Building Indoor Localization with Deep Learning using Unconstrained Images

Submitted September 2023, in partial fulfillment of
the conditions for the award of the degree **MSc Computer Science.**

Joel Braganza
20508981

Supervised by Professor Andrew French

School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Date 08/09/2023

I hereby declare that I have all necessary rights and consents to publicly distribute this dissertation via the University of Nottingham's e-dissertation archive.

Abstract

Indoor navigation is an interesting problem that involves a computer learning about its physical environment. The scope of technologies and techniques applied is very wide in scope with many attempts in the hope of creating a robust and reliable navigation system. The applications of such a system would be seen in robotics, healthcare with assistive devices, augmented reality, surveillance, Internet of Things and many other domains. With the advent of deep learning, new possibilities arises to solve indoor localisation with more simplicity and less cost than previous approaches. In this project, a unique approach to the problem of indoor localisation is proposed with a convolutional neural network (CNN) plus a hidden Markov model (HMM) structure using only monocular, 2D images. Two datasets are created, one with 288 and the other with 576 images covering 24 locations in one building. The CNN acts as a classification network with each location acting as an individuals class containing a set of images. A lightweight neural architecture of 11 layers is used in achieving accuracies of 34.48% in the 288-dataset and 46.53% in the 576-dataset. On examination of the CNN results, it is determined that the poor performance were mainly due to incorrect classifications between classes with very similar visual information, like two locations along the same corridor. However, since we can assume that navigation in indoor environments follows a sequential path, a statistical approach using a HMM is implemented to improve the performance of the CNN. In this project the HMM is tested on a smaller scale with fewer classes demonstrating that it can successfully correct errors from the CNN as well as maintain correct classifications from the CNN.

Keywords: Convolutional Neural Network, Hidden Markov Mode, Indoor Localisation, Indoor Navigation, Deep Learning, Indoor Positioning.

Acknowledgements

I would like to extend a huge thank you to my supervisor Professor Andrew French for all his passion, support, and great ideas I received throughout my dissertation. His guidance and feedback was valuable. I always left every meeting more inspired to learn and open my mind up to new possibilities.

Of course I could not imagine the support, guidance and influence my parents have had on me as I am only here because of their hard work and sacrifices. I will always remember and be grateful for what they have done for me and will strive to honour them.

I would also like to thank all my friends for their joy and support.

Finally, I would like to thank God for the strength, wisdom and perseverance He has given me in completing this dissertation. I am eternally grateful for his mercy, grace and love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In a world where we have seen the proliferation of smartphones and exponential growth of computing power, a host of new possibilities are becoming available to build new applications. We have all seen the benefits of using navigation technology in our daily lives as well as in industry. The ease at which we can find the best route to a destination of our choice, almost anywhere in the world with great efficiency has changed they way we travel and communicate. However one of the current limitations to navigation technologies is that they do not extend to indoor environments very well and provide little information about our indoor surroundings.

The ability of computers to position themselves in an indoor environment is fundamental to the field of augmented reality (AR), robotics, surveillance and many others which must provide a 3D representation of the real world with little delay and great precision in order to be viable [3]. With the ability to make sense of their surround environments, computers will be able to act much more effectively and intelligently in our world. With the improvement of camera technology and decreasing cost, the potential for computer vision to exceed the capabilities of the human eye, autonomously and with great speed becomes possible. Improving the ability of software to recognise its location within an indoor en-

vironment will provide a strong stepping stone to creating more intelligent systems that can carry out higher order reasoning.

1.2 Overview and Background

The initial idea of this project was to develop an indoor navigation or positioning system that could allow a user or robot to navigate their indoor environment whilst having a mobile device with them. One of the main impediments to creating a system has been solving the indoor localisation problem. Out of the multitude of studies attempting to solve indoor localisation in the context of navigation, only a small minority have attempted to develop a navigation system based purely on 2D images with a deep learning method as the core component of the solution. In addition many approaches were quite complex and challenging to set up and maintain [4] [5].

Given the previous approaches to solving indoor localisation, would it be possible to develop a much more simpler and lightweight indoor navigation system to maintain or scale up if necessary? Since images are a relatively easy and intuitive form of data for humans and the vast majority of people have personal smartphones, it was proposed to use a smartphone device to retrieve a set of images in an indoor environment along with applying a deep learning method given its success in other domains [2], to develop a reliable solution to indoor localisation.

This could be accomplished with a classification network that would require a dataset where each class is a location in an indoor environment and the classification network would train to try and optimally recognise images within each class (location) from the dataset. This approach however has the obvious drawback of having an increasingly larger number of classes as more and more locations are added to the network, as well as issues with sub meter localisation as the network may find this confusing. Much literature has explored the use of 3D environments and forming 3D coordinate systems to solve this problem [4]. But for the purpose of this project given the scope, time and complexity,

a classification network approach to indoor localisation was pursued to investigate the results of this deep learning method.

One potential issue with a deep learning approach is the large dataset size required to adequately train a network. Given the time expensive process of data collection and constraints with processing power, a solution was needed to account for limited or poorer performance in the deep learning model due to the small dataset size. The addition of a hidden Markov model (HMM) to this navigation system was proposed because of the Markov model's success in a wide domain of use and its predictive capabilities with linear sequences related to categorical problems. The hidden Markov model allows the CNN's output to act as variables in the observed sequence of the model while the hidden sequence ideally represents the correct classification (location) label sequence [6].

1.3 Aims and Objectives

The aim of this project is to research, design and implement a solution to the indoor localization problem using a deep learning approach combined with a statistical model. This will involve training a network to classify 2D images from a smartphone with an appropriate accuracy using a self made dataset.

In order to achieve the aims of this project the following steps to be completed include:

- Create dataset by capturing and collecting images in an indoor environment at multiple locations.
- Preprocess and organize the dataset accounting for the challenges of identifying accurate locations as well as adding data augmentation.
- Design and implement a convolutional neural network. Train the network and evaluate the results.
- Design and implement a hidden Markov model and evaluate performance.

- Record and reflect on results.

1.4 Description of the Work

In this project we explore the use of an indoor navigation system for mobile users or autonomous robots that will provide information to most optimally navigate indoor environments.

A key part of this system involves indoor localisation which by applying computer vision techniques and statistical methods we investigate whether the location of a user indoors can be solved using a convolutional neural network (CNN) and a hidden Markov model(HMM) based using only 2D images from a monocular camera. An image dataset can be created to train the CNN and optimised through data augmentation to account for the limited size of the dataset.

Once trained, this system will be able to accept an image as input and classify at which location this image was taken. After a few images are inputted and classified, the HMM will correct any classification errors found in the CNN using logic either manually implemented or based on probabilities from the CNN model.

1.5 Dissertation Overview

This dissertation consists of 6 chapters. In chapter 1, the problem of indoor localisation and developing an indoor navigation system will be introduced along with the key aims and objectives of this project. In chapter 2 the surrounding literature around indoor localisation will be explored as well as the research behind deep learning networks and neural architectures in order to gain a better understanding of how to approach problems. Chapter 3 will outline the ideas behind creating the dataset and the process established to do so as well as the design and implementation of the deep learning network and the

hidden markov model. In chapter 4 the results of the network system will be presented along with a discussion into the findings. Chapter 5 will summarise the project before chapter 6 discusses further work that can be done.

Chapter 2

Literature Review

2.1 Indoor Localisation

2.1.1 Introduction

Global positioning systems (GPS) are not suitable to indoor spaces due to the structures and objects that would block or weaken signals used to determine location. The creation of a suitable indoor localisation system is a widely sought after problem with a large variety of techniques, technologies and approaches to solving [7]. The ability to extract a precise location from devices could be very valuable as it could add greatly to the context and circumstances of data. This could have great benefits to enterprise as well as building new adaptive systems around individuals and devices in their locations. A pertinent example of this would be the Internet of Things (IoT) and smart technologies [8].

Given that traditional positioning technologies have had fairly good localising capabilities, they may be able to be combined with newer long range IoT technologies to obtain the global location from indoor environments since only their relative, local locations can be obtained using current technology [9]. Localisation technologies can solve issues related to security and privacy of IoT devices by helping recover devices lost, tracking irregularities in device locations indicating a security risk, introducing authentication schemes based on proximity thereby increasing trust.

A lot of the traditional techniques of localisation revolve around signal measurements like signal strength, or the time that a signal propagates between a transmitter and receiver, or angle of signal arrival to name a few. These techniques are calculated using statistical methods, machine learning or with deep learning models like artificial neural networks (ANN) [9]. Technologies used to apply the previously mentioned localisation techniques include WiFi, Bluetooth, Zigbee, RFID (Radio Frequency Identification), UWB (Ultra Wide Band)[7] [9].

Indoor localisation is a field that has many applications requiring different levels of location precision. While most indoor localisation system would require a premade map of the indoor environment to be built, some approaches can account for a new environment by identifying objects and their physical characteristics in order to enable the user to navigate successfully [4]. Previous work in this field has included many variations of devices and methods to solve indoor localization. Some studies combined sensors with image data such as accelerometers and gyroscopes to increase accuracy, others used traditional image analysis choosing various features to extract important information [4].

2.1.2 Differences in Approaches

As classified by Brenna et al. the various technologies of indoor localisation include optical (infrared, visible light), sound (ultrasound, audible sound), radio frequency (WiFi, Bluetooth, UWB, RFID, Zigbee), passive technologies (magnetic field, inertial technology, computer vision) or hybrid technologies [8]. Using infrared signals has been negatively affected by many issues with obtaining a clear line of sight, dealing with light interference, and detecting small obstacles [5].

Ultrasound navigation based systems have the advantage of being undetectable to human hearing and can determine direction but provide a privacy risk to users. With many active

technologies that emit signals, performance issues appear as these systems are scaled up. This is due to the collision of signals in the indoor environment. This stands in contrast to passive technologies where the increase in devices receiving signals has not affected overall performance when these systems are scaled up [8]. One popular localisation system used in many studies is WiFi fingerprinting given the need for only software modification to configure. The low cost, quality of results as well as the existing infrastructure of access points available makes it attractive. However this approach requires a tedious data collection process with specialist input, a reliable internet connection as well as a recalibration process to update location mapping. Often this approach is dependent on the number of beacons available to gain accurate measurements [8] [10].

Data collected for configuring a system can differ from: using unique locations and sensors, purely images collected or 3D representation's of an indoor setting. In terms of the sensing devices there is a split between studies using 3D cameras or 3D generated images to map indoor environments versus 2D cameras or 2D images. There are also differences in using cameras in fixed positions, or mobile cameras, or cameras combined with other sensors like gyroscopes [4]. Traditional image analysis such as Scale Invariant Feature Transform (SIFT) or Speeded Up Robust Features (SURF) are used in some studies as localisation methods as opposed to machine learning methods like support vector machine (SVM) or deep learning methods convolutional neural networks (CNN) [4].

2.2 Deep Learning

2.2.1 Introduction

Deep learning (DL) is a subfield of machine learning (ML) that has become more popular due to the availability of data and lowering cost of computational power. Deep learning like machine learning learns and optimises itself using data, but it is also able to automatically extract features as part of the learning process eliminating manual feature engineering [2]. The most well known type of deep learning network is the convolu-

tional neural network (CNN). There are 4 types of learning that can take place including: supervised, unsupervised, semi-supervised and reinforcement. Supervised learning will used labeled data to train the network, gain accuracy and then test images on its own. Unsupervised learning uses only unlabelled data in training and produces patterns and relationships that humans would not otherwise be able to identify with large amounts of data. Semi-supervised data uses few labelled samples with many unlabelled samples of data aiming to improve the predictive abilities of the network. Reinforcement learning will employ a trial and error approach to create the optimal behavior of the model by getting the best reward [2] [11].

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) consist of multiple layers which extract useful information known as features from the input data. CNNs leverage the use of local spatial information allowing the model to gain invariance to translation and scale [12]. A key characteristic of CNNs include convolution which is a process that involves sliding a window known as a kernel filter over an image as seen in figure 2.1, multiplying every single value in the grid, adding the result and outputting a single value as a pixel on the new grid. So a 3x3 filter will insert the value in the pixel on the second row and second column

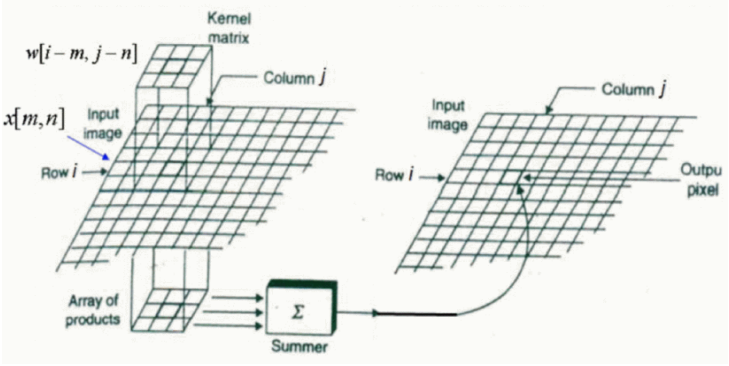

Figure 2.1: Kernel Filter Convolving Over Input Image And Outputting New Pixel Grid. [1]

of that filter window. Since the convolutional layers convolve over the kernel matrix region outputting this to the next layer, the number of connections is much less compared to a

fully connected layer where each neuron is connected to every neuron in the next layer. So as the pixel data progresses through the hidden layers of the convolutional model, more higher order abstract features are able to be extracted from the dataset. The values in the kernel filter are altered during the training process to reduce the loss function and select the best features [1].

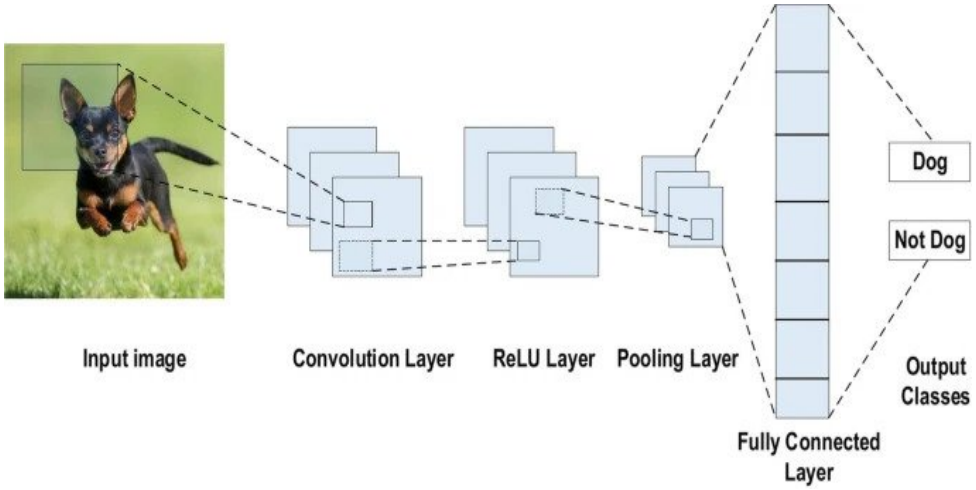

Figure 2.2: Example of CNN Architecture with Binary Classification [2]

CNNs often include a pooling layer which accesses larger areas of the image, reducing the resolution of the output of the convolutional layers and the amount of parameters in the model. There are a number of pooling methods like max pooling or average pooling which can reduce the spatial size of the image representation [2]. An activation function is also typically added after a convolution layer to add non linearity to the network. The most common activation function is known as Rectified Linear Unit (ReLU). The final layers will consist of a fully connected layer (FCN) similar to the older multilayer perceptron (MLP) network with each node connected to eachother between each layer leading to the final classification output. A dropout layer can be added before the FCN to randomly drop neurons, increasing uncertainty and reducing overfitting [1]. A conceptual diagram of the overall network architecture of a CNN is seen in figure 2.2

2.2.3 Computer Vision Based Localisation

Computer vision in localisation, can be accomplished through the use of CNNs. Some of the advantages of using a passive approach with a vision localisation system whereby the camera is located on the user/robot and images are taken from a first person point of view, is that it is low cost, private, user friendly and able to achieve high accuracy [8]. Crowdsourcing data is a popular approach to building and maintaining complex models with incentive schemes to ensure adequate data quality. Incorporating more complex hardware like binocular cameras have also been explored in order to deal with the problem of computer vision model inadequacies from a more pure computer vision, image dataset construction process [3].

Vision based methods can be divided into those that employ markers or non-marker methods. Marker methods are quite robust with great performance but require the setting up and producing of many distinct markers in the environment [5]. Khan et al. has listed many challenges with developing an indoor navigation system using computer vision. These include making a scalable system that is generalisable and able to be applied to new environments. Efficiency when it comes to memory storage and computational requirements are important as applications often require processing in real time with limited battery power. Dealing with changes in brightness and objects that block others can also lower the performance of the system [8] [5].

There is an inherent advantage to vision based approaches given the technological infrastructure in place with smartphones equipped with cameras and sensors as well as their low power usage and cost. Also from a technical standpoint with the digital camera, because it has developed in a natural way similar to the human eye, developing methods to learn about indoor environments is easier [5].

This study [13] focuses on the application of indoor localisation on humanoid robots using a visual place recognition algorithm for navigation. They think of the environment

as a graph structure with nodes representing a unique location and connections called edges between the nodes. An advantage with vision is that it easily solves loop closure (when a robot returns to a place it has been before) by easily comparing images. This study uses an RGB camera to collect images and the authors compare manual feature engineering with deep learning feature extraction using a transfer learning approach with the VGG network[13]. Transfer learning is a process by which a pretrained network on a larger dataset is adapted to a smaller dataset often in a different domain; this is known as fine tuning. Sometimes only parts of the transfer model are used and adapted with other classifiers like SVM (Support Vector Machine). VGG-F is an 8 layer CNN requiring 224x244x3 input image size. For the study the 7th layer was replaced with a fully connected layer, softmax and classification layer as output. These datasets consisted of 16 rooms with 8000 images in total. This paper showed that the accuracy of the VGG network was much higher in comparison to handcrafted feature methods like SIFT or Bag of Words (BoW) as well as the F1-Score which was 96% with data augmentation compared to 73.10% for BoW or SURF (Speeded Up Robust Features) [13].

This paper [13] then goes on to describe their network architecture system using a CNN with an LSTM (Long-Short Term Memory) as it is resilient against blurring and changes in brightness. The authors intend to outperform PoseNet by using LSTM units in the fully connected layer of the CNN in order to reduce dimensionality and prevent overfitting. The goal is to "regress camera poses directly from images"[14]. The CNN architecture used is an existing network called GoogLeNet with average pooling and fully connected (FC) layer which leads to a FC regression layer that outputs pose and orientation. The LSTM is placed after the FC layer and before the FC regression layer reducing dimensionality. The dataset contains 1095 images with six images at each location in five directions covering 360 degrees, with one pointing up. The locations are spaced 1m apart and the authors mentioned textural and structural similarities between locations adding difficulty. Results showed that the novel architecture is 37.5% more accurate than PoseNet with position and 19% more accurate than PoseNet with orientation. However a SIFT method known

as Active Search produced even better results than the CNN plus LSTM architecture, however it was not always able to calculate a pose result making it less reliable than the CNN method in some cases[14].

The next two studies Han et al. and Zhao, Xu et al. are similar in that they involve using a CNN with SLAM to identify and eliminate problems due to dynamic changes captured in the images. In Han et al., a monocular camera is used with Simultaneous Localisation and Mapping (SLAM), an algorithm commonly used to help a robot localise and map out unknown environments at the same time. In this study a Mask-RCNN locates and removes an obstacle and returns the new image to the SLAM algorithm. The model is pretrained on the MS COCO dataset[15]. Zhao, Xu et al. built a vision based indoor navigation system based on SLAM (Simulatenous Localization and Mapping) but uses a deep learning network to fix the brightness, textural and dynamic object changes in the environment that worsen the performance of SLAM. This was implemented using a single shot model (SSD) with ResNet50 that would detect new objects in the environment deleting them, then sending the images back to the SLAM algorithm [16].

This study mentions the importance of an improved indoor localisation method in the field of augmented reality given the demands of obtaining real time tracking and localisation of objects. Whilst computer vision methods provide a possible solution, problems with obtaining large dataset for 3D environments and computational costs lead the authors to propose a lightweight system [3]. The system proposed aims to achieve: sub meter accuracy, reduced overhead only requiring a smartphone and increasing accuracy in a 3D coordinate environment by combining fingerprint and image methods. This was designed by using WiFi to create subspaces to divide the "feature space into sub-area"[3]. The localisation system called iStart had an average error of 0.6m, view error of 6 degrees and able to localise in 4 seconds. This study [17] also relates to the AR domain by developed a guided AR based map with audio feedback for navigating a shopping mall using a CNN. The CNN was trained to recognise and classify shopping signs. A 2D map was created along with audio feedback to help the user reach their destination. In order to determine

the most optimal path, data is extracted from sensors related to motion and orientation [17].

Previous studies often used a combination of DL with traditional methods and ML such as this study [18] that used multiscale local binary pattern features (MSLBP) to extract features with a deep learning model and a support vector machine model to classify images into 42 locations[18]. Another example of combining DL with older methods is seen in Zhao, Zhu et al. [19] where a combination of a different type of signal feature called CSI (Channel State Information) with a new Kalman filter (filters noise) and a CNN predict position based on the CSI fingerprint image [19]. Their neural network design inspired by AlexNet was 6 layers with 2 convolutional layers 3x3 filters, 2 max pooling layers, 2 fully connected layers, ReLu (Rectified Linear Unit) activation function and an L2 loss function. Compared to other neural architectures like ResNet the accuracy of the position estimate from the CSI fingerprint images improved by 47.2% with a 0.93 reduction in mean error [19].

Some papers approach the idea of indoor localisation from a behavioral standpoint like Xiao et al. [20] describes an indoor localization approach that focuses on common objects to gain positioning in large indoor spaces. Modelling after the human brain using certain types of nerve cells activated when a person is in a certain place as well as positional cells that use a coordinate system, the authors of this study used object recognition and position computation to calculate a user's position using a faster-RCNN network [20]. Another study [21] uses semantic information rather than just visual features to solve the localisation problem with a deep learning approach [21]. A semantic database is created using a mask R-CNN network for object classification and SURF for keypoints and descriptors. Parts of ResNet50 are used in the CNN model with a focus on object appearance in the middle layers and changes in viewpoints in the top layers of the model. A coarse approach looks at the class of the object while a more detailed subsequent step extracts CNN image features. The mask R-CNN network was pretrained on the COCO

dataset containing common objects found in indoor environments. Results showed 90% of images localised to 30 cm as well as better performance than other studies using PoseNet or CNN and LSTM combinations [21].

As mentioned before some possible solutions involve using different camera hardware or better input image datatypes as seen in the next two studies [10] and [22]. The indoor localisation systems proposed in Sun et al. are concerned with locating humans. One of the proposed systems uses a single panoramic camera on a ceiling to first detect a person in the image. Then the pixel location that best matches with the location of the person is extracted and mapped onto a room map to construct a coordinate system using an artificial neural network [10]. The ANN used is a three layer perceptron trained with a back propagation algorithm using different locations of a human in images with the ground truth of the location coordinates known. The panoramic camera localisation system performed much better with a mean error of 0.84 m compared to WiFi approaches fingerprinting localisation using K nearest neighbour (KNN) or a propagation model (PM) with mean errors of 1.70 m and 2.33 m respectively [10]. Akall et al. examines merging images from 4 cameras on a robot to create a compound image to provide a wider field of view and help interpret the location of the robot. A CNN is trained in this study for each camera with a separate one for the compound images. The results show that the CNN with compound images performs better [22].

Chapter 3

Design and Implementation

3.1 Dataset

Figure 3.1: Sample Image of the Dataset

3.1.1 Planning

In order to train a deep learning network to recognize a user's location in an indoor environment, a dataset needed to be created based on what a user or robot would experi-

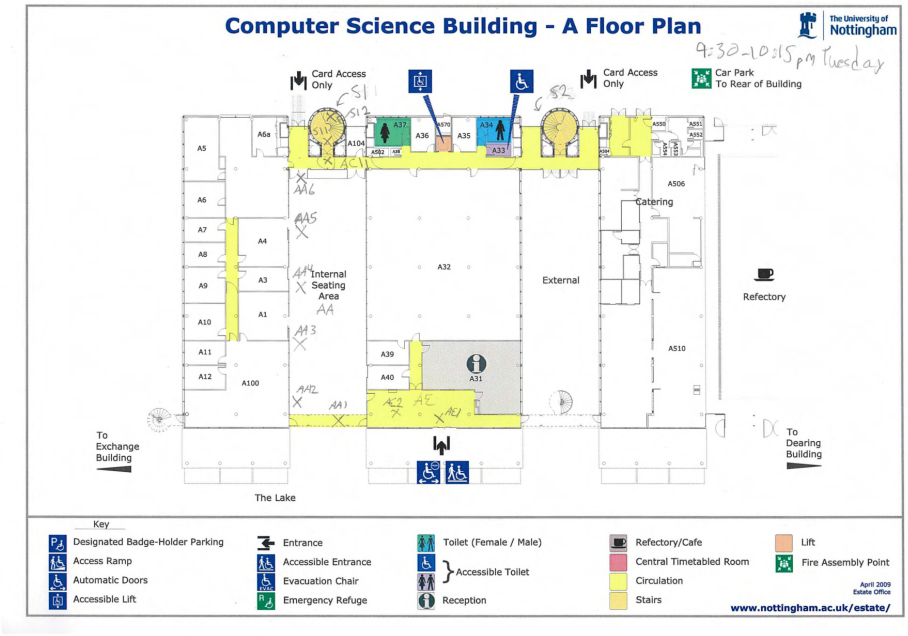

Figure 3.2: Floorplan for Level A

ence when navigating an indoor environment. Naturally CNN's can work well with large amounts of data to perform classification tasks quite effectively, but this would provide a challenge, given the limited amount of data that could be collected in this project. The School of Computer Science building in the Jubilee Campus at the University of Nottingham 3.2 was the test case for this indoor localisation network given the relatively large size, multiple floors and variety of locations.

In order effectively classify the location in an indoor environment, different viewpoints and angles would need to be accounted for as well as an appropriate spacing between individual locations. Given the limited datasize and use of a smartphone to capture images, spacing between locations was approximately 5 metres to avoid overlapping locations too much. For the purpose of this project, given that the dataset would need to model after a user taking images on a mobile device such as a smartphone the images would need to be taken at human level height.

As seen in 3.2 a floor plan of the ground level (level A) of the building is shown with crosses marking the locations where the images would be taken. Each of the locations marked with a cross represents the place where a set of images were taken and stored in a single class. So each marked location represents a different class. The path taken through the building: started at the entrance, moved through the atrium, up the stairs and then to the next floor (level B) 3.3. Then a left at the top of the stairs down the corridor all the way to the right most corridor on the map finishing at the office at the end of that corridor.

Each location was labelled according to the floor level it was on, the room which it was in and at what stage of the step sequence it was located at. For example AA6 represented a location on level A (ground floor), in the atrium (A) and at the 6th step within the atrium, or BC24 represented a location on level B (1st floor), in a corridor (C), in corridor number 2, and at the 4th step in that particular corridor.

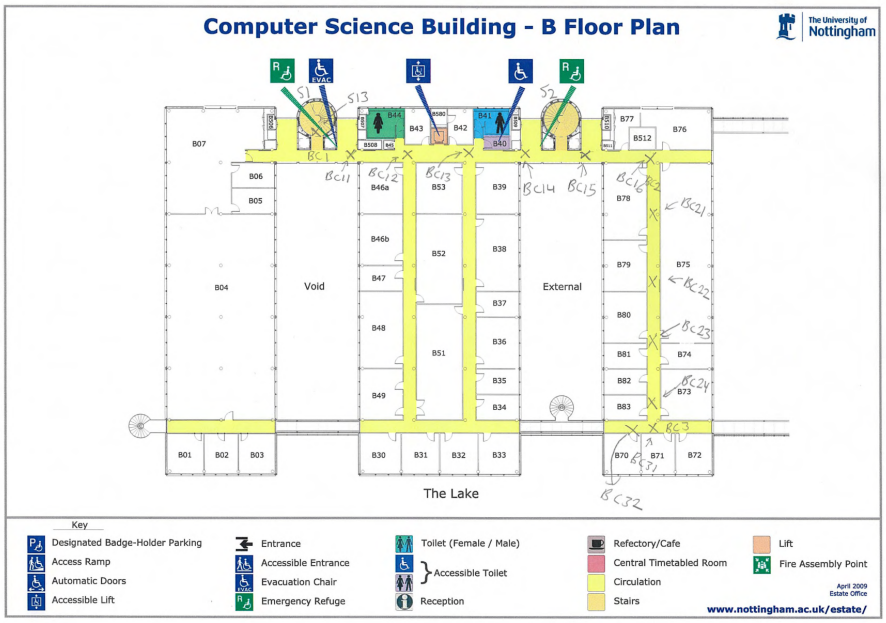

Figure 3.3: Floorplan for Level B

The initial concept was to implement the system on a route from the entrance of the building to a office on level B, and then extend it with alternative routes through other corridors and rooms such as A32, the computer lab.

3.1.2 Data Preprocessing and Organisation

The creation of the dataset involved collecting two datasets, with the second being an improved version of the first. The data collection process involved taking images vertically on a iphone 10SE using the smartphone camera with an aspect ratio of 4:3. Images were first converted from HEIC (high efficiency Image File Format) to JPG then rotated 90 degrees clockwise to position them right side up. Using a coordinate method the images were cropped to remove the black bars on the left and right of each image. For both datasets, image locations started from the main entrance of the building and followed the path a user would take to a office on the second floor with each image collection location spaced by 10 steps.

The images were then divided into folders of 12 images numbering 24 directories in total, then further subdivided into training and testing folders. The folders were named with abbreviations according to the room/corridor/staircase the images were situated in as well as numbered with the sequence according to the path one would take going through the building. The organisation and naming of folders was implemented with a bash file and a software suite called ImageMagick was used to convert images from HEIC to JPG. For each dataset the training and testing images were split into two separate folders organised by class name.

3.1.3 Data Collection Process

The first dataset (288-dataset) image collection process involved first facing the westerly direction towards the front of the building and then rotating in a clockwise fashion cap-

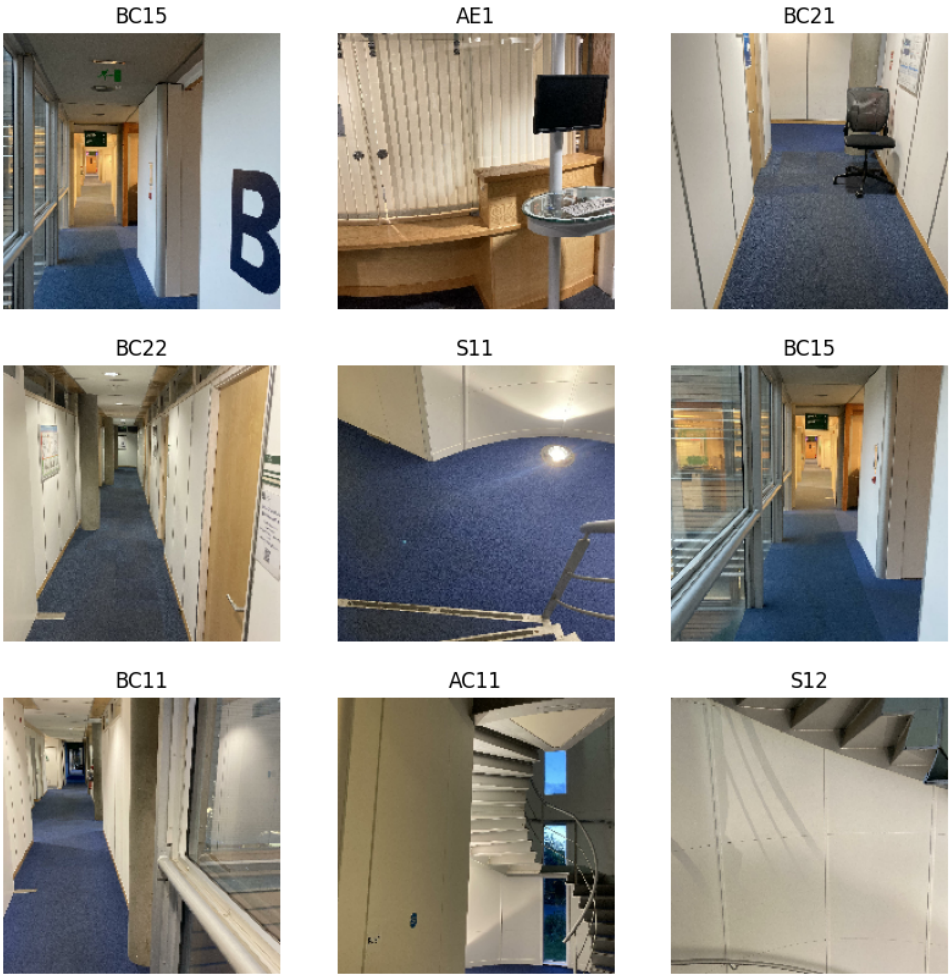

Figure 3.4: Sample of Images in Dataset after Preprocessing

turing 3 images every 90 degrees until a full 360 turn had been achieved with a total of 12 images collected, totalling 288 images. The train test split was 8 images for training and 4 images for testing in each class. In order to train and test the model on the full 360 degree view of each location, 2 out of 3 images in each 90 degree slice were chosen for training and the remaining 1 out of 3 images were chosen for testing; totaling 8 out of 12 images for training and 4 out of 12 images for testing in each location giving a 66%-33% split.

The second improved version of the dataset (576-dataset) was created by capturing 24 images in two viewpoints at each location with 576 images in total. The two viewpoints included a front view and a backward view with 12 images in each direction as if to simulate a user or robot experience while progressing along a path indoors. The images captured

the same scene in each view with slight horizontal and vertical translation variation. An exception was given to a T junction where 8 images were taken in three directions. The train test split was 75%-25% by choosing 3 images in front view and 3 in a backward view to be in the testing set, leaving 9 images in front view and 9 in backward view for training. For T junctions, 2 images in each location was chosen for testing in each direction totalling 6 images with the rest being used for training. There were 5 locations in the dataset that were T junctions.

Given the images were taken in an ordered fashion, the training and testing images sorted were always chosen in the same direction, mainly in the first dataset. Since the images of the second dataset changed direction depending on the path of the user and did not contain a standard direction from which image collection process always started. There was a great overlap of many images within each view (front and backward) and this same location in the ordered sequence of images taken in each class/location to sort through training and testing images allowed the training and testing images to be more varied.

3.2 Convolutional Neural Network

The convolutional neural network was implemented in python using the Keras interface with the TensorFlow Library. The computational power the CNN is trained on is an Intel Core i7-3520M CPU @ 2.90GHZx4 with 16 GB of RAM. Some of the libraries used for preprocessing and evaluation metrics include matplotlib.pyplot, numpy, PIL (Python Imaging Library), and pandas.

3.2.1 Neural Architecture

According to floor plans for both levels in figure 3.2 and figure 3.3, the locations marked with a cross as previously mentioned represent the individual classes in the dataset. Subsequently the objective of the CNN model is to predict the class or X marked location an

image comes from. The CNN model trains on the dataset consisting of 66% (288-dataset) or 75% (576-dataset) of images in each classes and testing on the remaining images in each class as well.

The specific choice of which images in each class must be in the training and testing set is important as we don't want to shuffle images between classes, given the great similarity with some classes, techniques such as cross validation or nested cross validation were excluded from the preprocessing process. Before inputting the data into the CNN, data augmentation was implemented using keras layers with random translation as seen in figure 3.5. The random translation horizontally shifted images 10%-30% to the left

Figure 3.5: Sample of Random Translation Applied to Training Images in 576-Dataset

and right, tripling the size of the training set resulting in 576 training images in the 288-dataset and 1296 images in the 576-dataset.

As shown in figure 3.6, the CNN consists of 11 layers. The first layer rescales the images from an original dimension of 3012 x 3004 pixels to 200 x 200 pixels. This layer also normalises the RGB values that range from 0 to 255 to the range of 0 to 1 and the 3

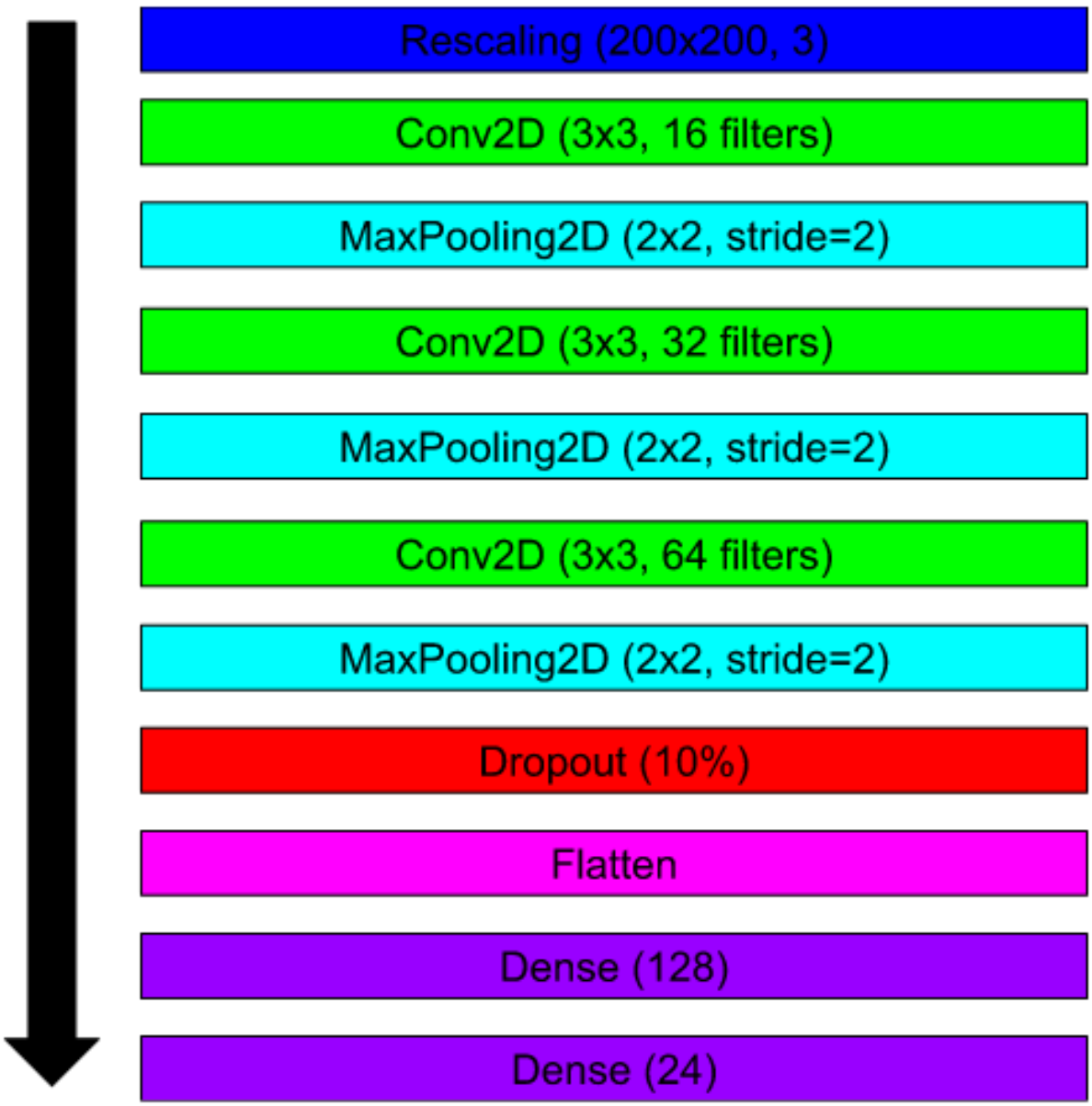

Figure 3.6: CNN architecture

represents three channels, RGB. Since the rescaling is incorporated as layer, images will be rescaled and normalised after preprocessing and data augmentation.

The Conv2D stands for a 2D convolutional layer and consists of the filter size and dimensionality of the output or number of filters produced by the layer. The filter size remains 3x3 throughout the neural architecture but the number of filters increases by a factor of 2 as the model gets deeper. This is with the aim of extracting higher order features that contain more useful information. For the convolutional layers, a 3x3 filter size has the best performance compared to 5x5 or 7x7 filter sizes [23][19].

The MaxPooling2D represents the maxpooling operation which summarises the features from part of the feature map created in the convolutional layers. This is referred to as a downsampling technique making the model more insensitive to local translation (invariance) so a lower resolution of the input is created but it will contain more important features thus increasing the receptive field size. Like a kernel filter it has a pool window size which is 2x2 and stride of 2 specifying how far the pool window moves.

The dropout layer will ignore certain nodes in a layer at random each iteration. In this case 10% so if there were 1000 nodes, 100 nodes would be randomly dropped in every iteration. For this reason dropout is used to help with the problem of overfitting. The flatten layer takes the input data and flattens it into a one dimensional array because the dense layer only take one dimensional input and the convolutional layers have a high dimensionality.

The dense layer is the same as a fully connected layer as previously mentioned in Chapter 2. This means that every neuron receives and gives input from the previous and next layer. The number specified in figure 3.6 refers to the input size or number of nodes with the last dense layer being 24 representing the number of classes in the dataset.

After each convolutional layer an activation function known as ReLU (Rectified Linear Unit) was used to add non linearity to the network. This function allows us to change the shape of the response by specifying a threshold by which neurons with a negative input are set to 0 while those with a positive number are passed through.

Before the model can be trained, it is compiled and tuned with hyperparameters, including the adam optimisation algorithm is used and the sparse categorical cross entropy. A sparse categorical variable is one used when a value is represented as an integer and the sparse categorical cross entropy is a loss function that will use this sparse categorical variable.

The loss function compares the predicted values with the target values providing a metric of how well the network is training. The aim is always to minimize the loss function.

3.3 Hidden Markov Model

The Hidden Markov Model was implemented from scratch using code from: `https://towardsdatascience.com/hidden-markov-model-implemented-from-scratch-72865bda430e` The libraries used in the implementation included maplotlib, numpy, pandas, itertools and functools.

HMM was tested on only four classes due to time limitations and simplicity. The four locations were all located in the atrium ("Internal Seating Area" as in figure 3.2) on level A:

`AA1, AA2, AA3, AA4`

The confusion matrix values in figure 4.4 were adapted to include any incorrect test classifications to stay within the bounds of AA1, AA2, AA3, or AA4 as seen in figure 3.7.

$$\begin{bmatrix} 0.52 & 0.16 & 0.16 & 0.16 \\ 0.16 & 0.52 & 0.16 & 0.16 \\ 0.01 & 0.65 & 0.01 & 0.33 \\ 0.33 & 0.01 & 0.01 & 0.65 \end{bmatrix}$$

Figure 3.7: Emission Probability Matrix (B) based on 576-dataset confusion matrix

The main components of the hidden Markov model are the state transition matrix (A) 3.8, emission probability matrix (B) 3.7 and the initial state probability distribution (π) 3.9. Each row of the matrices and vector must add up to 1.0 as the values are a probability distribution across different states.

$$\begin{bmatrix} 0.20 & 0.80 & 0.00 & 0.00 \\ 0.45 & 0.10 & 0.45 & 0.00 \\ 0.00 & 0.45 & 0.10 & 0.45 \\ 0.00 & 0.00 & 0.80 & 0.20 \end{bmatrix}$$

Figure 3.8: State Transition Matrix (A)

$$\begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

Figure 3.9: Initial State Probability Distribution (π)

In a markov chain the next state only depends on the previous state and its probabilities. A hidden Markov model (HMM) is made up of an observed sequence and a hidden sequence. So the markov chain becomes the hidden sequence which is not known by the user but the observed sequence are a set of variables that are dependent on the hidden states.

So the probabilities of transitioning between the hidden states (classes AA1, AA2, AA3, AA4) are represented by the state transition matrix (A) and the probabilities between the hidden states and observed variables are represented by the emission probability matrix (B). Notice there are no probabilities between the observed variables. The current observation is dependent only on the current hidden state not the previous observed variable in the observed sequence. The initial state probability distribution is used at the beginning of the hidden sequence to assign a probability to the first hidden state.

In the implementation, a class representing a vector and a class representing a matrix were created to serve as fundamental blocks of the model. Once these were constructed, a hidden markov chain is implemented given that:

$$\lambda = (A, B, \pi)$$

Using inheritance to extend the functionality of the hidden markov chain the model uses a forward-backwards algorithm to find the most likley hidden sequence produced by an observed sequence using the .uncover method. The overall idea was to input class labels from the CNN into the HMM as an observed sequence.

Chapter 4

Results and Evaluation

4.1 Introduction

The previous chapter explained how the indoor localisation problem was approached and the system designed. This chapter will go through the results of the CNN with the initial, smaller dataset and the larger improved dataset as well as present the results of the HMM. The following graphs are representative of the training process of the network. An epoch represents that all the samples in the training set have influenced the internal parameters of the CNN.

The confusion matrix is a table which shows how many of the samples in each class were correctly classified and how many were not and in which location were they incorrectly classified. Read horizontally the rows (y axis) represent the correct classes with the columns (x axis) indicating the misclassified test sample.

4.2 Convolutional Neural Network

4.2.1 Metrics

Various evaluation metrics are presented in 4.1 comparing the 288 dataset with the 576 dataset. There is a clear improvement in the CA (classification accuracy) score from

34.38% to 46.53% representing a 12.15% increase based on just changing the dataset as the neural architecture, hyperparameters and number of epochs during training the model remained the same.

Metrics	288-Dataset	576-Dataset
CA	34.38%	46.53%
Precision	50.56%	50%
Recall	52%	50.98%
F1	51.27%	50.49%

Table 4.1: Evaluation Metrics for CNN

The precision, recall and F1 scores are similar between the two datasets with a slight decrease in the recall and F1 scores in the 576-dataset. Interestingly the F1 score decreases from 288-dataset to the 576-dataset while the classification accuracy increases. The F1 score is also generally higher than the accuracy score in both datasets 4.1. The relative equality between precision and recall could suggest that the number of false positives is approximately equal to the number of false negatives. The precision score (P) and recall score (R) is calculated:

$$\mathtt{P} = \frac{\mathtt{TP}}{\mathtt{TP} + \mathtt{FP}} \qquad \mathtt{R} = \frac{\mathtt{TP}}{\mathtt{TP} + \mathtt{FN}}$$

$$\mathtt{TP} = \mathtt{True\ Positive} \quad \mathtt{FP} = \mathtt{False\ Positive} \quad \mathtt{FN} = \mathtt{False\ Negative}$$

The F1 score is calculated:

$$\frac{2 * \mathtt{Precision} * \mathtt{Recall}}{\mathtt{Precision} + \mathtt{Recall}}$$

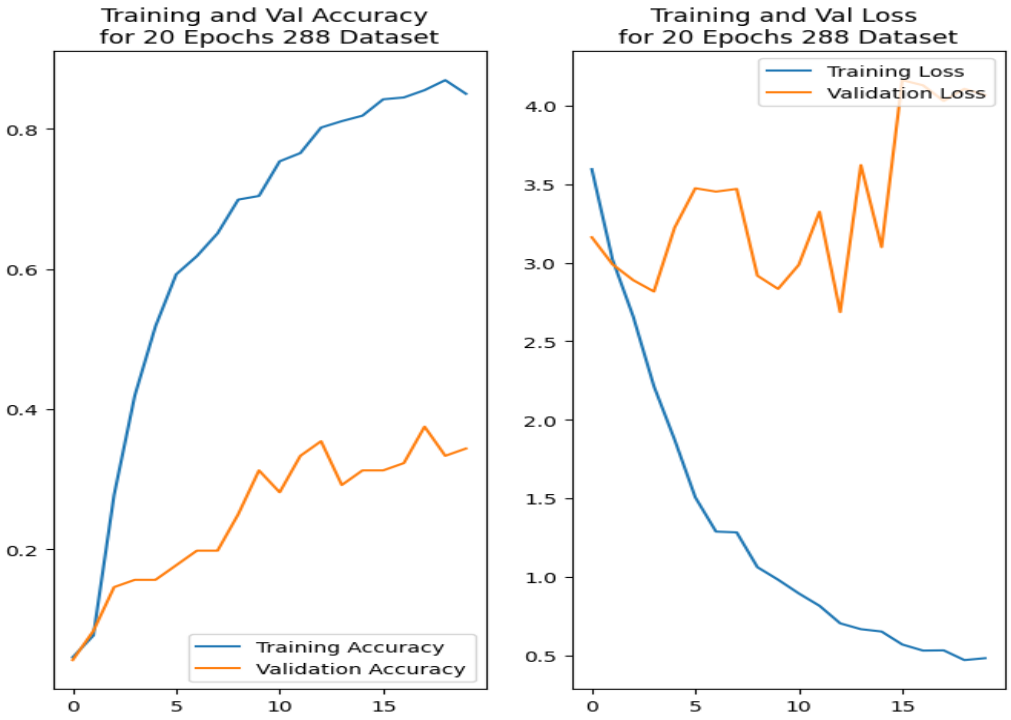

Figure 4.1: Training and Validation Accuracy and Loss Over 20 Epochs for 288 Dataset

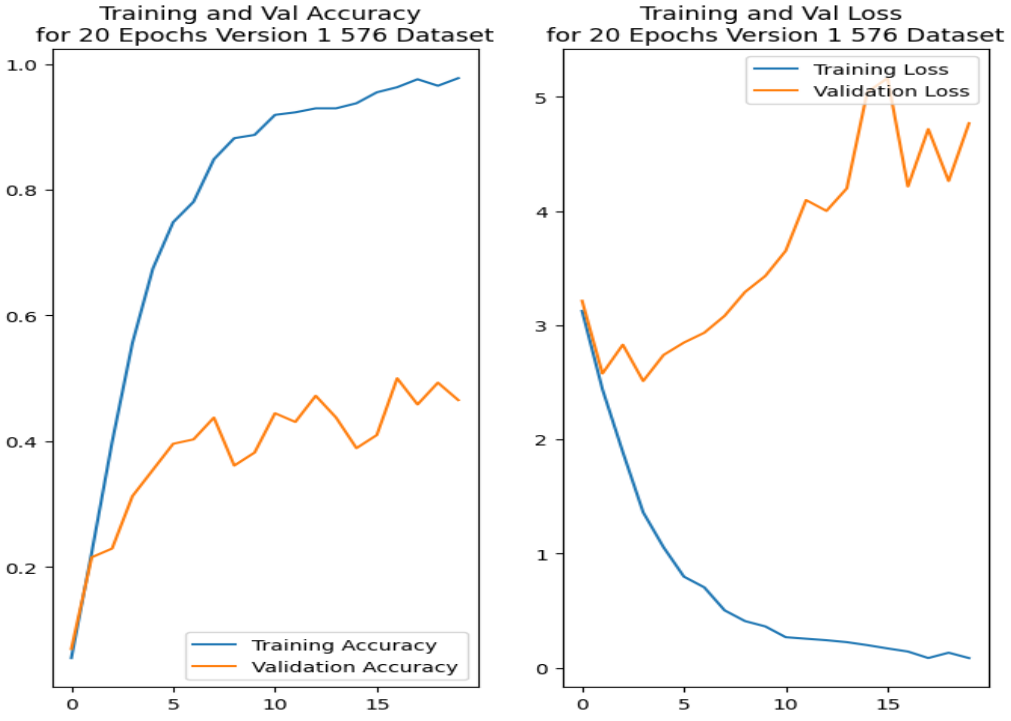

Figure 4.2: Training and Validation Accuracy and Loss Over 20 Epochs for 576 Dataset

The training and validation accuracy and loss is displayed over 20 epochs for the 288-dataset 4.1. On first glance it appears to be an overfitting problem with the training

accuracy rising much higher than the validation accuracy. This seems to be further confirmed with an oscillating then increasing validation loss starting around epoch 10 to epoch 20.

The training and validation accuracy and loss displayed over 20 epochs for the 576-dataset 4.2 showed similar results to the 288 dataset 4.1 but with a higher validation and training accuracy. The problem of overfitting is also evident in this example with a much higher training accuracy compared to validation accuracy as well as a higher validation loss shown in the 576 dataset. In comparison to the 288 dataset the validation accuracy seemed to have a steeper incline between 0-5 epochs suggesting that overfitting occurred as the model progressed through the training set.

Given that the dropout layer was only 10% and placed after the last hidden layer, near the top layers before the fully connected layers, this could have contributed to the overfitting. Even with data augmentation the dataset size is still limited, contributing to the poor performance in validation accuracy. The change in viewpoints by focusing on a forward and backward view, or three directional view in the case of a T junction compared to just a 360 degree turn, improved the model as seen with the classification accuracy in 576-dataset.

4.2.2 Confusion Matrix

Looking at the confusion matrix below figure 4.3 illustrates the performance within each class allowing us to gain a deeper understanding the CNN model.

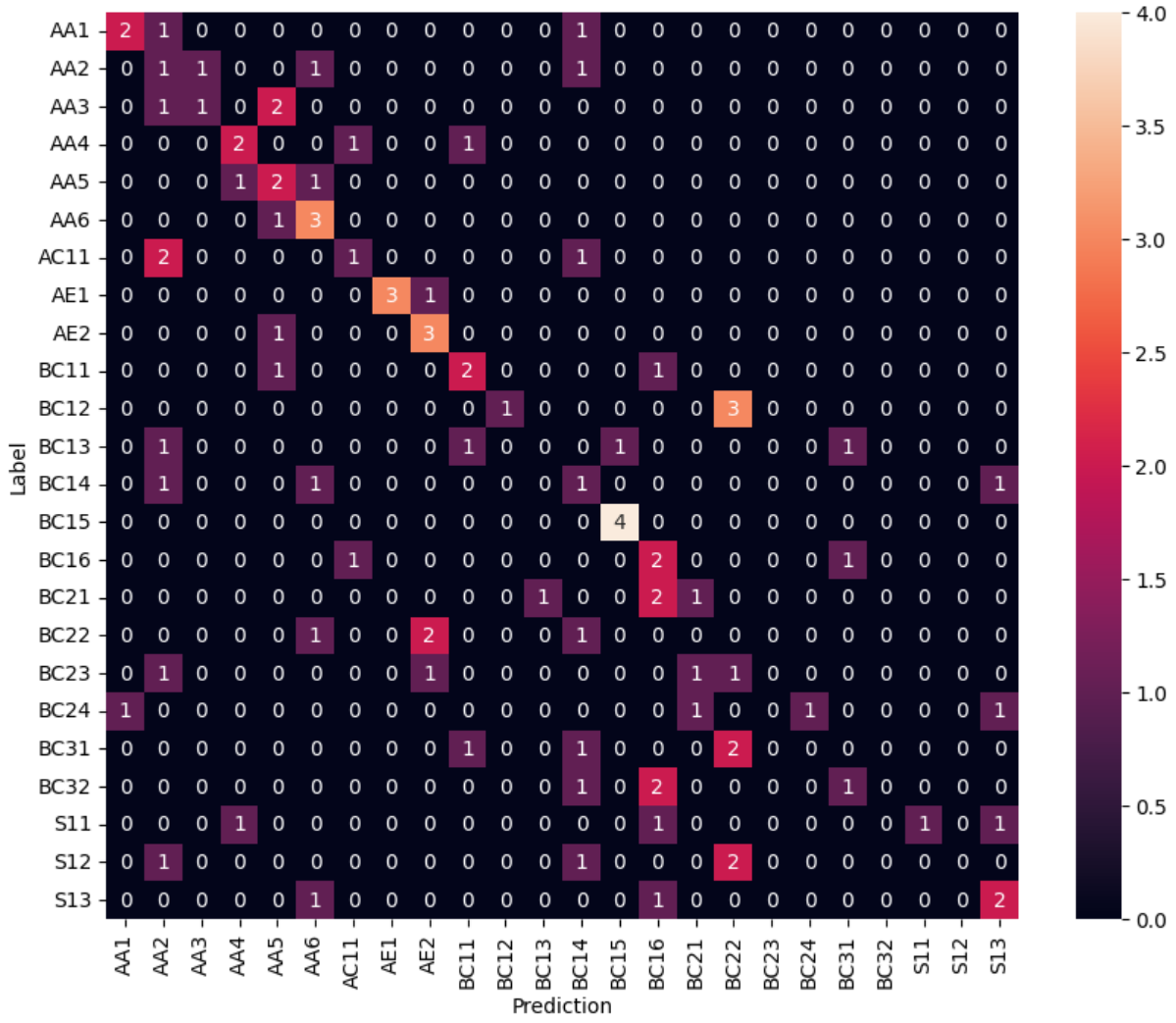

Figure 4.3: 288-Dataset Confusion Matrix based on Testing Images Dataset, 4 images per class. For the class labels, the first letter refers to the floor level (with the exception of stairs) A = ground floor and B = first floor. The second letter refers to the area, so A = Atrium, E = Entrance, C = Corridor, S = Stairs. The 3rd and sometimes 4th character refers to the relative location within its area, so AA3 = 3rd step location in the atrium. For class names with 4 characters the 3rd character identifies a specific corridor and the 4th character specifies the relative location.

The confusion matrix in figure 4.3 shows the labelled or ground truth on the vertical axis and the predicted classes on the horizontal axis. The 288-dataset contained 4 test images per class. The ideal results seen in a robust model would show a sequence of a high numbers in a diagonal line starting from the intersection of AA1 with AA1 to the intersection of S13 with S13.

Most of the locations on level A seemed to classify better compared to many of the loca-

tions on level B, in particular the corridors. Whilst BC15 achieving 4/4, this appears to be an outlier, it is possible that writing on posters, notices or signs found in the scene, improved the classification score.

On closer examination, the performance at the start or end of a corridor was moderately better than in the middle of the corridor. This makes sense given the greater similarity between images taken along the length of the corridor whilst located in the middle portion of a corridor.

It is evident that many of the incorrect classifications for the corridors are spread out quite randomly among the other classes often confusing with other corridor classes, however the errors in the atrium and entrance are usually within their own rooms.

Interestingly the incorrect classification from the entrance or atrium classes to non atrium or non entrance classes far away, seemed to occur due to similar images of glass walls captured in corridor classes like AC11, BC11 or BC14.

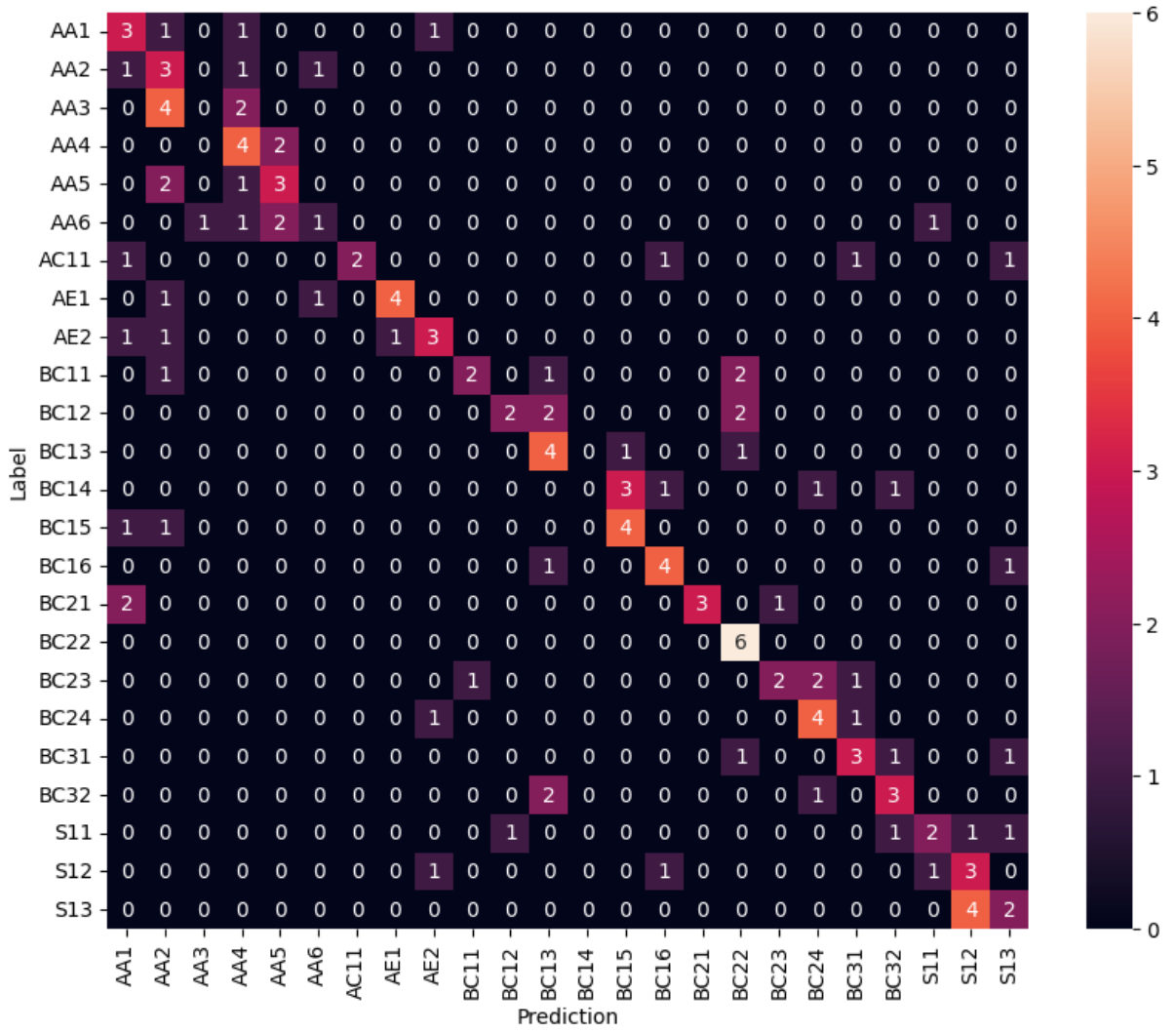

Figure 4.4: 576-Dataset Confusion Matrix based on Testing Images Dataset, 6 images per class. For the class labels, the first letter refers to the floor level (with the exception of stairs) A = ground floor and B = first floor. The second letter refers to the area, so A = Atrium, E = Entrance, C = Corridor, S = Stairs. The 3rd and sometimes 4th character refers to the relative location within its area, so AA3 = 3rd step location in the atrium. For class names with 4 characters the 3rd character identifies a specific corridor and the 4th character specifies the relative location.

Examining the confusion matrix in figure 4.4 is similar to figure 4.3 with the exception of an increase testing class size of 6 images. On general inspection the performance appears drastically better in both level A and level B classes, notably better in the corridors of level B.

The best score was in class BC22 with 6/6 images correctly classified. Whilst this may be an outlier in the data, this particular class contained clear fire signage in many of its

images suggesting that clear, color coded signs could improve performance.

On closer inspection class AA3 which incorrectly classified 4 images as AA2 and 2 images as AA4 (leaving 0/6 correctly classified) compared to class AA1, AA2 and AA4 which correctly classified 3/6, 3/6 and 4/6 respectively.

Given the symmetry and repetition present through many of the images in the atrium, the images in AA2 and AA3 look the same even to the human eye when viewing images taken approximately in front or behind of the user. Whilst the focus on increasing the viewpoint of images directly in front or behind the user may have had a negative effect for this class, it greatly improved the performance of many corridor classes found on level B. Compared to a 360 degree view of a location, the forward/backward view with a minimal degree rotation worked better.

4.3 Hidden Markov Model

Results of the hidden Markov model (HMM) are displayed using sequences.
Given the HMM was tested using four classes: AA1, AA2, AA3 and AA4. Using the implemented hidden Markov model class with the forward-backwards algorithm, a test observation sequence was inputted, outputting the most likely hidden sequence known as the uncovered sequence.
Two test cases were devised to verify the validity of this model. The first sequence followed a normal, control path through the atrium:

$$\texttt{AA1} \rightarrow \texttt{AA2} \rightarrow \texttt{AA3} \rightarrow \texttt{AA4}$$

The second sequence followed a path most likely to be chosen by the trained CNN model based on the probabilities described in the confusion matrix in figure 4.6:

$$\texttt{AA1} \rightarrow \texttt{AA2} \rightarrow \texttt{AA2} \rightarrow \texttt{AA4}$$

The state transition matrix 4.5 remained the same for both set of tests and was manually coded based on the most likely direction a user would transition between the states. At the start and end of the sequence it was assumed that the user would be more likely to move to the state next to it rather than stay in the same location. Middle state location were treated with equal probability of moving in either direction and a small probability (10%) of stay in the same location.

$$\begin{bmatrix} 0.20 & 0.80 & 0.00 & 0.00 \\ 0.45 & 0.10 & 0.45 & 0.00 \\ 0.00 & 0.45 & 0.10 & 0.45 \\ 0.00 & 0.00 & 0.80 & 0.20 \end{bmatrix}$$

Figure 4.5: State Transition Matrix

The first set of tests involve using the emission probability matrix derived from the confusion matrix values of 576-dataset CNN model (CM EPM).

$$\begin{bmatrix} 0.52 & 0.16 & 0.16 & 0.16 \\ 0.16 & 0.52 & 0.16 & 0.16 \\ 0.01 & 0.65 & 0.01 & 0.33 \\ 0.33 & 0.01 & 0.01 & 0.65 \end{bmatrix}$$

Figure 4.6: Emission Probability Matrix (CM EPM) based on 576-dataset confusion matrix

CM EPM Test case 1:

`Observation Sequence :` `[AA1, AA2, AA3, AA4]`

`Hidden Sequence :` `[AA1, AA2, AA3, AA4]`

`Uncovered Sequence :` `[AA1, AA3, AA2, AA3]`

CM EPM Test case 2:

`Observation Sequence :` `[AA1, AA2, AA2, AA4]`

`Hidden Sequence :` `[AA1, AA2, AA3, AA4]`

`Uncovered Sequence :` `[AA1, AA2, AA3, AA4]`

$$\begin{bmatrix} 0.52 & 0.16 & 0.16 & 0.16 \\ 0.16 & 0.52 & 0.16 & 0.16 \\ 0.16 & 0.16 & 0.52 & 0.16 \\ 0.16 & 0.16 & 0.16 & 0.52 \end{bmatrix}$$

Figure 4.7: Manually Coded Emission Probability Matrix (MC EPM)

Manually Coded EPM Test case 1:

`Observation Sequence :` `[AA1, AA2, AA3, AA4]`

`Hidden Sequence :` `[AA1, AA2, AA3, AA4]`

`Uncovered Sequence :` `[AA1, AA2, AA3, AA4]`

Manually Coded EPM Test case 2:

`Observation Sequence :` `[AA1, AA2, AA2, AA4]`

`Hidden Sequence :` `[AA1, AA2, AA3, AA4]`

`Uncovered Sequence :` `[AA1, AA2, AA3, AA4]`

As seen above, two test sets/versions of the emission probability matrix were tested with two possible sequence paths in the atrium. The first emission probability matrix was based on the results of the confusion matrix of the CNN but adapted on a smaller scale to the first four locations in the atrium: AA1, AA2, AA3, and AA4. The second emission probability matrix was manually coded giving a 52% probability that an observed class would be the same as its hidden class and a 16% probability that it could be any other hidden class.

With the CM EPM, inputting an observed sequence of [AA1, AA2, AA3, AA4] and a hidden sequence of [AA1, AA2, AA3, AA4] outputted [AA1, AA3, AA2, AA3] as the most likely hidden sequence (uncovered sequence). Given the 4-class confusion matrix in figure 4.6, it was clear that the most likely observed sequence to be outputted by the CNN based on the ground truth of [AA1, AA2, AA3, AA4] would be [AA1, AA2, AA2, AA4]. Since 65% of the time AA3 was incorrectly classified as AA2; the model doesn't leave much chance to confirm a correct sequence as seen in test case 1. So when [AA1,

AA2, AA2, AA4] was inputted as an observed sequence into the HMM with the CM EPM it correctly outputted [AA1, AA2, AA3, AA4] as this was the most likely hidden sequence.

The MC EPM outputted [AA1, AA2, AA3, AA4] as the most likely hidden sequence with both [AA1, AA2, AA3, AA4] and [AA1, AA2, AA2, AA4] as inputs for the observed sequence. With an equal weighting between each class, a majority would predict that an observed class is the same as a hidden class (observed AA3 is 52% chance it is hidden AA3). The MC EPM is balanced enough to account for more significant incorrect classifications in AA3 and AA4 but equally distributed between those possible incorrect options (16% each). This matrix allows for versatility by correcting mistakes from the CNN classifier as well as accepting correct sequences.

So far we have discussed constructing an emission probability matrix based on the trained probabilities or confusion matrix values of a CNN model or manually coding matrix values through expert opinion. A third option to produce emission probability matrix values involves extracting the spread of probabilities from the final output layer of the CNN model.

Chapter 5

Conclusion

In this project, it has been demonstrated that solving indoor localisation is a challenge when using a small dataset with a lightweight neural architecture and limited computational power. However the implementation and results of the hidden Markov model present an interesting solution to the issue of inter-class similarity.

By using locations as classes, the accuracy of the model is limited by the spatial extent of the training samples. Even the direction and angle that was taken in comparison with the 288-dataset and 576-dataset had a significant impact on the results. Another limitation of this project was whether the CNN would perform well under different lighting conditions as illumination changes was seen as a common issue in the literature. As well as changing environments and motion blur would also worsen the performance of the CNN.

The results displayed a 12.15% increase with the improved 576-dataset giving a strong emphasis on the particular angles of images taken in the data collection process and the size of the dataset on performance of the CNN. The findings of this project provide insight into the combination of using a CNN and HMM to accomplish a robust indoor localisation technique that could form the foundation of a novel indoor navigation system.

Chapter 6

Further Work

6.1 Convolutional Neural Network

Given the constrained dataset, a future project could involve fine tuning a pretrained network with the 288 or 576 dataset in order to increase accuracy. As suggested by [13] a fine tuning technique useful for smaller datasets is by attaching a linear classifier (SVM) to the output of the layer directly preceding the fully connected layer.

In addition to improving the CNN another objective could be to determine orientation by exploring the use of 3D models and creating a coordinate system to map out the environment.

6.2 Other Deep Learning Approaches

A couple of papers mentioned the use of Recurrent Neural Networks (RNN) or Long Short Term Memory (LSTM) as methods for better solving indoor localisation [14]. Given the fact that they they incorporate temporal and sequential information this may be useful in the context of navigation in terms of developing a memory and preventing confusion between locations that look similar and are not adjacent to eachother.

6.3 Hidden Markov Model

Larger test cases of sequences for the Hidden Markov model would be needed as well as expanding the number of possible states to 24 instead of 4 in order to observed how the model scales up. As mentioned at the end of chapter 4, another method of creating the emission probability matrix would be to input values directly from the spread of probabilities found in the fully connected layer over the 24 classes and comparing it with the manually inputted expert method.

Bibliography

[1] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, August 2017.

[2] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1):53, March 2021.

[3] Jing Guo, Shaobo Zhang, Wanqing Zhao, and Jinye Peng. Fusion of wifi and vision based on smart devices for indoor localization. In *Proceedings of the 16th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*, VRCAI '18, pages 1–8, New York, NY, USA, December 2018. Association for Computing Machinery.

[4] Anca Morar, Alin Moldoveanu, Irina Mocanu, Florica Moldoveanu, Ion Emilian Radoi, Victor Asavei, Alexandru Gradinaru, and Alex Butean. A Comprehensive Survey of Indoor Localization Methods Based on Computer Vision. *Sensors*, 20(9):2641, January 2020. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute.

[5] Dawar Khan, Zhanglin Cheng, Hideaki Uchiyama, Sikandar Ali, Muhammad Asshad, and Kiyoshi Kiyokawa. Recent advances in vision-based indoor navigation: A systematic literature review. *Computers & Graphics*, 104:24–45, May 2022.

[6] Sean R. Eddy. What is a hidden Markov model? *Nature Biotechnology*, 22(10):1315–1316, October 2004. Number: 10 Publisher: Nature Publishing Group.

[7] Priya Roy and Chandreyee Chowdhury. A Survey of Machine Learning Techniques for Indoor Localization and Navigation Systems. *Journal of Intelligent & Robotic Systems*, 101(3):63, March 2021.

[8] Ramon F. Brena, Juan Pablo García-Vázquez, Carlos E. Galván-Tejada, David Muñoz-Rodriguez, Cesar Vargas-Rosales, and James Fangmeyer. Evolution of Indoor Positioning Technologies: A Survey. *Journal of Sensors*, 2017:e2630413, March 2017. Publisher: Hindawi.

[9] Faheem Zafari, Athanasios Gkelias, and Kin K. Leung. A Survey of Indoor Localization Systems and Technologies. *IEEE Communications Surveys & Tutorials*, 21(3):2568–2599, 2019. Conference Name: IEEE Communications Surveys & Tutorials.

[10] Yongliang Sun, Weixiao Meng, Cheng Li, Nan Zhao, Kanglian Zhao, and Naitong Zhang. Human Localization Using Multi-Source Heterogeneous Data in Indoor Environments. *IEEE Access*, 5:812–822, 2017. Conference Name: IEEE Access.

[11] Yassine Ouali, Céline Hudelot, and Myriam Tami. An Overview of Deep Semi-Supervised Learning, July 2020. arXiv:2006.05278 [cs, stat].

[12] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, December 2022. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[13] Piotr Wozniak, Hadha Afrisal, Rigel Galindo Esparza, and Bogdan Kwolek. Scene Recognition for Indoor Localization of Mobile Robots Using Deep CNN. In Leszek J. Chmielewski, Ryszard Kozera, Arkadiusz Orłowski, Konrad Wojciechowski, Alfred M. Bruckstein, and Nicolai Petkov, editors, *Computer Vision and Graphics*,

Lecture Notes in Computer Science, pages 137–147, Cham, 2018. Springer International Publishing.

[14] Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-Based Localization Using LSTMs for Structured Feature Correlation. pages 627–637, 2017.

[15] Shibo Han, Minhaz Uddin Ahmed, and Phill Kyu Rhee. Monocular SLAM and Obstacle Removal for Indoor Navigation. In *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*, pages 67–76, December 2018.

[16] Wei Zhao, Liangjie Xu, Bozhao Qi, Jia Hu, Teng Wang, and Troy Runge. Vivid: Augmenting Vision-Based Indoor Navigation System With Edge Computing. *IEEE Access*, 8:42909–42923, 2020. Conference Name: IEEE Access.

[17] G. S.T. Perera, K. W.R. Madhubhashini, Dilani Lunugalage, D. V.S. Piyathilaka, W. H.U. Lakshani, and Dharshana Kasthurirathna. Computer Vision Based Indoor Navigation for Shopping Complexes. In *Proceedings of the 2020 4th International Conference on Vision, Image and Signal Processing*, ICVISP 2020, pages 1–6, New York, NY, USA, March 2021. Association for Computing Machinery.

[18] Abdel Ghani Karkar, Somaya Al-Maadeed, Jayakanth Kunhoth, and Ahmed Bouridane. CamNav: a computer-vision indoor navigation system. *The Journal of Supercomputing*, 77(7):7737–7756, July 2021.

[19] Bobai Zhao, Dali Zhu, Tong Xi, Chenggang Jia, Shang Jiang, and Siye Wang. Convolutional neural network and dual-factor enhanced variational Bayes adaptive Kalman filter based indoor localization with Wi-Fi. *Computer Networks*, 162:106864, October 2019.

[20] Aoran Xiao, Ruizhi Chen, Deren Li, Yujin Chen, and Dewen Wu. An Indoor Positioning System Based on Static Objects in Large Indoor Scenes by Using Smartphone Cameras. *Sensors*, 18(7):2229, July 2018. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute.

[21] Wei Zhang, Guoliang Liu, and Guohui Tian. A Coarse to Fine Indoor Visual Localization Method Using Environmental Semantic Information. *IEEE Access*, 7:21963–21970, 2019. Conference Name: IEEE Access.

[22] Orhan Akal, Tathagata Mukherjee, Adrian Barbu, Jared Paquet, Kevin George, and Eduardo Pasiliao. A Distributed Sensing Approach for Single Platform Image-Based Localization. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 643–649, December 2018.

[23] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, April 2015. arXiv:1409.1556 [cs].