LoVi App: Android Application-based Image Classification for Low Vision

Mitra Sofiyati, Fandi Azam Wiranata, Wervyan Shalanannda*, Eueung Mulyana, Isa Anshori & Ardianto Satriawan

School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Jalan Ganesa No. 10, Bandung 40132, Indonesia

*E-mail: wervyan@telecom.stei.itb.ac.id

Abstract. In Indonesia, many people with visual impairments are drawing public attention to their rights as fellow humans. One of the limitations that individuals with low vision face is their ability to recognize objects and navigate their surroundings due to difficulties in visual perception. In this modern era, deep learning technologies, especially in image classification, can help people with low vision overcome these challenges. In this paper, we discuss a deep learning system that optimizes image classification on users' smartphones to enhance visual support for individuals with low vision. We present an Android-based app, LoVi, designed to assist users with low vision. Powered by core systems within Sherpa models (TrotoarNet, IndoorNet, and CurrencyNet), LoVi has three modes: outdoor, indoor, and currency. The LoVi application provides over 80% accuracy for navigation on sidewalks, indoor object recognition, and currency identification. TrotoarNet aids in sidewalk navigation, IndoorNet assists with indoor object identification, and CurrencyNet recognizes Rupiah banknotes. Additionally, low-vision users can receive voice feedback for further accessibility.

Keywords: *convolutional neural network; deep learning; image classification; low vision; smartphone*.

1 Introduction

As defined by the Ministry of Health of Indonesia, visually impaired people are those with partial vision impairments that cannot be fixed by ordinary means such as glasses. Low vision refers to individuals with partial vision impairments as members of the visually impaired, or *diffable netra* (DN). According to estimates, 1.5% of Indonesians are blind [1]. Further, a study conducted by the Vision Loss Expert Group found that Indonesia has the highest percentage of people with vision impairment of any country in the world [2]. It has been reported that 20% of all Indonesians with disabilities in the categories of ‘very poor’, ‘poor’, and ‘almost poor’ were blind, according to data from the Center for the Study of Disabilities, University of Indonesia (2010) [3]. The limitations caused by disabilities are possessed by people at the lower middle economic level.

Received January 19th, 2024, Revised June 27th, 2024., Accepted for publication August 17th, 2024.
 ISSN: 2337-5787, DOI: 10.5614/itbj.ict.res.appl.2023.18.2.3

Therefore, visually impaired people lack access to special tools to facilitate their daily lives. In the 2003 Indonesian census, 24.45% of disabilities were found in children and adolescents aged 0-18 and 21.4% in school-age children [4].

There are several smart sticks available to assist DNs, including WeWalk [5] and BriCane [6]. Both WeWalk and BriCane use ultrasonic sensors to detect obstacles within a 3 meter detection radius. Although this is effective for near-distance sensing, they cannot be used by DNs to navigate and perform remote sensing (>3 m) on roads. The lack of tools to recognize the objects around them also reduces the level of independence of DNs. Students, teachers, and professional musicians are some examples of DN productive activities that require high levels of independence. Because of limited support and facilities from their families or communities, DNs in Indonesia still cannot engage in these activities. Compared to similar products from abroad, BriCane is relatively inexpensive. For BriCane, the price ranges from IDR 2 to 3 million, while for imported canes like WeWalk the price is double that [6]. For people in a lower middle economic level society, IDR 2 to 3 million is not an insignificant amount. Based on data from the Ministry of Health's Research and Development Agency, disability and blindness prevalence are higher in the lower-income index [7, 8].

On the other hand, the mobile phone is almost a 'primary' need for everyone, including people with low vision. Deep learning and mobile phone technology may be combined to develop an assistance system for low-vision patients. According to a survey conducted by a non-profit organization [9], 71.4% of visually impaired respondents use smartphones. Moreover, the survey suggests that assistive devices are primarily developed on smartphones but still require sensors and embedded systems, which are more costly. Table 1 shows some systems that facilitate DN from previous studies. These systems serve several purposes, such as navigation and environmental recognition.

Based on the YOLO3 model, Lee *et al.* [10] developed an assistant system for the visually impaired. Object recognition accuracy was 96.46%, Korean text detection accuracy was 98%, and face detection accuracy was 72.6%. However, each subtask is performed independently. In [11], Nguyen et al. used a web-based system to detect objects with sound feedback for low vision. With SSD-MobileNetV2, the system gets a fast, small, and specific model optimized for mobile implementation. The model's confidence score decreased as the number of detected objects increased. Won *et al.* [12] developed a transfer learning-based object detection system that uses Faster Region-Convolutional Neural Network Inception V2 (Faster R-CNN) and Single Shot Detector MobileNetV1. In the analysis, the R-CNN model produced better results for mAP but took longer to infer than SSD according to the results.

Table 1 Previous work of assistant system for visually impaired people.

Category	Technology	Accuracy	Reference
Object detection, text detection, face detection	YOLO3	96.46%; 98%; 72.6%	[10]
Object detection (people, cars, dogs)	SSD-MobileNetV2	>90%	[13]
Object detection (40 objects)	SSD-MobileNetV1, Faster R-CNN Inception V2	89.61%	[14]
Object detection (technology, text, person, plant, color)	CNN	91.5%	[15]
Object recognition (outdoor objects)	ResNet50, Inception V3, VGG19	94.78%; 96.39%; 90.88%	[16]
Object detection	YOLO	85.5%	[17]
Obstacle detection	Adaboost	84.7%	[18]
Obstacle detection, navigation	Sensor, RFID, GPS (NavCane)	–	[19]
Navigation: crosswalk detection	YOLOv4, RGB Camera	93.46%	[20]
Navigation system	NFC, RFID, iBeacon	–	[21]

Reference [15] shows that CNN can be used indoors and outdoors to classify objects with high precision. Reference [16] only used CNN to detect outdoor objects. Inception V3, ResNet50, and VGG19 were the CNN models used to build the system. The accuracy of ResNet50 was 94.78%, that of Inception V3 was 96.39%, and that of VGG19 was 90.88%. Many pre-trained models are available for object detection and classification, including MobileNet, VGG, ResNet, Inception, and YOLO. A pre-trained model may perform differently in terms of accuracy, latency, size, etc. Model architecture parameters, such as Adam, RMSprop, and SGD [16], also affect accuracy. According to [22], YOLACT image classification library provides a higher confidence level than TensorFlow but with a large delay.

In addition to object detection, deep learning can also be used to develop a navigation system for low vision. Developers can explore deep learning, particularly object detection, to warn of obstacles while navigating [19, 23] and it can be used for navigation as well. Reference [20] detects crosswalks and recognizes traffic lights to create a navigation system for the visually impaired. A navigation system called TARSIUS [21] also provides code information via vibration, sounds and vocal instructions like ‘turn right’. Reference [24] used the same concept to develop an assistive cane with three ultrasonic sensors.

We aimed to increase DN’s independence by developing the LoVi application, a smart assistant for low vision (LoVi), utilizing deep convolutional neural networks. Sherpa refers to Tibetans who act as climbing guides in the Himalayas

[25]. Based on the same spirit, the LoVi application aims to be a reliable guide for DNs or LoVi because it is embedded with advanced technologies from the present era of machine learning (ML).

2 LoVi Application Design

This paper presents the LoVi image classification application for Android. Using Android and artificial intelligence technology, LoVi was designed to help people with low vision perform daily tasks. Figure 1 illustrates its two subsystems: the LoVi mobile interface and the Sherpa model based on LoVi deep convolutional neural networks. LoVi displays three modes: outdoor, indoor, and currency. In addition, the Sherpa model needs to be capable of classifying images based on the mode selected in the application. Using TrotoarNet for outdoor use, IndoorNet for indoor use, and CurrencyNet for currency use, smartphones can run LoVi applications with lower power consumption. LoVi displays the three modes it offers users when they open it for the first time. The LoVi app's first interface guides the user through the process of classifying images (videos split into frames) detected by the smartphone. Based on the selected mode and real-time, the Sherpa model will then generate predictions based on the accuracy values and categories. Sound is used to communicate prediction results that meet a minimum accuracy threshold. In the event the prediction accuracy exceeds the threshold, a sound that mentions the category with the highest accuracy is played. Then, if the user presses the back or exit button, the LoVi application is terminated.

2.1 User Interface of Lovi App

Figure 2 illustrates the LoVi application user interface. The first interface displays the LoVi application logo. The second interface displays three buttons based on the user's mode option. It appears for a few seconds and switches to the second interface when the application is opened. For mode selection, the application activates the guidance voice. The application will switch to the next activity after pressing the mode button. This activity uses the smartphone camera to capture real-time frames that the Sherpa model uses as input for inference. The secondary window (right side) reflects only the top three accuracies of the categories. If the category with the highest accuracy meets the threshold, the feedback sound will be activated. The TrotoarNet, IndoorNet, and CurrencyNet models were first evaluated with the same threshold value. As a result of observing the trial, we discovered that the results tended to fall into a particular category. Thus, the thresholds were adjusted accordingly.

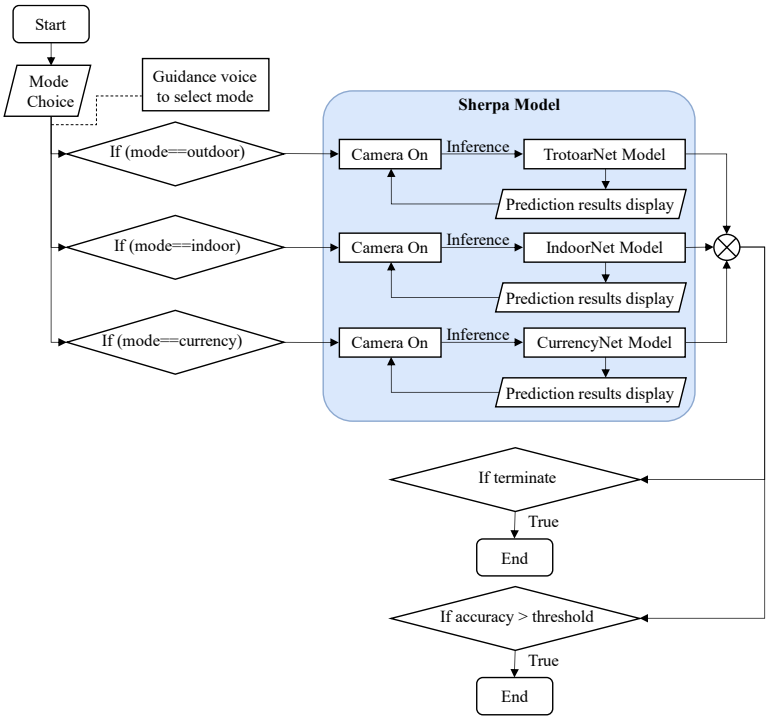


Figure 1 Flowchart of LoVi app.

Figure 2 User interface of LoVi app.

2.2 Preprocessing Dataset

The dataset was preprocessed into 224 x 224 dimensions and divided into 60% training, 30% validation, and 10% testing data. Tests were performed on the pre-trained Sherpa model to establish a baseline model for testing the NN-

Architecture. The NN-Architecture Sherpa model and pre-trained model testing process are shown in Figure 3.

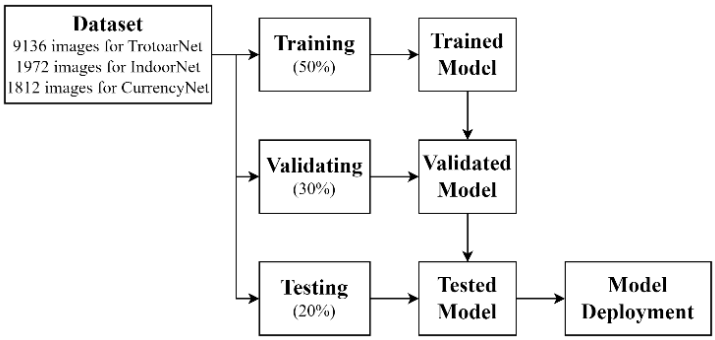


Figure 3 The flow of testing the Sherpa model along with the dataset distribution scheme, from training and testing to the final architecture.

In LoVi, the final architecture of the Sherpa model was derived from these test types. Table 2 shows variations in testing the pre-trained and Sherpa models, including test accuracy, number of parameters, latency, frames per second (FPS), and model size. The only parameters that were used to measure the NN-Architecture were accuracy and size. Table 3 shows a list of labels for each model and the average accuracy of the testing dataset.

Table 2 Variations in testing pre-trained models and NN-architecture.

Testing	Parameter	Variation
Pre-trained model	-	EfficientNet, DenseNet 169, Inception-Resnet version 2, Inception version 3, MobileNet small, ResNet 50 version 1, ResNet 101 version 1, ResNet 152 version 1, VGG 16, VGG 19, dan Xception
NN-Architecture	Unit size	16, 32, 64, 128, 256, 512, 1024, 2048
	Activation function	hard_sigmoid, softsign, softmax, sigmoid, ReLU
	Learning-rate	0.00001, 0.0001, 0.001, 0.01, 0.1
	Optimizer	SGD, RMSprop, Adam, Adadelta, Adagrad, Adamax, Nadam
	Loss function	mean_squared_error, mean_absolute_error, mean_absolute_percentage_error, mean_squared_logarithmic_error, kullback_leibler_divergence, cosine_proximity, squared_hinge, hinge, categorical_hinge, logcosh, categorical_crossentropy, binary_crossentropy, poisson

Table 3 Details of the deep learning model on Sherpa.

ModelID	Function	Label	Dataset
IndoorNet	Helping LoVis to navigate outdoors, especially on sidewalks	bottle, fruit, glass, scissors, keyboard, chair, laptop, table, monitor, mouse, person, toothpaste, plate, knife, mobile phone, spoon, toothbrush, bag, and unrecognizable object	1972 images
TrotoarNet	Helping LoVis to identify objects in the room, such as bottles, smartphones, plates, etc.	turn right, turn left, stop, and straight	9136 images
CurrencyNet	Helping LoVis to classify rupiah currency (paper type) according to its nominal value	IDR 1.000, IDR 2.000, IDR 5.000, IDR 10.000, IDR 20.000, IDR 50.000, IDR 100.000, and unrecognizable object	1842 images

A total of five variables were tested in the NN-Architecture test: units, unit size, activation function, learning rate, optimizer, and loss function. The pre-trained model and the NN-Architecture models were tested on all ModelIDs (TrotoarNet, IndoorNet, and CurrencyNet). In order to create the Sherpa model, 150x training and 150x testing were required. The training-to-testing process was built into the same program to train and test Sherpa models, import libraries, and then read preprocessed datasets for training and validation. This dataset was inserted into the training process. A deep convolutional neural network (CNN) model architecture was created before training began, and a fully connected layer was added before the output layer. As a result of completing the training process, the program saved the model in *.h5 format and the training process graph in the specified directory and then proceeded to test the model.

2.3 Sherpa Model Deployment

The developed Sherpa model became the LoVi application's core, consisting of TrotoarNet, IndoorNet, and CurrencyNet. TrotoarNet categorizes sidewalks into four categories, i.e., turn right, turn left, go straight, and stop. IndoorNet classifies objects in a room, including laptops, scissors, glasses, and more. Then CurrencyNet classifies rupiah banknotes (paper type) from IDR 1,000 to 100,000, and one additional category 'no money'. Coins were not included in the LoVi application due to their relative ease of identification. In order to use the Sherpa model in the LoVi application, we converted the model to an edge-supported format following the steps in Figure 4. A dotted line in Figure 4 indicates that this process does not occur continuously but is only considered in optimizing application functions. Using a CNN model architecture, we built the Sherpa

model. We converted the model into a lite version using TensorFlow Lite. Additionally, the model was optimized to reduce the deployment size in LoVi. Our optimization consisted of changing the data type of the model that was deployed in the application to an 8-bit integer to reduce the size and processing time.

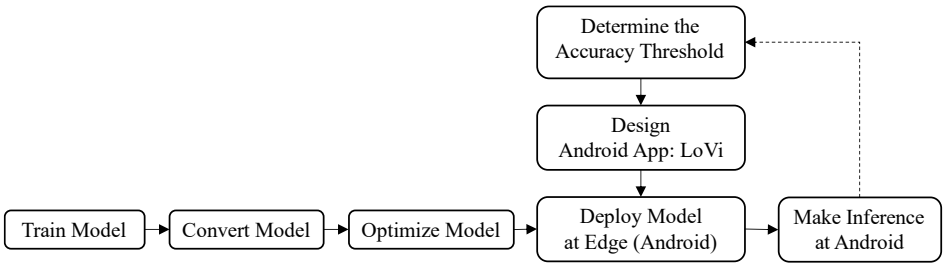


Figure 4 The flow of Sherpa model deployment into the LoVi application.

3 Results and Discussion

3.1 Sherpa Model Testing

The Sherpa model was tested in three stages: (1) determining the best pre-trained model and the baseline, (2) determining the best neural network architecture (NN-Architecture), and (3) testing the final architecture of the Sherpa model. The first stage is detailed in Appendix A, while the rest is included in this subsection.

3.1.1 NN-Architecture

In the NN-Architecture test, Table 4 shows the baseline Sherpa model. This baseline was obtained from the analysis of the pre-trained model test so that the optimum pre-trained model was obtained, which was VGG16, taking into account test accuracy, latency, model size, and frame rate. This subsection of the test used the default parameters of the NN-Architecture. As described in Appendix A, this parameter corresponds to the fully connected layer architecture. In testing the NN-Architecture, we sought to determine the optimal unit size, activation function, learning rate, optimizer, and loss function parameters to outperform the Sherpa baseline. Figure A.1 illustrates the accuracy and size of the model when unit size variation was carried out on ModelID, using the pre-trained VGG16 model that was determined in the previous test. As the unit size increased, the model accuracy and size increased as well. Depending on the type of ModelID, accuracy varied, while model size tended to remain the same. We trained each model 24 times to get this data. Compared to the baseline Sherpa model (unit size = 1024), unit size 128 was most optimal because accuracy was maintained at >90% while model size decreased by 69% (~180 MB) to 81 MB.

Figure 5 shows the activation functions: sigmoid, hard_sigmoid, softsign, and softmax. The test showed that sigmoids on fully connected layers resulted in accuracy better than the baseline Sherpa model (activation function = ReLU-ReLU- Softmax), which were +4.5% in IndoorNet, +6% in TrotoarNet, and +3% in CurrencyNet. In addition to the baseline Sherpa model, none of the other activation functions increased the accuracy for the three ModelIDs. Also, we conducted 12 times model training. With 66 times total model training, Figure 6 shows the learning rate, optimizer, and loss function tests. The accuracy from the variation of the learning rate did not exceed the accuracy from the baseline Sherpa model (learning rate = 0.0001). Furthermore, the variations in the optimizer and loss function in Figure 6(b) and 6(c) did not result in better accuracy than the baseline Sherpa model (optimizer = RMSprop, loss function = categorical_cross-entropy).

Table 4 Baseline Sherpa model.

ModelID	Pre-trained model	Val_Acc (%)	Test_Acc (%)	Latency (ms)	imag es/s	NOPs (million)	Model size (MB)
IndoorNet	VGG 16	98.08	94.56	76	13	41.4	261
TrotoarNet	VGG 16	85.70	91.20	26	38	41.4	261
CurrencyNet	VGG 16	96.32	94.79	42	24	41.4	261

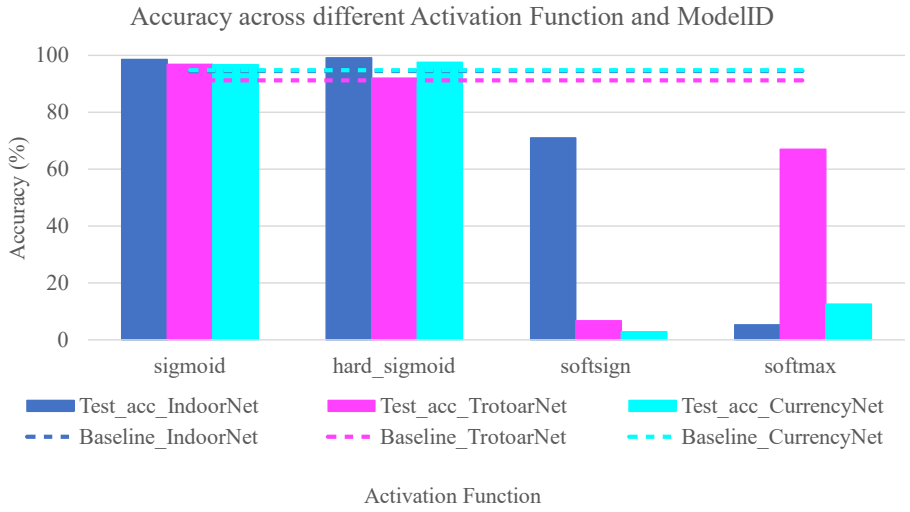


Figure 5 Accuracy of the activation function test results.

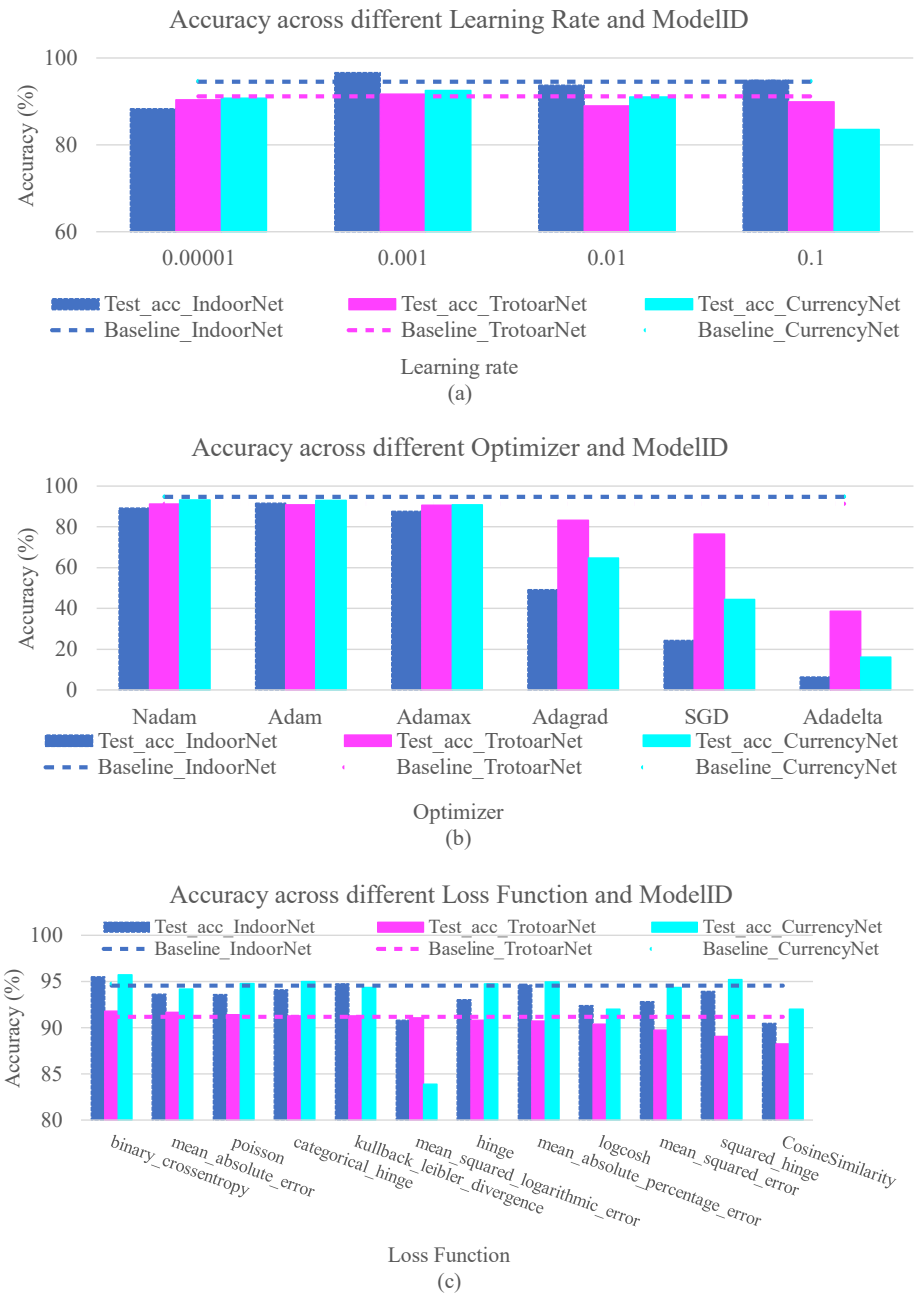


Figure 6 Accuracy of (a) learning rate, (b) optimizer, (c) loss function, against ModelID.

3.1.2 Sherpa Model Analysis

In order to get the baseline Sherpa model, we made a ranking based on four aspects: model size, latency versus NoPs, latency versus accuracy, and FPS, then averaged them to get the final ranking. A Sherpa model with the smallest NoP size is needed to prevent overloading the smartphone's processing. The NoPs in Figure A.4(b) are proportional to the model size. Table 5 shows the top 3 ranking models, which were VGG16, VGG19, and MobileNet.

Table 5 Ranking results of pre-trained models for determining the baseline Sherpa model.

Pre-trained model	Model size	*Latency* vs NoPs	*Latency* vs Accuracy	FPS	Average
VGG 16	1	1	1	1	1.00
VGG 19	2	2	2	2	2.00
MobileNet small	3	3	4	3	3.25
Inception V3	5	5	7	4	5.25
Xception	8	6	5	5	6.00
EfficientNet	4	4	10	7	6.25
ResNet 50 v1	9	7	3	6	6.25
DenseNet 169	7	8	9	9	8.25
ResNet 101 v1	10	9	6	8	8.25
Inception-*Resnet* V2	6	10	11	11	9.50
ResNet 152 V1	11	11	8	10	10.00

Pre-trained models are better if they are more accurate and have lower latency. As shown in Figure A.2, 6 out of the 11 pre-trained models used in the Sherpa model yielded greater than 80% accuracy, i.e. MobileNet, ResNet50, ResNet101, ResNet152, VGG 16, and VGG 19. Table 5 shows the ranking results based on latency versus accuracy. If we want at least 10 frames per second, we can tolerate 100 ms of latency with the Sherpa model. In accordance with these specifications, only two pre-trained models met the specifications for all ModelIDs (Figure A.1). VGG16 performed the best in all aspects. Testing the pre-trained model was used as a baseline for testing the Sherpa model, which is shown in Table 6 with the architecture and parameters of the fully connected layer shown in Figure A.4(a) and Table 6.

Table 6 NN-Architecture from the baseline Sherpa model.

Parameters	Architecture
Pre-trained model	VGG16
Unit size	1024
Activation function	ReLU, ReLu, Softmax
Learning-rate	0.0001
Optimizer	RMSprop
Loss function	categorical_crossentropy

The final architecture was obtained from the results of the NN-Architecture test, which can be seen in Table 7. The final architecture was the reference for training the final Sherpa model, which became the core of the LoVi application. Based on the final architecture training results, a significant increase in the performance of the Sherpa model was obtained when compared to the baseline (Table 8), especially in the accuracy and size of the model. We used two metrics in terms of accuracy: testing accuracy and validation accuracy. Training accuracy measures how well the model is learning from the training data. It was calculated as the percentage of correctly predicted instances. As with training accuracy, validation accuracy evaluates the model's generalization ability to new, unseen data. It was calculated similarly to training accuracy.

Table 7 Comparison of the final and baseline Sherpa model architecture.

Parameters	Final	Baseline
Pre-trained model	VGG16	VGG16
Units	128	1024
ActivationFunc	Sigmoid (3x)	ReLU, ReLu, Softmax
learning-rate	0.0001	0.0001
Optimizer	RMSprop	RMSprop
loss function	categorical_crossentropy	categorical_crossentropy

Table 8 The results of the final and baseline architecture of the Sherpa model.

Result	IndoorNet		TrotoarNet		CurrencyNet	
	Final	Baseline	Final	Baseline	Final	Baseline
Testing accuracy (%)	95.5	94.56	96.79	91.20	93.35	94.79
Validation accuracy (%)	96.64	98.08	86.45	85.70	96.32	96.32
Latency (ms)	77.08	76.42	24.75	26.36	38.16	41.88
Images/s	13	13	40	38	26	24
NoPs (million)	17.9	41.5	17.9	41.5	17.9	41.5
h5 size (MB)	81	261	81	261	81	261

3.2 LoVi App Testing

Figure 7 shows the LoVi application's test result with TrotoarNet, IndoorNet, and CurrencyNet. As shown in Figure 8, the Sherpa model's accuracy decreased after conversion to the lite version before being deployed in LoVi. We know from Figure 8 (a) that TrotoarNet was accurate in all four categories before converting to the lite version. While the lite version model yielded 5.13% accuracy loss, the quantization process resulted in 2.66% accuracy loss. However, TrotoarNet still had an accuracy value above 80% on average. Figure 8 (b) shows that IndoorNet only lost 1.38% accuracy when converted to the lite version model. IndoorNet's accuracy was also reduced by 2.10 % by quantization, from 91.74% to 89.65%. According to Figure 8 (c), the average accuracy of CurrencyNet after conversion

and optimization was still over 80%. Only the image of an IDR 20,000 bill had an accuracy below 80%.

Hence, based on three results of testing the model's accuracy before deployment, it is evident that quantification did not significantly decrease the accuracy value. As a result, the quantized model was still suitable for TrotoarNet, IndoorNet, and CurrencyNet, as the accuracy per category remained above 80% in almost all categories.

Our performance test included reviewing the model's size and inference time after quantization in the LoVi application and its accuracy. It is possible to see the model size directly in the existing model details. Figure 9 (a) shows how the Sherpa model (VGG16 and MobileNet) is used in the LoVi application to calculate the inference time. In the graph, the model's size is also shown before and after quantization. The figure shows a 73.56% reduction in model size before and after quantization. Furthermore, the inference time was reduced by 67.65%. According to Figure 9 (b), LoVi's performance on a smartphone was linear over time. For approximately five minutes, the smartphone consumed about 17 to 22 mAh. The TrotoarNet, IndoorNet, and CurrencyNet models, trained on MobileNet and quantized into an 8-bit integer model, consumed the same amount of smartphone power. However, the inference time differed by more than 50%.

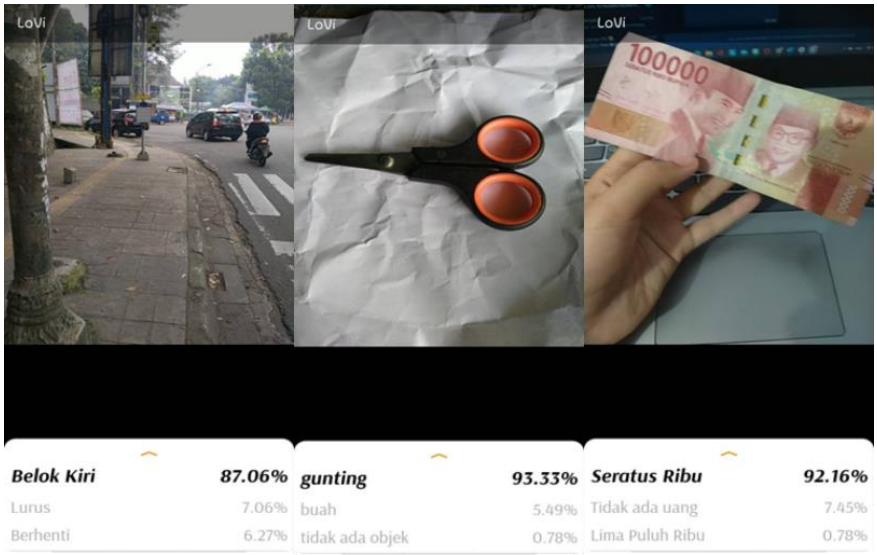


Figure 7 Recognition by LoVi app. Translations of the text from the left image: *Belok Kiri* [Turn Left], *Lurus* [Straight Ahead], *Berhenti* [Stop]; from the middle image: *gunting* [scissors], *buah* [fruit], *tidak ada objek* [no object detected]; from the right image: *Seratus Ribu* [One Hundred Thousand], *Tidak ada uang* [No money detected]; *Lima Puluh Ribu* [Fifty Thousand].

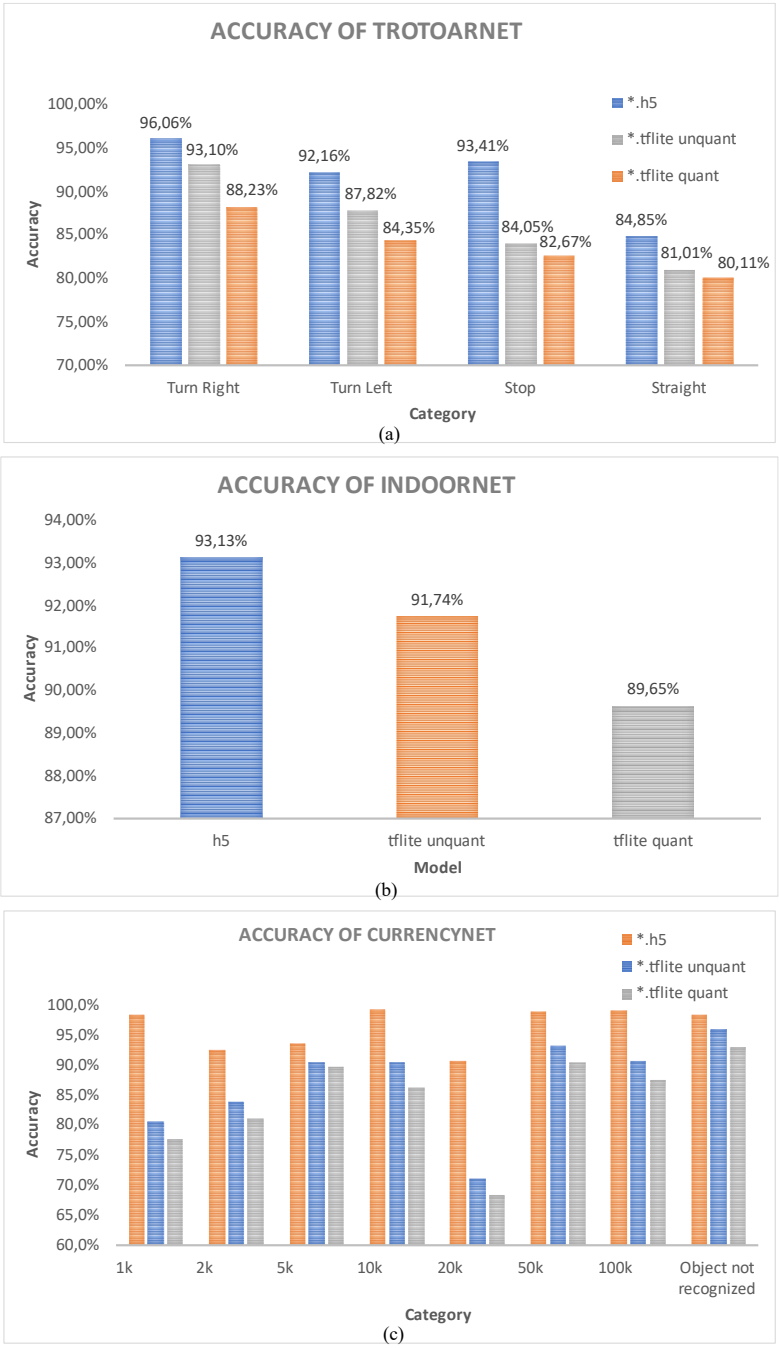
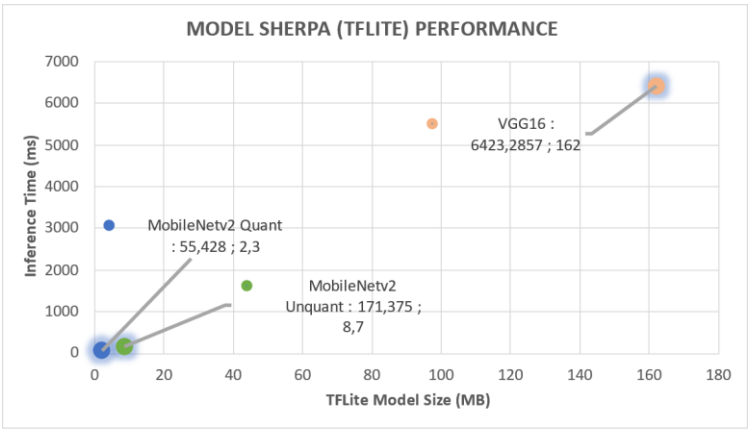
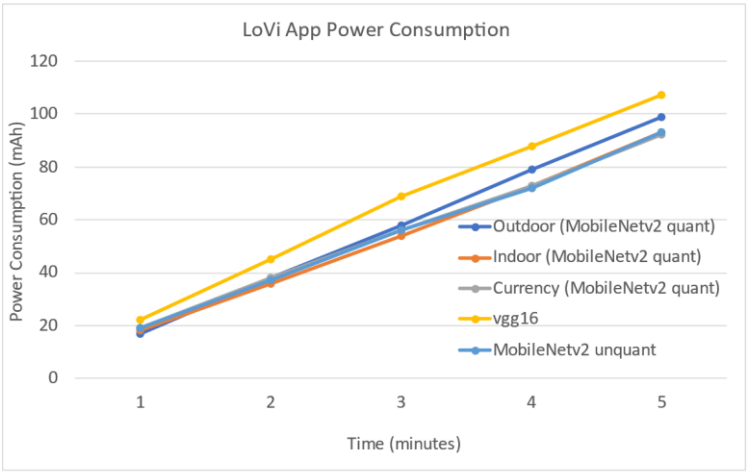


Figure 8 Decreased accuracy of the Sherpa model due to the conversion process for: (a) TrotoarNet, (b) IndoorNet, (c) CurrencyNet.



a



b

Figure 9 Test results of (a) the inference time and size of the Sherpa model, and (b) the smartphone power consumption by the LoVi app.

4 Conclusion

LoVi was developed to make outdoor navigation easier for people with low vision. It can also identify indoor objects and Indonesian banknotes (paper type). Approximately 80% of the model used in the LoVi application was accurate. Moreover, the LoVi application allows users to hear prediction results via voice, making it easy for low-vision users.

In terms of performance, the pre-trained model and NN-Architecture test variations differed depending on the ModelID (IndoorNet, TrotoarNet, and

CurrencyNet) created. Based on the NN-Architecture test, there were five variations: (1) units/unit size, with optimum performance when unit size = 128, which reduced 69% (-180MB) of the baseline model size while maintaining a test accuracy of >90% (91.42% on IndoorNet, 91.25% on TrotoarNet, and 93.76% on CurrencyNet); (2) activation function, with optimum performance when the activation function = sigmoid, with test accuracy increasing significantly compared to the Sherpa baseline model, i.e., 98.64% (+4.08%) on IndoorNet, 96.88% (+5.68%) on TrotoarNet, and 96.8% (+2.01%) on CurrencyNet. The Sherpa model performed better when the baseline parameters were used, i.e., learning rate = 0.0001, optimizer = RMSprop, and loss function = categorical_cross-entropy, in testing the variables (3) learning rate, (4) optimizer, and (5) loss function.

Inference time was reduced by 67.65% as a result of the optimization results used in the development of the TFLite model. It is therefore a relatively good choice for the LoVi application. The chosen model is based on transfer learning from MobileNet, quantized into 8-bit integers.

Acknowledgments

This work was partially supported by the School of Electrical Engineering and Informatics, Bandung Institute of Technology.

References

[1] *Press Release: Pertuni's Strategic Role in Empowering the Blind in Indonesia)*, Pertuni (Persatuan Tunanetra Indonesia), 04 Maret 2017. (Text in Indonesian) Available: https://pertuni.or.id/siaran-pers-peran-strategis-pertuni-dalam-memberdayakan-tunanetra-di-indonesia/ (08 July 2023).

[2] Vision Loss Expert Group, *Magnitude, Temporal Trends, and Projections of the Global Prevalence of Blindness and Distance and Near Vision Impairment: A Systematic Review and Meta-analysis*, Lancet Glob Healt, **5**(9), pp. e888-e897, 2017.

[3] Irwanto, E.R., Kasim, A., Fransiska, M., Lusli & Siradj, O., *Situation Analysis of People with Disabilities in Indonesia: A Desk-Review*, Pusat Kajian Disabilitas Fakultas Ilmu-ilmu Sosial dan Politik, Universitas Indonesia, Depok, 2010. (Text in Indonesian)

[4] Garina, L., *Prevalence, Characteristics, and Health Services for Children with Special Needs in Indonesia*, 2012.

[5] WeWALK, *WeWalk*, Available: https://wewalk.io/en/.

[6] Tempo, *Introducing BriCane, Modern Cane for the Blind Made in Bandung*, tempo.co, 31 October 2020. (Text in Indonesian). Available: https://difabel.tempo.co/read/1400939/perkenalkan-bricane-tongkat-difabel-netra-kekinian-buatan-bandung (08 July 2023).

[7] Ministry of Health of the Republic of Indonesia, *Disability Situation*, December 2014. (Text in Indonesian) Available: https://pusdatin.kemkes.go.id/download.php?file=download/pusdatin/buletin/buletin-disabilitas.pdf (07 July 2023).

[8] Ministry of Health of the Republic of Indonesia, *Situation of Vision Impairment and Blindness*, October 2014. (Text in Indonesian). Available: https://pusdatin.kemkes.go.id/download.php?file=download/pusdatin/infodatin/infodatin-Gangguan-penglihatan-2018.pdf (08 July 2023).

[9] Karkar, A. & Al-Maadeed, S., *Mobile Assistive Technologies for Visual Impaired Users: A Survey*, in International Conference on Computer and Applications (ICCA), Doha, 2018.

[10] Lee, C.S., Lee, J.I. & Han, S.E. *Deep Learning Based Mobile Assistive Device for Visually Impaired People*, in International Conference on Consumer Electronics-Asia (IOCE-Asia), 2021. DOI: 10.1109/ICCE-Asia53811.2021.9641925

[11] Nguyen, H., Nguyen, M., Yang, S. & Le, H., *Web-based Object Detection and Sound Feedback System for Visually Impaired People*, in International Conference on Multimedia Analysis and Pattern Recognition (MAPR), 2020. DOI: 10.1109/MAPR49794.2020.9237770

[12] Wei-Cheng, W., Yoke-Leng, Y. & Kok-Chin, K., *Object Detection and Recognition for Visually Impaired Users: A Transfer Learning Approach*, in 2nd International Conference on Artificial Intelligence and Data Sciences (AiDAS), 2021. DOI: 10.1109/AIDAS53897.2021.9574220

[13] Nguyen, H., Nguyen, M., Nguyen, Q., Yang, S. & Le, H. *Web-based Object Detection and Sound Feedback System for Visually Impaired People*, in International Conference on Multimedia Analysis and Pattern Recognition (MAPR), 2020. DOI: 10.1109/MAPR49794.2020.9237770

[14] Won, W.-C., Yong Y.-L. & Khor, K.-C., *Object Detection and Recognition for Visually Impaired Users: A Transfer Learning Approach*, in 2nd International Conference on Artificial Intelligence and Data Sciences (AiDAS), 2021. DOI: 10.1109/AIDAS53897.2021.9574220

[15] Caballero, A.R. Catli, K.E.I. & Babierra, A.G.F., *Object Recognition and Hearing Assistive Technology Mobile Application using Convolutional Neural Network*, in International Conference on Wireless Communication and Sensor Networks (icWCSN), New York, 2020.

[16] Parikh, N., Shah, I. & Vahora, S., *Android Smartphone based Visual Object Recognition for Visually Impaired using Deep Learning*, in International Conference on Communication and Signal Processing, 2018. DOI: 10.1109/ICCSP.2018.8524493.

[17] Vaidya, S., Shah, N., Shah, S. & Shankarmani, R., *Real-Time Object Detection for Visually Challanged People*, in 4th International Conference on Intelligent Computing and Control Systems (ICICCS), 2020. DOI: 10.1109/ICICCS48265.2020.9121085.

[18] Sunitha, M. R, F. Khan, G. Ghatge R & H. S, *Object Detection and Human Identification using Rasberry Pi*, 1st International Conference on Advances in Information Technology (ICAIT), 2019. DOI: 10.1109/ICAIT47043.2019.8987398.

[19] Meshram, V.V., Patil, K., Meshram V.A. & Shu, F.C. *An Astute Assistive Device for Mobility and Object Recognition for Visually Impaired People*, IEEE Transactions on Human-Machine Systems, **49**(5), pp. 449-460, 2019.

[20] Tian, S., Zheng, M., Zou, W., Li. X. & L. Zhang, *Dynamic Crosswalk Scene Understanding for the Visually Impaired*, IEEE Transactions on Neural Systems and Rehabilitation Engineering, **29**, pp. 1478-1486, 2021.

[21] Mataro, T.V., Masulli, F., Rovetta, S., Cabri, A., Traverso, C., Capris, E., & Torretta, S., *An Assistive Mobile System Supporting Blind and Visual Impaired People when Are Outdoor*, in IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI), 2017. DOI: 10.1109/RTSI.2017.8065886.

[22] Sivate, T.M., Pillay, N., Moorgas, K. & Singh, N., *Autonomous Classification and Spatial Location of Objects from Stereoscopic Image Sequences for the Visually Impaired*, in 2022 International Conference on Electrical, Computer and Energy Technologies (ICECET), 2022.

[23] Widayani, A., Kusuma, H. & Purwanto, D., *Visually Impaired Person Detection Using Deep Learning for Dangerous Area Warning System*, in 2022 International Seminar on Intelligent Technology and Its Applications (ISITIA), 2022.

[24] Basheshankar, A., Lande, A., Nandeshwar, A., Sarkar, Udapure, A. & Chandankhede, D.H., *Assistive Cane for Visually Impaired People*, in 2022 10th IEEE International Conference on Emerging Trends in Engineering & Technology Signal and Information Processing (ICETET-SIP-22), 2022.

[25] Wikipedia, *Sherpa People*, Available: https://en.wikipedia.org/wiki/Sherpa_people. (8 March 2022).

[26] *TensorFlow*, Available: https://www.tensorflow.org/lite. (01 March 2022)

[27] *Keras*, Available: https://keras.io/. (01 March 2022)

[28] Ioffe, S. & Szegedy, C., *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, in 32nd International Conference on International Conference on Machine Learning, 2015. DOI: 10.48550/arXiv.1502.03167.

Appendix A. Pre-Trained Model Evaluation

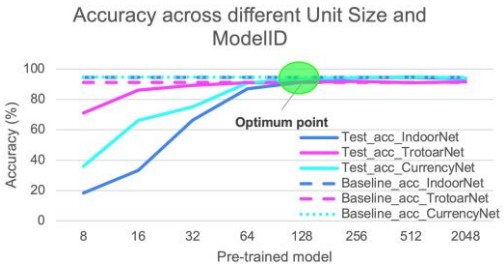
The images used in this experiment had size 224 x 224 pixels, batch_size 128, and epoch 25. The pre-trained models were all trained with image size 224 x 224 pixels. To prevent overfitting, we used relatively small epoch values and relatively large batch sizes, since batch size and epoch cannot be determined from the data. NN-Architecture and Sherpa's final architecture were benchmarked using a list of pre-trained models, as provided in Table A.1. In building the pre-trained models, we used ImageNet weights and TensorFlow libraries, excluding EfficientNet. Sherpa offers three models: IndoorNet, TrotoarNet, and CurrencyNet, each with a distinct function, label, and dataset. IndoorNet recognizes 19 different types of objects. TrotoarNet and CurrencyNet categorize images into four and eight categories, respectively.

Table A.1. List of pre-trained image classification models from ImageNet for model training [26][27].

Pre-trained Models	Abbreviation	ML Library	Input Size
EfficientNet	EfcNet	Efficientnet	224 x 224
DenseNet 169	DnsNet_169	Tensorflow	224 x 224
Inception-Resnet version 2	Inc_Res_v2	Tensorflow	224 x 224
Inception version 3	Inc_v3	Tensorflow	224 x 224
MobileNet small	Mob_v3_sml	Tensorflow	224 x 224
ResNet 50 version 1	Rs_v1_50	Tensorflow	224 x 224
ResNet 101 version 1	Rs_v1_101	Tensorflow	224 x 224
ResNet 152 version 1	Rs_v1_152	Tensorflow	224 x 224
VGG 16	Vgg_16	Tensorflow	224 x 224
VGG 19	Vgg_19	Tensorflow	224 x 224
Xception	Xcp_v1	Tensorflow	224 x 224

In Figure A.1(a), we compare pre-trained models to static variables for all ModelIDs. The NoPs were based on the pre-trained model used and the testing results. All ModelIDs had the same NoP value. ResNet152 had the highest NoPs at 162.3 million, while VGG16 had the lowest, at 41.5 million. There was a difference in complexity and classification ability. Figure A.1(b) illustrates that IndoorNet, TrotoarNet, and CurrencyNet use different pre-trained models. IndoorNet, TrotoarNet, and CurrencyNet have identical models with the same pre-trained models at 261 MB, 281 MB, and 407 MB, respectively, while ResNet152 has a larger model, 1017 MB. NoPs and model sizes are generally similar. The more complex the model, the larger the model size.

Figure A.2 illustrates the results of the latency versus accuracy test on all pre-trained models. According to the ModelID tests, the top-three optimum models, VGG16, VGG19, and MobileNet, had over 90% accuracy and varying latency. It is worth mentioning that the VGG16 and VGG19 generally had a latency of 100 ms.



(a)

(b)

Figure A.1. (a) Accuracy, (b) model size from the unit size test, and Model ID.

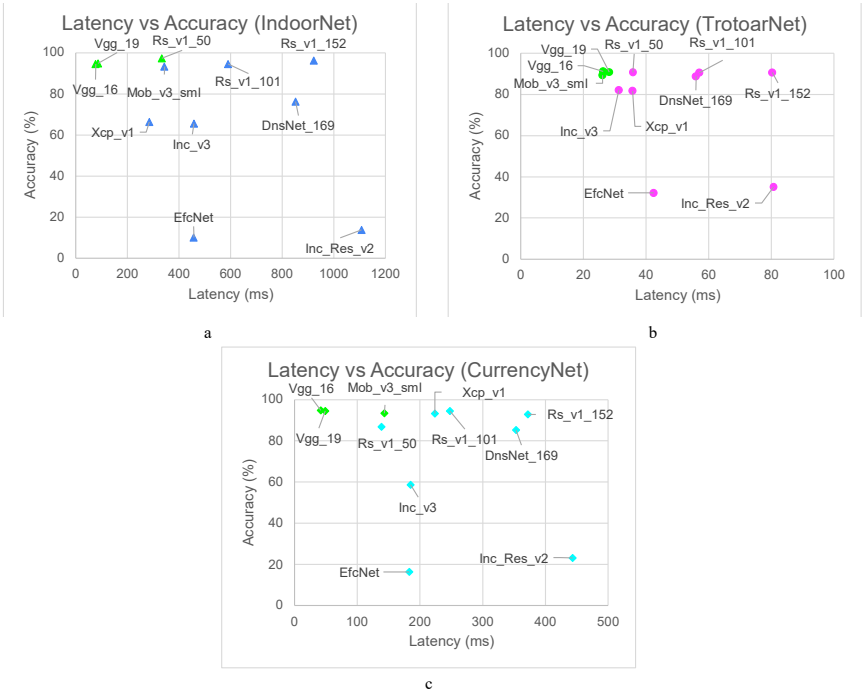
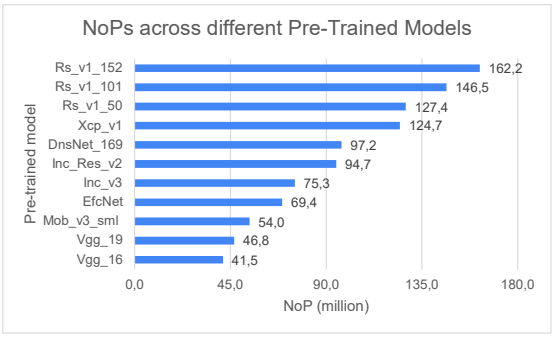
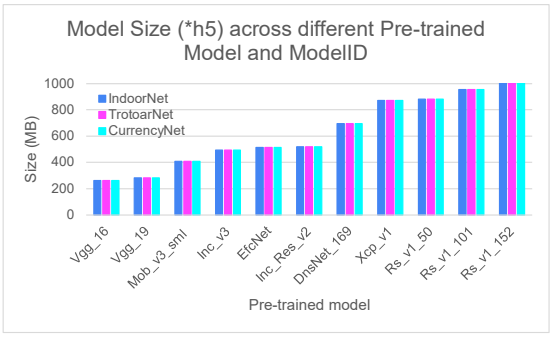


Figure A.2. The results of the latency vs accuracy test on the pre-trained model.

Figure A.3 (a) illustrates the test of a fully connected layer using the same parameters. Feature maps created by feature extraction are still multidimensional arrays, so they must be flattened or reshaped into vectors in order to be input into a fully connected layer. All neurons from the previous layer are connected to neurons in the next layer in the fully connected layer. Before the fully connected layer can be connected to all neurons, each activity must be converted to one-dimensional data. To classify data from the previous layer, we used a fully connected layer. The difference between the fully connected layer and the ordinary convolutional layer is that the neurons in the convolutional layer are connected only to certain input regions. Conversely, the neurons in the fully connected layer are all connected. A batch normalization operation is also used in Figure A.4(a) to speed up the training process and increase learning rates. It equalizes the distribution of every input value during the training process because parameters change in the previous layer [28].



a

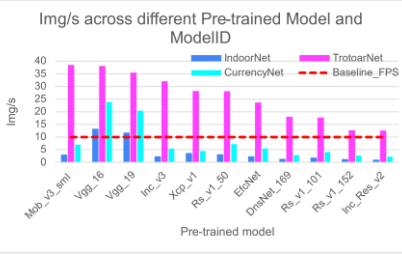


b

Figure A.3. Test results of (a) NoPs on pre-trained models, (b) size across pre-trained models on ModelID.

```
1.  output = Dense(1024,activation =
    "relu")(output)
2.  output = BatchNormalization()(output)
3.  output = Dropout(0.2)(output)
4.  output = Dense(1024,activation =
    "relu")(output)
5.  output = BatchNormalization()(output)
6.  output = Dropout(0.2)(output)
7.  output = Dense(19,
    activation='softmax')(output)
```

(a)



(b)

Figure A.4. (a) Fully connected layer, (b) Img/s test results on pre-trained models on ModelID.

Several neurons were selected at random and not used during training to regularize the neural network after normalization. They were suspended during backpropagation, preventing overfitting and speeding up the learning process. Dropout will be used to remove neurons from hidden and visible layers using random selection. Each neuron will have a probability between 0 and 1. A fully connected layer architecture also uses activation functions, such as ReLU and Softmax, which determine whether neurons are active based on the weighted sum of their inputs. In this way, Softmax and ReLU provide non-linear decision boundaries that optimize training. According to Figure A.4(b), pre-trained models differ in their inference latency. The smaller the latency, the higher the frames per second. More specifically, the optimal pre-trained models that produce high FPS are VGG16 and VGG19, with FPS values > 10, which both already meet the minimum FPS standard for the determined Sherpa model.