A comparison of visual place recognition methods using a mobile robot in an indoor environment

Beatrice van Eden[1,*], *Natasha* Botha[1] *and Benjamin* Rosman[2]

[1]Centre for Robotics and Future Production, Manufacturing Cluster, Council for Scientific and Industrial Research, South Africa

[2]School of Computer Science and Applied Mathematics, University of the Witwatersrand, South Africa

Abstract. Spatial awareness is an important competence for a mobile robotic system. A robot needs to localise and perform context interpretation to provide any meaningful service. With the deep learning tools and readily available sensors, visual place recognition is a first step towards identifying the environment to bring a robot closer to spatial awareness. In this paper, we implement place recognition on a mobile robot considering a deep learning approach. For simple place classification, where the task involves classifying images into a limited number of categories, all three architectures; VGG16, Inception-v3 and ResNet50, perform well. However, considering the pros and cons, the choice may depend on available computational resources and deployment constraints.

1 Introduction

Place classification for mobile robots refers to the task of automatically categorising and recognising different types of places or environments that a robot encounters during its navigation. The goal is to enable the robot to understand and interpret its surroundings, allowing it to make informed decisions and perform appropriate actions based on the identified place type.

Place classification plays a crucial role in various robot applications, including autonomous navigation, environment mapping, and human-robot interaction. By accurately identifying places such as offices, kitchens, bedrooms, studies, or other areas, the robot can adapt its behaviour, perform specific tasks, and navigate effectively in different environments.

There are many indoor applications for visual place recognition. In Afif et al. [1] they evaluate efficientNet for Visual Place Recognition (VPR) to assist visually impaired individuals by providing them with information about their surroundings and helping them navigate indoor spaces. Another example is Augmented Reality (AR) and Virtual Reality (VR) applications as seen in Kim et al. [2].

Place recognition is a fundamental topic of research and development in the field of computer vision. In the past, visual place recognition has been limited to only considering

[*] Corresponding author: bveden@csir.co.za

processed images produced with manually identified features. These could be local features such as Scale Invariant Feature Transformation (SHIFT), Speeded Up Robust Features (SURF) or global features such as Histogram of Oriented Gradients (HOG). However, the last two decades have seen the rise of deep learning approaches in place recognition applications in Zhang et al. [3].

In Zhang et al. [3] there is a nice representation of place recognition milestones of visual place recognition methods, see Figure 1. We see that over the past few decades, methods transitioned from traditional handcrafted feature-based methods to more recently learned feature-based methods.

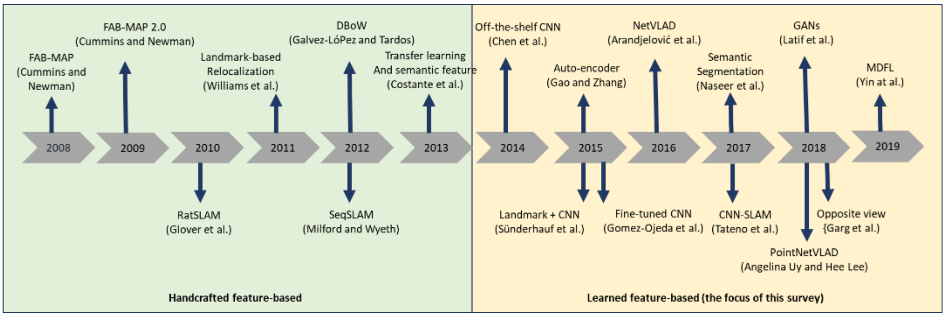

FAB-MAP (Cummins and Newman)
FAB-MAP 2.0 (Cummins and Newman)
Landmark-based Relocalization (Williams et al.)
DBoW (Galvez-LóPez and Tardos)
Transfer learning And semantic feature (Costante et al.)
Off-the-shelf CNN (Chen et al.)
Auto-encoder (Gao and Zhang)
NetVLAD (Arandjelović et al.)
Semantic Segmentation (Naseer et al.)
GANs (Latif et al.)
MDFL (Yin at al.)
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
RatSLAM (Glover et al.)
SeqSLAM (Milford and Wyeth)
Landmark + CNN (Sünderhauf et al.)
Fine-tuned CNN (Gomez-Ojeda et al.)
CNN-SLAM (Tateno et al.)
Opposite view {Garg et al.)
PointNetVLAD (Angelina Uy and Hee Lee)
Handcrafted feature-based
Learned feature-based (the focus of this survey)

Fig. 1. Milestones of visual place recognition methods. (Adapted from Zhang et al. [3])

In Schubert et al. [4] they discuss state-of-the-art deep learning methods that can be used to address challenges in VPR. It provides an introduction to how VPR is implemented and evaluates different approaches. We also see another approach to the VPR problem when Chen et al. [5] implements You Only Look Once (YOLO).

To explore the use of deep learning in place recognition, we consider a convolutional neural network (CNN) approach for place recognition on a mobile robot. CNNs are trained on labelled datasets of images that represent different places, allowing them to learn discriminative features and patterns for each place category. To achieve accurate place classification for mobile robots, it is important to curate a diverse and representative dataset of labelled images encompassing different place categories. The dataset should capture variations in lighting conditions, viewpoints, and environmental factors to ensure robustness in real-world scenarios.

Another survey on deep visual place recognition from Masone et al. [6], includes a summary of datasets that are used in visual place recognition tasks. Deep learning approaches rely on big and clean datasets. Various datasets are of outdoor scenes with a few nice indoor datasets like Zhou et al. [7] and Quattoni et al. [8].

Normally CNNs require a very large dataset, a thousand images or more for each room in the environment, for training. To reduce the number of images needed, while retaining a decent accuracy, we consider a transfer learning approach to CNN's. Transfer learning speeds up learning while transferring information from already trained weights on a different dataset explained in Zhuang et al. [9]. In Tammina et al. [10] we see how transfer learning and the VGG16 architecture were used to classify images.

For this work, we train three different CNN architectures; VGG16, Inception v3 and ResNet50. We use transfer learning for each CNN using weights trained on the ImageNet dataset in Deng et al. [11].

In Section 2 we expand on the different CNN architectures followed by a discussion of the Imagnet dataset as well as the data we collected to train our custom place classification algorithms. In Section 3 we see the results of our trained networks and we test our trained weights on a robot. This work is concluded in Section 4.

2 Methods

Different CNN architectures have been developed to address various challenges and requirements in the field of computer vision. Different architectures aim to strike a balance between depth, complexity, and computational resources, catering to various use cases and constraints. Their suitability for a specific use case depends on factors such as dataset size, computational resources, and implementation constraints.

Three common architectures have been chosen to compare the effectiveness of our place recognition on a robot. VGG16, Inception-v3, and ResNet50 are all powerful CNN architectures that have demonstrated excellent performance on various image classification tasks, including both complex and simple classification problems as seen in Jiang et al. [12], Joshi et al. [13], Shabbir et al. [14].

The selection of hyperparameters was performed manually on a trial-and-error basis, as there is no mathematical formulation for the calculation of appropriate parameters for a specific dataset. The overall computational architecture of the proposed CNNs are shown in the next sections.

2.1 CNN Layers Explained

2.1.1 Input layer

Input layer specifications are important to consider when preparing data for these CNN models. They must be pre-processed to match the expected format before feeding them into the networks for training.

For example, the first layer takes an image of h (height) × w (width) × c (number of channels) pixels as input and passes it through convolutional and max pooling layers to reduce its spatial size.

After passing through the designed convolutional and max-pooling layers of the chosen architecture, the final feature vector is obtained at the fully connected layer and is input into the classifier for class prediction [15].

A convolutional layer, often referred to as a "conv layer," is a fundamental building block in a CNN. It is a key component of CNNs used for image and spatial data analysis. The primary purpose of a convolutional layer is to perform feature extraction from the input data through a process called convolution.

2.1.2 Convolutional Layer

A convolutional layer is a fundamental building block in a CNN. It is a key component of CNNs used for image and spatial data analysis. The primary purpose of a convolutional layer is to perform feature extraction from the input data (Equation 1 [15]).

$$Feature\ Vextor = \sum (Input_{n\times n} + Weight_{n\times n}) + Bias \tag{1}$$

The convolutional layer applies a set of learnable filters (kernels) to the input data. These filters are small, typically 3x3 or 5x5 grids of numbers represented by n x n. The layer slides these filters over the input data to compute a dot product between the filter and a local region of the data. This process captures features present in the input.

The result of the convolution operation is a feature map. Each filter produces one feature map. Feature maps represent the presence of specific patterns or features in the input data. These patterns can be simple, such as edges, or more complex, like textures or object parts. The operation is shown in Figure 2.

2.1.3 Activation Layer

After convolution, a non-linear activation function, typically ReLU (Rectified Linear Unit), is applied to the feature maps, see Nair et al. [15]. ReLU is computationally faster than other activation functions. This introduces non-linearity into the model, allowing it to capture complex relationships in the data. The operation is shown in Figure 2.

2.1.4 Max-Pooling Layer

Pooling layers follow convolutional layers. These layers reduce the spatial dimensions of the feature maps while retaining the most important information. Max pooling is the most popular pooling operation. A window (typically 2x2 or 3x3) is applied to each location in the input feature map, and the maximum value within that window is retained in the output feature map.

It helps capture the most prominent features and reduce the sensitivity of the network to small changes in the input data. The operation is shown in Figure 2.

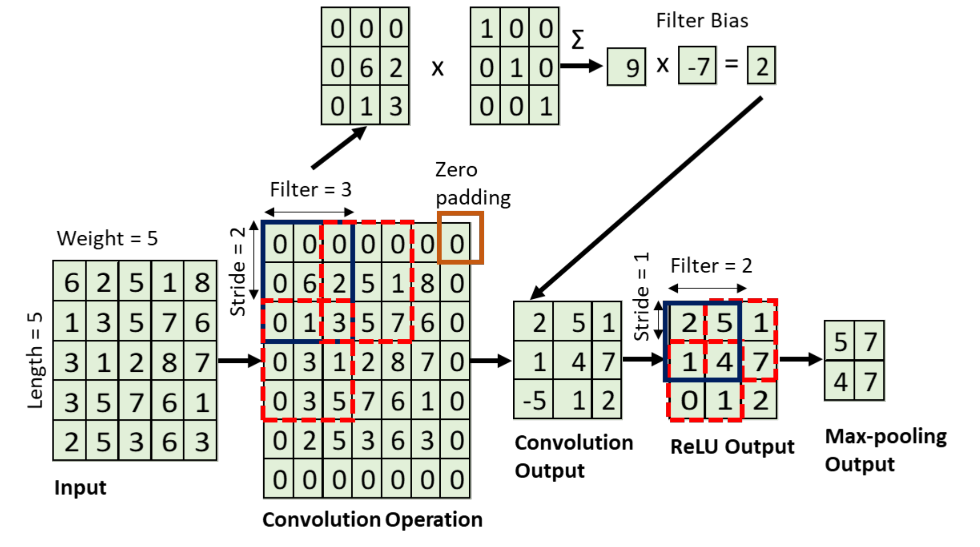


Fig.2. Mathematical operations in convolutional, rectified linear unit (ReLu), and max-pooling layers. (Adapted from et al. [15])

2.1.5 Fully connected Layer

A fully connected layer is a type of layer in a neural network that connects every neuron from the previous layer to every neuron in the current layer. Fully connected layers are typically found in the later stages of neural. These layers play a crucial role in transforming the learned features into final output predictions or decisions.

This layer converts the three-dimensional matrix from the previous layers to a one-dimensional vector. Equation 2 shows the operation, [15].

$$Z_{Vo\times 1} = Weight_{Vo\times Vi}\ I_{Vi\times 1}.Bias_{Vo\times 1} \tag{2}$$

Vo and Vi is the input and output vector size while Z is the output.

2.1.6 Softmax Layer

The softmax layer is the final layer in the neural network. It is important in most neural network architectures, even more so when the network is used for multiclass classification tasks. It is responsible for transforming the raw output scores from the preceding layers into probability distributions over multiple classes.

If the previous output vector is ‘z’, the softmax function for a class’i’ is calculated as in equation 3.

$$P(i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{3}$$

The output will be a vector with values between 0 and1.

2.2 VGG16

The VGG16 (Visual Geometry Group 16) is a CNN architecture that was introduced by the Visual Geometry Group at the University of Oxford. It was developed for image classification tasks.

VGG16 is known for its simplicity and effectiveness in capturing features from images, Simonyan et al. [16]. However, VGG16 is a relatively deep architecture with many parameters, which may make it computationally expensive and resource-intensive for deployment on resource-constrained devices like robots.

The VGG16 architecture consists of a total of 16 weight layers, including 13 convolutional layers and 3 fully connected layers. It follows a straightforward design principle of using small-sized convolutional filters repeatedly along with max-pooling layers to downsample the spatial dimensions. Figure 3 shows the architecture of the VGG16 model.

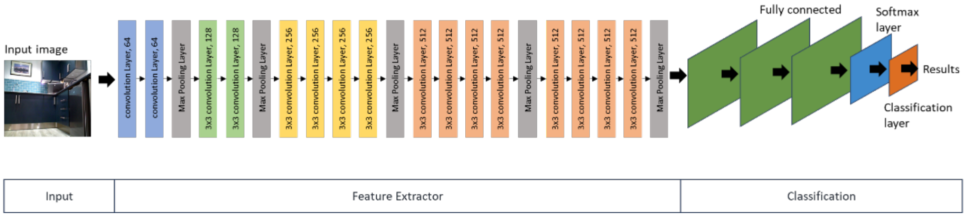

Input image
convolution Layer, 64
convolution Layer, 64
Max Pooling Layer
3x3 convolution Layer, 128
3x3 convolution Layer, 128
Max Pooling Layer
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 256
Max Pooling Layer
3x3 convolution Layer, 512
3x3 convolution Layer, 512
3x3 convolution Layer, 512
3x3 convolution Layer, 512
Max Pooling Layer
3x3 convolution Layer, 512
3x3 convolution Layer, 512
3x3 convolution Layer, 512
3x3 convolution Layer, 512
Max Pooling Layer
Fully connected
Softmax layer
Results
Classification layer
Input
Feature Extractor
Classification

Fig. 3. Architecture of VGG16 model, (Adapted from Pardede et al. [17])

2.3 Inception-v3

The Inception-v3 is a CNN architecture developed by researchers at Google. It is designed for image classification. The architecture builds upon the original Inception model.

Inception-v3 is designed to capture features at different scales in Szegedy et al. [18]. By using different filter sizes and pooling operations in the Inception modules, Inception-v3 can capture features at multiple spatial scales. This allows it to effectively learn both local and global features.

It balances model complexity and performance, making it a suitable choice for a wide range of applications. Inception-v3 offers good accuracy while being computationally efficient compared to deeper architectures like VGG16 or ResNet50, this can also be seen in Table 1 in the result section. It can handle both complex and simple image classification tasks effectively.

The Inception-v3 architecture combines a variety of convolutional filters with different sizes and strides to capture features at various scales. It also employs a technique called factorisation to reduce the computational complexity of convolutions. The idea behind factorization is to break down a large convolutional operation into a series of smaller operations, which can be computationally more efficient. Figure 4 shows the architecture of the Inception-v3 model.

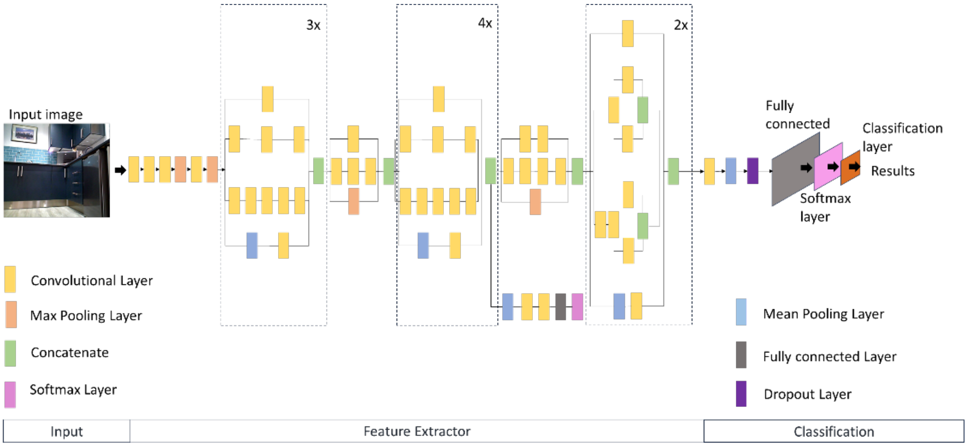


Fig. 4. Architecture of Inception-v3 model, (Adapted from Ali et al. [19])

2.4 ResNet-50

The ResNet50 (Residual Network-50) is a CNN architecture that was introduced by researchers at Microsoft Research Asia. It is specifically designed to address the challenge of training very deep neural networks.

ResNet50 addresses the challenge of training deep networks by introducing residual connections in He et al. [20]. A residual connection is like a shortcut that allows the network to skip some layers and directly pass information from one layer to another. The primary motivation behind ResNet50 was to overcome the degradation problem associated with training deep neural networks. The degradation problem refers to the observation that increasing the network depth leads to reduced accuracy due to difficulties in training the deep layers.

ResNet50 can handle both simple and complex image classification effectively. However, it is a deeper architecture than Inception-v3 and may require more computational resources during training and inference. Figure 5 shows the architecture of the ResNet50 model.

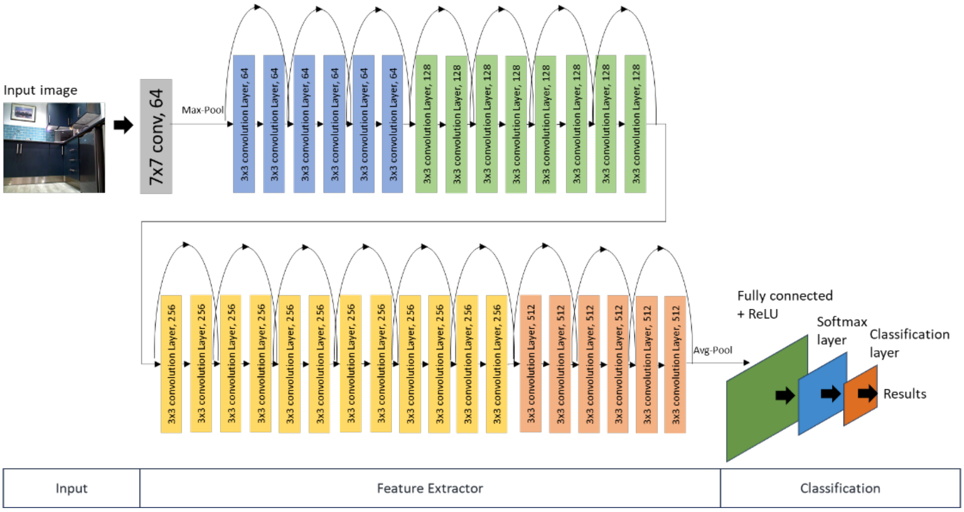

Input image
7x7 conv, 64
Max-Pool
3x3 convolution Layer, 64
3x3 convolution Layer, 64
3x3 convolution Layer, 64
3x3 convolution Layer, 64
3x3 convolution Layer, 64
3x3 convolution Layer, 64
3x3 convolution Layer, 128
3x3 convolution Layer, 128
3x3 convolution Layer, 128
3x3 convolution Layer, 128
3x3 convolution Layer, 128
3x3 convolution Layer, 128
3x3 convolution Layer, 128
3x3 convolution Layer, 128
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 256
3x3 convolution Layer, 512
3x3 convolution Layer, 512
3x3 convolution Layer, 512
3x3 convolution Layer, 512
3x3 convolution Layer, 512
3x3 convolution Layer, 512
Avg-Pool
Fully connected + ReLU
Softmax layer
Classification layer
Results
Input
Feature Extractor
Classification

Fig. 5. Architecture of ResNet50 model, (Adapted from Shyamalee et al. [21])

2.5 The Dataset

ImageNet is a large-scale visual database designed for visual object recognition and classification tasks. It serves as a benchmark for training and evaluating computer vision algorithms and models in Russakovsky et al. [22]. The database contains millions of labelled images across a wide range of object categories. All three CNN architectures have pre-trained Tensorflow weights based on the ImageNet database that we used in transfer learning when training our models.

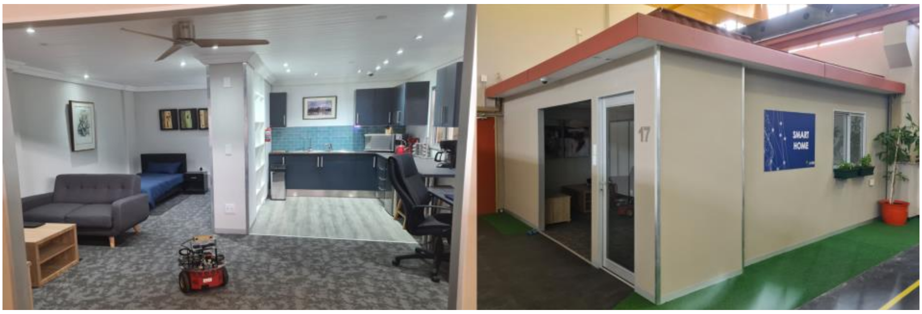
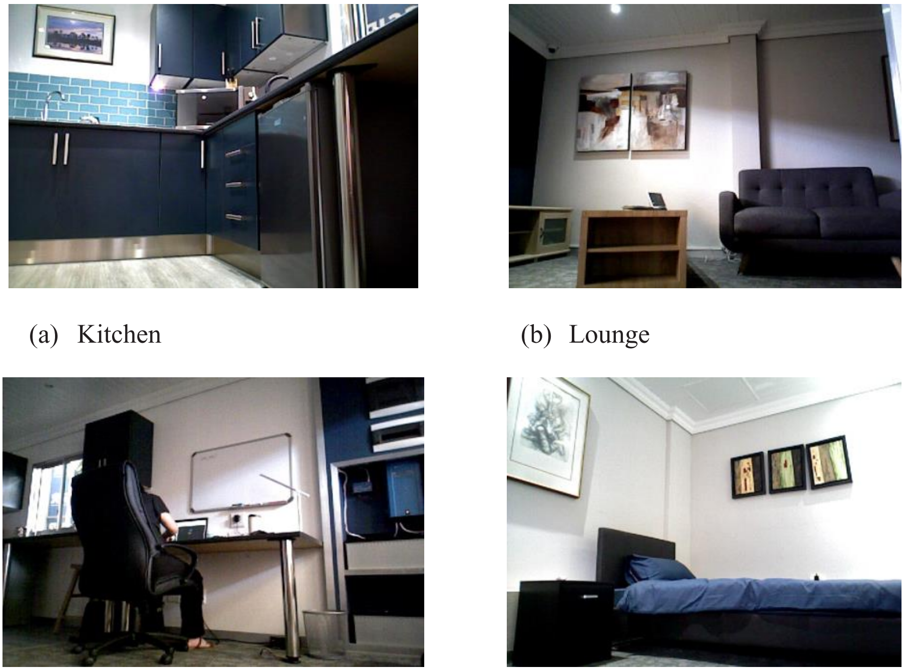


Fig. 6. Smart home environment for robot

To collect our database with four different rooms the Pioneer mobile robot was manually driven around the smart home, Figure 6. Data were collected in each room, namely the kitchen, lounge, study and bedroom. A few *.bag* files were captured which contain all the sensors and shared data. The *.bag* file contains all the images of each of the four different rooms in the smart home. The RGB images were extracted from these *.bag* files for training with sample images shown in Figure 7. This resulted in 181 train and 59 test RGB images for the kitchen, 124 train and 41 test RGB images for the lounge, 204 train and 67 test RGB images for the bedroom and 181 train and 62 test RGB images for the office. This is a small dataset and we therefore used images from the ImageNet dataset to supplement the training using a transfer learning approach.



(a) Kitchen

(b) Lounge

(c) Study

(d) Bedroom

Fig. 7. Mobile robot data sample for kitchen, lounge, study, and bedroom - robot's point of view

2.6 Experimental Setup - Implementation on Voyager mobile robot

The Voyager is a mobile robot, developed in-house, able to generate a map of its environment, see Figure 8. Place recognition will be added to the functionality so that the map provides more meaningful insights. This is a building block for detecting objects that belong in a certain room and the actions (or activities) generally undertaken in that room.

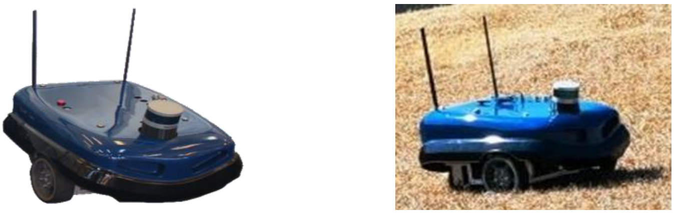


Fig. 8. Voyager mobile robot

The Voyager mobile robot software is built using robot operating software (ROS2) in Macenski et al. [23]. To implement the classification, ROS2 needs to work with Tensorflow which was used to train our CNN models. A custom ROS2 node was developed to implement the place recognition functionality, using the weights obtained during training for the three different CNN models, which would allow Tensorflow to communicate with ROS2.

In this custom ROS2 node, images are received through the subscriber from a specific topic. The subscriber then converts the received image message using *cv_bridge* and passes it to the TensorFlow model for classification. The result is then published by the publisher to a designated topic for further processing or visualisation. Note that the specific implementation details may vary depending on your chosen neural network architecture, dataset, and ROS2 setup. Both publishers/subscribers and client/server can be used for implementing image classification in ROS 2. The choice between them depends on the specific requirements and design of your system. Here are some considerations for each approach:

- Publishers/Subscribers follow an asynchronous communication model, where the publisher node publishes messages to a topic, and any interested subscriber nodes receive and process those messages. This can be useful when you have multiple subscribers that need to independently process the image data without waiting for each other. The pairs are loosely coupled. The publisher doesn't need to know the specific subscribers, and subscribers don't need to know the publishers. This can make development easier.

- Client/Server follows a synchronous communication model. The client node sends a request to the server node, which then processes the request and returns the result. This synchronous communication can be beneficial when you need a direct response from the server for image classification, especially in real-time or interactive scenarios. With a client/server approach, the server node acts as a central entity responsible for processing the image classification requests.

This implementation is tested using a test *.bag* file which was captured in the smart home. It is an open-plan house with four different rooms: an office (or study), kitchen, bedroom, and lounge. Voyager first generates a map of the indoor environment, using Simultaneous Localisation and Mapping (SLAM). It then sends the images obtained through its camera to the place recognition algorithm through the custom ROS2 node. The custom ROS2 node passes these images to Tensorflow for classification, before sending the result back to the custom ROS2 node. This uses a publisher and subscriber. Figure 9 is a simplified rqt graph including the recognition node interacting with the Voyager stack. This rqt graph provides a visual representation of the communication graph among ROS 2 nodes.

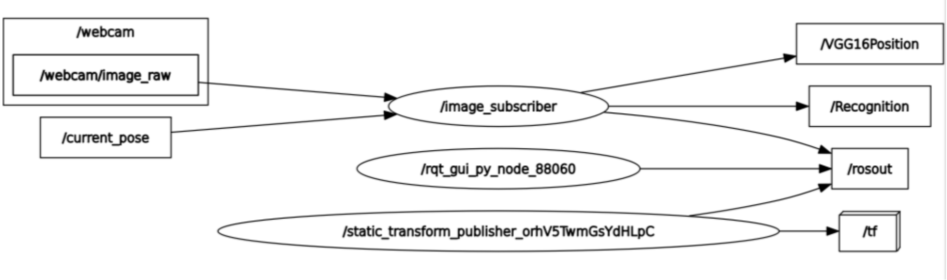


Fig. 9. Simplified rqt graph including the recognition node

3 Results and Discussion

When implementing classification on a robot, it is important to consider the computational resources available on the robot and the real-time constraints of the application. If your robot has sufficient computational power and memory, implementing VGG16, Inception-v3, or ResNet50 can yield accurate image classification results. However, if computational resources are limited, you may need to explore more lightweight architectures such as MobileNet or SqueezeNet, which are specifically designed for resource-constrained environments. Ultimately, the choice of architecture should be based on a trade-off between accuracy and computational efficiency, considering the specific requirements and constraints of the robot application.

We first train each CNN architecture using the smart home dataset discussed in Section 2.4 before implementing and testing it on the Voyager platform. When comparing VGG16, Inception-v3, and ResNet50 CNN architectures, it is important to note that the "better" option depends on specific factors such as the task requirements, available computational resources, and dataset characteristics.

3.1 CNN Architecture Training

In this section we present and discuss the results from training and validating the three CNN architectures using the dataset described in Section 2.4.

3.1.1 VGG 16

This CNN architecture is relatively straightforward, it is easy to implement and understand. The computational requirements for this CNN are not as efficient as other architectures, due to a large number of parameters. Care should therefore be taken to avoid overfitting when the dataset is small.

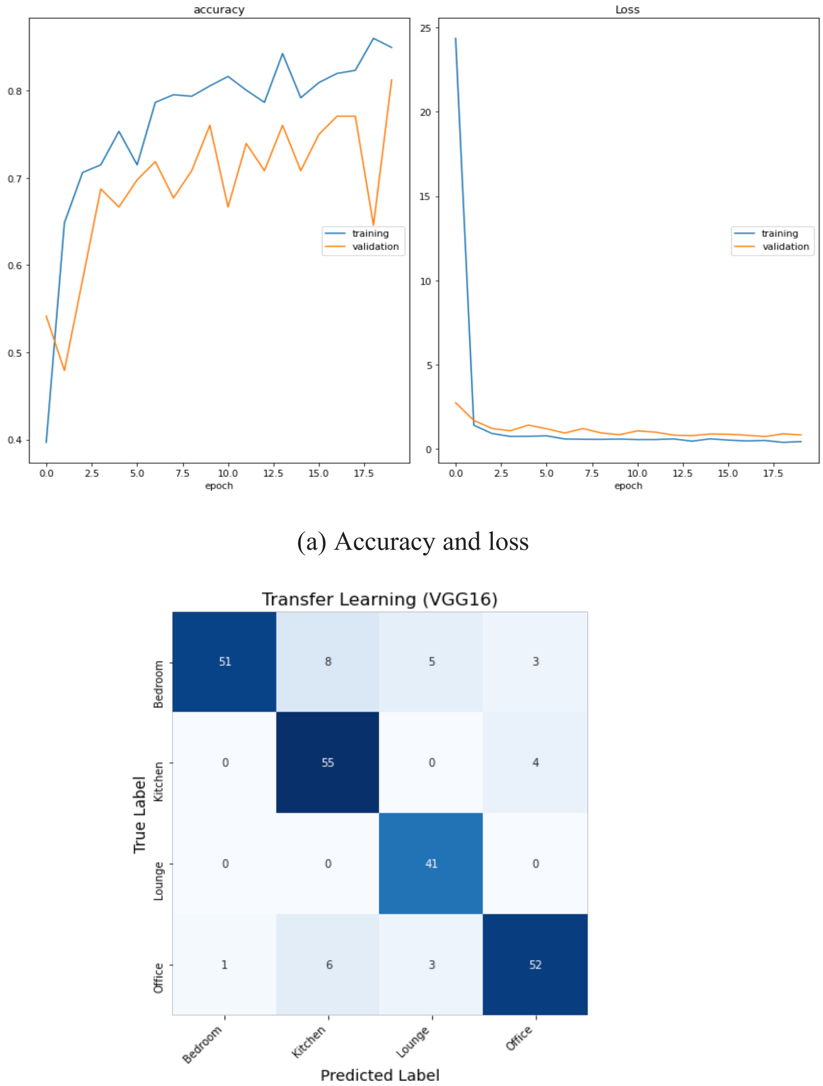
The results from training on our dataset and validating with a separate set of validation images are shown in Figure 11. In Figure 10(a) we see the accuracy and loss when training the VGG16 CNN, with an accuracy of 86.90%.

The performance of the VGG16 CNN model is illustrated in Figure 10(b) with a confusion matrix. A confusion matrix is a table used to evaluate the performance of a machine learning model, particularly in classification tasks. It provides a summary of the predictions made by the model compared to the actual labels of the data. The confusion matrix organises the predictions into four categories: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).VGG16 demonstrates good performance on our data.

- True Positives (TP): The number of instances correctly predicted as positive by the model.
- True Negatives (TN): The number of instances correctly predicted as negative by the model.
- False Positives (FP): The number of instances incorrectly predicted as positive by the model when the actual label is negative.
- False Negatives (FN): The number of instances incorrectly predicted as negative by the model when the actual label is positive.

From the confusion matrix, various evaluation metrics can be derived, including accuracy, precision, recall, and F1 score, which further quantify the model's performance.

The VGG16 model performed well seen in the confusion matrix.



(a) Accuracy and loss

	Bedroom	Kitchen	Lounge	Office
Bedroom	51	8	5	3
Kitchen	0	55	0	4
Lounge	0	0	41	0
Office	1	6	3	52

(b) Confusion matrix

Fig. 10. Results for training and validating the VGG16 CNN model

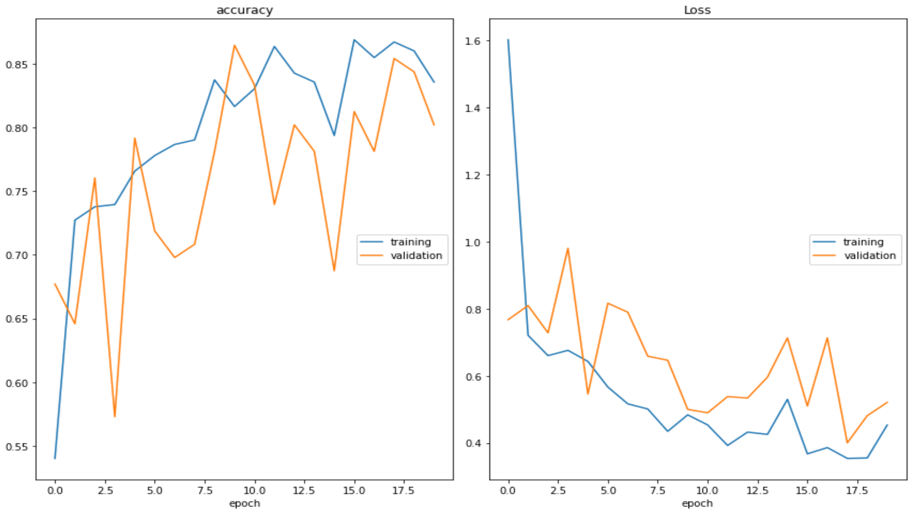
3.1.2 Inception-v3

Th Inception-v3 CNN architecture makes efficient use of computational resources due to the use of factorised convolutions, see Table 1. It captures features at multiple spatial scales, enabling it to handle complex patterns in images. However, it is relatively complex to understand and implement compared to simpler architectures. In Figure 3, you can see the intricacies of the architecture of the Inception-v3 model compared to that of the VGG16 model in Figure 3.

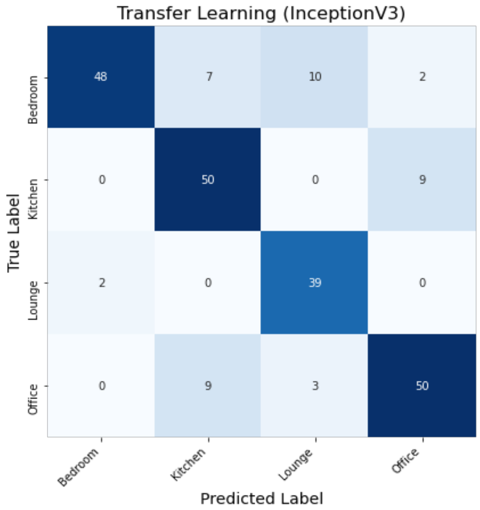
Table 1. Comparison of models' computational time and size

Model	**Patch size**	**Single image computation time (Seconds)**	**Total image processed per bag (394.630s/bag)**	**Model size**
VGG16	224x224x3	0.230	1715	1.25GB
Inception-v3	224x224x3	0.980	402	124.1MB
ResNet5	224x224x3	0.725	544	96.5MB

The results from training on our dataset and validating with a separate set of validation images are shown in Figure 11, with an accuracy of 81.66%. The confusion matrix shows only slightly less accuracy than the results for the VGG16 model.



(a) Accuracy and loss



(b) Confusion matrix

Fig. 11. Results for training and validation of the Inception-v3 CNN model

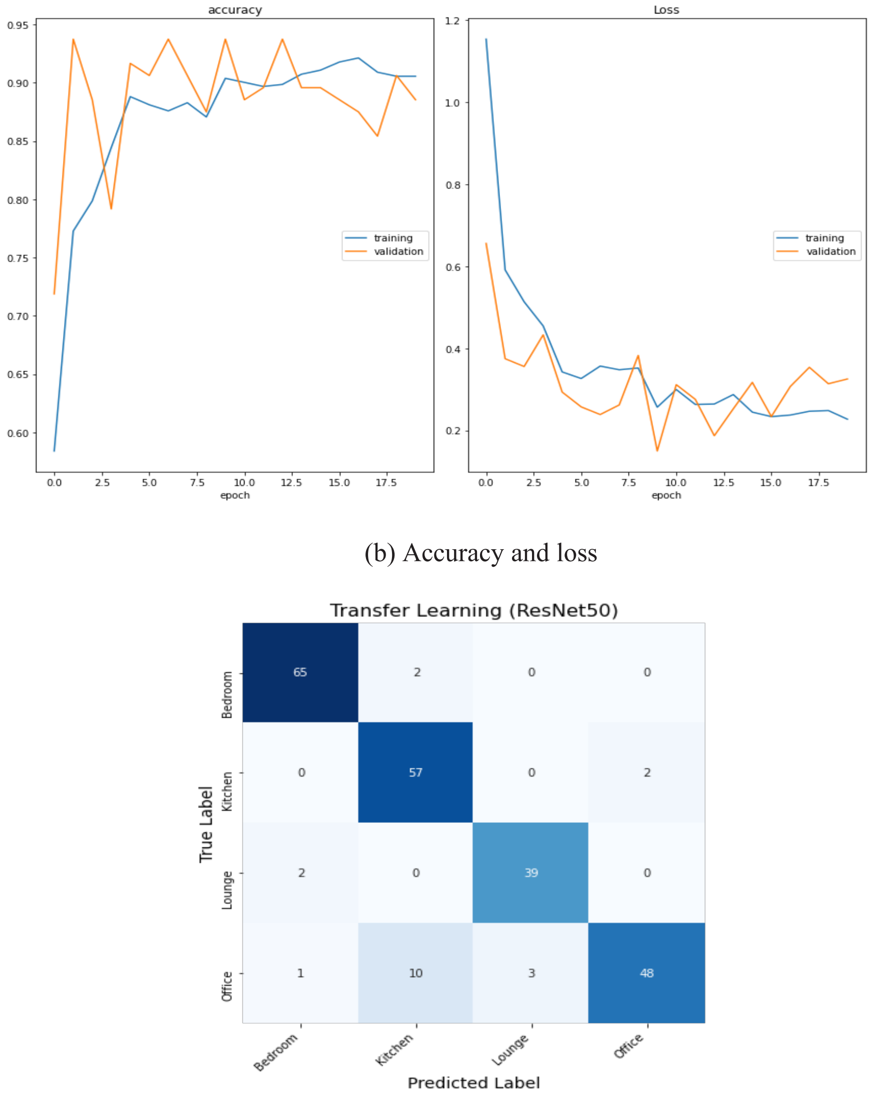
3.1.3 ResNet50

The ResNet50 CNN architecture addresses the issue of vanishing gradients in deep neural networks, allowing for effective training of very deep models. It provides good accuracy with relatively fewer parameters compared to VGG16, see Table 2. However, it is still computationally demanding, particularly during training.

Table 2. Deep Learning Models, with the Number of Convolutional Layers and Parameters

Deep learning model	**Number of convolutional layers**	**Number of parameters (Millions)**
VGG16	16	138
Inception-v	48	21.80
ResNet5	50	23.78

The results from training on our dataset and validating with a separate set of validation images are shown in Figure 12, with an accuracy of 91.27%. From the confusion matrix, we do not see a significant change in performance compared to the VGG16 or Inception-v3 models.



(b) Accuracy and loss

(b) Confusion matrix

Fig. 12. Results for training and validation of the ResNet50 CNN model

3.1.4 Conclusion

For simple place classification, where the task involves classifying images into a limited number of categories, all three architectures can perform well. However, considering the pros and cons, the choice may depend on available computational resources and deployment constraints.

If computational resources are limited, Inception-v3 might be a good option due to its computational efficiency while still achieving reasonable accuracy. If computational resources are more abundant, ResNet50 could be a suitable choice as it provides excellent accuracy, particularly when dealing with more complex image patterns.

VGG16 can be considered if simplicity and ease of implementation are important factors, although it may require more computational resources to deploy compared to Inception-v3.

In conclusion, there is no definitive "better" architecture among VGG16, Inception-v3, and ResNet50. The choice should be made based on a trade-off between computational efficiency, model complexity, and available resources, considering the specific requirements and constraints of your simple place classification task.

3.2 Implementation on Voyager mobile robot

In Figure 13 we see the 3-dimensional map created by the Voyager mobile robot. The lines were added to indicate the different rooms of the open area. The green line indicates the path the voyager robot travelled.

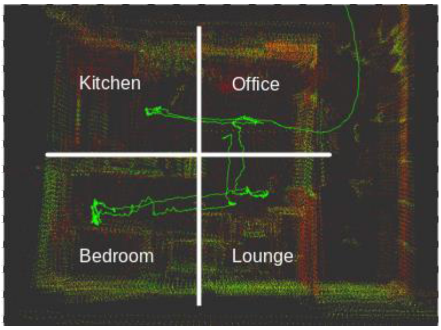
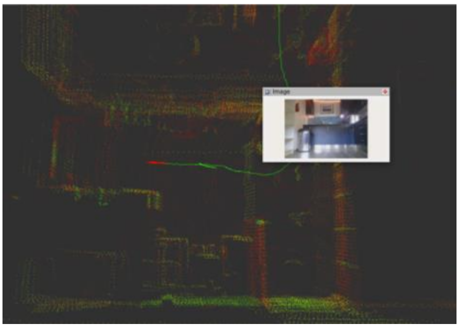
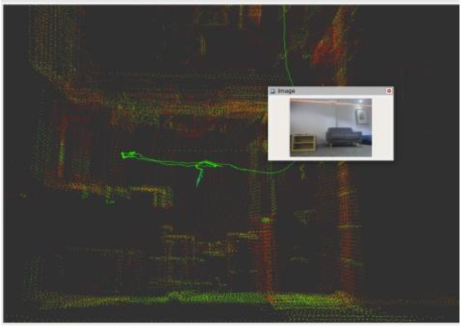
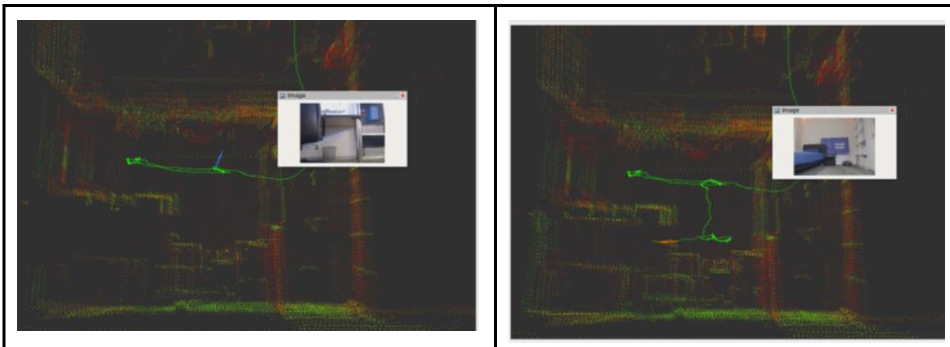


Fig. 13. Map generated of smart home by Voyager mobile robot

Voyager then provides an indication of which room it is in. The place recognition results, considering all three CNN models, are shown in Figure 14.

	
(a) Kitchen (red arrow)	(b) Lounge (green arrow)



(c) Study (blue arrow)

(d) Bedroom (orange arrow)

Fig. 14. Map generated of the smart home for place recognition at a given position

From the experimental implementation in section 2.4 we can see that the accuracies for the three models that we trained are close to each other with the ResNet model ranking the best out of three with an accuracy of 91.27% followed by VGG16 model with an accuracy of 86.90% and the Inception-v3 model with 81.66% accuracy. The VGG16 model is big to load. A single recognition can take up to 0.200s on average as seen in Table 1, this means that only 4-5 out of 30 frames per second (fps) can be captured and classified before running into processing issues.

The inception model can classify 10 out of every 30fps and the ResNet50 model can classify 2 out of 30fps. However, they are less accurate with the real-time implementation with ResNet50 performing slightly better than Inception v3. The Inception-v3 model is the least computationally intense model.

When the frames are not selected at this processing speed the system cannot deal with all the data tath it has to classify.

4 Conclusion

In this study we implemented Visual Place Recognition (VPR) as an additional functionality on the Voyager mobile robot, a platform developed in-house. To this end, we used the smart house within the CSIR robotics laboratory to capture the required dataset. The smart house is an open-plan small house with four rooms: lounge, office, kitchen and bedroom. This dataset was supplemented with additional images from ImageNet using a transfer learning approach.

For the VPR we investigated three convolutional neural networks (CNN): VGG16, Inception-v3 and ResNet50. Each CNN was trained and validated using the dataset from the smart house on Tensorflow. The results indicated that all three architectures are suitable for simple place recognition with computational resources as the deciding factor. Taking computation and accuracy into account ResNet50 performed the best followed by VGG16 and then Inception-v3. ResNet50 would be a better option if there are no restrictions on the computational resources as it provides excellent accuracy, especially for more complex images.

The place recognition algorithm was implemented on the Voyager mobile robot through a custom ROS2 node which allows communication with Tensorflow. Place recognition was tested using one of the *.bag* files. Voyager was able to successfully recognise all four rooms in the smart house for all three CNN models.

The additional place recognition functionality will allow Voyager to improve upon its localisation capabilities while also better interpreting its surroundings. This is a first step towards implementing an autonomous capability for the platform which will improve its adaptability and functionality within different environments.

References

1. Afif, M., Ayachi, R., Said, Y. and Atri, M., 2020. Deep learning based application for indoor scene recognition. Neural Processing Letters, 51, pp.2827-2837.
2. Kim, J. and Jun, H., 2008. Vision-based location positioning using augmented reality for indoor navigation. IEEE Transactions on Consumer Electronics, 54(3), pp.954-962.
3. Zhang, X., Wang, L. and Su, Y., 2021. Visual place recognition: A survey from deep learning perspective. Pattern Recognition, 113, p.107760.
4. Schubert, S., Neubert, P., Garg, S., Milford, M. and Fischer, T., 2023. Visual Place Recognition: A Tutorial. arXiv preprint arXiv:2303.03281.
5. . Chen, Z., Maffra, F., Sa, I. and Chli, M., 2017, September. Only look once, mining distinctive landmarks from convnet for visual place recognition. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 9-16). IEEE.
6. Masone, C. and Caputo, B., 2021. A survey on deep visual place recognition. IEEE Access, 9, pp.19516-19547.
7. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A. and Torralba, A., 2017. Places: A 10 million image database for scene recognition. IEEE transactions on pattern analysis and machine intelligence, 40(6), pp.1452-1464.
8. IEEE. Quattoni, A. and Torralba, A., 2009, June. Recognizing indoor scenes. In 2009 IEEE conference on computer vision and pattern recognition (pp. 413-420). IEEE.
9. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H. and He, Q., 2020. A comprehensive survey on transfer learning. Proceedings of the IEEE, 109(1), pp.43-76.
10. Tammina, S., 2019. Transfer learning using vgg-16 with deep convolutional neural network for classifying images. International Journal of Scientific and Research Publications (IJSRP), 9(10), pp.143-150.
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L., 2009, June. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.
12. . Jiang, Z.P., Liu, Y.Y., Shao, Z.E. and Huang, K.W., 2021. An improved VGG16 model for pneumonia image classification. Applied Sciences, 11(23), p.11185.
13. Joshi, K., Tripathi, V., Bose, C. and Bhardwaj, C., 2020. Robust sports image classification using InceptionV3 and neural networks. Procedia Computer Science, 167, pp.2374-2381.
14. Shabbir, A., Ali, N., Ahmed, J., Zafar, B., Rasheed, A., Sajid, M., Ahmed, A. and Dar, S.H., 2021. Satellite and scene image classification based on transfer learning and fine tuning of ResNet50. Mathematical Problems in Engineering, 2021, pp.1-18.
15. Ali, L., Alnajjar, F., Jassmi, H.A., Gocho, M., Khan, W. and Serhani, M.A., 2021. Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures. Sensors, 21(5), p.1688.
16. Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

17. Pardede, J., Sitohang, B., Akbar, S. and Khodra, M.L., 2021. Implementation of transfer learning using VGG16 on fruit ripeness detection. Int. J. Intell. Syst. Appl, 13(2), pp.52-61.
18. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).
19. Ali, L., Alnajjar, F., Jassmi, H.A., Gocho, M., Khan, W. and Serhani, M.A., 2021. Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures. Sensors, 21(5), p.1688.
20. He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
21. Shyamalee, T. and Meedeniya, D., 2022. Glaucoma detection with retinal fundus images using segmentation and classification. Machine Intelligence Research, 19(6), pp.563-580.
22. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. and Berg, A.C., 2015. Imagenet large scale visual recognition challenge. International journal of computer vision, 115, pp.211-252.
23. Macenski, S., Martín, F., White, R. and Clavero, J.G., 2020, October. The marathon 2: A navigation system. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 2718-2725). IEEE.