Training and Deploying Computer Vision Models for Indoor Localisation

Mika Senghaas
IT University of Copenhagen
jsen@itu.dk

A Thesis presented for the Degree of
Bachelor of Science in Data Science

IT UNIVERSITY OF CPH

IT University of Copenhagen
Computer Science Department

May, 15th 2023

Contents

Abstract

In our increasingly urbanised world, indoor localisation is becoming a necessity for a wide variety of applications, ranging from personal navigation to augmented reality. However, despite extensive research efforts, indoor localisation remains a challenging task and no single solution is widely adopted. Motivated by the success of deep learning in numerous computer vision tasks, this study explores the feasibility of deep learning for accurate room-level localisation in indoor spaces. Various neural network architectures are trained and evaluated on a novel video dataset tailored for indoor localisation. The findings reveal that deep learning approaches can provide reasonable localisation results, even when trained on a small dataset. The approach is currently limited by its inability to distinguish between visually similar and adjacent areas, as well as biases within the training data. Despite these shortcomings, the results are encouraging and inspire optimism about the method's practical viability.

1 Introduction

With the introduction of the satellite-based Global Positioning System (GPS), localisation in outdoor spaces has become more efficient and accurate than ever before. Gradual commercialisation led to the technology rapidly transforming industries and personal navigation. Today, outdoor localisation is widely considered a *solved problem*.

The same cannot be said for indoor localisation. Because the transmitted radio signals sent out by the satellites in GPS systems are not strong enough to penetrate through walls and struggle with reflections from large buildings, the technology yields inaccurate results at best, and often becomes dysfunctional in indoor spaces [15, 21].

With the ongoing urbanisation and the emergence of autonomous robots and vehicles, the need for indoor localisation technologies is growing. Over the past decade, a wide variety of solutions have been proposed. Infrastructure-based systems use radio signals transmitted by beacons, like Bluetooth [4, 6], Ultra-Wideband (UWB) [2, 3] or Wi-Fi [15, 21, 31], to localise an agent in a known environment. Infrastructure-less systems, like simultaneous localisation and mapping (SLAM) algorithms, rely solely on sensors, like cameras [7, 22, 28] or distance-measuring lasers [30] to localise an agent in an unknown environment.

While these approaches have produced remarkable results, localising agent's with centimetre accuracy, they are limited for various reasons: Infrastructure-based systems require initial setup and maintenance of the installed hardware, which makes them costly, time-intensive and difficult to implement in large environments. Infrastructure-less systems, on the other hand, require complex processing of sensory information and need to be fine-tuned by experts for each indoor space, to achieve outstanding results. These limitations have prevented existing solutions from being widely adopted in use-cases like personal navigation in indoor spaces. Previous approaches were designed assuming centimetre accuracy is categorically required. However, not all use-cases require such a level of precision. For example, in a museum or shopping mall, it might be sufficient to know in which area a visitor is. In these cases, the constraint of centimetre accuracy can be relaxed, in favour of a simpler and more versatile solution.

Deep learning, which is part of a broader family of machine learning methods, has recently gained a lot of attention in the field of computer vision and proven to be a powerful tool for solving a wide variety of tasks. Common computer vision tasks are image and video classification, where the goal is to predict a label from a set of pre-defined labels for a given image or video.

Given this, it is natural to ask (a) whether indoor localisation can be phrased as a coarse-grained classification task, where labels correspond to areas in an indoor space, and (b) whether deep learning

techniques can be used to produce accurate localisation results in this setting. Therefore, this study investigates the applicability of modern deep learning techniques to indoor localisation. The main contributions can be summarised as:

1. A novel, small single-frame and video classification dataset for indoor localisation based on 40 minutes of video footage in 20 different rooms.
2. A rigorous evaluation of several modern deep learning architectures for the task of indoor localisation, when viewed as a classification task.
3. A discussion of the results and an outlook on the applicability of a pure deep learning pipeline for indoor localisation.

2 Background

Phrasing the problem of indoor localisation as a classification problem and solving it with a pure deep learning approach requires a brief introduction to some of the fundamentals that underlie the methods used in this study. This section, therefore, introduces the fundamental concepts of deep learning relevant to this study.

2.1 Fundamentals of Machine and Deep Learning

Machine learning is a subfield of artificial intelligence (AI) that describes a series of techniques and algorithms that allow computers to learn from data without being explicitly programmed. One example of a machine learning task is classification, which aims to assign a discrete label $\hat{y} \in \{y_1, \ldots, y_n\}$ to an input x. The mapping from the input x to the label $\hat{y}$ is called a classifier, and often denoted as $\hat{f}(x) = \hat{y}$. Typically such a classifier is trained on a large set of labelled data, called the training set, which consists of true instances x_i and their labels y_i. Using numeric optimisation algorithms, like gradient-descent, the classifier iteratively improves its approximation $\hat{f}$ of the true mapping $f(x) = y$ by minimising a loss function $\mathcal{L}(\hat{f}(x), y)$ that quantifies the error of the machine learning model.

Deep learning is a subfield of machine learning, and describes a specific class of machine learning algorithms that are based on the theory of artificial neural networks (ANNs). ANNs are inspired by and loosely related to the structure and functioning of the neurons in the human brain. They are structured in a series of fully-connected layers, where each layer consists of nodes. Figure 1 shows an ANN with an input layer with three nodes, three hidden layers with seven nodes each and an output layer with three nodes. Information flows from the input layer through the hidden layers to the output layer. The information flow happens through sequential linear transformations of the input data, which are performed by the nodes in the network. Specifically, each node's output is a linear transformation of the outputs of all nodes in the previous layer,

$$z_i = \sum_{j=1}^{n} w_{ij} x_j + b_i \tag{1}$$

where x_j is the output of the j-th node in the previous layer, w_{ij} is the weight of the connection between the j-th node in the previous layer and the i-th node in the current layer, b_i is the bias of the i-th node in the current layer, and z_i is the output of the node.

Before the output z_i is passed to the next layer, it is transformed by a differentiable, non-linear activation function σ to produce an activation $a_i = \sigma(z_i)$. Once all activations in the j-th layer are

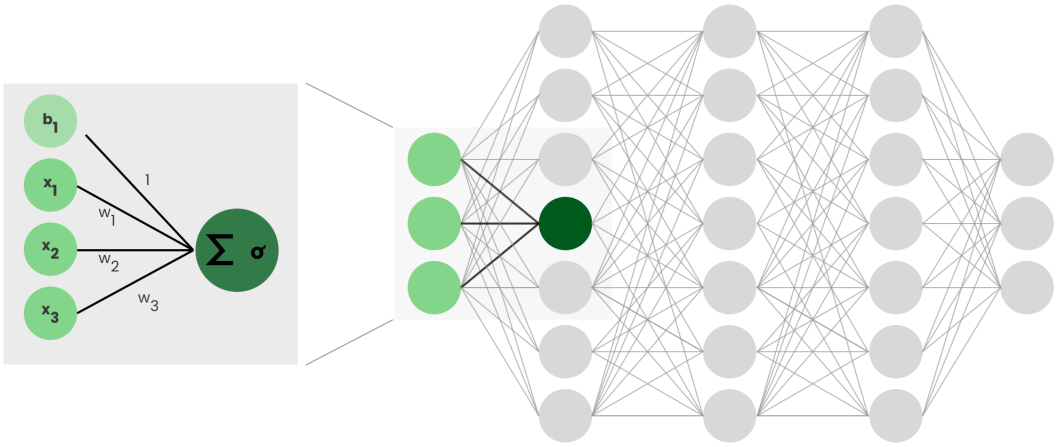


Figure 1: **Artificial Neural Network.** A schematic of the fundamental building blocks of an artificial neural network (ANN). The right figure shows the macro structure of an exemplary ANN with an input layer, three hidden layers, and an output layer. The left figure shows the micro structure of a single node in the network. The node performs a linear transformation of the inputs x_i and the weights w_i and adds a bias b_i. The output of the node is the result of a non-linear activation function σ applied to the linear transformation.

computed, the next layer's activations can be computed in the same way. This process, referred to as forward propagation, is iteratively repeated until the activations in the output layer are computed, which are the final outputs of the network.

Critically, the linear transformations performed by each node are parametrised by weights, which are optimised during training. This allows ANNs to learn complex non-linear mappings given enough samples of the input-output relationship without explicitly programming the mapping. The generality and scalability of the method have proven ANNs to be a powerful tool for modelling complex data relationships in a wide variety of domains.

2.2 Image Classification

Image classification is one of the most fundamental and widely studied tasks in computer vision and describes the process of assigning a label $y \in \{y_1, \ldots, y_n\}$ to an image x. Extracting information from images to assign a label is not straight-forward, because of the high-dimensional and unstructured nature of images. These characteristics make it challenging for traditional heuristic-based algorithms to extract meaningful information, which has long limited the capabilities of computer vision systems. However, with the advent of deep learning and the introduction of a special type of neural network, called convolutional neural network (CNN), this has changed.

CNNs are a type of neural network specifically designed to process visual information. Traditionally, they are organised hierarchically and consist of convolutional, pooling and fully-connected layers, as shown in Figure 2. Convolutional layers are the core of CNNs and are responsible for extracting features from the input. Each convolutional layer consists of a set of filters, where each filter, sometimes called kernel k, is a three-dimensional matrix of weights with dimensions $c_i \times h_k \times w_k$,

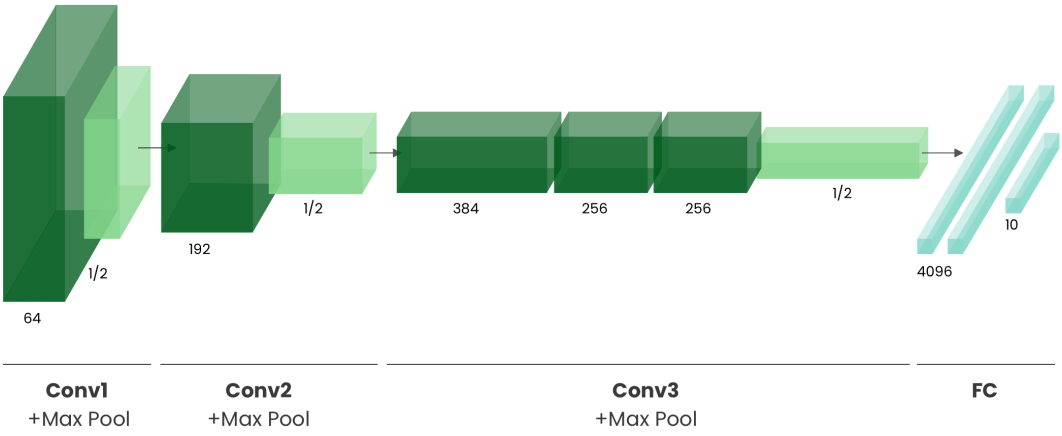


Figure 2: **Convolutional Neural Network.** A schematic of a traditional convolutional neural network (CNN), designed for image classification (here: AlexNet [23]). The network consists of a series of 2D convolutional layers, pooling layers and fully-connected layers, whose outputs are displayed as dark-green, light-green and blue boxes, respectively.

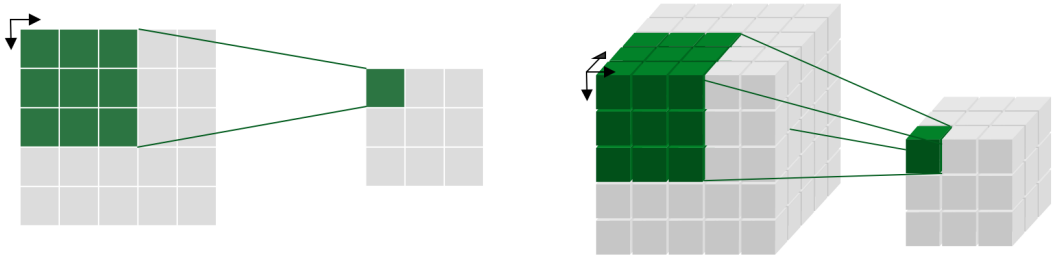
where c_i is the number of channels in the input, and h_k and w_k are the height and width of the filter. Given an input x with dimensions $c_i \times h_i \times w_i$, a single convolutional filter k produces a feature map z by sliding the filter across the spatial dimensions of the input, as shown in Figure 3a, and computing the convolution at each position i, j (Equation: 2).

$$z_{i,j} = \sum_{c=1}^{c_i} \sum_{m=1}^{h_k} \sum_{n=1}^{w_k} k_{c,m,n} x_{c,i+m,j+n} \tag{2}$$

The output z is a two-dimensional matrix, called feature-map. Within a single convolutional layer, multiple filters are applied to the input, which results in multiple feature-maps. After applying a non-linear activation to each feature map, they are stacked along the channel dimension to form the output of the convolutional layer, which is passed to the next layer. Pooling layers are often interleaved with convolutional layers to reduce the dimensionality of feature maps. The pooling operation is similar to convolution, as a kernel is slid However, instead of computing a weighted sum, the pooling operation computes a statistic, such as the maximum (Max Pooling) or average (Average Pooling), of the values in the kernel.

CNNs have been found to work specifically well for data with a spatial structure such as images, because convolutions model the inherent nature of images: While a stand-alone pixel is not as informative, the value of a pixel in the context of its neighbouring pixels is. This characteristic is naturally captured by the convolution operation. Furthermore, in sliding filters across the input, they can capture the same feature independently of its location in the image, which makes CNNs invariant to translation. This is a desirable property for image classification, as the location of features in an image may not be relevant for the final classification.

The first CNN-based architectures date back to the 1960s and have shown successes in simple image classification tasks [24]. However, it was not until 2012, when the CNN-based architecture AlexNet [23] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [8], that CNNs arrived in the mainstream of computer vision. Since then CNNs have been proposed in various dif-



(a) 2d Convolution (b) 3D Convolution

Figure 3: **2D and 3D Convolution.** A simplified illustration striding of 2D and 3D convolutional filters across an input. For visualisation purposes, a single channel is shown. A 2D convolutional filter (a) slides across the spatial dimension (height and width) of an input. A 3D convolutional filter (b) slides across the spatio-temporal dimension (height, width, and time) of an input.

ferent forms, mostly differing in the network's structure and depth and the width and resolution of the filters. To this day, CNNs still rank amongst the top performing methods in image classification benchmarks.

2.3 Video Classification

Video classification can be seen as a generalisation of image classification: Instead of assigning a label to a single image, video classifiers assign a label y or a sequence of labels $y_1, \ldots, y_t$ to a video sequence $x = (x_1, \ldots, x_T)$, where x_t is a frame of the video.

The difference is subtle, yet important, because it introduces a temporal dimension to the task. It is generally assumed that the temporal dimension is critical for video classification, because it provides additional information about the video. For example, it might be difficult to determine whether a person is running or walking from a single frame, but trivial given the motion captured by a sequence of frames.

Motivated by the need for a powerful model to capture the semantic content of video data and the success of CNNs in image classification, researchers have started to apply CNNs to video classification [19, 5, 34, 11, 12], where the networks have access to the complex spatio-temporal evolution of the video. Over the course of the last decade, various approaches based on CNNs have been proposed that exhibit different connectivity patterns for modelling the temporal dimension of the video [19].

A naive solution, which ignores the temporal dimension entirely, is to predict a label for each frame and then average the predictions. Despite their simplicity, these models have been shown to perform surprisingly well [19]. Different methods for aggregation have been proposed, such as averaging [19], majority votes, or using a neural architectures, such as recurrent neural networks (RNNs) to learn the importance of each frame [9] for the final prediction.

CNN-based architectures that model the temporal dimension directly usually leverage 3D convolutions [34, 5]. 3D convolutions are a natural extension of 2D convolutions, as defined in Section 2.2. Instead of sliding a convolutional filter across the spatial dimension of the input (Figure 3a), a 3d-convolutional filter slides across the spatio-temporal dimension of the input (Figure 3b), which

produces feature maps in spatio-temporal space that are learned jointly. 3D convolutions have been used in many different variants. Architectures range from pure 3D convolution networks [5, 34] to hybrid architectures that combine 2D and 3D convolutions [11, 12]. Overall, the literature on video classification is vast and complex. Despite the introduction of large-scale datasets for video classification [20], the research field has not yet converged to a consensus about the best approach for video classification.

3 Methodology

In this light, this study aims to understand how different CNN-based architectures perform when faced with the task of assigning location labels to a continuous stream of frames, a video, in an indoor space.

To this end, the study considers a wide-variety of different models that are detailed in Section 3.3. One major distinction between the different models, is whether or not they operate on a single-frame or a sequence of frames. Because this distinction impacts all steps of the study, from data processing to model evaluation, the following two different problem settings are distinguished throughout the entire study:

1. **Single-frame classification**: Given a continuous stream of frames, the task is to classify each frame individually.
2. **Video classification**: Given a continuous stream of frames, the task is to classify fixed-sized clips of the video.

3.1 Raw Data

Framing the problem of indoor localisation as a classification task requires a labelled data set, which consists of sequentially-arranged pairs of inputs and outputs, that map visual information to location labels.

The raw data was collected from a single camera of a mobile device at a frame rate of 30 FPS in high resolution (2426×1125). The mobile device was hand-held by a human agent while walking around the main building of the Southern Campus of the Copenhagen University (Danish: Københavns Universitet, KU) in Copenhagen S, Denmark. The large multi-storey building consists of a total of six floors. The location was deemed compatible with this study, as it showcases distinctive learnable indoor features (e.g. coloured walls, architectural unique structures, etc.), but also challenges the model, for example, due to areas that visually similar to each other (like libraries and corridors). For the scope of this project, the data collection was limited to the first two floors. This process yielded a set of videos $V = \{v_1, ..., v_n\}$, where each video v_i is a sequence of k frames $f_1, ..., f_k$. Each frame f_i is a RGB image with dimensions $3 \times 2426 \times 1125$.

Each video v_i is associated with a set of location labels $L = \{l_1, ..., l_n\}$, where each location label l_i identifies the location of the agent at a specific time in the video. For the scope of this project 20 different location labels were considered, which are identified by a descriptive name and integer. Figure 4 shows the floor plan of the first two floors of the building, where each location class is mapped to a coloured region. The location labels were assigned in close correspondence to the original floor plan. This, however, led to some classes being a lot larger than others, which is likely to result in a class imbalance. Annotation was performed manually by a single human agent. Because changes in the location labels only occur at the transition of rooms, the annotation process was simplified by

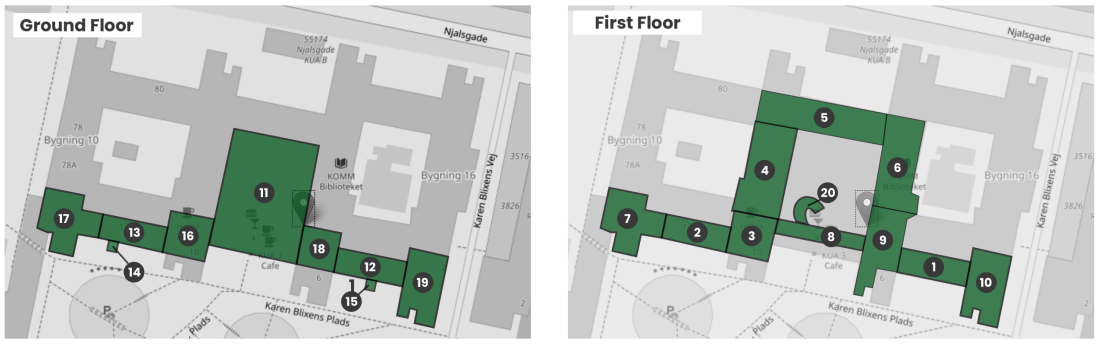


Figure 4: **Location Labels.** The figure shows the floor plan for the ground floor (left) and first floor (right) of the main building of the Southern Campus of the Copenhagen University. The 20 location labels considered in this study are numbered and mapped to a coloured region on the floor plan.

annotating the starting and ending time stamps of a location label, which were later pre-processed to frame-by-frame annotations.

A total of $n = 53$ videos of varying length were recorded, with an average duration of ~ 57s, amounting to a total number of ~ 50 minutes of footage, or an equivalent of $\sim 90K$ frames. Out of the total 53 videos that were recorded, 37 were used for training and 16 were used for testing. Importantly, the videos in the training split were recorded in a single session, while the videos in the test split were recorded on four separated days, in a span of two to four weeks after the training data had been recorded. This was done to ensure that the models were tested against unseen data, to more accurately assess their generalisation capabilities. Indeed, the test data was recorded in different weather conditions, at different times of the day, with different lighting conditions and, by pure chance, one of the areas was repainted during the time between the recording of the training and test data. With all these changes in mind, the test data is expected to be as different from the training data, as it would be in a real-world scenario.

3.2 Processed Data

3.2.1 Single-Frame Dataset

Single-frame classification models expect a single frame f_i from a video v_i as input. Technically, all frames in a video v_i could be used as input, but because of the strong local correlation between adjacent frames, it was hypothesised that models would overfit to the training data. For this reason, and in an attempt to assimilate the single-frame and video datasets, frames from a video v_i were sampled at a sampling rate fixed r_f. The sampling rate is not tied to the architecture of single-frame models and was therefore chosen globally to be $r_f = 5$ for the training split. This means that only every 5th frame from the videos in the raw data split were used for training. A sampling rate was not adopted for the testing split, in order to test the model on the full set of frames.

Matching the location labels to each of the extracted frames was straight-forward. For each frame f_i in a video v_i, the location label l_i was found as the label currently active at the time of the frame.

Figure 5: **Data Extraction.** A raw video with $k = 10$ frames is processed into $k_f = 5$ frames for the single-frame dataset and $k_v = 2$ clips for the video dataset. Samples for each dataset type are indicated by a surrounding box. Frames are sampled at a rate of $r_f = 2$ for the single-frame dataset and $r_v = 3$ for the video dataset. A clip contains $s_v = 2$ frames. Frames are colour-coded in correspondence to the dataset they belong to (single-frame dataset in purple and video dataset in blue), if they are sampled to the respective dataset. Discarded frames are coloured in grey (light-grey for within video frames, dark-grey for trailing frames).

This was done by finding the location label l_i that satisfied condition $t_{l_i} \leq t_{f_i} \leq t_{l_{i+1}}$, where t_{l_i} and t_{f_i} are the time stamps of the location label l_i and frame f_i, respectively. This process yielded a set of single-frame samples $D_F = \{(f_1, l_1), ..., (f_{n_f}, l_{n_f})\}$ with $n_f =\sim 12K$ for the training split and $n_f = 20K$ for the testing split.

3.2.2 Video Dataset

A video classification models expects a single clip c_i as input, which is a fixed-sized sequence of s_v frames sampled from a video v_i at a sampling rate r_v. For this reason, the video classification dataset is constructed by extracting clips from the videos in the raw data split. Clips are extracted in uniformly spaced intervals. Within this study only non-overlapping clips were considered, meaning that in a single clip c_i, the location label l_i is the same for all frames $f_i \in c_i$. This is a simplification of the problem, as it is possible for a person to move from one location to another within a single clip, but was found more compatible with existing video classification architectures. In reality, the simplification is viable, because during the relatively small clip duration of ~ 2s, transitions between locations only occur in a small subset of clips.

Because clips do not overlap multiple location labels, each clip c_i can be associated with a single location label l_i. The location label l_i is found as the location label present during the entire time span of the clip. This process yielded a set of clip-location samples $D_V = \{(c_1, l_1), ..., (c_{n_v}, l_{n_v})\}$.

3.3 Models

Following the literature on single-frame and video classification tasks, a twelve different models were trained and evaluated in this work. Following the distinction made in Section 3, the models are grouped into single-frame and video classification models.

Table 1 gives a comprehensive overview of all models. The table shows the release year, the sampling rate r_f for single-frame classification models and the sampling rate r_v and clip size s_v for video classification models. The table also shows the input size, the number of parameters and the number

Table 1: **Model Overview.** The table shows all models that were evaluated in this work. The models are split into two categories: single-frame models and video models. For each model, the table reports the release year (Release), the frame rate (Rate) of the training data, the number of frames per clip (F/C; *only applicable to video classifiers*), the spatial resolution (Size) of the input images, the number of parameters in millions (Params), the number of floating point operations in billions (FLOPs) and the benchmark Top-1 accuracy (Acc1) on ImageNet [8] for single-frame classification models and the top-1 accuracy on the Kinetics [20] dataset for video classification models. The table is sorted by release date within each group.

	Model	**Release** (Y)	**Rate** (r_f/r_v)	**F/C** (s_v)	**Size** (h, w)	**Params** (M)	**FLOPs** (G)	**Acc@1** (%)
Single-Frame	AlexNet [23]	2012	5	-	224	61.1	0.71	56.52
	ResNet18 [14]	2015	5	-	224	11.7	1.81	69.76
	ResNet50 [14]	2015	5	-	224	25.6	4.09	76.13
	DenseNet 121 [17]	2016	5	-	224	7.0	2.88	74.43
	MobileNet V3 [16]	2019	5	-	224	3.5	0.32	71.88
	ViT-B-16 [10]	2020	5	-	224	86.7	17.56	81.07
	EfficientNet V2 S [32]	2021	5	-	224	21.5	8.37	84.23
	ConvNext Tiny [26]	2022	5	-	224	28.2	4.46	82.52
Video	R(2+1)D [35]	2018	4	16	182	28.11	76.45	76.01
	Slow R50 [12]	2018	8	8	224	32.45	54.52	74.58
	SlowFast R50 [12]	2018	8	8	224	34.57	65.71	76.94
	X3D S [11]	2020	6	13	182	3.5	2.96	73.33

of floating point operations (FLOPs) for a single forward pass. Finally, the table shows the top-1 accuracy on the ImageNet dataset [8] for single-frame classification models and the top-1 accuracy on the Kinetics dataset [20] for video classification as an indicator of the model's performance on a generic classification task.

The following section briefly describes the main architectural features of each model.

3.3.1 Single-Frame Classifiers

Alexnet [23] is a convolutional neural network that was introduced by Krizhevsky *et al.* in 2012. It was the first deep neural network to win the ImageNet Large Scale Visual Recognition Challenge [8] and is considered to be one of the first successful application of deep learning to image classification. Architecturally, it consists of five convolutional layers with occasional max-pooling layers in between, followed by three fully connected layers, as illustrated in Figure 6.

After the success of AlexNet, the speed of research in the field of deep neural networks significantly picked up. Networks with increasing depth, such as VGG [29], were found to perform better, but led to new research problem: As information about the input or gradient passes through many layers, it can vanish during gradient-descent based learning, leading to a stagnation during training. The phenomenon of "vanishing gradients" was visible in larger training errors for deeper networks, and coined the "degradation" problem [14].

The **ResNet** [14] architecture is considered a cornerstone of modern deep learning history as it introduced the concept of residual connections. In a residual subnetwork of L layers H_l, the output x_l is computed as the sum of the input to the subnetwork x_{l-1} and the output of the subnetwork

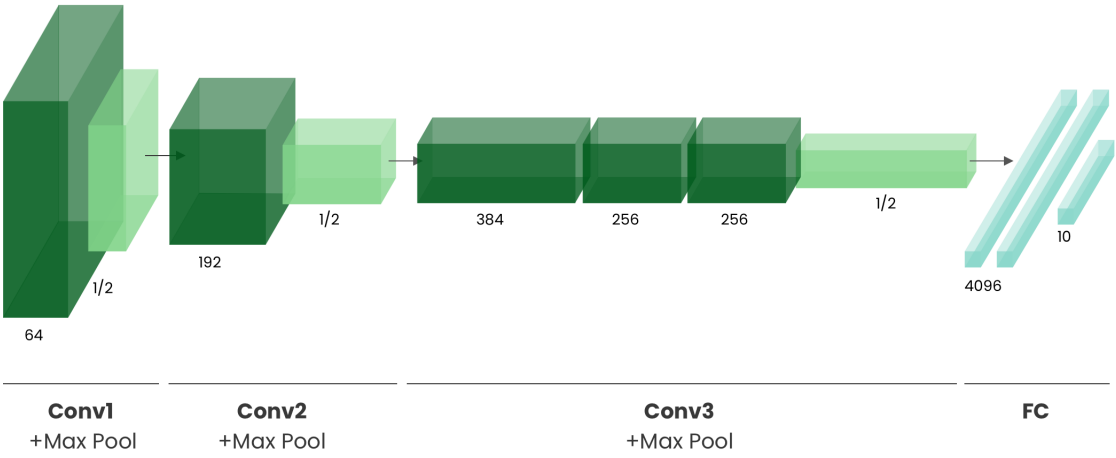


Figure 6: **AlexNet Architecture.** An illustration of the architecture of AlexNet [23]. Each box represents the output of a layer. Dark-green boxes are outputs of convolutional layers, light-green boxes are outputs of max-pooling layers and blue boxes are outputs of fully connected layers. The number of channels is denoted below each convolutional layer. Spatial downsampling is only performed by the first convolutional layer and all pooling layers. The downsampling factor is denoted below the pooling boxes.

$H_l(x_{l-1})$, i.e. $x_l = x_{l-1} + H_l(x_{l-1})$. The introduction of this identify mapping to the output of the subnetwork was shown to facilitate signal propagation in forward and backward paths, and, in connection with batch normalisation [18], allowed to train networks of unprecedented length. The original paper introduced several architectures with different depths, but in this work only ResNet18 and ResNet50, with 18 and 50 layers respectively, are considered.

DenseNets [17] extend the idea of residual connections. At the core of the DenseNet architecture is the concept of "dense blocks": Similar to a residual block, it is a subnetwork of L layers denoted as H_l. Unlike the residual block, each layer in the dense block receives the concatenation of all preceding layer outputs as an input, such that $x_l = H_l([x_0, x_1, \ldots, x_{l-1}])$. Dense blocks are compute intensive, but the strong connectivity between layers support feature reuse, which allowed the architecture to be shallower than predecessors. Again, the authors introduced several architectures with different depths. Here, only DenseNet 121 is considered.

MobileNet V3 [16] is the third iteration of the MobileNet architecture, which was introduced by Howard *et al.* 2019. The main contribution of MobileNets are so-called depth-wise separable convolutions. A depth-wise separable convolution factorises a regular 2D convolution into two separate steps: First, a single 2D filter is applied to each input channel separately, and then a 1x1 convolution is applied to the output of the previous step. This architectural change makes MobileNets significantly more efficient with only minor performance losses, which made them popular for application on low-compute devices, like mobile phones. MobileNet V3 is the most recent iteration of the architecture, and its smallest variant, MobileNet V3 Small, is used in this work.

EfficientNet V2 [32], proposed by Tan *et al.* in 2021, is searching for a compute-optimal CNN architecture. The main finding of the paper is a scaling law, which states that for a given baseline network, e.g. scaling up a network's depth, width and resolution with a constant factor leads to improved performance and efficiency. In the original paper the authors scale up previous state-of-the-art models like ResNet and MobileNet, which showcase state-of-the-art performance, which make up

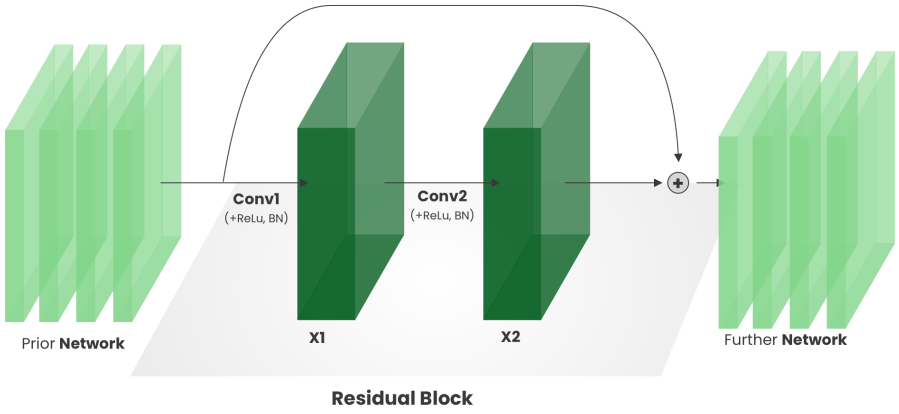


Figure 7: **Residual Connection.** An illustration of a generic residual block, as defined by He *et al.* [14]. A subnetwork of $L = 2$ layers (consisting of convolution, batch normalization and ReLU layers) is applied to an input x_0. The first layer produces an output x_1, and the second layer an output x_2. The output of the residual block is given by $x_2 + x_0$ through the skip connection.

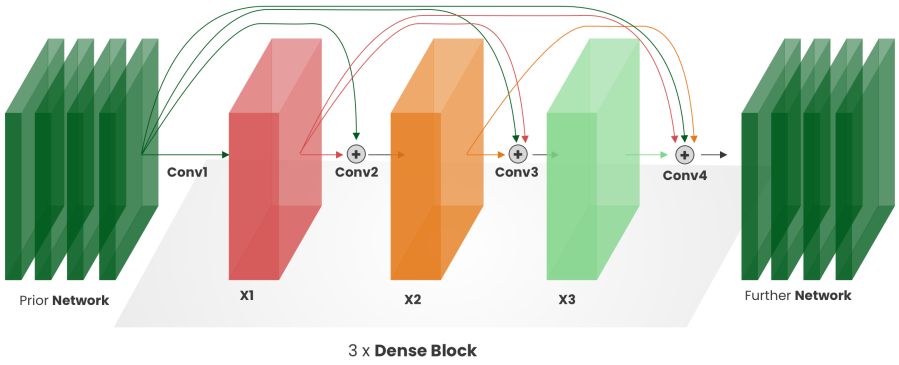


Figure 8: **Dense Block.** An illustration of a generic dense block, as defined by Huang *et al.* [17]. A subnetwork of $L = 4$ layers (consisting of convolution, batch normalization and ReLU layers) is applied to an input x_0. The i-th layer produces an output x_i. The input to the i-th layer is the concatenation of the output of all previous layers, so $H_i(x_i) = H_i([x_0, x_1, \ldots, x_{i-1}])$. The final output of the denseblock is the concatenation of the output of all layers, so $x_4 = [x_0, x_1, x_2, x_3]$.

a new family of models called EfficientNets. Here, EfficientNet V2 S is used, which is the smallest variant of the EfficientNet V2 family.

With the ground-breaking paper "Attention is all you need" [36], Vaswani *et al.* introduced the Transformer architecture in 2017. Although being initially designed for machine translation, the architecture was quickly adapted to other tasks in the domain of natural language processing, superseding previous state-of-the-art models on various benchmarks.

In 2020, **Vision Transformer** [10] was introduced as one of the first Transformer-based model adapted for the computer vision tasks. The main contribution of the model is the patch embedding mechanism, which overcomes the architectural mismatch between the two-dimensional input of images and the one-dimensional input of Transformer architectures. Dosovitskiy *et al.* propose to split a 2d image into a sequence of fixed-sized patches, which are then flattened and embedded into a sequence of "tokens". Together with positional encodings for the position of the patch in the original image, this sequence of tokens can be fed into a regular Transformer architecture. The authors introduce several configuration of the architecture. In this work, the base Vision Transformer, ViT-B-16, which is among the smaller variants of the model is used.

Since then, many variants of the Vision Transformer have been proposed. They either decrease the computational complexity of the architecture [33, 13], or improve the performance by recovering some of the inherent architectural advantages of convolutional neural networks [25]. However, these architectures are not considered in this work.

Finally, **ConvNext** [26] is one of the most modern convolutional neural network architectures considered in this work and was proposed by Liu *et al.* [26] in 2021. Against the trend of Transformer-based architectures in computer vision tasks, ConvNext is a pure convolutional architecture. ConvNext uses the ResNet 50 [14] architecture as a baseline and performs gradual modernisation steps, ranging from different optimisation algorithms, filter dimension to network depth and width. The authors show that the resulting convolutional architecture is competitive with the Transformer-based architectures, and coined in ConvNext.

3.3.2 Video Classifiers

Early attempts of using 3D convolutional filters for video classification tasks [5, 34] show potential for modelling the temporal dimension of video data. However, the authors also show that simple single-frame models perform surprisingly well, and that the temporal dimension only provides marginal gains. This finding and the fact that one would expect the temporal information and motion patterns that emerge in sequences of images to be important for video motivated Tran et. al to investigate different architectural designs for video classification tasks. They contrast regular 2d convolution, mixed 2d and 3d convolution, 3d convolution and 3d convolution with factorised filters, which they coin **R(2+1)D** [35]. They find that the R(2+1)D architecture improves the performance on the Kinetics dataset by a noticeable margin.

In the same year, Feichtenhofer *et al.* [12] proposed **SlowFast** [12], which introduces a two-stream architecture for video classification tasks. SlowFast has two separate pathways for processing video data. The first pathway, the slow pathway, processes the video data at a low frame rate, which allows it to capture the spatial information of the video data. The second pathway, the fast pathway, processes the video data at a high frame rate, which allows it to capture the temporal information of the video data. The authors show that the combination of the two pathways improves the performance over single-stream architectures. In this study, both the single-stream slow architecture **Slow R50** and the two-stream architecture **SlowFast R50** are considered.

Finally, the **X3D** architecture reviews the design choices of previous architectures that were proposed. The authors find that previous approaches differ mainly by varying the temporal dimension (frame rate and number of frames) and the spatial dimension (resolution, depth and width of filters). Jointly studying the effects of scaling the temporal and spatial dimensions, the authors propose a family of architectures that gradually scales up the depth, width and resolution of the network, while keeping the computational complexity constant. In this work, the smallest variant of the X3D family, X3D S, is used.

3.4 Training

All models were implemented using the deep learning framework `PyTorch` [1]. The model architectures were taken from the `torchvision` and `pytorchvideo` libraries for single-frame and video classification models, respectively and initialised with the default pre-trained weights provided by the libraries. For all single-frame pre-training was performed on the ImageNet dataset [8] and for all video classification models pre-training was performed on the Kinetics dataset [20]. Fine-tuning was then performed remotely on the high performance cluster (HPC; Appendix 6.2) of the IT University of Copenhagen using GPU-accelerated training.

All models were optimised to minimise the cross-entropy loss function $\mathcal{L}$ (Equation 3),

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=1}^{L} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) \quad , \tag{3}$$

where $\hat{\mathbf{y}}$ is the predicted probability distribution over the L location labels and $\mathbf{y}$ is the one-hot encoded ground truth location label for a single frame or video. The function penalises the model for predicting a low probability for the ground truth label. For the models to learn the relationship between frames/ clips and location labels, the gradient of the loss function with respect to all model parameters is computed and AdamW [27], the de facto standard optimiser for deep learning models, is employed to update the model parameters at a constant learning rate of 10^{-4}. No learning rate warm-up or scheduling was performed during training.

Mini-batch training was used for all models with a batch size of 16 for single-frame models and 4 for video classification models due to memory constraints. Because of the high computational complexity of 3D convolutions, the video classification models take longer to converge and were therefore trained for 15 epochs, while the single-frame models were trained for 10 epochs.

3.5 Evaluation

To assess all models in terms of their performance and efficiency on the task of location classification, a series of different performance and efficiency metrics are computed. All metrics are computed on the test set, which was separated from the original dataset before training, as described in Section 3.1.

Quantifying Performance. In the context of indoor location classification, a model is considered to perform well if it is able to accurately predict the location given a single frame or video clip. An intuitive way to quantify the performance of a model is to compute the Top-K multi-class accuracy:

$$\text{Top-K Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(y_i \in \mathcal{Y}_i) \quad , \tag{4}$$

Here, N is the number of samples, y_i is the ground truth label for sample i and $\mathcal{Y}_i$ is the set of the top-k predictions for sample i. The indicator function $\mathbb{I}$ is 1 if the ground truth label is in the set of top-k predictions and 0 otherwise. Within the context of this report, the Top-1 and Top-3 accuracy are computed, which are measuring the ratio of samples for which the true label is the top prediction, or within the top-3 predictions, respectively.

Furthermore the Macro F1-score is computed. The F1 score is a class-specific metric that measures the harmonic mean of precision and recall. The precision P_i of a classifier towards some class i can be interpreted as the probability that a sample that is classified as class i is actually from class i. The recall R_i is the probability that a sample from class i is classified as class i. High precision and recall values are desirable, so an ideal classifier would have a high precision and recall of 1. The F1-score is a way to combine both metrics into a single metric by computing their harmonic mean.

$$P_i = \frac{TP_i}{TP_i + FP_i} \quad , \quad R_i = \frac{TP_i}{TP_i + FN_i} \quad , \quad F1_i = \frac{2 \cdot P_i \cdot R_i}{P_i + R_i} \quad , \tag{5}$$

Finally, the Macro F1-score (Equation 6) is computed by averaging the F1-scores for each location label.

$$\text{Macro F1-score} = \frac{1}{L} \sum_{i=1}^{L} F1_i \quad . \tag{6}$$

Here, each location label is weighted equally, regardless of the number of samples that belong to the label. For the scope of this report the metric was found to be a good extension to the Top-K accuracy, as it gives insights into potential class imbalance issues.

Quantifying Efficiency. Efficiency is critical for real-time inference, especially on mobile devices. For this reason, direct proxies for a model's efficiency are computed using the `PyTorch Benchmark` library. The library allows to track various metrics during inference on a single CPU. The metrics that are computed are the number of floating point operations (FLOPs), mean inference time per sample (latency), the mean number of samples per second (throughput).

All benchmarks were computed on a desktop CPU. While the compute resources are likely to be different on a mobile device, the results still give a good indication of the relative efficiency of the models and allow to extrapolate the insights to performance on a mobile device. It is to be noted that latency and throughput are inversely proportional to each other: Less inference time per sample (low latency) leads to a higher number of inferences per second (high throughput). However, as inference times vary greatly between models, it is useful to compute and visualise both metrics to accentuate either high or low throughput models.

Understanding Model Behaviour. To understand the model behaviour in more detail, the confusion matrix and a subset of the misclassified samples were manually inspected for the best performing single-frame and video classification model.

A confusion matrix is a $L \times L$ matrix, where L is the number of location labels. The entry at index (i, j) is the number of samples that were predicted to belong to location label j, given the ground truth label i. Confusion matrices are a traditional tool in classification tasks, as they give a good overview of the models performance on the different classes and can highlight regularly confused classes visually.

Table 2: **Results.** The table the performance and efficiency metrics for all trained models. The models are grouped by their type (single-frame or video). The performance metrics are the top-1 accuracy (Acc@1), top-3 accuracy (Acc@3) and Macro F1-Score (Ma.-F1). The efficiency metrics are the number of floating point operations (FLOPs) per inference, the mean inference time in milliseconds per prediction (Latency) and the mean number of predictions per second (Throughput). The best performing model in each category is highlighted in bold. The metrics are computed on the test set of the respective dataset. *SlowFast R50 could not be benchmarked because of limitations of the `PyTorch Benchmark` library.*

	Model	Acc@1 (%)	Acc@3 (%)	Ma.-F1 (%)	FLOPs (G)	Latency (ms/Pred)	Throughput (Preds/s)
Single Frame	AlexNet	75.00	89.28	74.74	0.71	17.13	58.43
	ResNet18	79.91	93.38	77.54	1.82	32.794	30.49
	ResNet50	82.54	93.75	79.34	4.12	56.149	17.81
	DenseNet121	78.21	91.62	77.10	2.88	56.022	17.87
	MobileNet V3 Small	78.06	91.11	76.88	**0.06**	**9.168**	**109.07**
	EfficientNet V2 Small	80.92	94.55	77.54	2.88	50.795	19.69
	ViT B-16	78.29	93.18	77.76	17.59	125.113	7.99
	ConvNext Tiny	**83.59**	94.50	**79.78**	4.47	49.345	20.27
Video	R(2+1)D	80.78	**97.15**	73.66	**93.72**	**1237.00**	**0.81**
	Slow R50	77.46	94.80	69.97	42.00	791.54	1.26
	SlowFast R50	77.46	91.33	73.32	-	-	-
	X3D	**58.72**	**68.68**	**31.43**	2.85	336.81	2.97

Given the insights from the confusion matrix, a subset of the misclassified samples was manually inspected, to understand why the model failed to predict the correct location label. This analysis should unveil what type of situations are particularly challenging for the models and highlight potential areas for improvement.

Finally, the best performing single-frame and video classification model were used to continuously predict the location label of a subset of the raw videos in the test split to mimic a real-world deployment scenario. Again, the analysis of the results should drive focus to potential areas of improvement that are not visible from looking at independent mispredicted samples.

4 Results

Computer vision models, when carefully designed and trained, are generally capable of providing reasonable accurate predictions on the task of indoor localisation when phrased as a classification task. The detailed results of the evaluation of the performance and efficiency of the different models are presented in Table 2 and discussed in the following.

4.1 Performance Analysis

Surprisingly, even simple single-frame classification models are capable of providing a reasonable solution to the task of indoor localisation. Despite the lack of information about the temporal context of the frames, the best-performing single-frame classifier (ConvNext Tiny) achieves a top-1 accuracy of 83.59% and a top-3 accuracy of 94.50%. This is a significant finding as it proves that

in most cases the static information of the frames is sufficient to accurately determine location in indoor spaces.

The choice of the model architecture seems to affect the overall performance, but only to a small degree. The Top-1 accuracy of all single-frame classifiers ranges within 75% and 84% - only a 9% difference between the worst-performing model (AlexNet) and the best-performing model (ConvNext Tiny). Similarly, the Top-3 accuracy ranges within 89% and 95%, which is only a 6% difference. Generally, the results follow the performance that is to be expected given the ImageNet benchmarking results (Table 1). ResNet50 outperforms ResNet18, which outperforms AlexNet. The most modern model considered in this study, ConvNext Tiny, performs best overall. The only exception is ViT B-16, which performs worse than expected given its dominance in most computer vision benchmarks in recent years. It is likely that given its significantly larger size, it struggles with undertraining. Overall, the results show that the more complex models generally improve the Top-1 and Top-3 Accuracy, but only marginally so. For all models, the Macro F1-Score was observed to be lower, but only marginally so. This suggests that the models generally do not seem to be affected negatively by the class imbalance of the dataset.

It is generally assumed that knowledge about the temporal context of the video is beneficial for video classification tasks, as it allows the model to understand the motion of objects in the scene. In the context of indoor localisation, the temporal context of the video could for example be useful when a subset of frames is occluded by a person walking through the scene, the camera transitions between rooms, or when the camera is moved very suddenly.

However, the results suggest that video classifiers do not improve the Top-1 accuracy. The best performing video classifier, R(2+1)D, achieves a Top-1 accuracy of 80.78%, which is 2.81% lower than the best performing single-frame classifier, ConvNext Tiny. The same goes for the Slow R50 Family, which achieve a Top-1 accuracy of 77.46%. The X3D model is an outlier in all models, achieving a Top-1 accuracy of only 58.72%. The model was found to converge a lot slower than all other models, and could therefore not reach the same level of performance in 15 training epochs. It is likely that the model would perform better given more training time.

However, an interesting tendency is visible when analysing the Top-3 accuracy. The best video classifier, R(2+1)D, achieves a Top-3 accuracy of 97.15%, which is the best result out of all models. This finding hints at the fact that the temporal context of the video helps the robustness of the model: While single-frame classifiers make almost random predictions (the predicted class is not in the top-3 predictions) in 9.5% of all cases, video classifiers only make random predictions in 2.85% of all cases. This is a significant improvement, and speaks to a higher prediction robustness.

4.2 Efficiency Analysis

In deep learning, more complex models generally outperform simpler ones if provided with sufficient training data and compute resources. However, there is a trade-off between model complexity and efficiency, especially when efficiency is critical like when deployed on a mobile device.

The efficiency of the single-frame and video classifiers generally varies more than the performance - both within the groups and between the groups. The single-frame classifiers range from from a throughput rate of 7.99 predictions per second (ViT-B-16) to 109.07 predictions per second (MobileNet V3 S). This means that the most efficient model is 13.65 times more efficient than the least efficient model. The video classifiers require more compute and memory resources during inference, and are there therefore less efficient. They are only able to provide predictions between 1.26 predictions per second (Slow R50 Family) and 3.01 predictions per second (R(2+1)D). However, as each prediction considers a history of frames, the predictions are more robust and valid for a longer time.

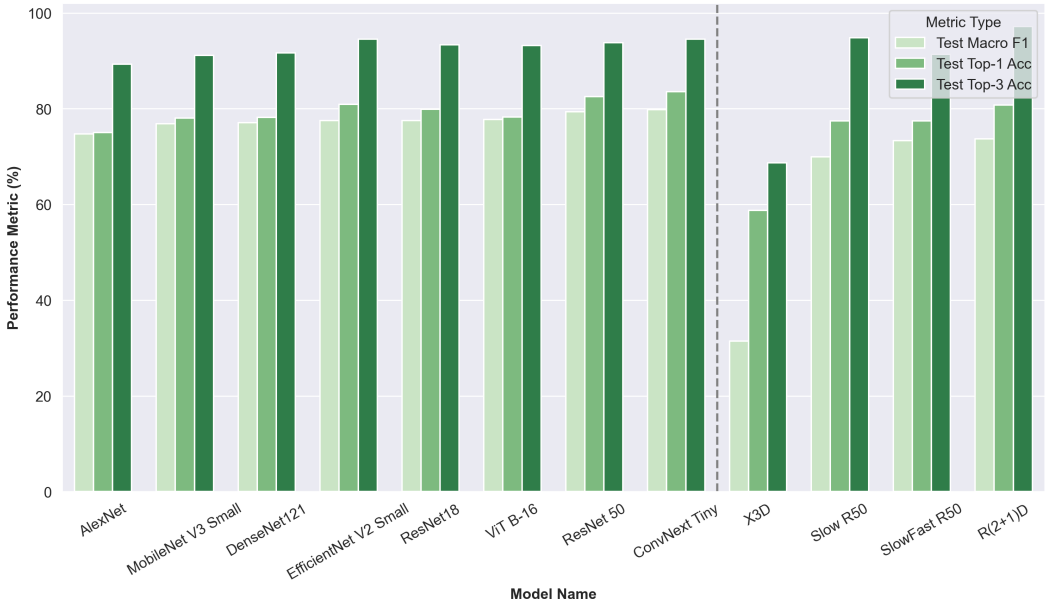


Figure 9: **Performance Metrics.** The Figure shows the performance metrics, Macro F1, Top-1 Accuracy and Top-3 Accuracy, for all trained models on the test split. A grey, dotted line separates the single-frame classifiers (left) from the video classifiers (right).

Given these findings, it is clear that performance gains can be achieved with an increase in model complexity. However, small gains come at a high cost in terms of efficiency. Figure 10 visualises the performance-efficiency trade-off for all models by plotting the relationship between the Top-1 Accuracy against the latency (inference time in milliseconds per prediction) and the throughput (predictions per second). Depending on the specific deployment requirements, different models are more suitable. If low memory consumption and high throughput are critical, high efficiency models like MobileNet V3 S are most suitable. If real-time inference is not critical, but high accuracy is required, models like ResNet50 or ConvNext Tiny are best suited. If more robust, but less frequent predictions are required, video classifiers like R(2+1)D can be valid options.

4.3 Understanding Model Behaviour

To better understand the complex dynamics, strengths and weaknesses of deep learning models in tackling the task of indoor classification, the two best performing models in each group, ConvNext Tiny and R(2+1)D, are analysed in more detail.

4.3.1 Confusion Patterns

Figure 11 shows the confusion matrices for (a) ConvNext Tiny and (b) R(2+1)D. Despite the fact that the two models have very different architectures, they exhibit similar confusion patterns.

Both models show a tendency for confusing visually similar classes. The first example of this are the two corridors on the ground floor (Ground Floor Corridor 1 and Ground Floor Corridor 2 and the Atrium (K). The architecture, interior design and lighting of the corridors are very similar, making it more challenging for both models to distinguish between the two. Figure 12a shows the confusion

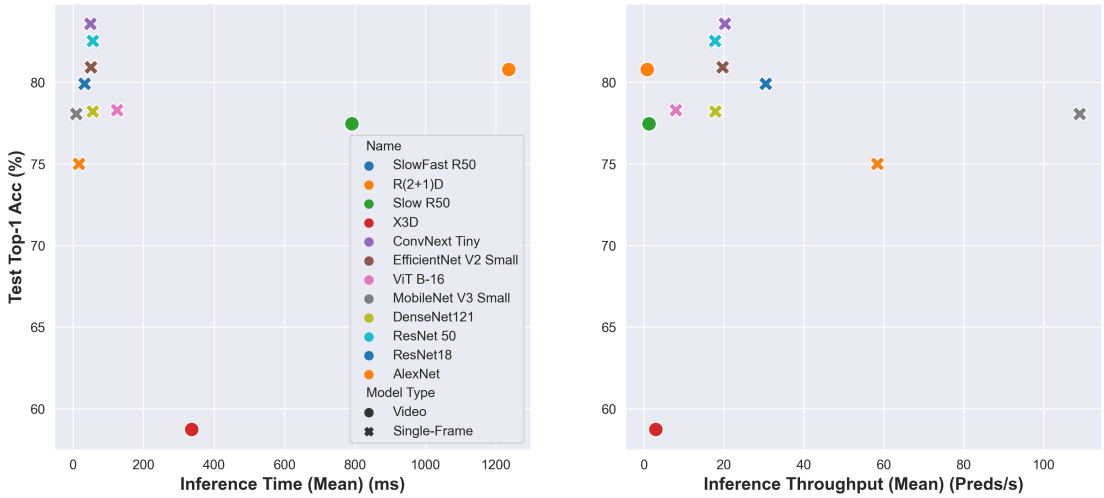
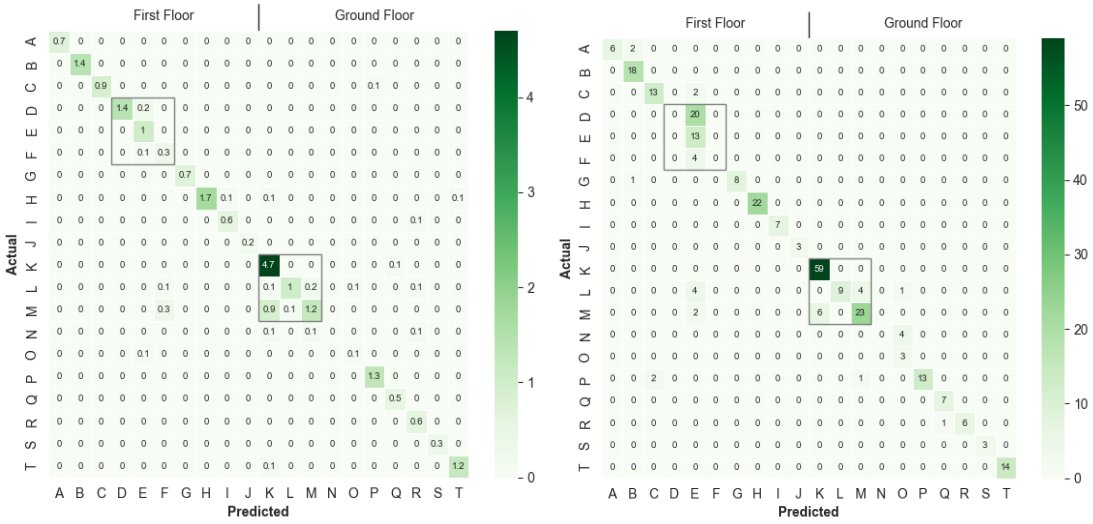


Figure 10: **Performance-Efficiency Trade-Off.** The Figure visualises the performance-efficiency trade-off for all models by plotting the relationship between the Top-1 Accuracy against the latency (inference time in milliseconds per prediction) and throughput (predictions per second). Each model is given unique colour, and the marker type indicates whether the model is a single-frame classifier (circle) or a video classifier (cross).



(a) ConvNext Tiny

(b) R(2+1)D

Figure 11: **Confusion Matrices.** The Figure shows the confusion matrix for (a) ConvNext Tiny and (b) R(2+1)D. The confusion matrix of ConvNext Tiny is normalised to show 1K samples per class for visual purposes. For both matrices, the entry at row i and column j shows the number of samples that belong to class i but were predicted to be class j. Gray rectangles indicate challenging classes, that are often confused, which are displayed in more detail in Figure 12.

between these classes for both models in detail. ConvNext Tiny has a tendency for predicting Atrium instead of Corridor and regularly mixes up the two corridors. R(2+1)D is less prone to mixing up the two corridors, but also mispredicts Corridor 2 to be the Atrium in a few instances.

A second example are the three libraries (First Floor Library 1, First Floor Library 2 and First Floor Library 2). The libraries are visually similar, as they share similar characteristics, like the bookshelves, the tables and the chairs. Figure 12b shows the confusion between these classes for both models. ConvNext Tiny is more capable of differentiating the three libraries, but still confuses Library 1 for Library 2. Weirdly, R(2+1)D is unable to differentiate between the libraries and naively predicts Library 2 for all three libraries. This might be, because Library 2 is the most common library in the training data, and the model is therefore biased towards this class.

The above failure cases highlight another intricacy of the models: While locations that are less present in the training data are generally handled well (robustness towards class imbalance), the models have a tendency for predicting the majority class in cases of uncertainty.

4.3.2 Misprediction Cases

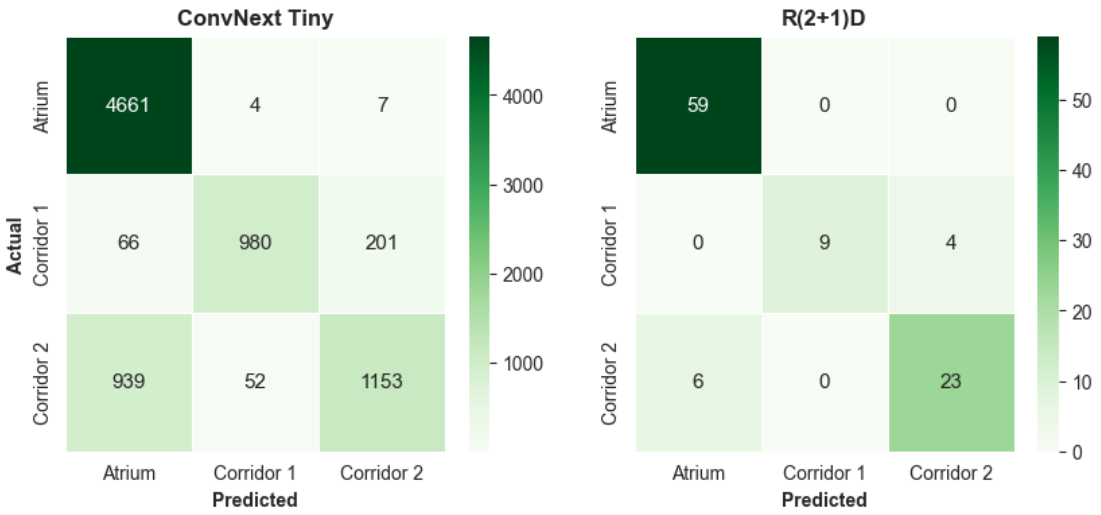
To further analyse and confirm the above findings, exemplary mispredictions are analysed for both models. Beyond the common misprediction patterns that were visible from the confusion matrices, this analyses found two more common failure cases.

Figure 13 displays six exemplary mispredictions for ConvNext Tiny (Figure 13a) and R(2+1)D (Figure 13b) each. The six mispredicted samples are grouped into three commonly found misprediction patterns:

Visual Similarities. This pattern describes mispredictions between visually similar classes. In the case of this study, there are several pairs or triples of classes (e.g. the two corridors, three libraries), which share a lot of visual characteristics. The two samples in this group for ConvNext Tiny uncover another of such groupings, which are the coloured areas on the ground and first floor. The model is visibly less sure about the prediction and the true class is typically the second most confident prediction. For R(2+1)D, the first sample shows a confusion between the libraries, and the second sample shows a confusion between the two corridors on the ground floor.

Location Transitions. This pattern describes mispredictions between classes that occur at the transition between areas. The first sample for ConvNext Tiny is a clip that just passes the door between two libraries. The model predicts the library before the door is passed, and the ground truth label is the library after the door is passed. The second sample shows a similar case, where the model predicts to be in the red area on the first floor, but the ground truth label is still the class before. This confusion pattern naturally arises, because locations may not be sharply separable only from the imagery of a camera. For example, if the camera only captures the features of the location that lies a few meters ahead, it is likely that the model predicts the location that lies ahead, but the ground truth label is still the location that lies behind.

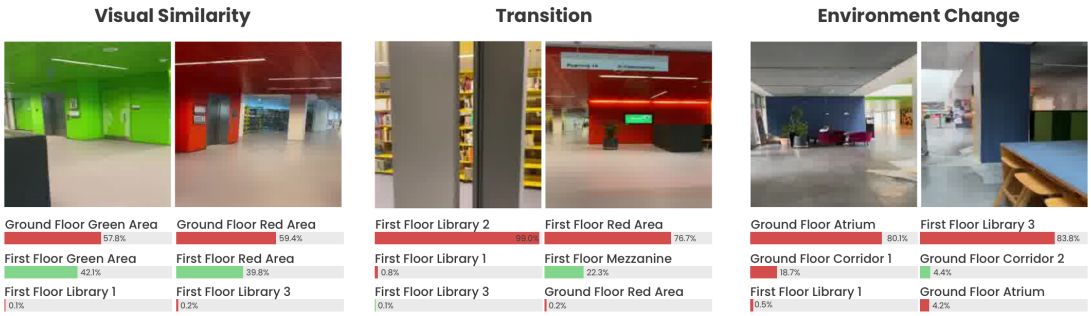
Environment Change. A third, interesting pattern was observed for both models in the area that was renovated in between the data collection periods for the training and testing split. Parts of Corridor 2 on the ground floor were renovated and repainted in blue. Both models fail to recognise this area and typically fall back to the two most common classes in the training data, which is the Atrium and Library 2.



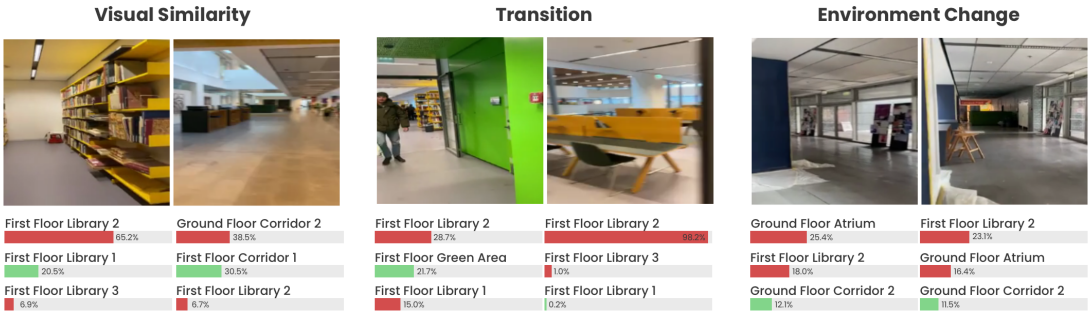
(a) Corridors

(b) Libraries

Figure 12: **Common Misprediction Patterns.** The figure shows subsets of the confusion matrices that highlight classes that are commonly confused by ConvNext Tiny (left) and R(2+1)D (right). Figure (a) shows the confusion between the two corridors on the ground floor and the Atrium. Figure (b) shows the confusion between the three libraries on the first floor.



(a) Misprediction Patterns for ConvNext Tiny



(b) Misprediction Patterns R(2+1)D

Figure 13: **Mispredicted Samples.** Exemplary mispredicted samples for (a) ConvNext Tiny and (b) R(2+1)D. For each sample, the top 3 predicted classes are shown in order of confidence. The true class, if among the top 3 predictions, is highlighted through a green bar. The mispredicted samples are grouped in the three commonly observed misprediction patterns: (1) Visual Similarity (2) Transition and (3) Environment Change. For the clips from the video classifier, the first frame of the clip is shown.

4.3.3 Model Behaviour Analysis

Finally, to emulate the behaviour of the models in a real-world scenario, both ConvNext Tiny and R(2+1)D are used to continuously predict the location on all test splits. The top predictions and the confidence scores are logged for each predictions. The following observations were made:

Prediction Robustness. Both models show a high robustness to regularly occurring changes in the environment, such as varying lighting conditions, occlusions or people in the scene. This is especially true for R(2+1)D, which is likely due to the temporal modelling capabilities of the model. However, drastic changes in the environment, such as the repainting of an entire region, lead to consistent mispredictions of the affected area. This is a general drawback of localisation systems that solely rely on the modality of vision

Prediction Consistency. The single-frame classifier, ConvNext Tiny, was found to be more inconsistent in its predictions than the video classifier, R(2+1)D. In difficult cases, like the ones described in the previous section, ConvNext Tiny would often predict different classes in a short time span, leading to "flicker" in the predictions. This behaviour was almost entirely absent in R(2+1)D, due to the lower throughput rate and its temporal modelling capabilities, which allow the model to smooth out predictions over time.

Training Data Bias. A bias toward the training data is present in both models. If features that are not representative of a class, but are overrepresented in the training data of that class, there is a chance for the model to learn these features as being indicative of a class. This was the case for the libraries: Most video clips that were taken while walking through bookshelves were filmed in library 2, while the other libraries were filmed in a more open space. This led to model associating in-between bookshelves clips to library 2, even though they are also present in the other libraries. When used in practice, data collection has to be carefully designed to avoid such biases.

4.4 Deployment on Mobile Devices

As a proof of concept, the most efficient and mobile-optimised model, MobileNet V3 Small, was deployed on a mobile device. The trained model was quantised to 8-bit float precision and converted to `TorchScript` format to be run more efficiently on mobile devices. Deployment was done using the `PlayTorch` framework, which is a port of the `PyTorch` Mobile SDK for native iOS and Android to Javascript. The deployed model can be tested by downloading the PlayTorch app from the App Store or Google Play Store and scanning the QR code found in the README of this project's GitHub repository. This will open the application, download the model and run it locally on the device.

5 Conclusion

5.1 Limitations & Future Work

This study has shown the potential of using common deep learning methods for providing accurate room-level localisation.

Arguably, the biggest limitation of this study is the small data set that was used for training and evaluation. While the setting of this study allowed to draw insightful conclusion about the data efficiency of the models and the general feasibility of the methodology, it is an open question how similar systems scale to larger and more complex indoor spaces when more training data is available.

Specifically, it would be interesting to investigate if a training dataset that captures the full variation of visual inputs from a indoor location over a longer period of time would lead to higher localisation

accuracy, and most importantly, can help to overcome the misprediction patterns observed in this study. Interesting follow-up questions, that also speak to the practicality of the method, are how much data needs to be collected to produce a representative training dataset, that can train models that do not suffer from the misprediction patterns observed in this study.

Another consequence of the small data size is that the test split might not be representative of the true variation of visual inputs from a indoor location over the course of a year in the current setup. In that is the case, then the evaluation metrics reported in this study are likely to be over-optimistic.

Finally, the detailed analysis of the mispredicted samples has shown that most errors, because the models that are visually similar, e.g. the libraries. As the vast majority of the mispredictions are due to this reason, future work should focus on improving the models' ability to distinguish between visually similar rooms. As said, this could be achieved by collecting more data from these rooms. Other starting points might be to directly incorporate about the relative position of the rooms in the building, which could limit the number of possible predictions. For example, if a model is confident in the prediction of the current room, it could use this information to limit the number of possible predictions for the next frame, if it has information about the current adjacent rooms.

5.2 Conclusion

This study has demonstrated that it is possible to use a pure deep learning pipeline for providing accurate room-level localisation in indoor spaces.

Both single-frame classifiers, that do not model temporal information, and video classifiers, that do model temporal information, were found to be capable of learning discriminative features from the visual input that allow them to localise a human agent in an indoor space from a set of rooms with a test accuracy of up to 83.59%.

A detailed analysis of the mispredictions patterns and intricate behaviors of the models unveiled three common failure modes: All models struggle with visually similar classes, transition between classes and when major visual changes, such as renovations, occur in the environment. All of these issues are grounded in the nature of the data set and the difficulty of the problem itself, but might be mitigated by collecting more data or by incorporating additional information in the models.

References

[1] Pytorch. https://pytorch.org.

[2] N. Alsindi, B. Alavi, and K. Pahlavan. Measurement and modeling of ultrawideband toa-based ranging in indoor multipath environments. *Vehicular Technology, IEEE Transactions on*, 58:1046 – 1058, 04 2009.

[3] N. Alsindi and K. Pahlavan. Cooperative localization bounds for indoor ultra-wideband wireless sensor networks. *EURASIP Journal on Advances in Signal Processing*, 2008, 04 2008.

[4] U. Bandara, M. Hasegawa, M. Inoue, H. Morikawa, and T. Aoyama. Design and implementation of a bluetooth signal strength based location sensing system. In *Proceedings. 2004 IEEE Radio and Wireless Conference (IEEE Cat. No.04TH8746)*, pages 319–322, 2004.

[5] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017.

[6] S. S. Chawathe. Beacon placement for indoor localization using bluetooth. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 980–985, 2008.

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[9] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389, 2014.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[11] C. Feichtenhofer. X3D: expanding architectures for efficient video recognition. *CoRR*, abs/2004.04730, 2020.

[12] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. *CoRR*, abs/1812.03982, 2018.

[13] A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi. Escaping the big data paradigm with compact transformers. *CoRR*, abs/2104.05704, 2021.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[15] S. He and S.-H. G. Chan. Wi-fi fingerprint-based indoor positioning: Recent advances and comparisons. *IEEE Communications Surveys and Tutorials*, 18(1):466–490, 2016.

[16] A. Howard, M. Sandler, G. Chu, L. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam. Searching for mobilenetv3. *CoRR*, abs/1905.02244, 2019.

[17] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[20] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017.

[21] A. Khalajmehrabadi, N. Gatsis, and D. Akopian. Modern WLAN fingerprinting indoor positioning methods and deployment challenges. *CoRR*, abs/1610.05424, 2016.

[22] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[25] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030, 2021.

[26] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. *CoRR*, abs/2201.03545, 2022.

[27] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[28] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *CoRR*, abs/1502.00956, 2015.

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[30] K. Sugiura and H. Matsutani. Particle filter-based vs. graph-based: SLAM acceleration on low-end fpgas. *CoRR*, abs/2103.09523, 2021.

[31] A. Tahat, G. Kaddoum, S. Yousefi, S. Valaee, and F. Gagnon. A look at the recent wireless positioning techniques with a focus on algorithms for moving receivers. *IEEE Access*, 4:6652 – 6680, 09 2016.

[32] M. Tan and Q. V. Le. Efficientnetv2: Smaller models and faster training. *CoRR*, abs/2104.00298, 2021.

[33] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877, 2020.

[34] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014.

[35] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. *CoRR*, abs/1711.11248, 2017.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

	Specification	Value
Sys.	Name	Darwin
	Node	MacBook Pro
CPU	Model	Apple M1
	Architecture	ARM64
	Physical Cores	8
	Frequency	2.4 GHz
Mem.	Total Capacity	16 GB
	Avg. Used Capacity	~ 7.4 GB

(a) Local Machine

	Specification	Value
Sys.	Name	Linux
	Node	Desktop 24
CPU	Model	Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz
	Architecture	x86_64
	Physical Cores	20
	Frequency	3.3 GHz
GPU	Model	NVIDIA GeForce GTX 1080 Ti
	Memory	11.2 GB
Mem.	Total Capacity	250 GB
	Avg. Used Capacity	~ 7.4 GB

(b) HPC (Remote Server)

Table 3: **Machine Specifications.** The Table shows relevant hardware specifications for (a) the local machine and (b) the remote server that were used for conducting experiments within this study.

6 Appendix

6.1 Reproducibility

All code and data used in this project is available on GitHub. The project's `README` file contains detailed instructions on how to reproduce the results of this project.

Further, the precise configuration and results of the experiments that are reported here are publicly available as a public Weights & Biases experiments.

6.2 Machine Specifications

Table 3 lists the two machines, alongside relevant specifications, that were used for training and evaluation of the models. The HPC cluster was used for training and evaluation of all models. Analyses and visualisations were performed on the local machine, as well as running the real-time inference demo.

Table 4: Encoding of Classes

Class	**Encoding**
First_Floor_Corridor_1	A
First_Floor_Corridor_2	B
First_Floor_Green_Area	C
First_Floor_Library_1	D
First_Floor_Library_2	E
First_Floor_Library_3	F
First_Floor_Magenta_Area	G
First_Floor_Mezzanine	H
First_Floor_Red_Area	I
First_Floor_Yellow_Area	J
Ground_Floor_Atrium	K
Ground_Floor_Corridor_1	L
Ground_Floor_Corridor_2	M
Ground_Floor_Entrance_Magenta	N
Ground_Floor_Entrance_Yellow	O
Ground_Floor_Green_Area	P
Ground_Floor_Magenta_Area	Q
Ground_Floor_Red_Area	R
Ground_Floor_Yellow_Area	S
Stairs_Atrium	T