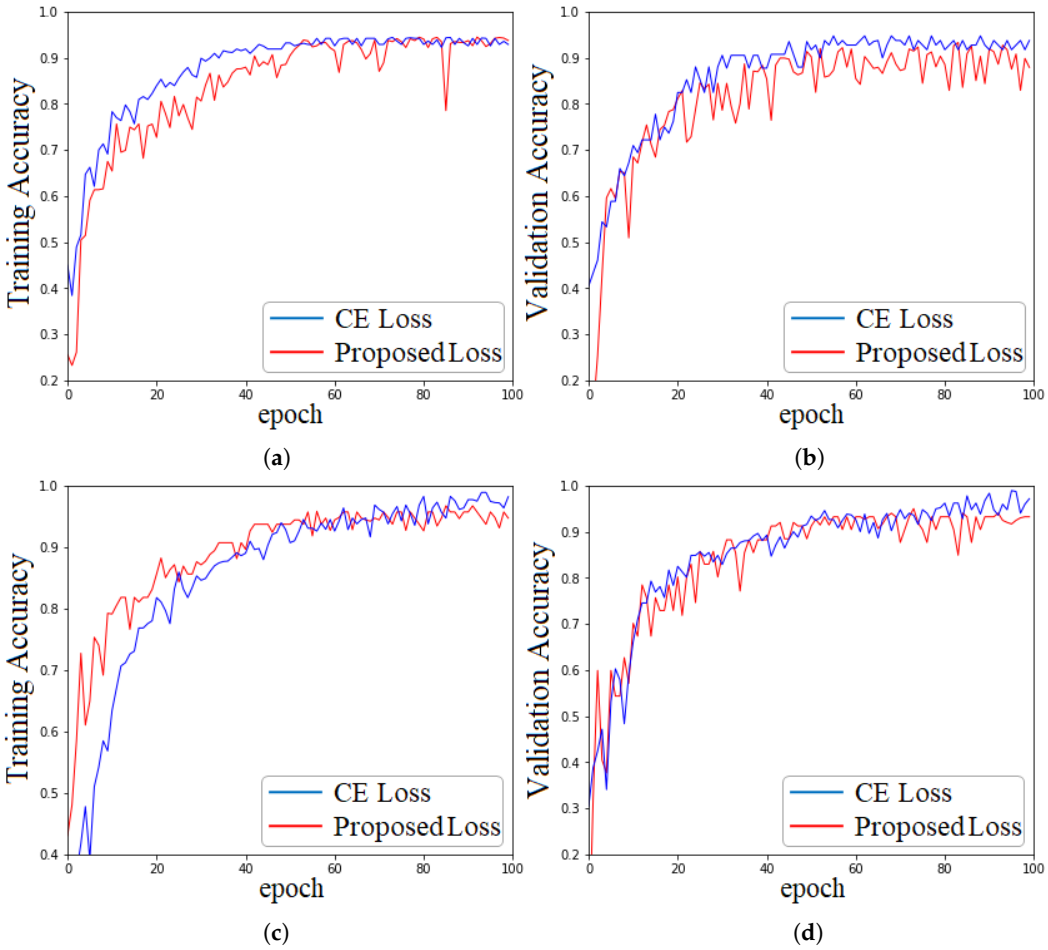


Figure 8. Comparison between results of the BC and the proposed loss. (**a**) training curves of classifier model; (**b**) validation curves of classifier model; (**c**) training curves of controller model; (**d**) validation curves of the controller model.

5. Empirical Results

In this section, the results of the proposed classifier and controller models, along with the proposed loss, are presented. First, evaluation parameters are described, and then the experimental setup, which is used for the execution and implementation of the proposed system, is detailed. Finally, the performance of the proposed classifier and controller models is compared with state-of-the-art models.

5.1. Evaluation Parameters

To evaluate the performance of the proposed classifier and controller models, the standard metrics are used, including accuracy and F1-Score. Let us define TP as True Positive, TN as True Negative, FP as False Positive, and FN as False Negative. Mathematically, these evaluation metrics are defined as Equations (7) and (8):

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{7}$$

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{8}$$

Accuracy is one of the most standard metrics to evaluate the performance of the CNN-based classification models. Accuracy is a metric of how close a measured value is to its standard or reference value. It means that the measurements for a given object are relative to the known value, but the measurements are far from each other, meaning that the accuracy

is without precision. The F1-Score, another standard metric for evaluating the classification problem, is a weighted mean of precision and recall. Precision $(TP/(TP+FP))$ is the quality of being exact. It refers to how two or more measurements are close to each other, regardless of whether those measurements are accurate or not. Recall $(TP/(TP+FN))$, also known as sensitivity, measures the ratio of the relevant results predicted correctly by the model. In the case where both precision and recall are influential, the F1-Score can be used.

5.2. Training Strategy and Parameter Tuning

In this step, the effects of different parameters on the proposed classifier and controller models are investigated. These parameters are including, input image size, training testing split ratio, and elements of proposed loss. Whenever an experiment is conducted, one parameter is replaced while the other parameters stay unchanged. In addition, each experiment is taking in 10-fold cross-validation (run ten times) method on both prepared datasets.

The tools and technologies employed for implementing and developing the proposed system primarily consist of Python developmental language V3.5. For deep learning model creation, the Keras framework based on Tensorflow 2.1 library is used. All the training models and procedures are executed on NVIDIA GeForce GTX 1080TI with 8 GB memory. The Stochastic Gradient Descent (SGD) [32] with a momentum of 0.9 is adopted as the optimization. The SGD is an efficient algorithm developed to reduce the loss function in the quickest way possible. The initial learning rate is set to the 3×10^{-5} reducing mechanism when validation loss stopped improving. The dataset is divided into a number of mini-batches during the training process. The batch size is set to 16, which means that the input data are divided into 16 batches and processed each batch in parallel. To prevent the possible overfitting, an early-stop strategy is used to monitor the validation loss improvement.

In addition, to make data more robust, an image data generator feature (data augmentation) is used. Data augmentation is a powerful feature and lets the data be enhanced in real-time while the deep learning model is still training or in the execution phase. Using the Keras library, there are a number of options, such as contrast random normalization and geometric transformations, which can be selected to perform the different transformations on data. Any transformation can be applied to any image present in the dataset while it is transferred to the model during training. Utilizing this feature offers two major benefits: saving time and memory overhead during the training phase and making the model more robust. Table 5 presents the parameters and their values, which are selected for the train data generator.

Table 5. Augmentation parameters and their values.

Parameters	Value	Parameters	Value
Input	20	zoom_range	0.2
Conv2D	0.2	rescale	1/255
MaxPooling2D	0.2	horizontal_flip	True
Conv2D	0.2	fill_mode	‘nearest’

5.2.1. Image Size

The classification performance of the proposed classifier and controller models is compared by given different input image sizes, including 128×128, 228×128, 256×256, and 465×256. Both models are trained on prepared datasets with these sizes, and results on the test set are illustrated in Table 6.

Table 6. The classification results of the proposed models with different input image sizes (mean ± std).

Image Size	Classifier Model		Controller Model		Time (s) per Epoch
	F1-Score	Accuracy (%)	F1-Score	Accuracy (%)	
128 × 128	0.91 ± 0.01	91.8 ± 1.7	0.92 ± 0.01	92.4 ± 1.4	10
228 × 128	0.92 ± 0.01	93.9 ± 1.2	0.93 ± 0.01	93.7 ± 1.3	11
256 × 256	**0.96 ± 0.01**	**96.2 ± 1.5**	**0.97 ± 0.01**	98.6 ± 1.1	15
465 × 256	**0.96 ± 0.01**	**96.2 ± 1.5**	**0.97 ± 0.01**	**98.7 ± 1.1**	19

It can be observed from Table 6 that, for both models, the input image with sizes of 256 × 256 and 465 × 256 outperform the other input sizes in terms of accuracy. Although the input image with sizes of 256 × 256 and 465 × 256 is increasing the training time, it can be ignored due to their significant improvement of the model's performance comparing to input image with sizes of 128 × 128 and 228 × 128.

The classifier model with input sizes of 256 × 256 and 465 × 256 attained the same result by achieving 96.2% accuracy. The controller model, when trained with an input size of 465 × 256, achieved an accuracy of 98.7% that is only 0.1% higher than when the input size is 256 × 256. Although the input image with the size of 465 × 256 has the same effect on the performance of the controller model (just improved accuracy 0.1%) as input size of 256 × 256 has, it significantly increases the computation cost by up to four seconds for each epoch training time. Therefore, the input image size of 256 × 256 is selected for further experiments.

5.2.2. Training Testing Split Ratio

The training-test split method is a technique to evaluate the performance of deep learning models for prediction on the data that has never been used during the training process. In order to achieve the best classification performance of the proposed models, the dataset is divided into five different training and testing ratios, including 90:10, 80:20, 70:30, 40:60, and 50:50. The X:Y split ratio means the X% of the data are used for training of the deep learning model while the rest of the Y% data are used to test the model. As each prepared dataset has a total number of 534 images when split into 90:10 ratio, 481 images are selected for training, and 53 images are selected for testing the model. Both proposed classifier and controller models are trained on the prepared datasets with the mentioned split ratio, and the results on the test set are presented in Table 7.

Table 7. The classification results of the proposed models with different input image sizes (mean ± std).

Training-Test Split Ratio	Classifier Model		Controller Model	
	F1-Score	Accuracy (%)	F1-Score	Accuracy (%)
90:10	0.95 ± 0.01	95.9 ± 1.4	**0.97 ± 0.01**	98.2 ± 0.9
80:20	**0.96 ± 0.01**	**96.2 ± 1.5**	**0.97 ± 0.01**	**98.6 ± 1.1**
70:30	0.95 ± 0.02	95.7 ± 1.7	0.96 ± 0.01	96.8 ± 1.3
60:40	0.90 ± 0.02	91.3 ± 2.3	0.91 ± 0.02	92.3 ± 2.1
50:50	0.84 ± 0.02	84.2 ± 2.2	0.84 ± 0.02	84.4 ± 2.7

As presented in Table 7, the models, when trained with split ratios of 90:10 and 80:20 obtain almost similar results, which is better than when the model trained with split ratios of 70:30, 60:40, and 50:50. However, both proposed models attained the best classification results when trained with a split ratio of 80:20 by achieving an accuracy of 96.2% and 98.6% for classifier and controller models, respectively.

5.2.3. Effect of α and β of the Proposed Loss Function

As expressed in the methodology section, the parameters α and β are defined to regularize the proposed loss function. The α is defined to control the scale of loss and β for controlling the scale of class weight. In order to evaluate the effect of α and β on the performance of the proposed loss, different values are given to these two variables. Furthermore, the proposed models are trained on both prepared datasets with the various values of $\alpha = \{0, \ldots, 5\}$ and $\beta = \{1, \ldots, 5\}$, and results on the test set are presented in Figure 9.

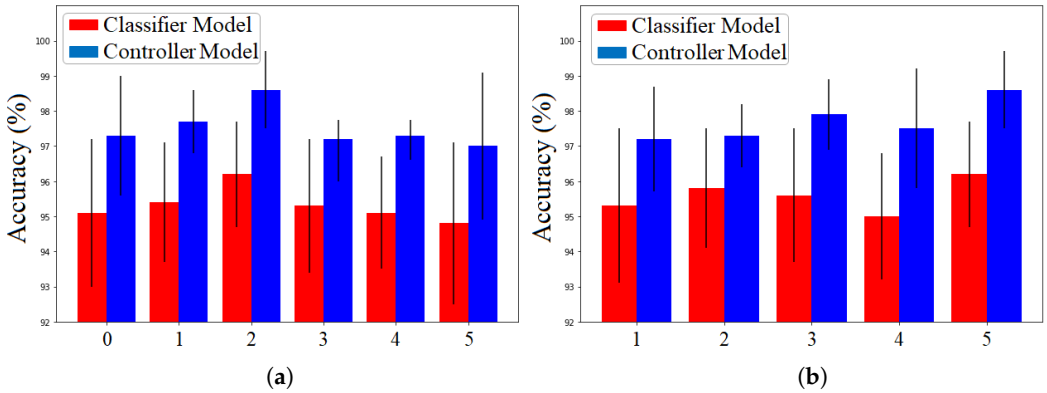


Figure 9. Performance of the proposed classifier and controller models with various values for α and β. Error bars illustrate ±1 standard deviation. (**a**) the effect of different values of α; (**b**) the effect of different values of β.

When $\alpha = 0$, the proposed loss will be the same as weighted CE loss, which only considers the probability of the input data when it is allocated to its true class. The proposed loss also considers the probability of allocating the input data to other classes rather than its actual class. Therefore, when the α has a smaller value, the loss focuses more on the data that the model correctly predicts their labels, and when the α has a bigger value, the second part of the loss function (which is responsible when the label of the data are not correctly predicted) becoming the main part. The best value for α and β can be achieved empirically and must be adjusted for each dataset separately. On the prepared datasets, both proposed models show their best performance when α and β are set to 2 and 5, respectively.

5.3. Comparison of Proposed Loss and Cross-Entropy Loss

The performance of the proposed loss is further compared to the CE loss. As mentioned earlier, the proposed loss is an extended version of the CE loss. The CE loss only considers the probability of the input data when it is allocated to its true class, but the proposed loss, in addition to the advantages of the CE, also considers the probability of allocating the input data to other classes rather than its true class. For comparison, the proposed classifier and controller models are trained in 10-fold cross-validation (run ten times) method once with employing the proposed loss and once with CE loss. In order to have a fair comparison, in both experiences, all the parameters and training strategies are the same, and the only difference is the loss function. The classification outcomes on the test set are monitored, and the best results are picked until the 25th, 50th, 75th, and 100th epochs. The results are shown in Figure 10.

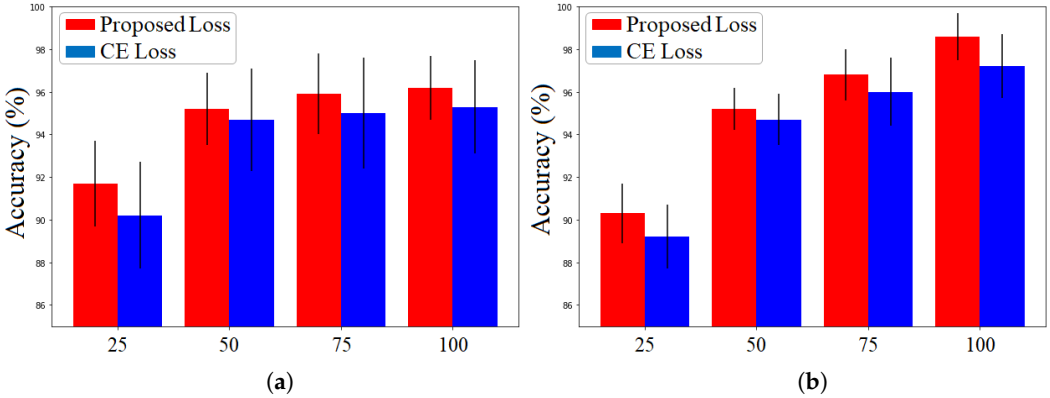


Figure 10. Comparison of the classification accuracies after 25th, 50th, 75th, and 100th epochs achieved by the proposed CNN model when trained with proposed loss and CE loss on the prepared dataset. Error bars illustrate ±1 standard deviation (**a**) results on the RGB image dataset; (**b**) results on the laser image dataset.

As shown in Figure 10, the proposed loss leads the network to obtain a better performance compared to CE loss for both classifier and controller models. It can also be seen from Figure 10 that the classification accuracies for both experiments are increasing as the number of epochs increases, and both proposed classifier and controller models achieved their best performance at epoch 100. In addition, almost in all experiments, the error bar indicates that, when the model is equipped with a proposed loss, it has a smaller standard deviation than when the model is equipped with CE. In total, based on the presented results, it can be concluded that the proposed loss not only improves the performance of the models by increasing the accuracy but also speeds up the optimization procedure and subsequently decreases the training time. The results for 100 epochs are presented in more detail in Table 8.

Table 8. Comparison of the effect of CE and proposed loss on the classifier and controller models when trained for 100 epochs. The results are taken on the test sets (mean ± std).

Cost Function	Classifier Model		Controller Model	
	F1-Score	Accuracy (%)	F1-Score	Accuracy (%)
CE Loss	0.95 ± 0.02	95.9 ± 2.2	0.96 ± 0.01	87.2 ± 1.5
Proposed Loss	**0.96 ± 0.01**	**96.2 ± 1.5**	**0.97 ± 0.01**	**98.6 ± 1.1**

5.4. Comparison of the Proposed CNN Model with State-of-the-Art Models

After key parameters evaluation, the performance of the proposed classifier and controller models are compared with three state-of-the-art models, including AlexNet, MobileNetV2, and VGG-16. As discussed in the methodology section, state-of-the-art models are designed to perform on the giant dataset, and, due to their convolution size and depth, overfitting occurs if they perform on the small dataset. Therefore, to train and make these models work properly on the prepared datasets, all of them are modified, and their hyperparameters are tuned according to the prepared datasets. The main modification is replacing the original fully-connected layers of these models with the fully-connected layers as designed for the proposed model. This significantly reduces the number of parameters. In addition, layer normalization is added to the main body of these models after the convolution layers. Finally, a strong dropout [33], with $p = 0.5$, is applied after the first fully connected layer.

The proposed classifier and controller models and adjusted AlexNet, MobileNet, and VGG-16 are trained on the prepared dataset with batch size 16 for 100 epochs. All the models are trained once with CE and once with proposed loss. For the proposed loss, the values of α and β are set to 2 and 5, respectively. SGD is employed as the optimizer with an initial rate of 3×10^{-5} and a momentum of 0.9. Data are split into a training and test set with an 80:20 ratio. All the models are trained by providing them with the augmented training dataset. The size of input data for the state-of-the-art models have not changed, and the original size has been used, which is 256 × 256 for AlexNet, 224 × 224 for MobilenetV2 and VGG-16.

As the results of the test set of the prepared datasets are presented in Table 9, the proposed models, when equipped with proposed loss, outperformed the other models by obtaining the accuracy of 96.2% and 89.6% for classifier and controller models, respectively. Moreover, all the models show slightly better performance when they are equipped with the proposed loss comparison to when they employed CE as the loss function. More well-known classification models were intended to be compared with the models as mentioned above, but they had pretty similar performance as presented models and therefore not been mentioned for comparison. These comparisons are not to demonstrate that the proposed classifier and controller models achieved better performance than the state-of-the-art models, but to illustrate that shallow networks such as the proposed classification model are simple yet accurate and effective for a small dataset.

Table 9. Performance comparison between the proposed models with AlexNet, MobileNetV2, and VGG16. The results on the test sets are measured by accuracy and F1-Score (mean ± std). For each measurement, the best result is bold. PR-L and CE-L indicate the proposed loss and Cross-Entropy loss, respectively.

Backbone	Cost Function	Classifier Model		Controller Model	
		F1-Score	Accuracy (%)	F1-Score	Accuracy (%)
AlexNet	Pr L	0.91 ± 0.01	91.2 ± 1.8	0.91 ± 0.02	91.3 ± 2.1
	CE L	0.91 ± 0.02	91.2 ± 2.6	0.90 ± 0.02	90.9 ± 2.3
MobileNet	Pr L	0.95 ± 0.02	95.4 ± 2.5	0.94 ± 0.02	95.5 ± 1.9
	CE L	0.93 ± 0.02	93.2 ± 2.6	0.94 ± 0.01	94.7 ± 1.6
VGG-16	Pr L	0.94 ± 0.01	95.8 ± 1.3	**0.97 ± 0.01**	97.8 ± 1.2
	CE L	0.94 ± 0.02	95.3 ± 2.4	0.96 ± 0.01	97.1 ± 1.7
Ours	Pr L	**0.96 ± 0.01**	**96.2 ± 1.5**	**0.97 ± 0.01**	**98.6 ± 1.1**
	CE L	0.95 ± 0.02	95.3 ± 2.2	**0.97 ± 0.01**	97.2 ± 1.5

5.5. Comparison of the Proposed CNN Model with State-of-the-Art Models

To evaluate the effectiveness of our approach, we examined the performance of the proposed system for mobile robot localization and navigation in the real environment. As mentioned, a real apartment (Figure 2) is chosen to take the experiments. Five different routes (see Table 2) are designed to evaluate the navigation system of the mobile robots.

For each route, if the robot executes all the commands correctly from the starting point and reaches the destination, it is considered a 100% success. Otherwise, if it makes a mistake during taking the route, the success rate will be calculated until the last command is done correctly. For example, if the robot is to move from workstation 1 to workstation 8 (route 1), first it should rotate to the angle of zero (T0), then move straight (Go), and finally stop at workstation 8 (SW8). In addition, before each "Go" command, the system checks the existence of the doorway and only allows the "Go" command to be executed if there is no obstacle in the way. Therefore, to complete path one, four commands must be executed correctly. If the proposed system correctly detects that there is an obstacle on the pathway (closed or semi-closed door), it stops the operation. In this case, only two commands are performed, rotation to the angle of zero (T0) and the obstacle check, which is considered as a 100% success. However, if the system mistakenly detects an obstacle in the path and stops

the operation, then only rotation to the angle of zero (T0) command is executed correctly, which is a 25% success rate. We tested each of the five routes fifteen times. We left all the doors open five times for the robot to pass through without any obstacles. We also left at least one of the doors halfway open five times and closed at least one of the doors in the path five times. We tested each designed route fifteen times. Five times, we left all the doors open for the robot to move through the pathway without facing any obstacles. Five times, we also left at least one of the pathway's doors halfway open, and another five times left at least one of the pathway's doors fully closed.

We took the experiments of the system, with the proposed and VGG-16 backbone for classifier and controller models, along with proposed and CE loss. The experiment results of the proposed mobile robot localization and navigation system in the real apartment environment are presented in Table 10.

As the experimental results in Table 10 show, the designed system performs better when adopting the proposed loss than when employing the CE loss. In addition, the best performance of the proposed localization and navigation system is obtained when the proposed backbone is used for the classifier and controller models along with the proposed loss function.

Table 10. The experiment results of the proposed system for mobile robot localization and navigation in the real apartment. PR-L and CE-L indicate the proposed loss and Cross-Entropy loss, respectively.

Backbone of Classifier & Controller	Cost Function	Route 1 Success Rate (%)	Route 2 Success Rate (%)	Route 3 Success Rate (%)	Route 4 Success Rate (%)	Route 5 Success Rate (%)	Success Rate Average (%)
VGG-16	Pr L	**100.0**	98.33	97.50	100.0	**93.30**	97.87
	CE L	95.00	93.33	88.00	80.83	77.73	86.98
Ours	Pr L	**100.0**	**98.33**	**98.03**	**100.0**	**96.10**	**98.50**
	CE L	**100.0**	93.33	90.86	80.83	81.62	89.23

6. Discussion

In this work, we proposed the CNN-based approach to address the image classification problem, which is a part of the proposed system for autonomous mobile robot localization and navigation in an indoor environment. We prepared a dataset with a small number of images. However, training deep learning models on a small data set is challenging, as they require a significant amount of data. Training the deep learning models on the small dataset may result in overfitting or performance limitations such as bounded accuracy. Therefore, to overcome these problems, we employed the following strategies:

1. Instead of the complicated deep learning-based models, which usually are designed to perform on large datasets, we proposed a simple CNN-based model to deal with small prepared datasets. According to the results, the intended strategy for building the CNN model was an effective way for classification with a small dataset.
2. We proposed a loss function that not only deals with the data that the model predicts their labels correctly, but also considered those data that are assigned to other classes instead of their actual class. In addition, the loss function has two effective parameters of $\aleph$ and β which control the scale of the loss and the scale of class weight, respectively. The results have demonstrated that the proposed loss function improved the performance of the proposed classifier and controller models, speeds up the optimization procedure, and subsequently decreases the training time.
3. Tuning of the hyper-parameters has led the networks to perform better by achieving higher accuracy rates. Data augmentation effectively improved the performance of the models by increasing the diversity of the training sets. Layer normalization made the models more reliable by reducing the noise of the training carves, and the SGD optimizer sped up the training process.

7. Conclusions

In this paper, we have introduced an approach based on the CNN model via image classification to address the localization and navigation of the mobile robot in an indoor environment. In combination with the proposed method and the topological map of the workspace, the mobile robot can find its location and navigate autonomously. Appropriate loss function has been proposed to improve the generalization capability since the prepared dataset has a small number of training images. The empirical studies demonstrated that the proposed techniques can effectively improve the performance of the neural networks, and thus benefit the navigation system. In future work, we would like to improve the proposed system so that the robot navigates autonomously in more complex environments. The limitation in presented work is when the environment has fewer features or has several areas with the same features, such as hospitals or hotels where there are many rooms with the same shape and characteristics. We strongly believe that more hardware and sensors such as Laser or RGB depth camera are required to overcome such a problem.

Author Contributions: Investigation, led the experimentation, performed data analysis, and drafted the manuscript, F.F.; Technical support and manuscript revision, J.W.; Project administration and supervision; Z.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (Grant No. 91848111), the National Natural Science Foundation of China (Grant No. 62103393), and the Chinese Academy of Science-The World Academy of Sciences (CAS-TWAS) President's Fellowship.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The prepared dataset for this study is available at https://rec.ustc.edu.cn/share/4dd7bfe0-f221-11eb-92fb-c5f9ff8e18c3, accessed on August 2021, password:1234.

Acknowledgments: This work is supported by the Chinese Academy of Science—The World Academy of Sciences (CAS-TWAS) President's Fellowship.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Guang, X.; Gao, Y.; Leung, H.; Liu, P.; Li, G. An autonomous vehicle navigation system based on inertial and visual Sensors. *Sensors* **2018**, *18*, 2952–2964. [CrossRef]
2. Konrad, T.; Gehrt, J.J.; Lin, J.; Zweigel, R.; Abel, D. Advanced state estimation for navigation of automated vehicles. *Annu. Rev. Control* **2018**, *46*, 181–195. [CrossRef]
3. Onyekpe, U.; Palade, V.; Kanarachos, S. Learning to localise automated vehicles in challenging environments using Inertial Navigation Systems (INS). *Appl. Sci.* **2021**, *11*, 1270–1292. [CrossRef]
4. Foroughi, F.; Zong, P. Controlling servo motor angle by exploiting Kinect SDK. *Int. J. Comput. Appl.* **2015**, *116*, 1–6. [CrossRef]
5. Rodrigues, M.L.; Vieira, L.F.M.; Campos, M.F. Mobile robot localization in indoor environments using multiple wireless technologies. In Proceedings of the 2012 Brazilian Robotics Symposium and Latin American Robotics Symposium, Ceará, Brazil, 16–19 October 2012; pp. 79–84.
6. Morales, Y.; Tsubouchi, T. DGPS, RTK-GPS and StarFire DGPS performance under tree shading environments. In Proceedings of the 2007 IEEE International Conference on Integration Technology, Shenzhen, China, 20–24 March 2007; pp. 519–524.
7. Song, Z.; Jiang, G.; Huang, C. A survey on indoor positioning technologies. In *International Conference on Theoretical and Mathematical Foundations of Computer Science*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 198–206.
8. Sibai, F.N.; Trigui, H.; Zanini, P.C.; Al-Odail, A.R. Evaluation of indoor mobile robot localization techniques. In Proceedings of the 2012 International Conference on Computer Systems and Industrial Informatics, Sharjah, United Arab Emirates, 18–20 December 2012; pp. 1–6.
9. Raghavan, A.N.; Ananthapadmanaban, H.; Sivamurugan, M.S.; Ravindran, B. Accurate mobile robot localization in indoor environments using bluetooth. In Proceedings of the 2010 IEEE international conference on robotics and automation, Anchorage, AK, USA, 3–8 May 2010; pp. 4391–4396.
10. Ferris, B.; Fox, D.; Lawrence, N.D. Wifi-slam using gaussian process latent variable models. *IJCAI* **2007**, *7*, 2480–2485.
11. Elbasiony, R.; Gomaa, W. WiFi localization for mobile robots based on random forests and GPLVM. In Proceedings of the 2014 13th International Conference on Machine Learning and Applications, Detroit, MI, USA, 3–5 December 2014; pp. 225–230.
12. Kim, H.; Lee, D.; Oh, T.; Choi, H.T.; Myung, H. A probabilistic feature map-based localization system using a monocular camera. *Sensors* **2015**, *15*, 21636–21659. [CrossRef]
13. Sattler, T.; Leibe, B.; Kobbelt, L. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1744–1756.
14. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
15. Agarap, A.F. Deep learning using rectified linear units (relu). *arXiv* **2018**, arXiv:1803.08375.
16. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
17. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
18. Dourado, C.M.D., Jr.; da Silva, S.P.; da Nobrega, R.V.; Barros, A.C.; Sangaiah, A.K.; Reboucas Filho, P.P.; de Albuquerque, V.H.C. A new approach for mobile robot localization based on an online IoT system. *Future Gener. Comput. Syst.* **2019**, *100*, 859–881.
19. Foroughi, F.; Wang, J.; Chen, Z. Indoor Robot Localization in Hand-Drawn Maps by using Convolutional Neural Networks and Monte Carlo Method. In Proceedings of the 2019 4th International Conference on Automation, Control and Robotics Engineering, Shenzhen, China, 19–21 July 2019; pp. 1–7.
20. Kendall, A.; Grimes, M.; Cipolla, R. Posenet: A convolutional network for real-time 6-dof camera relocalization. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2938–2946.
21. Radwan, N.; Valada, A.; Burgard, W. Vlocnet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robot. Autom. Lett.* **2018**, *3*, 4407–4414. [CrossRef]
22. Wu, H.; Wu, X.; Tian, G. Indoor robot localization based on single RFID tag. *Artif. Life Robot.* **2018**, *23*, 373–379. [CrossRef]
23. Lin, S.; Wang, J.; Xu, M.; Zhao, H.; Chen, Z. Topology Aware Object-Level Semantic Mapping Towards More Robust Loop Closure. *IEEE Robot. Autom. Lett.* **2021**, *6*, 7041–7048. [CrossRef]
24. Saravanan, M.; Kumar, P.S.; Sharma, A. IoT enabled indoor autonomous mobile robot using CNN and Q-learning. In Proceedings of the 2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), Bali, Indonesia, 1–3 July 2019; pp. 7–13.
25. Ran, T.; Yuan, L.; Zhang, J. Scene perception based visual navigation of mobile robot in indoor environment. *ISA Trans.* **2021**, *109*, 389–400. [CrossRef] [PubMed]
26. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 22–24 June 2009; pp. 248–255.
27. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
28. Kukačka, J.; Golkov, V.; Cremers, D. Regularization for deep learning: A taxonomy. *arXiv* **2017**, arXiv:1710.10686.
29. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 7–9 July 2015; pp. 448–456.

30. Ba, L.J.; Kiros, R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450.
31. Ren, M.; Liao, R.; Urtasun, R.; Sinz, F.H.; Zemel, R.S. Normalizing the normalizers: Comparing and extending network normalization schemes. *arXiv* **2016**, arXiv:1611.04520.
32. Yang, J.; Yang, G. Modified convolutional neural network based on dropout and the stochastic gradient descent optimizer. *Algorithms* **2018**, *11*, 28–42. [CrossRef]
33. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.