

Convolutional Neural Networks and Ensembles for Visually Impaired Aid

Fabricio Breve(✉)

São Paulo State University, Rio Claro, SP 13506-900, Brazil
fabricio.breve@unesp.br
https://www.fabriciobreve.com

Abstract. Recent surveys show that smartphone-based computer vision tools for visually impaired individuals often rely on outdated computer vision algorithms. Deep-learning approaches have been explored, but many require high-end or specialized hardware that is not practical for users. Therefore, developing deep learning systems that can make inferences using only the smartphone is desirable. This paper presents a comprehensive study of 25 different convolutional neural network (CNN) architectures to tackle the challenge of identifying obstacles in images captured by a smartphone positioned at chest height for visually impaired individuals. A transfer learning approach is employed, with the CNN models initialized with weights pre-trained on the vast ImageNet dataset. The study employs k-fold cross-validation with $k = 10$ and five repetitions to ensure the robustness of the results. Various configurations are explored for each CNN architecture, including different optimizers (Adam and RMSprop), freezing or fine-tuning convolutional layer weights, and different learning rates for convolutional and dense layers. Moreover, CNN ensembles are investigated, where multiple instances of the same or different CNN architectures are combined to enhance the overall performance. The highest accuracy achieved by an individual CNN is 94.56% using EfficientNetB4, surpassing the previous best result of 92.11%. With the use of ensembles, the accuracy is further improved to 96.55% using multiple instances of EfficientNetB4, EfficientNetB0, and MobileNet. Overall, the study contributes to the development of advanced deep-learning models that can enhance the mobility and independence of visually impaired individuals.

Keywords: Convolutional Neural Networks · Deep Learning · Computer Vision · Visually Impaired Aid

1 Introduction

According to the World Health Organization, approximately 2.2 billion people suffer from some form of visual impairment, including at least 1 billion with moderate or severe distance vision impairment [40]. The prevalence of distance vision impairment is significantly higher in low- and middle-income areas compared to high-income regions [34]. This population faces numerous

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023, LNCS 13956, pp. 520–534, 2023.
https://doi.org/10.1007/978-3-031-36805-9_34

difficulties in their daily routines, mostly linked to mobility and navigation. White canes and guide dogs are currently the most commonly utilized tools to aid visually impaired (VI) individuals [15]. With advancements in computer vision and related technologies, numerous navigation systems have been proposed [1,3,4,11,15–17,20,21,24,25,29,39]. However, many of these systems have limitations [15], such as requiring costly, bulky, and/or custom equipment [11,25–27,29,39] or being too computationally intensive to run on portable devices and requiring a network connection to a more powerful remote server [16,23].

A systematic literature review conducted by Budrionis et al. [3] found that smartphone-based computer vision tools for the VI often employ outdated image and video processing techniques. Another systematic review, conducted by Mandia et al. [24], discovered that researchers have started to adopt deep learning approaches [6,22,32] and that these techniques have grown with the advent of increased computational power in machines. However, carrying high-powered computational devices for vision-based assistive solutions is not practical for users. Hence, a deep learning system that can make inferences using only an edge device such as a smartphone is desirable. Ideally, this system should not require network connectivity or additional accessories.

In a prior study, Breve et al. [2] proposed a framework that leverages Convolutional Neural Networks (CNNs), transfer learning, and semi-supervised learning (SSL). The focus of the framework was to minimize computational costs and make it feasible for implementation on smartphones without requiring additional hardware. The framework uses a smartphone camera to capture images of the user's path and immediately classifies them, providing real-time feedback to the user. A dataset was created to train the classifiers, encompassing various indoor and outdoor environments with different lighting, flooring, and obstacles. The effectiveness of various CNN architectures was evaluated by fine-tuning pretrained weights from the ImageNet dataset [28]. A prototype of the framework running on a smartphone was recently presented [31].

In this study, previous works are significantly expanded upon with the following key contributions:

1. Eight additional CNN models were added to the study, based on the cutting-edge EfficientNet architecture [37], bringing the total number of networks evaluated to 25.
2. The K-Fold Cross Validation process was repeated five times, providing more robust results.
3. Image pre-processing functions were introduced to enhance image preparation for each network type, resulting in improved accuracy in most cases.
4. Ensembles of CNNs were employed to boost the overall accuracy by leveraging the strengths of multiple CNN architectures.

The rest of the paper is structured as follows: Sect. 2 shows some related work on visually impaired aid (VIA). Section 3 presents the VIA dataset. Section 4 displays the CNN architectures employed in this paper. Section 5 presents experimental results and analysis comparing the performance of these models on the

VIA dataset. Section 6 shows simulations with CNN ensembles, which enhance the accuracy of individual models. Finally, the conclusions are summarized in Sect. 7.

2 Related Work

Several endeavors have been undertaken to integrate computer vision into aiding visually challenged individuals. Mandia et al. [24] conducted a review of current vision-based assistive solutions for VI individuals. The review primarily focuses on camera-based systems and summarizes the sensors, image processing algorithms, and communication protocols used. The use of acoustic output devices, RFID, and GPS in addition to cameras is also discussed. The evolution from traditional image processing techniques to deep learning for assistance for the VI is highlighted. The paper concludes that the literature does not fully optimize deep learning models for edge devices.

Budrionis et al. [3] provides an overview of recent research prototypes of electronic travel aids (ETA) that use smartphones to assist VI people in orientation, navigation, and wayfinding. The authors systematically review scientific achievements in the field and compare various smartphone-based ETA prototypes. The meta-analysis found a few attempts to use state-of-the-art computer vision methods based on deep neural networks. The study contrasts these findings with a survey of blind expert users to reveal a major mismatch between user needs and academic development in the field. The authors conclude that the development of affordable smartphone-based ETAs is crucial for VI people in low-income countries and highlight the need for further research to address the identified gaps.

Islam et al. [15] reviews the development of walking assistants for VI individuals and highlights the recent advancements, including their benefits and limitations. The authors aim to provide a comprehensive overview of the current state of walking assistants and suggest areas for future development in sensors, computer vision, and smartphone-based technology.

Kuriakose et al. [20] propose an EfficientNet-Lite based scene recognition model for use in a smartphone application that supports navigation for the blind and VI. The model is trained and tested using a custom dataset of indoor and outdoor scenes. A proof of concept prototype app was developed on the Android platform.

Bai et al. [1] present a wearable assistive device for VI people to help them navigate and recognize objects in indoor and outdoor environments. The device consists of a RGB-D camera, an inertial measurement unit, a smartphone, and an audio module. It uses a CNN-based object recognition system for perception and navigation. The system provides semantic information about the surroundings and interacts with the user through audio.

Jiang et al. [16] proposed a wearable system that uses stereo vision. The system leverages binocular vision sensors to capture images and selects the best images based on stereo image quality assessment. The selected images are then

sent to the cloud for processing using a CNN and it returns information to the user to assist with decision-making.

Paul et al. [17] proposes a system consisting of a camera, GPS, infrared and light sensors connected to a microprocessor (Raspberry Pi) to process and relay information about the user's surroundings. The camera captures images, while the microprocessor uses image processing techniques to analyze the images and identify objects and obstacles. The control unit then relays this information to the user through audio output. The authors did not specify which processing techniques they used, stating only that they have used the OpenCV library.

Hoang et al. [11] developed a system that employs a Kinect camera mounted on a belt to capture and analyze the surroundings. The system detects obstacles and conveys this information to the user via audio feedback. A laptop computer must be carried in a backpack to process the captured information using the Point Cloud Library.

In 2013, Tapu et al. [38] introduced a real-time obstacle detection and classification system that utilized video from a smartphone camera. They created a framework consisting of tracking, motion estimation, and clustering methods. Four years later, Tapu et al. [39] introduced a more advanced framework based on CNNs for detecting, tracking, and recognizing objects in outdoor settings. However, this system requires the use of a laptop computer, carried in a backpack, as the processing unit.

Lin et al. [23] proposed a guiding system that utilizes a smartphone. The system incorporates CNNs for object recognition but relies on a desktop server with a GPU and Compute Unified Device Architecture (CUDA) to handle the computational intensive object recognition task. Although the system has an offline mode, it only offers recognition for faces and stairs.

Kumar and Meher [19] introduced an object recognition system that employs a mixture of CNN (Convolutional Neural Network) and RNN (Recurrent Neural Network). It can identify everyday indoor objects and their hues and generates auditory responses to the user. Saffoury et al. [29] put forward a system that uses a smartphone and laser pointer. The system makes use of laser triangulation to establish a collision avoidance protocol, and also delivers auditory feedback to the user.

Poggi and Mattoccia [25] developed a wearable device that consists of glasses with a custom RGBD sensor and FPGA onboard processing, a glove with micromotors for tactile feedback, a pocket battery, a bone-conductive headset, and a smartphone. The system employs deep learning techniques to categorize detected obstacles semantically. Previously, Poggi et al. [26] introduced a similar system for recognizing crosswalks.

Rizzo et al. [27] propose a fusion framework for combining signals from a stereo camera and infrared sensor for obstacle detection using a multi-scale CNN, with plans to implement the framework into a wearable vest.

Islam and Sadi [14] applied a CNN to detect path holes and obtained impressive results. However, it should be noted that they utilized a separate dataset for "path hole" images and another for "non-path hole" images, with the latter

consisting of road images taken with a wider angle. This raises the possibility that the network may have learned differences in style that are not relevant to the task at hand, thereby simplifying its job.

3 Dataset

The VIA dataset[1] includes 342 images separated into two categories: 175 "clear-path" and 167 "non-clear path". The images were taken with a smartphone camera and resized to 750 × 1000 pixels. The smartphone was positioned at chest height and inclined at an angle of 30° to 60° to capture several meters of the path ahead, including areas beyond the reach of a standard white cane.

Despite its small size, the dataset covers various indoor and outdoor environments with different floor types, including dry and wet, light conditions with both natural and artificial lighting, and obstacles like stairs, trees, holes, animals, and traffic cones. See Fig. 1 for examples of images in the proposed dataset.

4 CNN Architectures

This section showcases the CNN architectures explored in this study. It also outlines the layers added to complete the models and classify the VIA dataset images.

Table 1 displays the 25 evaluated architectures, along with some of their characteristics and references from literature.

The original CNN architectures were used with their existing structures and weights for ImageNet classification [28], except for the dense classification layers which were removed. Instead, an average global pooling layer was added, followed by a dense layer with 128 neurons and ReLU (Rectified Linear Unit) activation, then followed by a softmax classification layer. Figure 2 illustrates this proposed architecture, where x represents the CNN's horizontal and vertical input size (image size), and w, y, and z represent the size of the CNN's output in the last convolutional layer, which depends on the original CNN architecture and is specified in Table 1. The table also displays the number of trainable parameters in each CNN structure, including both the original layers and the added dense layers for VIA dataset classification."

5 CNN Comparison

This section presents the results of computer simulations that compared various CNN models applied to the VIA dataset. The simulations were carried out using Python and TensorFlow on three desktop computers equipped with NVIDIA GeForce GPU boards: GTX 970, GTX 1080, and RTX 2060 SUPER, respectively.

[1] Available at: https://github.com/fbreve/via-dataset.



(a) “clear path”



(b) “non-clear path”

Fig. 1. Examples of images from the VIA dataset: (a) “clear path” category; and (b) “non-clear path” category.

Table 1. CNN architectures, selected characteristics, and references.

Model	Input Image Resolution	Output of Last Conv. Layer	Trainable Parameters	Reference
DenseNet121	224×224	$7 \times 7 \times 1024$	7,085,314	[13]
DenseNet169	224×224	$7 \times 7 \times 1664$	12,697,858	[13]
DenseNet201	224×224	$7 \times 7 \times 1920$	18,339,074	[13]
EfficientNetB0	224×224	$7 \times 7 \times 1280$	4,171,774	[37]
EfficientNetB1	240×240	$8 \times 8 \times 1280$	6,677,410	[37]
EfficientNetB2	260×260	$9 \times 9 \times 1408$	7,881,604	[37]
EfficientNetB3	300×300	$10 \times 10 \times 1536$	10,893,226	[37]
EfficientNetB4	380×380	$12 \times 12 \times 1792$	17,778,378	[37]
EfficientNetB5	456×456	$15 \times 15 \times 2048$	28,603,314	[37]
EfficientNetB6	528×528	$17 \times 17 \times 2304$	41,031,002	[37]
EfficientNetB7	600×600	$19 \times 19 \times 2560$	64,115,026	[37]
InceptionResNetV2	299×299	$8 \times 8 \times 1536$	54,473,186	[35]
InceptionV3	299×299	$8 \times 8 \times 2048$	22,030,882	[36]
MobileNet	224×224	$7 \times 7 \times 1024$	3,338,434	[12]
MobileNetV2	224×224	$7 \times 7 \times 1280$	2,388,098	[30]
NASNetMobile	224×224	$7 \times 7 \times 1056$	4,368,532	[41]
ResNet101	224×224	$7 \times 7 \times 2048$	42,815,362	[8]
ResNet101V2	224×224	$7 \times 7 \times 2048$	42,791,426	[9]
ResNet152	224×224	$7 \times 7 \times 2048$	58,482,050	[8]
ResNet152V2	224×224	$7 \times 7 \times 2048$	58,450,434	[9]
ResNet50	224×224	$7 \times 7 \times 2048$	23,797,122	[8]
ResNet50V2	224×224	$7 \times 7 \times 2048$	23,781,890	[9]
VGG16	224×224	$7 \times 7 \times 512$	14,780,610	[33]
VGG19	224×224	$7 \times 7 \times 512$	20,090,306	[33]
Xception	299×299	$10 \times 10 \times 2048$	21,069,482	[5]

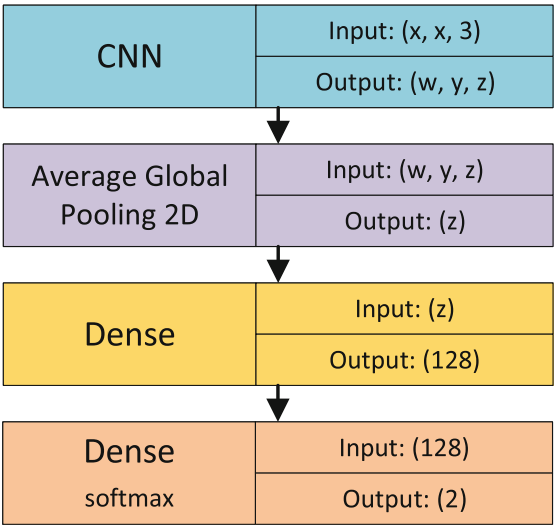

Fig. 2. Diagram of the proposed CNN networks.

The images were prepared for each CNN architecture by resizing them to the CNN input size and normalizing their range, with no other pre-processing applied. The networks were initialized with pre-trained weights from the Imagenet dataset [28], which has millions of images and hundreds of classes, and is a commonly used source for transfer learning. These pre-trained weights are available in Tensorflow. For the dense layers, the He uniform variance scaling initializer [7] was used.

The training phase involved six different scenarios with varying optimizers, learning rates, and frozen layers. The optimizers evaluated were RMSprop [10] and Adam [18]. In two of the scenarios, only the dense layer weights were trainable, while the convolutional layer weights remained frozen. In the other four scenarios, all layers were trainable, with two scenarios using the same adaptive learning rates for all layers and the other two using different fixed learning rates for the convolutional and dense layers. These scenarios are summarized in Table 2.

Table 2. The various scenarios in which each CNN was tested.

Config.	Fine-Tuning	Different Learning Rates	Optimizer
A	No	No	RMSprop
B	No	No	Adam
C	Yes	No	RMSprop
D	Yes	No	Adam
E	Yes	Yes	RMSprop
F	Yes	Yes	Adam

Configurations A and B were designed to preserve the weights trained on the large Imagenet dataset, while avoiding damaging them with the changes made using the smaller VIA dataset. In these scenarios, the CNNs acted as feature extractors for the dense layers, with an adaptive learning rate that varied from 10^{-3} to 10^{-5}. Configurations C and D explored the benefit of fine-tuning the weights in the convolutional layers for the target dataset, with an adaptive learning rate that varied from 10^{-3} to 10^{-5} applied to all layers. The learning rate was adjusted by a factor of 0.5 whenever the validation accuracy did not increase in the last two epochs. Configurations E and F explored the idea that the dense layers needed to be trained from scratch, while the convolutional layers received minor adjustments based on the weights learned from the Imagenet dataset. In these scenarios, the learning rate was fixed at 10^{-5} for the convolutional layers and 10^{-3} for the dense layers.

All models were trained for up to 50 epochs, with an early stopping criterion set to interrupt the training phase if the loss on the validation set did not decrease during the last 10 epochs. In most scenarios, the batch size was set to 16. However, for the scenarios involving EfficientNetB3 to EfficientNetB7, the

batch size was reduced to accommodate the memory constraints of the available GPUs with up to 8GB of RAM. The specific batch sizes for these exceptions can be found in Table 3.

Table 3. Batch sizes used for each method and configuration, based on the GPU memory constraints. For methods not listed in the table, the batch size was uniformly set to 16 for all configurations.

Method	Conf. A	Conf. B	Conf. C	Conf. D	Conf. E	Conf. F
EfficientNetB3	16	16	16	16	16 – 8	16
EfficientNetB4	16	16	4	8	4	8
EfficientNetB5	16	16	2	4	2	4
EfficientNetB6	16	16	2	2	2	2
EfficientNetB7	16	16	1	1	1	1

Table 4 shows the classification accuracy obtained through transfer learning with the 25 different CNN architectures. The results were obtained using K-Fold Cross Validation with $k = 10$ and repeated 5 times, so each value in the table represents the average of 50 executions. In all scenarios, 20% of the training instances were randomly selected as the validation subset to guide the learning rate adjustments and to determine the stopping criterion.

MobileNet was found to be the best architecture in three out of the six configurations and had the best average performance considering all configurations. Its success is noteworthy as MobileNet was specifically designed to operate on mobile devices with limited processing power, which are the target devices for this framework. EfficientNetB0, EfficientNetB4, and InceptionV3 were the best-performing architectures in the remaining configurations. The highest accuracy overall was achieved by EfficientNetB4 with configuration D (0.9456), closely followed by EfficientNetB0 with configuration C (0.9427). This suggests that fine-tuning the convolutional layers can improve accuracy for some networks. However, in general, most architectures saw a decrease in accuracy when fine-tuned, as indicated by the best performing configurations being A and B in average. Configurations E and F did not yield exceptional results, with their best results being worse than those achieved with other configurations. Nonetheless, their average performance was better than that achieved with configurations C and D. Regarding optimizers, Adam and RMSprop produced similar results, with Adam showing a slight advantage by achieving an average accuracy of 0.8490 (configurations B, D, and F) compared to 0.8453 of RMSprop (configurations A, C, and E).

Table 4. Comparison of 25 different CNN-based models applied to the VIA datasets with the six proposed configurations. The best results in each column are highlighted in bold and the best results in each row are highlighted in italics.

Method	Conf. A	Conf. B	Conf. C	Conf. D	Conf. E	Conf. F	Average
MobileNet	***0.9158***	**0.9152**	0.9299	0.9257	0.8729	**0.9105**	**0.9117**
Xception	0.8749	0.8755	*0.9374*	0.9252	0.8934	0.9029	0.9016
EfficientNetB0	0.8877	0.8876	**0.9427**	*0.9274*	0.8724	0.8819	0.9000
EfficientNetB3	0.8901	0.8889	*0.9404*	0.9291	0.8727	0.8761	0.8995
EfficientNetB2	0.8725	0.8660	0.9391	*0.9426*	0.8672	0.8679	0.8926
EfficientNetB4	0.8813	0.8807	0.9304	***0.9456***	0.8414	0.8632	0.8904
EfficientNetB1	0.8908	0.8855	*0.9369*	0.9341	0.8482	0.8463	0.8903
DenseNet201	0.8841	*0.8847*	*0.8847*	0.8807	0.8696	0.8809	0.8808
InceptionResNetV2	0.8650	0.8644	0.8954	*0.9217*	0.8632	0.8668	0.8794
InceptionV3	0.8691	0.8657	0.8611	*0.8965*	**0.8936**	0.8807	0.8778
DenseNet169	0.8789	0.8766	0.8779	*0.8854*	0.8679	0.8713	0.8763
DenseNet121	0.8730	0.8671	0.8867	*0.8912*	0.8715	0.8680	0.8763
ResNet50	*0.8947*	0.8901	0.8139	0.8392	0.8801	0.8761	0.8657
ResNet101	*0.8925*	0.8919	0.8130	0.7919	0.8731	0.8626	0.8542
EfficientNetB5	0.8953	0.8947	0.8760	*0.9233*	0.6398	0.8327	0.8436
MobileNetV2	*0.8924*	0.8912	0.8324	0.7954	0.8116	0.8042	0.8378
ResNet152	0.8755	0.8754	0.7418	0.7724	*0.8819*	0.8638	0.8351
ResNet50V2	0.8539	0.8581	0.7273	0.8263	0.8516	*0.8587*	0.8293
EfficientNetB6	0.8719	0.8690	0.8514	*0.8738*	0.7383	0.7516	0.8260
ResNet101V2	*0.8807*	0.8790	0.6184	0.7256	0.8778	0.8779	0.8099
ResNet152V2	*0.9106*	0.9077	0.5942	0.6663	0.8890	0.8885	0.8094
VGG19	0.8263	0.8118	0.7916	0.6707	*0.8746*	0.8680	0.8072
VGG16	0.8263	0.8175	0.7877	0.6316	*0.8759*	0.8543	0.7989
NASNetMobile	0.8560	*0.8578*	0.6671	0.6997	0.7092	0.7050	0.7491
EfficientNetB7	*0.8731*	0.8714	0.5248	0.4999	0.5242	0.5230	0.6361
Average	*0.8773*	0.8749	0.8241	0.8289	0.8344	0.8433	0.8472

6 CNN Ensembles

This section presents the computer simulations involving ensembles of multiple instances of CNN models, including ensembles of single and multiple instances of different architectures. In all ensemble experiments, the output of the last dense layer, just before the softmax activation function, was used. This resulted in two continuous values for each image, which represent the probability of each class. The ensemble output was then computed by taking the average of its members' output. The same folds used in the previous section were employed, and the results in this section represent the average of 50 executions, obtained through

K-Fold Cross Validation with $k = 10$ repeated 5 times. Ensemble outputs were computed for each fold individually and then averaged.

The first ensemble experiment involved creating ensembles using only instances of the same CNN model. For each configuration, the architecture that provided the best individual results in Table 4 was selected. Specifically, MobileNet was used for configurations A, B, and F, EfficientNetB0 for configuration C, EfficientNetB4 for configuration D, and InceptionV3 for configuration E. One to ten instances were created and initialized with different seeds, and the accuracy achieved with each ensemble was recorded in Table 5. Across all tested scenarios, ensembles achieved higher accuracies than a single instance, with the best results typically obtained using six instances. While all configurations benefited from the use of ensembles, configuration D achieved the best results overall. Specifically, the highest accuracy achieved was 0.9602, obtained using six and ten instances of EfficientNetB4 with configuration D.

The second ensemble experiment was similar to the first, but ensembles were formed using one to ten instances of the best architecture in each configuration. Configurations were added to the ensembles one by one according to their performance in Table 4, with configurations D, C, A, B, F, and E added in that specific order. The accuracy achieved with each ensemble is presented in Table 6, with the best overall accuracy of 0.9655 obtained using six instances from each of the three best configurations: EfficientNetB4 with configuration D, EfficientNetB0 with configuration C, and MobileNet with configuration A.

Table 5. Comparison of ensembles of single CNN-based models applied to the VIA datasets with the six proposed configurations. The best results in each column are highlighted in bold and the best results in each row are highlighted in italics.

Instances	Conf. A	Conf. B	Conf. C	Conf. D	Conf. E	Conf. F	Average
1	0.9158	0.9152	0.9427	*0.9456*	0.8936	0.9105	0.9206
2	0.9187	0.9135	0.9451	*0.9502*	0.9065	0.9170	0.9252
3	0.9170	0.9176	0.9445	*0.9532*	0.9112	0.9193	0.9271
4	0.9164	0.9158	0.9485	*0.9562*	0.9176	0.9182	0.9288
5	0.9187	0.9182	0.9491	*0.9585*	0.9171	**0.9217**	0.9306
6	0.9176	**0.9199**	**0.9549**	***0.9602***	0.9182	0.9199	**0.9318**
7	**0.9211**	0.9182	0.9543	*0.9579*	**0.9211**	0.9164	0.9315
8	0.9164	0.9158	0.9509	*0.9596*	0.9194	0.9158	0.9297
9	0.9182	0.9176	0.9538	*0.9596*	0.9200	0.9193	0.9314
10	0.9188	0.9159	0.9526	***0.9602***	0.9194	0.9176	0.9308
Average	0.9179	0.9168	0.9496	*0.9561*	0.9144	0.9176	0.9287

Table 6. Comparison of ensembles of multiple CNN-based models applied to the VIA datasets using the one to six of the proposed configurations. The best results in each column are highlighted in bold and the best results in each row are highlighted in italics.

Instances of each conf.	Conf.D	Conf.DC	Conf.DCA	Conf.DCAB	Conf. DCABF	All Conf.	**Average**
1	0.9456	0.9532	*0.9567*	0.9550	0.9503	0.9491	0.9517
2	0.9502	0.9491	*0.9544*	0.9538	0.9486	0.9521	0.9514
3	0.9532	0.9544	*0.9597*	0.9545	0.9509	0.9539	0.9544
4	0.9562	0.9555	*0.9614*	0.9527	0.9515	0.9556	0.9555
5	0.9585	0.9567	*0.9620*	0.9544	0.9544	**0.9573**	0.9572
6	**0.9602**	**0.9579**	***0.9655***	0.9562	0.9538	0.9556	**0.9582**
7	0.9579	0.9544	*0.9643*	**0.9574**	0.9544	0.9550	0.9572
8	0.9596	0.9532	*0.9626*	0.9562	0.9526	**0.9573**	0.9569
9	0.9596	0.9555	*0.9637*	0.9568	0.9568	0.9550	0.9579
10	**0.9602**	0.9561	*0.9626*	0.9556	**0.9579**	0.9550	0.9579
Average	0.9561	0.9546	*0.9613*	0.9553	0.9531	0.9546	0.9558

7 Conclusions

This paper explores the application of 25 different CNN architectures to identify obstacles in the path of visually impaired individuals. K-Fold Cross Validation was utilized with $k = 10$ and five repetitions to provide robust results. The architectures have low computational costs during inference, executing in milliseconds on current smartphones, allowing them to be implemented without relying on external equipment or remote servers. The CNN architectures were pre-trained on large datasets and evaluated first as feature extractors with pre-trained weights, then with fine-tuned weights for the proposed task. Fine-tuning an EfficientNetB4 network achieved the highest accuracy of 0.9456.

CNN ensembles were examined, comprising multiple instances of the single best architecture in each configuration, as well as instances of the best architectures in each configuration. In the first scenario, an ensemble of six instances was utilized, resulting in an accuracy improvement to 0.9602 for the fine-tuned EfficientNetB4. In the second scenario, the six instances of EfficientNetB4 were combined with six instances of EfficientNetB0, which were also fine-tuned to the proposed task, and six instances of MobileNet, which were used as a feature extractor. This approach resulted in a further accuracy increase to 0.9655.

The numerous computer simulations conducted in this study yielded promising results for some CNN architectures and investigated the use of different optimizers (Adam and RMSprop), learning strategies (single learning rate versus different rates for convolution and dense layers), and pre-trained weights (fixed versus fine-tuned). The study also demonstrated that ensembles could enhance accuracy by utilizing multiple instances of the same architecture and configuration or multiple instances of different architectures and configurations.

Future work includes expanding the proposed dataset by acquiring more images and exploring other approaches and modifications to the current framework to further enhance classification accuracy. Recently, a smartphone proto-

type application was developed to test real-world scenarios [31]. Furthermore, the findings presented in this paper can guide future research on related datasets as numerous CNN architectures were tested and compared.

Acknowledgements. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) and by the São Paulo Research Foundation - FAPESP (grant #2016/05669-4).

References

1. Bai, J., Liu, Z., Lin, Y., Li, Y., Lian, S., Liu, D.: Wearable travel aid for environment perception and navigation of visually impaired people. Electronics **8**(6), 697 (2019)
2. Breve, F., Fischer, C.N.: Visually impaired aid using convolutional neural networks, transfer learning, and particle competition and cooperation. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2020). https://doi.org/10.1109/IJCNN48605.2020.9207606
3. Budrionis, A., Plikynas, D., Daniušis, P., Indrulionis, A.: Smartphone-based computer vision travelling aids for blind and visually impaired individuals: A systematic review. Assistive Technology **34**(2), 178–194 (2022). https://doi.org/10.1080/10400435.2020.1743381,https://doi.org/10.1080/10400435.2020.1743381, pMID: 32207640
4. Cardillo, E., Caddemi, A.: Insight on electronic travel aids for visually impaired people: a review on the electromagnetic technology. Electronics **8**(11), 1281 (2019)
5. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1800–1807 (July 2017). https://doi.org/10.1109/CVPR.2017.195
6. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
7. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference On Computer Vision, pp. 1026–1034 (2015)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (June 2016)
9. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 630–645. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_38
10. Hinton, G., Srivastava, N., Swersky, K.: Neural networks for machine learning lecture 6a overview of mini-batch gradient descent (2012)
11. Hoang, V.-N., Nguyen, T.-H., Le, T.-L., Tran, T.-H., Vuong, T.-P., Vuillerme, N.: Obstacle detection and warning system for visually impaired people based on electrode matrix and mobile Kinect. Vietnam J. Comput. Sci. **4**(2), 71–83 (2016). https://doi.org/10.1007/s40595-016-0075-z
12. Howard, A.G., et al.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
13. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4700–4708 (July 2017)

14. Islam, M.M., Sadi, M.S.: Path hole detection to assist the visually impaired people in navigation. In: 2018 4th International Conference on Electrical Engineering and Information Communication Technology (iCEEiCT), pp. 268–273 (Sep 2018). https://doi.org/10.1109/CEEICT.2018.8628134
15. Islam, M.M., Sheikh Sadi, M., Zamli, K.Z., Ahmed, M.M.: Developing walking assistants for visually impaired people: a review. IEEE Sens. J. **19**(8), 2814–2828 (2019). https://doi.org/10.1109/JSEN.2018.2890423
16. Jiang, B., Yang, J., Lv, Z., Song, H.: Wearable vision assistance system based on binocular sensors for visually impaired users. IEEE Internet Things J. **6**(2), 1375–1383 (2019). https://doi.org/10.1109/JIOT.2018.2842229
17. Joe Louis Paul, I., Sasirekha, S., Mohanavalli, S., Jayashree, C., Moohana Priya, P., Monika, K.: Smart eye for visually impaired-an aid to help the blind people. In: 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), pp. 1–5 (2019). DOI: https://doi.org/10.1109/ICCIDS.2019.8862066
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014)
19. Kumar, R., Meher, S.: A novel method for visually impaired using object recognition. In: 2015 International Conference on Communications and Signal Processing (ICCSP), pp. 0772–0776 (April 2015). https://doi.org/10.1109/ICCSP.2015.7322596
20. Kuriakose, B., Shrestha, R., Sandnes, F.E.: Scenerecog: A deep learning scene recognition model for assisting blind and visually impaired navigate using smartphones. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 2464–2470 (2021). https://doi.org/10.1109/SMC52423.2021.9658913
21. Lakde, C.K., Prasad, P.S.: Review paper on navigation system for visually impaired people. Int. J. Adv. Res. Comput. Commun. Eng. **4**(1), (2015)
22. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
23. Lin, B.S., Lee, C.C., Chiang, P.Y.: Simple smartphone-based guiding system for visually impaired people. Sensors **17**(6), 1371 (2017)
24. Mandia, S., Kumar, A., Verma, K., Deegwal, J.K.: Vision-based assistive systems for visually impaired people: a review. In: Tiwari, M., Ismail, Y., Verma, K., Garg, A.K. (eds.) Optical and Wireless Technologies, pp. 163–172. Springer Nature Singapore, Singapore (2023)
25. Poggi, M., Mattoccia, S.: A wearable mobility aid for the visually impaired based on embedded 3d vision and deep learning. In: 2016 IEEE Symposium on Computers and Communication (ISCC), pp. 208–213. IEEE (2016)
26. Poggi, M., Nanni, L., Mattoccia, S.: Crosswalk recognition through point-cloud processing and deep-learning suited to a wearable mobility aid for the visually impaired. In: International Conference on Image Analysis and Processing, pp. 282–289. Springer (2015)
27. Rizzo, J.R., Pan, Y., Hudson, T., Wong, E.K., Fang, Y.: Sensor fusion for ecologically valid obstacle identification: Building a comprehensive assistive technology platform for the visually impaired. In: 2017 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO), pp. 1–5. IEEE (2017)
28. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vision **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
29. Saffoury, R., et al.: Blind path obstacle detector using smartphone camera and line laser emitter. In: 2016 1st International Conference on Technology and Innovation in Sports, Health and Wellbeing (TISHW), pp. 1–7. IEEE (2016)

30. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv 2: Inverted residuals and linear bottlenecks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520 (June 2018)
31. Sanga, G.M., Polo, J.M.G., Passerini, J.A.R.: Auxílio a deficientes visuais utilizando redes neurais convolucionais (2022)
32. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
33. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks For Large-scale Image Recognition, pp. 1–14. Computational and Biological Learning Society (2015)
34. Steinmetz, J.D., et al.: Causes of blindness and vision impairment in 2020 and trends over 30 years, and prevalence of avoidable blindness in relation to vision 2020: the right to sight: an analysis for the global burden of disease study. Lancet Glob. Health **9**(2), e144–e160 (2021)
35. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-first AAAI Conference On Artificial Intelligence (2017)
36. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2818–2826 (June 2016)
37. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR (09–15 Jun 2019), https://proceedings.mlr.press/v97/tan19a.html
38. Tapu, R., Mocanu, B., Bursuc, A., Zaharia, T.: A smartphone-based obstacle detection and classification system for assisting visually impaired people. In: The IEEE International Conference on Computer Vision (ICCV) Workshops (June 2013)
39. Tapu, R., Mocanu, B., Zaharia, T.: Deep-see: joint object detection, tracking and recognition with application to visually impaired navigational assistance. Sensors **17**(11), 2473 (2017)
40. World Health Organization: Vision impairment and blindness (Oct 2022), https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment Accessed: 2023-01-30
41. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8697–8710 (June 2018)