Arabian Journal for Science and Engineering (2020) 45:10511–10527
https://doi.org/10.1007/s13369-020-04799-7

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

Assisting the Visually Impaired in Multi-object Scene Description Using OWA-Based Fusion of CNN Models

Haikel Alhichri[1] · **Yakoub Bazi**[1] · **Naif Alajlan**[1]

Received: 13 March 2020 / Accepted: 12 July 2020 / Published online: 29 July 2020

Abstract
Advances in technology can provide a lot of support for visually impaired (VI) persons. In particular, computer vision and machine learning can provide solutions for object detection and recognition. In this work, we propose a multi-label image classification solution for assisting a VI person in recognizing the presence of multiple objects in a scene. The solution is based on the fusion of two deep CNN models using the induced ordered weighted averaging (OWA) approach. Namely, in this work, we fuse the outputs of two pre-trained CNN models, VGG16 and SqueezeNet. To use the induced OWA approach, we need to estimate a confidence measure in the outputs of the two CNN base models. To this end, we propose the residual error between the predicted output and the true output as a measure of confidence. We estimate this residual error using another dedicated CNN model that is trained on the residual errors computed from the main CNN models. Then, the OAW technique uses these estimated residual errors as confidence measures and fuses the decisions of the two main CNN models. When tested on four image datasets of indoor environments from two separate locations, the proposed novel method improves the detection accuracy compared to both base CNN models. The results are also significantly better than state-of-the-art methods reported in the literature.

Keywords Multi-label image classification · Convolutional neural networks (CNN) · VGG16 · SqueezeNet · Ordered weighted averaging (OWA) · Assistive technology for the visually impaired

1 Introduction

Artificial intelligence (AI) is one of the main areas of research and development in the world today. Many types of AI systems have been developed recently and have provided impressive solution in applications such as voice-powered virtual assistants (e.g., Siri and Alexa) [1], autonomous vehicles (e.g., Tesla) [2], robotics (e.g., in manufacturing cars) [3], and automatic translation (e.g., Google translate) [4]. Many AI-based solutions are also developed in the area of assistive technology, in particular for assisting the visually impaired (VI) persons. Most of these systems deal with the autonomous navigation problem using wearable assistive devices such as infrared sensors, ultrasound sensors, RFID, BLE beacon, and cameras [5, 6]. Besides autonomous navigation, VI persons need also other types of assistive technology such as people and object detection/recognition. Computer vision techniques combined with machine learning provide the most suitable solutions for this problem. For example, Hasanuzzaman et al. [7] present a computer vision system to recognize currency using speeded-up robust features (SURF). The system can recognize US notes with 100% true recognition rate and 0% false recognition rate. Other systems proposed in Refs. [8, 9], for example, assist the VI persons to do their shopping in the supermarket by detecting and reading barcodes and giving the VI person information (extracted from the shop database) about the product via voice communication. The authors in Ref. [10] proposed a system to assist the VI persons detect and read any text in their view. The system does this by first detecting candidate regions that may contain text using special

✉ Haikel Alhichri
hhichri@ksu.edu.sa; haike.alisr@gmail.com

Yakoub Bazi
ybazi@ksu.edu.sa; yakoub.bazi@gmail.com

Naif Alajlan
najlan@ksu.edu.sa; naif.ajlan@yahoo.com

[1] Advanced Lab for Intelligent Systems Research (ALISR), Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia

statistical features. Then, commercial optical character recognition (OCR) software is used to recognize text inside the candidate regions (or decide that the content of the regions is non-text). Another interesting application is a travel assistant presented by the authors in Ref. [11] that detects and recognizes text in public transportation domains. The system detects text writings on buses and stations and informs the VI person about station names and numbers, bus numbers and destinations, and so on. Another work by Jia et al. [12] addresses the problem of finding staircases inside buildings and informing the user when they are within five meters of any staircases. The method is based on an iterative preemptive RANSAC algorithm to detect steps of a staircase. But, the work by Yang and Tian [13] focuses on detecting doors inside buildings by detecting the most general and stable features of doors, namely edges and corners. Finally, a system for detecting restroom signage is presented in Ref. [14] based on scale-invariant feature transform (SIFT) features.

Object detection and recognition is a heavily studied problem in the computer vision field. The early object detection algorithms, such as the ones by Viola and Jones [15] and Dalal and Triggs [16], were built based on the extraction of handcrafted features before applying a classification algorithm. After the rebirth of neural networks in 2012 and the appearance of deep learning network and convolutional neural network (CNN) [17], more advanced object detection algorithms based on these methods have appeared, including region CNN (RCNN) [18, 19], you only look once (YOLO) [20, 21], single shot MultiBox detector (SDD) [22], pyramid networks [23], and Retina-Net networks [24]. These algorithms, while quite successful in detecting objects, have high computational costs and hence are difficult to execute on portable devices, unless one focuses on one object such as faces. This makes them less useful for the VI person because it is more convenient for him/her to use portable devices and also because she/he wants to detect a wider range of objects that can be seen in daily life. Therefore, some researchers proposed a compromise solution that can detect multiple objects in a short amount of time by solving the problem from a multi-label classification perspective [25–30]. In this approach, the presence of multiple objects can be detected, but not their exact location in the image. We believe that it is reasonable to assume that the VI person is more interested in detecting multiple objects in a fast way and then in knowing their exact locations.

Thus, in this work, we propose a scene description module for listing which objects are present in scene and inform the VI user about them using voice communications. The module is part of a larger assistive technology system for the visually impaired that is designed to assist them in (1) navigating indoor environments, (2) detecting and reading text information, (3) detecting and recognizing faces, and (4) detecting and listing objects present in the scene. The system uses a portable device connected to a camera placed on the VI person chest.

Our solution for the scene description module is based on a multi-label classification approach using deep CNN models. Usually, CNNs do not perform well for small datasets as they are prone to overfitting. In this case, it has been shown in many studies [31–34] that it is more suitable to employ a knowledge transfer approach by using pre-trained CNNs such as VGG CNN family [35] and GoogLeNet (inception CNN family) [36]. The pre-trained CNNs have been already trained on very large image datasets and all they need is small modifications to adapt them to our special dataset. One way to transfer knowledge from these pre-trained models is by using their learned feature representations as input to train another external classifier. The survey in [31] describes this option and discusses factors to consider in applying such approach. In some cases, it is possible and more worthwhile to retrain the whole model again but not starting from scratch, i.e., with random model weights. In other words, we start with the pre-trained weights and retrain the whole model on the new dataset, which is known as the fine-tuning approach.

Many CNN models have been proposed in the literature for image classification including VGG-VD [35], GoogLeNet [36], SqueezeNet [37], MobileNet [38], ResNet [39], and so on. These CNN models have different classification capabilities because of the architectural differences and because of the different datasets they are pre-trained on. Thus, it is wise to try to fuse them in a way that takes advantage of their respective strengths.

Fusion of ensemble of classifiers and/or multiple features are efficient techniques to achieve better results for applications like classification and recognition [40–47]. Usually, fusion is employed in three levels: data input level, feature level, and decision level [42]. In data input-level fusion, images from multiple sources are combined together to create a new image with a better signal-to-noise ratio than the input signals. Feature-level fusion consists of combining the features extracted from feature extraction algorithms to create a richer and more descriptive feature vector. In decision-level fusion, first the data are separately classified using different methods and then fusion consists of merging the output from the classification. For example, the work in [48] explores the fusion of images from different datasets to enhance the performance of a multi-task deep classifier. The authors in [44] propose a novel deep CNN model based on multiscale side output fusion in a deeply supervised network. The model fuses different feature vectors from the deep CNN at different scales in order to improve the salient object detection.

For decision-level fusion, the work in [45] investigated the use of multiple widely used ensemble methods in the context of deep neural network classifiers. The investigated methods include naive unweighted averaging, majority voting, the Bayes optimal classifier, etc. However, these methods are vulnerable to weak learners, are sensitive to overconfident learners and may lead to information loss [45].

Another recent work by Koh and Woo [46] presented a novel fusion approach of different multi-view classifications. The different multi-views are obtained by applying a classification model over different batches of the data. Then, the fusion of these different views involves computing co-occurrence matrices, weighted adjacency matrices, and Laplacian matrices, which are time-consuming.

In this work, we propose to improve the detection accuracy of CNN models by fusing their predictions using induced ordered weighted averaging (OWA) techniques. In particular, we use two base CNN models, namely the VGG16 model [35] and another light model called SqueezeNet [49]. We have selected these two models because they are not that deep and they are quite diverse. Our datasets are small (less than 160 images in the training set), and it is known from the literature that deeper models that have large number of weights need a huge dataset for training [35]. Furthermore, it is well known that fusion methods work best when we have diverse classifiers [45, 50]. Our chosen models are quite diverse because VGG16 uses a basic convolutional architecture with a large number of weights, whereas SqueezeNet is a much lighter network that uses an advanced squeeze/expand architecture. We also increase diversity through using a different training approach. For the SqueezeNet, we use a fine-tuning approach, where we update all its weights during training, whereas for the VGG16 model we only update the weights of the upper added layers only.

The induced OWA technique fuses the output predictions of the CNN models by computing their weighted average. However, the weights used are computed after ordering the predictions based on their importance or level of confidence. As a measure of confidence for each prediction, we propose to use the residual error between the predicted output and the true output. The residual error can be computed at training time, because we have the true outputs. But during test time, obviously, we do not have the true outputs. As a solution, we propose to estimate the residual errors from the input image directly by training another dedicated CNN for this purpose. In other words, for each dataset we train two CNN models: One is used to learn the actual output, while the other is used to learn the residual error. Using this approach for each predicted output, we have also an estimate of the residual error, which we can use as a measure of confidence in the prediction. It is important to note that unlike the regular weighted average scheme, where the weights are the same for all input images, the OWA technique computes different weights for each input image. Thus, the "optimal" weights are used for each input image, and this explains why OWA is able to improve on the accuracy of the two base models.

The contributions of this paper can be summarized in the following points:

- Proposing a deep learning solution for image multi-label classification based on the fusion of two CNN models using OWA theory.
- Proposing the residual errors between the model predictions and the true labels as measures of confidence in the model predictions and proposing using dedicated CNN models as a solution for their estimation from the input images.
- Developing the mathematical model that formulates the usage of the estimated residual errors to fuse the predictions of the CNN models using the induced OWA approach.

The rest of this paper is organized as follows. In Sect. 2, we provide a description of the proposed methods based on the fusion of CNN models using OWA. The experimental results and conclusions are presented in Sects. 3 and 4, respectively.

2 Materials and Methods

In this section, first, we describe the two pre-trained CNN models, VGG-16 and SqueezeNet, and the modification made on the architectures to adapt them to our multi-label classification problem. Then, Sect. 2.2 introduces the OWA theory. Finally, Sect. 2.3 describes the proposed method including the mathematical formulation of the proposed fusion approach.

2.1 Pre-trained CNN Models Description

As mentioned earlier, we propose to fuse the outputs of two base CNN models, namely the VGG16 model [35] and the SqueezeNet model [49], using the induced OWA approach. Figure 1 shows the architecture of the pre-trained CNN models used. For the VGG16 model, we remove the last two layers in the original model and then add an extra dense layer with LeakyReLU activation function [51] followed by a BatchNormalization layer [52].

As for the SqueezeNet CNN, we remove the layers after fire 9 block and replace them with an extra convolutional layer with LeakyReLU activation functions and a BatchNormalization layer. We have to also follow this by a GlobalAvgPooling2D layer before we end the network with the output layer. The output layer for both models has N_o neurons with linear activation functions, which is then converted to binary output (representing the presence and non-presence of the particular object) using a threshold T_p. For example, Fig. 1 shows a sample output for some input images, where

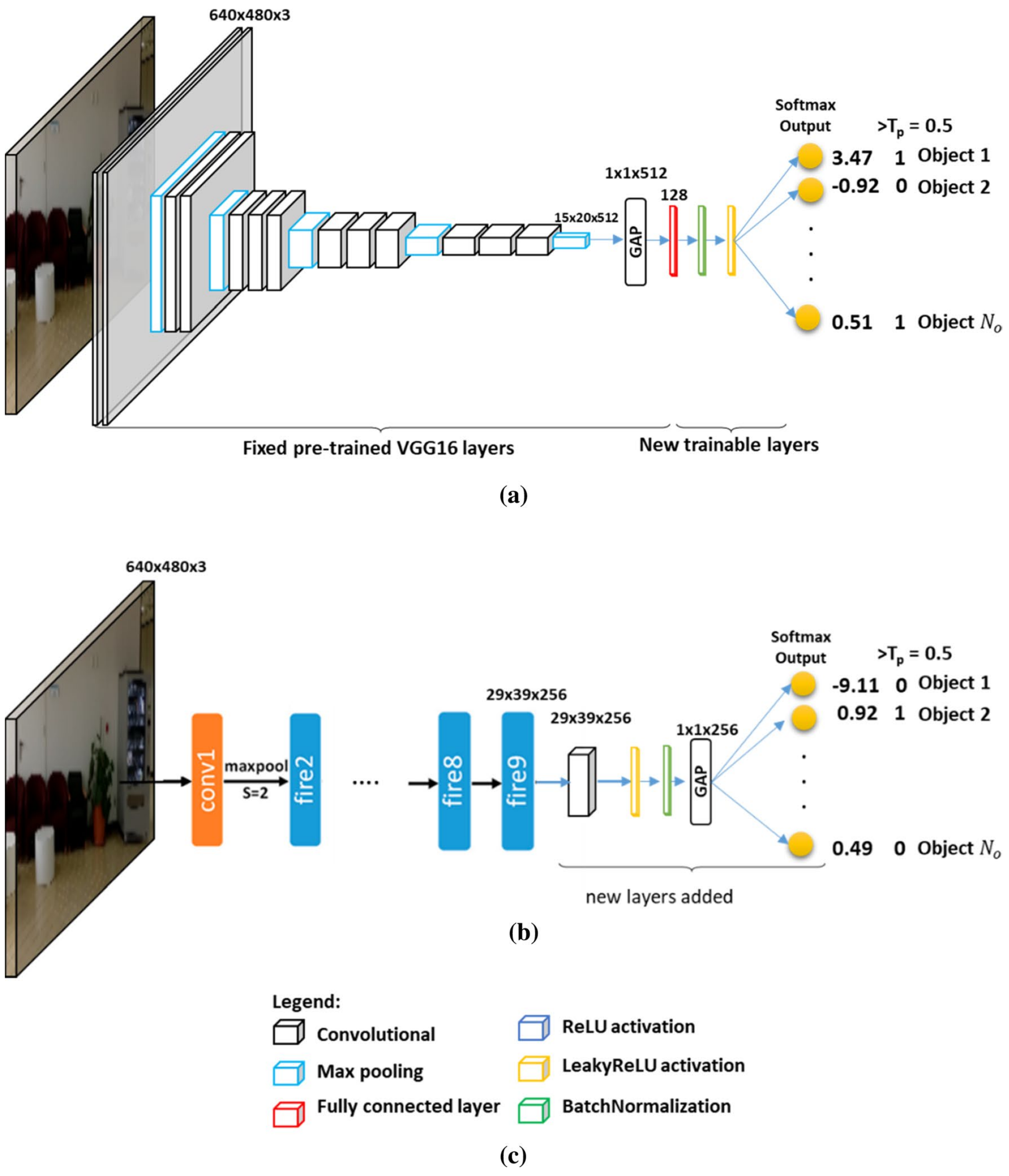

Fig. 1 The two pre-trained CNN models used in this work. **a** Base model 1 based on the pre-trained VGG16 CNN, **b** Base model 2 based on the pre-trained SqueezeNet CNN, **c** Legend of layers

the output is converted to binary values using a threshold $T_p = 0.5$. A binary output of one indicates the corresponding object is present in the image.

Another difference between the two base models is in the training. For the VGG16-based model, we fix the pre-trained layers because of the huge number of parameters in these layers (> 14 million), whereas for the SqueezeNet CNN, we employ a fine-tuning approach to train the network because it is a small network with less than one million parameters, which makes it easier to fine-tune using reasonable computational resources. Furthermore, even without fine-tuning the VGG16-based model can achieve good results, due to its rich architecture.

2.2 Induced OWA

Suppose we have a set of arguments a_j, representing outputs of multiple estimators which we want to fuse into one argument. One simple way to do this is using simple weighted averaging, where the weights represent our confidence in the corresponding estimator, i.e., if we have high confidence in a particular estimator, we can assign it a bigger weight and vice versa. However, sometimes the confidence is related to the output rather than the estimator itself. For example, we might favor outputs with large values over small values or vice versa. For these cases, the ordered weighted averaging (OWA) is defined.

An OWA operator of dimension P is a mapping $F_W : \mathcal{R}^P \rightarrow \mathcal{R}$ that has an associated weighting vector $\boldsymbol{w} = \left[w_1, w_2, \dots, w_P\right]^{\mathrm{T}}$ such that $w_j \in [0, 1]$ and $\sum_{j=1}^{P} w_j = 1$. The function $F_W\left(a_1, a_2, \dots, a_P\right)$ determines the aggregated value of the arguments $a_1, a_2, \dots, a_P$ such that:

$$F_W\left(a_1, a_2, \dots, a_P\right) = \sum_{j=1}^{P} w_j b_j, \tag{1}$$

where b_j is the jth largest of the a_j and P is the number of predictions.

It is important to observe the main difference between the definition of the OWA and a simple weighted average. The OWA involves the step of ordering the arguments a_j from largest or most confident to smallest or least confident. This makes the weights dependent on the position in the ordering rather than on the arguments themselves. This also makes the OWA operator a nonlinear operator that provides a very rich family of aggregation operators parameterized by the weighting vector. For example, if the weights are equal to $1/P$, then the OWA is simply the average operator. If the weight vector is [1, 0, …, 0], then the OWA becomes the maximum operator. Conversely, if the weight vector is [0, …, 0, 1], then OWA becomes the minimum operator.

By definition, the OWA operator performs an ordering of the arguments to be aggregated. The ordering is done based on the values of the arguments. In other application, the argument's value is not the one that makes the argument important. Instead, we have an auxiliary value which can give us a confidence level in the argument. In this scenario, we can use the *induced OWA* operator, which relies on an auxiliary value, called *order-inducing variable,* to order the arguments.

This scenario applies to us in this work, because the actual value of an estimator output does not give us any indication of its importance or confidence. (Recall that in our case, the arguments represent outputs of multiple estimators.) Thus, we need to use an order-inducing variable that measures the confidence in the estimator's output. We discuss this issue in Sect. 2.4.

2.3 Prioritized Aggregation Operator (PAO)

An issue of considerable interest, in applications of the induced OWA operator, is the determination of the weights to be used. To this end, various approaches have been suggested for obtaining these weights [53–56]. One elegant way is the prioritized aggregation operator (PAO), presented in [55]. Let the order-inducing variable be S_j (in our case this will be the predicted residual errors) and let $S_0 = 1$. The weights in the PAO approach are defined as follows. First, we define:

$$T_1 = 1 \quad T_2 = S_1 \quad T_3 = S_1 S_2. \tag{2}$$

Thus, in general we can write:

$$T_j = \prod_{k=1}^{j} S_{k-1}. \tag{3}$$

Finally, the weights are defined as follows:

$$w_j = \frac{T_j}{\sum_{j=1}^{P} T_j}. \tag{4}$$

The definition in (4) guarantees that $\sum_{j=1}^{P} w_j = 1$. Recall here that P is the number of arguments and hence the number of estimators.

2.4 Proposed Fusion Approach Using Induced OWA

Recall that in applications of the induced OWA operator, we need to define an order-inducing variable, which is a variable that can help us order the predicted outputs of the estimators from most confident to least confident. In other words, the order-inducing variable measures the confidence in the predicted output of an estimator.

Our idea is to use the *residual error*, between predicted and true outputs, as a measure of confidence in the predicted output. We can do that by analyzing and modeling the residual errors of the estimator in the input space. Obviously, to compute the residual error in the model output, we need to know the true output. However, recall that the true output is known during training, but not during testing of the model. Thus, the idea is to use another dedicated CNN model to learn how to predict the residual error from the input (image). In other words, we have a pair of CNN models, one to predict the actual output and another one to predict the residual error in the output. This pair of models is illustrated in Fig. 2, where Fig. 2a shows the main CNN models used for predicting the object presence, while Fig. 2b shows the two models used to predict the residual error in the outputs of the main model. In Fig. 2a, we also illustrate

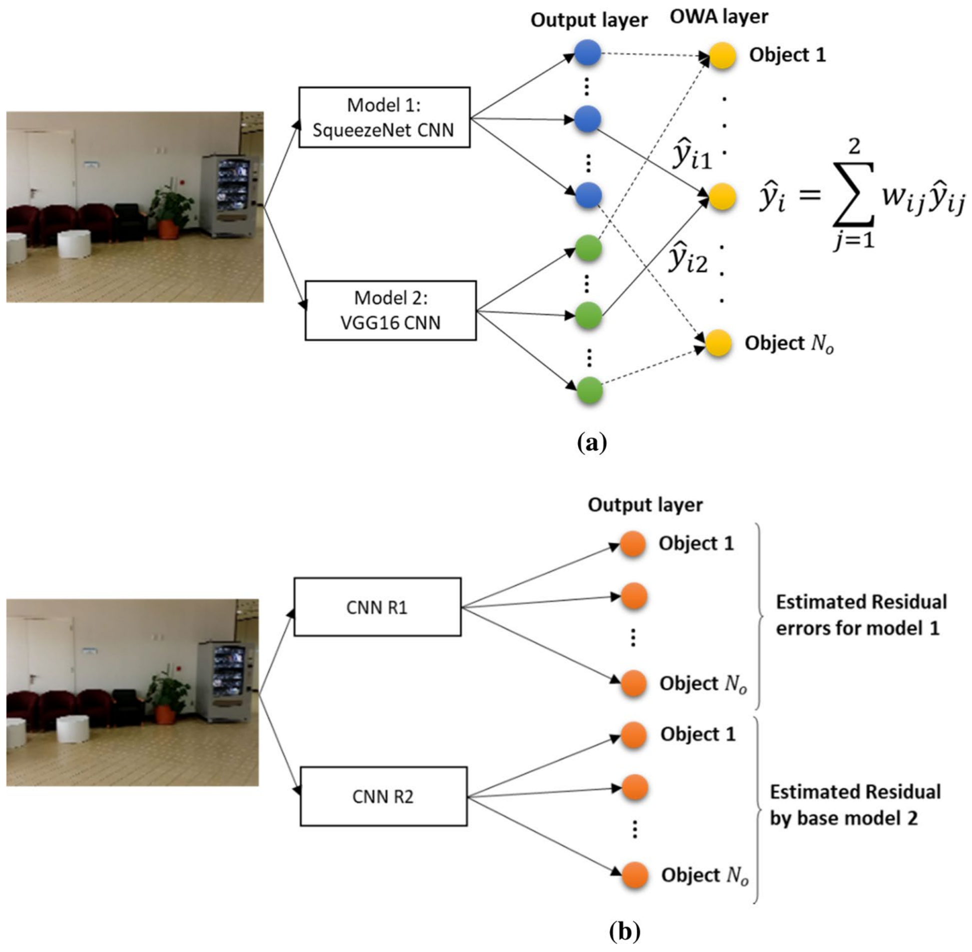

Fig. 2 Overview of proposed deep model with fusion of two pre-trained CNN models. **a** The two pre-trained models are used to predict the object presence and then fused using OWA, **b** Two models to predict the residual error in the outputs of the models in part (**a**)

the fusion operation performed at the decision level using the OWA approach.

The residual error models shown in Fig. 2b can perform the learning of the residual error, because we can compute the actual error during training from the predicted and true outputs. Accordingly, the proposed method involves the following steps:

1. Train the two main models shown in Fig. 2a.
2. Compute the actual residual errors for all training image samples.
3. Train two residual error models, shown in Fig. 2b, using the input images and the corresponding residual errors computed in step (2).
4. During test time, we use main CNN models in Fig. 2a to predict the object presence from the input image.
5. Then, we use the models in Fig. 2b to estimate the residual errors of both main CNN models.
6. Finally, we use the estimated residual errors to fuse the outputs of the two main CNN models based on the induced OWA approach.

Thus, given a sample input image, let y_j be the true object label (equal to one if object j is present). Furthermore, let $\hat{y}_{j1}$ and $\hat{y}_{j2}$ be the two predictions produced by the two main models for object j (These are real-valued numbers). Next, let $e_{j1} = \text{abs}\left(\hat{y}_{j1} - y_j\right)$ and $e_{j2} = \text{abs}\left(\hat{y}_{j2} - y_j\right)$ be the absolute values of the residual errors corresponding to the two main models. Then obviously, the lower residual error indicates higher confidence in the prediction. But recall that we need the order-inducing variable to order the predictions from most confident to least confident. Thus, we propose the following definition for the order-inducing variable:

$$S_{jk} = \frac{1}{1 + \text{abs}\left(\hat{y}_{jk} - y_{jk}\right)} \quad \text{for} \quad k = 1, 2. \tag{5}$$

Next, as explained earlier, the weights are defined by first ordering the predictions $\hat{y}_{j1}$ and $\hat{y}_{j2}$ based on their order-inducing variable S_{jk} and then computing the weights using the PAO approach. Therefore, we have two cases:

Case 1 $S_{j1} \geq S_{j2}$

$$w_{j1} = \frac{1}{1 + S_{j1}} \quad \text{and} \quad w_{j2} = \frac{S_{j1}}{1 + S_{j1}}. \tag{6}$$

Case 2 $S_{j2} \geq S_{j1}$

$$w_{j1} = \frac{S_{i2}}{1 + S_{j2}} \quad \text{and} \quad w_{j2} = \frac{1}{1 + S_{j2}}. \tag{7}$$

Finally, based on these weights, the final fused prediction for object j is computed as follows:

$$\hat{y}_j = \sum_{k=1}^{2} w_{jk}\hat{y}_{jk}. \tag{8}$$

It is worth observing here that the weights are not fixed for all objects; instead, they vary depending on residual errors predicted (from the test image) using the dedicated CNN models. This is why, this efficient fusion approach is able to select the better prediction for each object and hence produce improvement in the accuracy over the whole testing set.

Lastly, recall that the final predicted output $\hat{y}_j$ is compared to the presence threshold T_P to decide whether the object j is present or not.

3 Experimental Results

In the first part of this section, we describe the datasets used in this study and how they are collected. Then, we optimize the base CNN models for these datasets. In particular, we find experimentally the best presence threshold T_p. Finally, we present preliminary results of the proposed solution.

3.1 Dataset Description

The experiments in this paper use four datasets of multi-labeled images for evaluating the efficiency of the proposed deep learning solution. The first two datasets pertain to the college of computer and information sciences building at the King Saud University (KSU), Saudi Arabia. They have been collected by our team using a CMOS camera with the following features 87.2 fps, 752 × 480, 0.36 MPix, 1/3″, ON Semiconductor, Global Shutter, and connection via USB 2.0.

The second two datasets are collected by the authors of [29] in two different buildings of the University of Trento, Italy. The cameras used to capture these images are from a company called IDS-imaging [57]. The authors used the camera model UI-1240LE -C-HQ, which is CMOS-based camera with 25.8 fps, 1280 × 1024, 1.31 MPix, 1/1.8″, e2v, Global Shutter, Global Start Shutter, Rolling Shutter, and USB 2.0 support. The camera is equipped with KOWA LM4NCL 1/2″ 3.5-mm F1.4 manual IRIS C-Mount lens from RMA Electronics Inc. Company [58]. The details of these four datasets are given in Table 1, which also presents the list of objects considered in every dataset.

It is noteworthy that we have selected the objects deemed to be the most important ones in the considered indoor environments. Also note that the datasets are not randomly split into training and testing images, because we cannot guarantee that all objects will be represented in the training set. The split into training and test images is performed manually beforehand and is fixed for all experiments. Table 2 presents the number of occurrences of objects in the training set and the test set for each dataset, while Fig. 3 shows four sample images with the list of objects contained within.

3.2 Assessment Metrics

In order to assess the proposed solution, we need to define quantitative performance metrics. In single-label classification, we use metrics such as precision, sensitivity (recall), specificity, and accuracy. These metrics can also be used in the multi-label case, and they can be computed per label/object or as an overall metric.

Let x_i represent a sample image in the test set where $1 \leq i \leq N_{\text{test}}$, and let Y_i represent the set of true labels or objects associated with it. In addition, let P be a multi-label classifier that returns the set of predicted labels for x_i.

Table 1 Details of the four datasets of indoor scenes used in the work

Dataset	Train/test sizes	Objects considered
KSU1	161/159	Pillar, fire extinguisher/hose, trash can, chairs, external door, hallway, self-service, reception, didactic service machine, display screen, board, stairs, elevator, laboratory, internal door
KSU2	86/88	Board, fire extinguisher/hose, trash cans, chairs, external door, didactic service machine, self-service, reception, cafeteria, display screen, pillar, stairs, elevator, prayer room, internal door
UTrento1	58/72	External window, board, table, external door, stair door, access control reader, office, pillar, display screen, people, ATM, chairs, bins, internal door, and elevator
UTrento2	61/70	Stairs, heater, corridor, board, laboratories, bins, office, people, pillar, elevator, reception, chairs, self-service, external door, and display screen

Table 2 Number of occurrences of objects in the training set of each dataset

KSU1		KSU2		UTrento1		UTrento2	
Object	# Occ	Object	# Occ	Object	#	Object	#
Pillar	7	Board	38	External window	9	Stairs	14
Fire extinguisher/hose	65	Fire extinguisher/hose	28	Board	28	Heater	12
Trash can	20	Trash cans	22	Table	4	Corridor	33
Chairs	83	Chairs	37	External door	14	Board	45
External door	21	External door	18	Stair door	14	Laboratories	15
Hallway	44	Didactic service machine	7	Access control reader	5	Bins	15
Self-service	17	Self-service	17	Office	37	Office	13
Reception	5	Reception	8	Pillar	34	People	**8**
Didactic service machine	10	Cafeteria	5	Display screen	18	Pillar	11
Display screen	11	Display screen	14	People	1	Elevator	6
Board	24	Pillar	4	ATM	7	Reception	10
Stairs	16	Stairs	7	Chairs	28	Chairs	4
Elevator	9	Elevator	6	Bins	14	Self-service	6
Laboratory	5	Prayer room	5	Internal door	35	External door	6
Internal door	53	Internal door	32	Elevator	5	Display screen	8

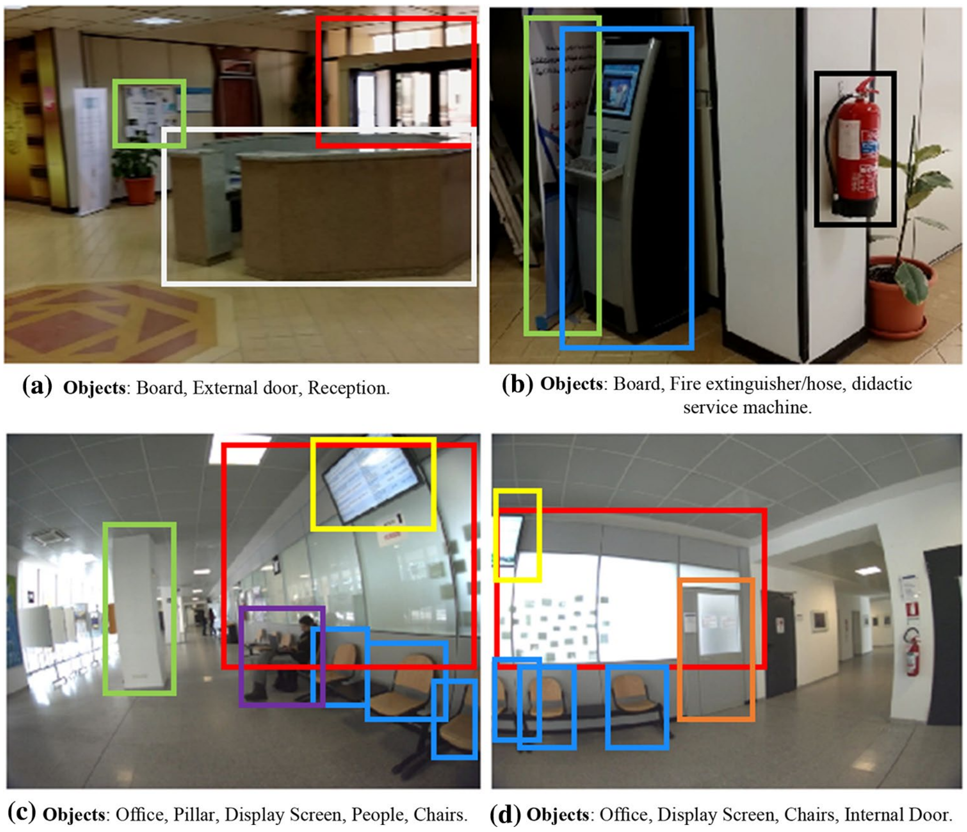
(a) **Objects**: Board, External door, Reception.
(b) **Objects**: Board, Fire extinguisher/hose, didactic service machine.
(c) **Objects**: Office, Pillar, Display Screen, People, Chairs.
(d) **Objects**: Office, Display Screen, Chairs, Internal Door.

Fig. 3 Sample images from the datasets and the set of objects present (image multi-labels). **a** KSU1 dataset, **b** KSU2 dataset, **c** UTrento1 dataset, and **d** UTrento2 dataset

For the label y_k, four basic quantities characterizing the binary classification performance on this label can be defined:

$$\mathrm{TP}_k = \left|\left\{x_i | y_k \in Y_i \text{ and } y_k \in P(x_i), 1 \le i \le N_{\text{test}}\right\}\right|;$$
$$\mathrm{FP}_k = \left|\left\{x_i | y_k \notin Y_i \text{ and } y_k \in P(x_i), 1 \le i \le N_{\text{test}}\right\}\right|;$$
$$\mathrm{FN}_k = \left|\left\{x_i | y_k \in Y_i \text{ and } y_k \notin P(x_i), 1 \le i \le N_{\text{test}}\right\}\right|;$$
$$\mathrm{TN}_k = \left|\left\{x_i | y_k \notin Y_i \text{ and } y_k \notin P(x_i), 1 \le i \le N_{\text{test}}\right\}\right|.$$

The quantities represent the number of true positives, false positives, false negatives, and true negatives with respect to label y_k, respectively. It can be easily checked that $\mathrm{TP}_k + \mathrm{FP}_k + \mathrm{FN}_k + \mathrm{TN}_k = N_{\text{test}}$. Based on these qualities, we define the metrics precision (PRE), sensitivity or recall (SEN), specificity (SPE), and accuracy (ACC) in Eqs. (9)–(12).

$$\mathrm{PRE}_k = \frac{\mathrm{TP}_k}{\mathrm{TP}_k + \mathrm{FP}_k}, \tag{9}$$

$$\mathrm{SEN}_k = \frac{\mathrm{TP}_k}{\mathrm{TP}_k + \mathrm{FN}_k}, \tag{10}$$

$$\mathrm{SPE}_k = \frac{\mathrm{TN}_k}{\mathrm{TN}_k + \mathrm{FP}_k}, \tag{11}$$

$$\mathrm{ACC}_k = \frac{\mathrm{TP}_k + \mathrm{TN}_k}{\mathrm{TP}_k + \mathrm{FP}_k + \mathrm{TN}_k + \mathrm{FN}_k}. \tag{12}$$

To get the overall metrics, we can just take the average of the individual metrics per label. In the multi-label case, ACC is ambiguous [59, 60]. However, balanced or average accuracy (AVG) can be used instead:

$$\mathrm{AVG}_k = \frac{\mathrm{SEN}_k + \mathrm{SPE}_k}{2}. \tag{13}$$

In regular classification, with a single label per image, the classification per image is either correct or wrong. However, multi-label classification problems have a chance of being partially correct, because some labels may be detected, while others not. Thus, there are other evaluation metrics specific to multi-label classification that takes this fact into account. The Hamming loss (HL) is probably the most widely used loss function in multi-label classification. The HL is used for measuring the fraction of incorrectly predicted labels. First, we define it per image sample HL_i:

$$\mathrm{HL}_i = \frac{1}{N_{\text{labels}}}\left|P(x_i)\Delta Y_i\right|. \tag{14}$$

Here, Δ stands for the symmetric difference between two sets and N_{labels} is the number of objects/labels. The overall HL can then be computed by taking the average over all sample images in the test set:

$$\mathrm{HL} = \frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}} \mathrm{HL}_i. \tag{15}$$

The mean average precision (mAP) is a ranking metric which refers to the average fraction of relevant labels ranked higher than the irrelevant ones. First, we compute the precision/recall curve for a particular label/object over the whole test set. Then, the average precision per object is computed as the area under the precision–recall curve:

$$\mathrm{AP}_k = \int_0^1 p(r)\mathrm{d}r. \tag{16}$$

Obviously, the precision/recall is discrete and thus AP needs to be approximated. Recently, it is approximated by finding the area under the precision–recall curve which is known as area under curve (AUC). Finally, mAP is computed the average of the individual AP_k.

$$\mathrm{mAP} = \frac{1}{N_{\text{labels}}}\sum_{k=1}^{N_{\text{labels}}} \mathrm{AP}_k. \tag{17}$$

The label ranking loss (RL) is another metric that computes the average number of label pairs that are incorrectly ordered. In other words, it computes the fraction of reversely ordered label pairs, i.e., an irrelevant label is ranked higher than a relevant label. For more details, we refer the reader to [61].

3.3 Training the CNN Models

We implement the proposed deep CNN models for image multi-label classification in the Keras environment with Tensorflow as a back end. Tensorflow is end-to-end open-source machine learning platform developed by Google and that can be programmed in Python. However, due to its peculiar programming style it is often used through a higher-level programming interface such as Keras (also written in Python).

All experiments are conducted on HP-laptop with an Intel Core i7-7700HQ CPU, the NVIDIA graphics card GeForce GTX 1060 Ti with 4 GB dedicated memory, and 8 GB of RAM. The size of the images contained in the datasets is 640×480. We set up the CNN base models so that they accept the images with their original size.

Figure 4 shows the plot of the curves of the loss versus epoch number for training the different CNN models on the first dataset (KSU1) as an example. We set the batch size to 16 and the learning rate to 0.001. From these plots, we can

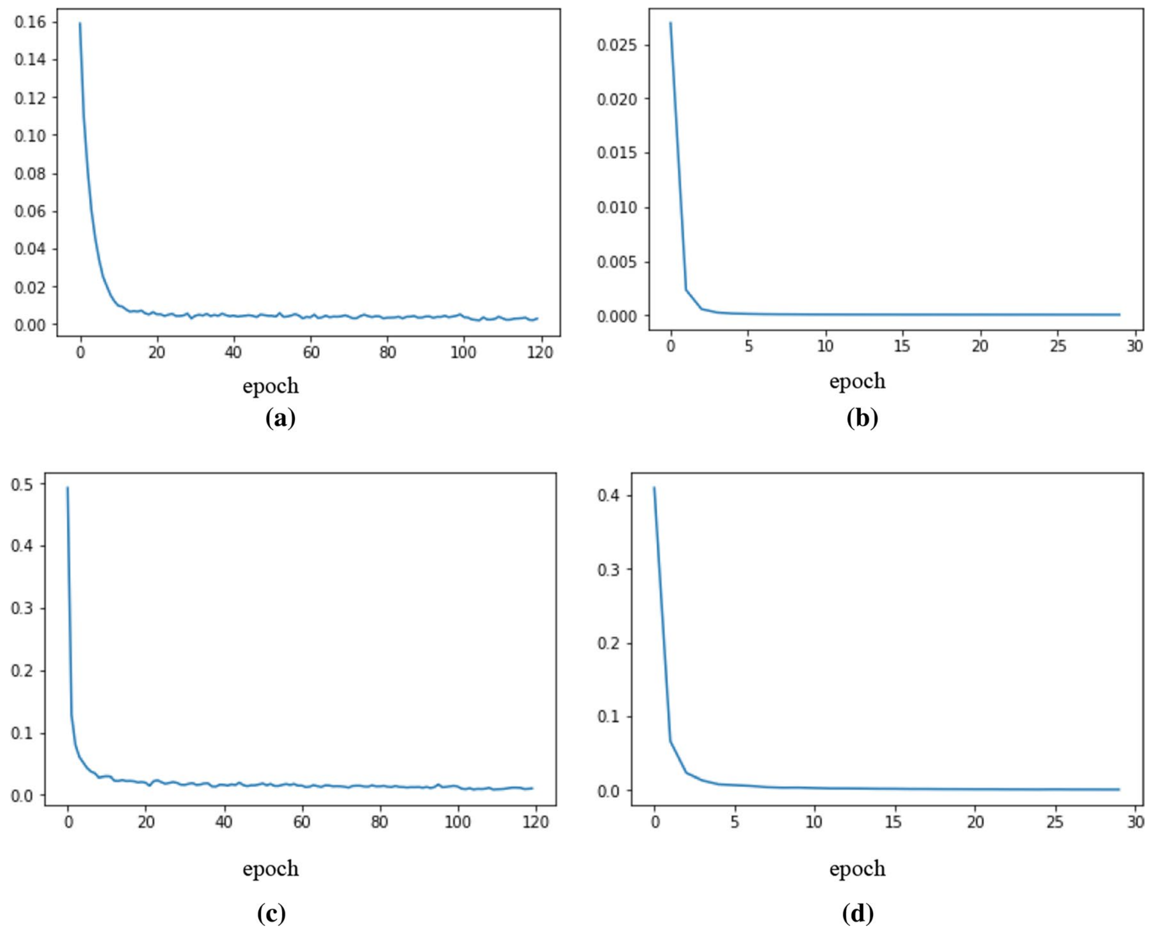

Fig. 4 Loss curves for training the CNN models. **a** Loss for SqueezeNet main model, **b** loss for SqueezeNet residual-error model, **c** loss for VGG-16 main model, and **d** loss for VGG-16 residual-error model

see that the loss function converges with 100 epochs for both models when training them to classify the actual images. In order to improve the stability of the models' convergence, we reduce the learning rate after epoch 100 from 0.001 to 0.0001 and train them for 20 epochs more. As for the CNN models dedicated to learn the residual error, they actually converge much quicker. Thus, we only trained them for 30 epochs each with a learning rate equal to 0.001 and a batch size of 16.

3.4 Determining the Optimal Presence Threshold T_P

Recall that an important parameter used in our CNN models is the presence threshold T_P which is used to convert the outputs into binary values. It is reasonable to assume that the best value for this parameter is 0.5, but an ablation study is needed to obtain the optimal value according to our defined metrics. Thus, in this set of experiments we examine the AVG accuracy of the base models with respect to the threshold T_P. The results are shown in Fig. 5 and clearly show that the best threshold value is 0.3 and not 0.5, and this is true for all datasets. We note here that we execute each training experiment ten times and plot the average of the AVG metric. The small bars at each point indicate the standard deviation of the ten runs.

3.5 Results for a Sample Run

The proposed fusion algorithm relies clearly on the correct estimation of the residual values between true and predicted outputs. To estimate the residual values, we can use any one of the two CNN models. Obviously, it is preferable to use the model that gives a better prediction of the residual values. To that end, we train both model types, SqueezeNet and VGG16, and evaluate them based on the mean squared error (MSE) between the true residuals and the one predicted. Table 3 shows the obtained MSE results. Based on this comparison, it is clear that SqueezeNet is more accurate in predicting the residual values. Thus, we use this model to estimate the residual values after each main model.

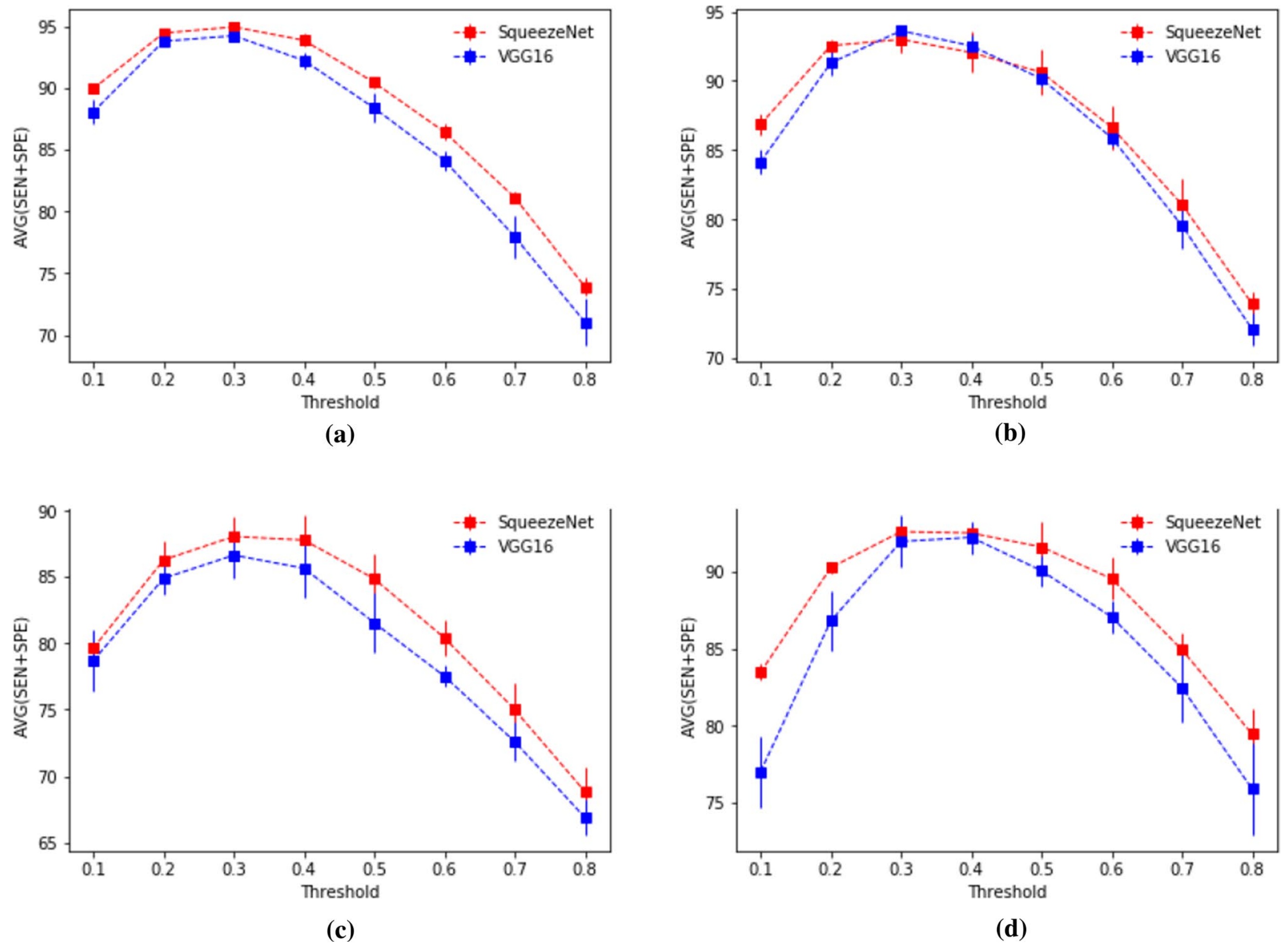

Fig. 5 A sensitivity of the base models with respect to the threshold parameter T_P, results for **a** KSU1, **b** KSU2, **c** UTrento1, and **d** UTrento2

Table 3 MSE between true residual and estimated residual on test sets

Model	Dataset			
	KSU1	KSU2	UTrento1	UTrento2
SqueezeNet	0.02975	0.04989	0.06509	0.04683
VGG16	0.05643	0.10568	0.15982	0.11981

Next, we tested the proposed method on the four datasets described in Section. Tables 4 and 5 present the results showing the AVG accuracies for the two based models and the fused model. Obviously, here we show the results of one sample run only. These results use the presence threshold $T_P = 0.3$.

From Tables 4 and 5, we can see that for all datasets, the proposed fused model produced improvement in the AVG accuracy, compared to both of the base models. We can also observe that in general SqueezeNet outperforms VGG16 in terms of AVG results. However, the fusion of the two models produces a good improvement on average for all datasets. That is, because even though SqueezeNet usually performs better, for some images VGG16 does a better job. For example, by looking at the "people" object in Table 5 (found in UTrento datasets), we clearly see that VGG16 outperforms SqueezeNet for this object type. Thus, we can say that the two models complement each other, as the VGG16 model in a sense corrects the SqueezeNet modes for those specific object classes.

When we look at the overall metrics, we can observe that the fused model outperforms the separate models for almost all metrics. There are few exceptions (highlighted in bold font) such as for the mAP metric in the KSU2 dataset, where the fused model achieved 80.69, whereas the SqueezeNet model achieved 81.43. The other exception is the UTrento1 dataset with respect to the RL metric. Here, the RL metric for the two separate models achieved quite different values, namely 0.185 and 0.125.

Table 4 A sample of detailed results for each class in the datasets KSU1 and KSU2 using $T_P = 0.3$

KSU1					KSU2				
Objects	# Occ	M1	M2	Fused	Objects	# Occ	M1	M2	Fused
Pillar	8	93.42	93.42	93.42	Board	35	83.94	87.71	86.77
Fire extinguisher/hose	66	95.04	87.63	94.94	Fire extinguisher/hose	25	94.83	92.44	92.44
Trash can	21	93.43	92.13	94.15	Trash cans	23	90.40	91.67	92.58
Chairs	74	97.06	94.03	95.88	Chairs	37	91.18	92.77	92.77
External door	29	91.95	94.06	94.06	External door	18	95.08	94.37	95.08
Hallway	42	95.82	94.63	95.48	Didactic service machine	10	95.00	100.00	100.00
Self-service	14	92.17	95.05	95.39	Self-service	16	95.49	93.40	94.79
Reception	5	99.68	100.00	100.00	Reception	6	100.00	98.78	99.39
Didactic service machine	7	92.53	92.20	92.53	Cafeteria	4	99.41	100.00	99.41
Display screen	12	99.66	95.15	99.66	Display screen	16	99.31	99.31	100.00
Board	23	94.18	90.91	94.18	Pillar	3	100.00	100.00	100.00
Stairs	24	98.15	95.09	97.18	Stairs	7	98.77	97.53	98.15
Elevator	12	100.00	100.00	100.00	Elevator	9	99.37	98.73	99.37
Laboratory	4	100.00	100.00	100.00	Prayer room	8	80.63	80.00	80.63
Internal door	41	86.97	86.49	86.92	Internal door	25	84.89	75.37	83.30
Overall	SEN	94.34	93.55	94.98	Overall	SEN	94.00	95.07	94.77
	SPE	96.33	94.56	96.20		SPE	93.77	91.88	93.85
	AVG	95.34	94.05	95.59		AVG	93.88	93.47	94.31
	mAP	82.80	80.65	84.97		**mAP**	**81.43**	**77.54**	**80.69**
	HL	0.035	0.046	0.034		HL	0.053	0.067	0.053
	RL	0.069	0.082	0.064		RL	0.112	0.121	0.110

M1 SqueezeNet, *M2* VGG16

Table 5 A sample of detailed results for each class in the datasets UTrento1 and UTrento2 using $T_P = 0.3$

UTrento1					UTrento2				
Objects	# Occ	M1	M2	Fusion	Objects	# Occ	M1	M2	Fusion
External window	9	85.83	71.20	75.74	Stairs	17	97.17	97.17	97.17
Board	28	84.76	74.88	79.88	Heater	16	93.52	80.56	92.59
Table	4	100.00	99.07	100.00	Corridor	40	91.67	97.50	97.08
External door	14	91.88	89.29	94.16	Board	46	61.41	78.08	67.66
Stair door	14	96.59	92.05	93.18	Laboratories	14	94.64	96.43	98.21
Access control reader	5	88.11	90.00	88.11	Bins	13	90.89	84.21	86.51
Office	37	88.10	97.62	97.62	Office	12	95.69	82.76	95.69
Pillar	34	86.03	94.98	92.89	People	6	72.40	92.97	89.84
Display screen	18	85.14	63.89	77.78	Pillar	12	98.28	99.14	99.14
People	1	47.37	98.25	99.12	Elevator	6	100.00	97.66	100.00
ATM	7	99.02	98.04	98.04	Reception	12	95.69	94.83	95.69
Chairs	28	95.00	91.19	94.88	Chairs	3	97.02	97.02	97.02
Bins	14	89.61	82.63	88.47	Self-service	7	100.00	84.13	100.00
Internal door	35	86.96	69.57	86.96	External door	7	100.00	99.21	100.00
Elevator	5	100.00	100.00	100.00	Display screen	9	98.36	93.63	99.18
Overall	SEN	87.41	86.19	90.92	Overall	SEN	96.65	96.88	99.18
	SPE	89.17	88.83	91.32		SPE	88.25	86.49	89.60
	AVG	88.29	87.51	91.12		AVG	92.45	91.68	94.39
	mAP	74.51	72.53	76.64		mAP	75.26	66.34	77.20
	HL	0.075	0.097	0.072		HL	0.074	0.104	0.069
	RL	**0.125**	**0.185**	**0.132**		RL	0.104	0.146	0.093

M1 SqueezeNet, *M2* VGG16

Table 6 Comparison to state-of-the-art methods

Dataset	Method	SEN	SPE	AVG	HL	Time (s)
KSU1	EDCS + GPR [28]	73.82	80.53	77.18		> 1
	MR random projection [30]	71.65	94.81	83.23		0.035
	Pre-trained GoogLeNet [36]	83.56	97.95	90.76		0.064
	Pre-trained ResNet [39]	81.16	95.84	88.50		0.107
	Convolutional SVM Net [26]	89.25	96.88	93.07		0.201
	Fine-tuning SqueezeNet [25]	93.88 ± 0.25	96.20 ± 0.31	95.04 ± 0.19	0.035 + 0.002	0.024
	Pre-trained VGG16 [ours]	92.71 ± 1.25	95.42 ± 0.53	94.07 ± 0.50	0.041 + 0.003	0.013
	Fusion using OWA [ours]	93.95 ± 0.58	96.28 ± 0.19	95.11 ± 0.30	0.034 + 0.001	0.071
KSU2	EDCS + GPR [28]	64.46	72.91	68.69		> 1
	MR random projection [30]	70.00	92.33	81.16		0.037
	Pre-trained GoogLeNet [36]	81.76	95.91	88.84		0.067
	Pre-trained ResNet [39]	82.16	95.84	89.00		0.110
	Convolutional SVM Net [26]	88.72	94.19	91.46		0.198
	Fine-tuning SqueezeNet [25]	92.72 ± 1.73	93.27 ± 0.45	93.00 ± 0.95	0.059 + 0.004	0.029
	Pre-trained VGG16 [ours]	94.84 ± 0.20	92.39 ± 0.77	93.62 ± 0.37	0.066 + 0.008	0.015
	Fusion using OWA [ours]	94.73 ± 0.29	93.98 ± 0.08	94.36 ± 0.16	0.054 + 0.003	0.075
UTrento1	SURF + GPR [28]	71.16	100.00	85.58		1.170
	EDCS + GPR [28]	69.66	80.19	74.93		1.080
	MR random projection [30]	71.16	83.20	77.18		0.037
	Pre-trained GoogLeNet [36]	78.65	94.34	86.49		0.068
	Pre-trained ResNet [39]	71.16	93.84	82.50		0.207
	Convolutional SVM Net [26]	84.26	89.14	86.70		0.206
	Fine-tuning SqueezeNet [25]	88.36 ± 3.07	87.64 ± 1.33	88.00 ± 1.48	0.085 + 0.007	0.013
	Pre-trained VGG16 [ours]	86.01 ± 4.57	87.19 ± 1.55	86.60 ± 1.73	0.109 + 0.011	0.017
	Fusion using OWA [ours]	91.37 ± 0.69	89.42 ± 1.17	90.39 ± 0.40	0.081 + 0.005	0.065
UTrento2	SURF + GPR [28]	77.72	100.00	88.86		1.170
	EDCS + GPR [28]	70.00	90.12	80.06		1.200
	MR random projection [30]	77.18	91.41	84.30		0.037
	Pre-trained GoogLeNet [36]	83.63	96.86	90.25		0.066
	Pre-trained ResNet [39]	89.54	96.38	92.96		0.208
	Convolutional SVM Net [26]	93.64	92.17	92.90		0.115
	Fine-tuning SqueezeNet [25]	96.05 ± 0.49	89.14 ± 0.75	92.59 ± 0.49	0.070 + 0.006	0.014
	Pre-trained VGG16 [ours]	97.20 ± 1.53	86.70 ± 2.62	91.95 ± 1.68	0.085 + 0.019	0.019
	Fusion using OWA [ours]	97.66 ± 1.21	89.86 ± 1.19	93.76 ± 0.56	0.062 + 0.006	0.084

The RL of the fused model should be lower than both RL of the two models, which is not the case. However, the RL value of 0.132 achieved by the fused model is significantly lower than the VGG16 model and closer to the optimal result of the SqueezeNet model.

3.6 Comparison to State-of-the-Art

We also present comparison with the state-of-the-art methods in Table 6. These state-of-the-art methods include: (1) SURF matching and Gaussian process regression (SURF + GPR) [28], (2) compressive sensing and Gaussian process regression (CS + GP) [28], (3) multi-resolution random projection (MR-random-proj) [30], (4) pre-trained GoogLeNet CNN [36], (5) pre-trained ResNet CNN [39], (6) fine-tuning of SqueezeNet CNN [25], and (7) a recent method based on convolutional SVM networks (convolutional SVM Net) [26].

Again, these results are obtained using $T_P = 0.3$ as the presence of threshold. The results in Table 6 clearly show that the proposed method produced significant improvements over the state-of-the-art methods for all datasets, except KSU1, where the improvement is insignificant.

From Tables 4, 5, and 6, we notice that SqueezeNet produces better results than VGG16 on average. It is very important to observe here that the fusion technique computes different weights for each image. In other words, for each image, a different weighting scheme is computed that favors the best model for the current image. Therefore, even though SqueezeNet does better than VGG16 on average, the latter does better for certain images. The proposed OWA fusion technique is able to discover and exploit those cases by adjusting its weights accordingly.

We also notice from Table 6 that the improvements are more significant for UTrento1 and UTrento2 datasets. These two datasets are more challenging as can be illustrated by their lower AVG accuracies achieved. One possible reason for that is that they contain barrel-like distortions as can be observed in Fig. 3. The challenging nature of these two datasets partly explains why the proposed method provides a more significant improvement for them. The two CNN models disagree more for these two datasets, and the fusion technique is able to exploit this disagreement in a complementary way to enhance the detection result.

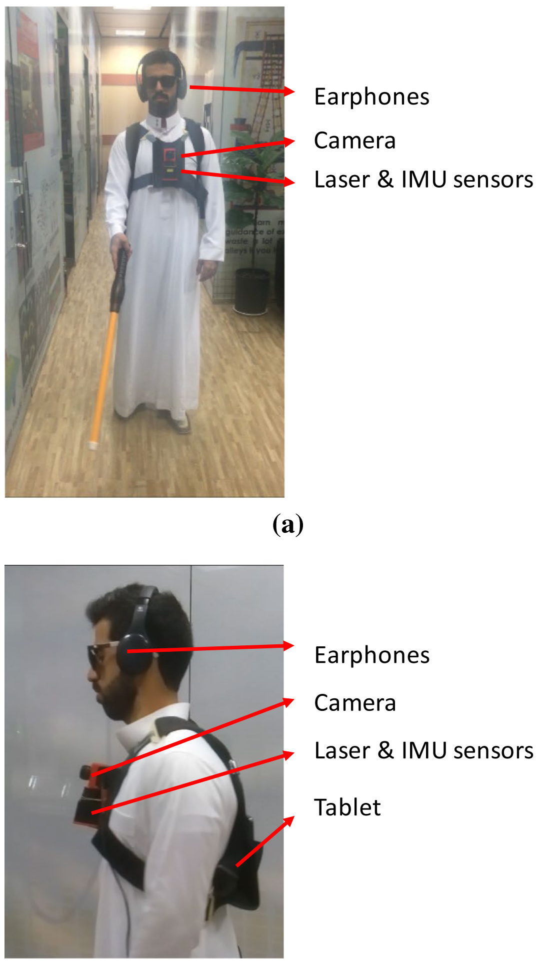

Fig. 6 BlindSys overview. **a** Frontal view, **b** Side view showing the camera on the chest, tablet in the back packet, and the headphones

3.7 Hardware Implementation Using FPGA

This work describes a module that is part of a bigger project, called BlindSys, to help the VI persons with vision tasks such as navigation, text recognition, object detection, and recognition of faces. The project involves a hardware part, which is illustrated in Fig. 6. The system hardware includes a wide-angle camera, earphone, laser and inertial measurement unit (IMU) sensors, and of a high-end 10-inch mobile device (tablet).

However, the tablet does not contain any graphical processing units, and thus, it is not able to handle real-time execution of some tasks. Thus, using a dedicated hardware circuitry to execute network models, such as field programmable gate arrays (FPGA), is an attractive solution.

Modern neural networks are computationally expensive and require specialized hardware, such as graphics processing units. The use of mobile devices without further optimization may not provide sufficient performance when high processing speed is required such as in our computer vision system to support the VI persons. We can speed up neural networks by moving CNN computation from software to hardware, namely an FPGA implementation, and by using fixed-point calculations instead of floating point.

The unique flexibility of the FPGA fabric allows the logic precision to be adjusted to the minimum that a particular network design requires. By limiting the bit precision of the CNN calculation, the number of images that can be processed per second can be significantly increased, improving performance and reducing power, while achieving the exact performance of the corresponding software implementation.

For example, the authors in [62] proposed an FPGA implementation of pre-trained deep neural networks from VGG16. They used dynamic precision quantization with 48-bit data representation and singular vector decomposition to reduce the size of fully connected layers, which led to smaller number of weights that had to be passed from the device the external memory. Another work by Zhang et al. [63] analyzed the throughput and required memory bandwidth for various CNNs using optimization techniques, such as loop tiling and transformation. They achieved 17.42× speedup for the AlexNet CNN [17]. Another work by Suda et al. [64] considers a higher-level solution, which uses the OpenGL compiler for deep networks. Using their method, they were able to implement two large-scale CNNs, namely AlexNet and VGG16, on two Altera Stratix-V FPGA platforms, DE5-Net and P395-D8 boards, which have different hardware resources.

Recently, the authors in [65] implemented a fully connected neural network with six layers and 64 neurons using the FPGA Cyclone IV GX FPGA DE2i-150 from Altera. The descriptions of the digital blocks are performed using fixed-point notation, which provides a high speed and a low cost of hardware resources. In this manner, $N = 32$ bits are used to represent a fixed-point format, where the most significant bit represents the sign, three bits are used to represent the integer part, and 28 bits are used to represent the fractional part. Recently, Duarte et al. [66] have proposed a protocol for automatic conversion of fully connected neural network implementations in high-level programming language to intermediate format (HLS) and then into FPGA implementation.

However, implementing even deeper networks with multiple dozens of layers is problematic, since all layer weights would not fit into the FPGA memory and will require the use of the external RAM, which can lead to the decrease in performance. Moreover, due to the large number of layers, error accumulation will increase and will require wider bit range to store fixed-point weight values. All of the above findings make our solution more advantageous because it is based on SqueezeNet and Vgg16 CNN which are shallow models with low number of layers compared to other more recent models in the literature. In fact, as we mentioned previously, FPGA implementations for the larger of the two models (VGG16) have already been successfully proposed in the literature.

4 Conclusions

In this work, we present a novel computer vision method for the detection of the presence of multiple objects in a scene. The method represents a module in a larger assistive technology system for the visually impaired. We propose an innovative idea to fuse two CNN models, namely VGG16 and SqueezeNet, based on dedicated CNN models to estimate the residual error in predicted outputs in combination with an OWA approach.

The experimental results on four image datasets of indoor environments from two separate locations show significant improvements compared to state-of-the-art methods. The proposed OWA approach, based on estimating the residual of the CNN outputs and using it as confidence values, is able to select the better of the two CNN outputs. One way to improve the results is to add more CNN models to the ensemble. However, this will also increase the computational time. Another more promising direction is to employ augmentations techniques to increase the dataset size, because usually CNN models require large datasets for training. Finally, an interesting idea is to divide the objects of interest among multiple CNN models. For example, we can use three CNN models of the same type or different types; each one is responsible for detecting five objects only out of 15. It is expected that reducing the number of objects per CNN should increase its performance.

Acknowledgements This work was supported by NSTIP strategic technologies programs, number (13-MED1343-02), in the Kingdom of Saudi Arabia.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Hoy, M.B.: Alexa, Siri, Cortana, and more: an introduction to voice assistants. Med. Ref. Serv. Q. **37**, 81–88 (2018)
2. Badue, C.; Guidolini, R.; Carneiro, R.V.; Azevedo, P.; Cardoso, V.B.; Forechi, A.; Jesus, L.F.R.; Berriel, R.F.; Paixão, T.M.; Mutz, F.W.; Oliveira-Santos, T.; de Souza, A.F.: Self-driving Cars: A Survey. arXiv (2019)
3. Atkinson, R.D.: Robotics and the Future of Production and Work. Information Technology and Innovation Foundation, Washington, DC (2019)
4. Alsohybe, N.T.; Dahan, N.A.; Ba-Alwi, F.M.: Machine-translation history and evolution: survey for Arabic–English translations. arXiv (2017)
5. Elmannai, W.; Elleithy, K.: Sensor-based assistive devices for visually-impaired people: current status, challenges, and future directions. Sensors **17**, 565 (2017)
6. Tapu, R.; Mocanu, B.; Zaharia, T.: Wearable assistive devices for visually impaired: a state of the art survey. Pattern Recognit. Lett. (2018)
7. Hasanuzzaman, F.M.; Yang, X.; Tian, Y.: Robust and effective component-based banknote recognition for the blind. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **42**, 1021–1030 (2012)
8. López-de-Ipiña, D.; Lorido, T.; López, U.: BlindShopping: enabling accessible shopping for visually impaired people through mobile technologies. In: Abdulrazak, B., Giroux, S., Bouchard, B., Pigot, H., Mokhtari, M. (eds.) Toward Useful Services for Elderly and People with Disabilities, pp. 266–270. Springer, Berlin (2011)
9. Tekin, E.; Coughlan, J.M.: An algorithm enabling blind users to find and read barcodes. In: Proceedings of IEEE Workshop on Application of Computer Vision 2009, pp. 1–8 (2009)
10. Chen, X.; Yuille, A.L.: Detecting and reading text in natural scenes. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004 CVPR 2004, p. II (2004)
11. Pan, H.; Yi, C.; Tian, Y.: A primary travelling assistant system of bus detection and recognition for visually impaired people. In:

2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), pp. 1–6 (2013)
12. Jia, T.; Tang, J.; Lik, W.; Lui, D.; Li, W.H.: Plane-based detection of staircases using inverse depth. In: ICRA 2012 (2012)
13. Yang, X.; Tian, Y.: Robust door detection in unfamiliar environments by combining edge and corner features. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition—Workshops, pp. 57–64 (2010)
14. Wang, S.; Tian, Y.: Camera-based signage detection and recognition for blind persons. In: Miesenberger, K., Karshmer, A., Penaz, P., Zagler, W. (eds.) Computers Helping People with Special Needs, pp. 17–24. Springer, Berlin (2012)
15. Viola, P.; Jones, M.: Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 2001, p. I (2001)
16. Dalal, N.; Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, pp. 886–893 (2005)
17. Krizhevsky, A.; Sutskever, I.; Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Commun. ACM **60**, 84–90 (2017)
18. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
19. Ren, S.; He, K.; Girshick, R.; Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031
20. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A.: You only look once: unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788 (2016)
21. Shafiee, M.J.; Chywl, B.; Li, F.; Wong, A.: Fast YOLO: a fast you only look once system for real-time embedded object detection in video. arXiv (2017)
22. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.E.; Fu, C.-Y.; Berg, A.C.: SSD: single shot MultiBox detector. In: ECCV (2016)
23. Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S.: Feature pyramid networks for object detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 936–944 (2017)
24. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P.: Focal loss for dense object detection. IEEE Trans. Pattern Anal. Mach. Intell. **42**, 318–327 (2020)
25. Alhichri, H.; Bazi, Y.; Alajlan, N.; Bin Jdira, B.: Helping the visually impaired see via image multi-labeling based on SqueezeNet CNN. Appl. Sci. **9**, 4656 (2019)
26. Bazi, Y.; Alhichri, H.; Alajlan, N.; Melgani, F.: Scene description for visually impaired people with multi-label convolutional SVM networks. Appl. Sci. **9**(23), 5062 (2019)
27. Malek, S.; Melgani, F.; Mekhalfi, M.L.; Bazi, Y.: Real-time indoor scene description for the visually impaired using autoencoder fusion strategies with visible cameras. Sensors **17**, 2641 (2017)
28. Mekhalfi, M.L.; Melgani, F.; Bazi, Y.; Alajlan, N.: A compressive sensing approach to describe indoor scenes for blind people. IEEE Trans. Circuits Syst. Video Technol. **25**, 1246–1257 (2015)
29. Mekhalfi, M.L.; Melgani, F.; Bazi, Y.; Alajlan, N.: Toward an assisted indoor scene perception for blind people with image multilabeling strategies. Expert Syst. Appl. **42**, 2907–2918 (2015)
30. Mekhalfi, M.L.; Melgani, F.; Bazi, Y.; Alajlan, N.: Fast indoor scene description for blind people with multiresolution random projections. J. Vis. Commun. Image Represent. **44**, 95–105 (2017)
31. Azizpour, H.; Razavian, A.S.; Sullivan, J.; Maki, A.; Carlsson, S.: Factors of transferability for a generic ConvNet representation. IEEE Trans. Pattern Anal. Mach. Intell. **38**, 1790–1802 (2014)
32. Chatfield, K.; Simonyan, K.; Vedaldi, A.; Zisserman, A.: Return of the Devil in the Details: Delving Deep into Convolutional Nets. arXiv:1405.3531 (2014)
33. Gao, C.; Li, P.; Zhang, Y.; Liu, J.; Wang, L.: People counting based on head detection combining Adaboost and CNN in crowded surveillance environment. Neurocomputing **208**, 108–116 (2016)
34. Nogueira, R.F.; de Alencar Lotufo, R.; Machado, R.C.: Fingerprint liveness detection using convolutional neural networks. IEEE Trans. Inf. Forensics Secur. **11**, 1206–1213 (2016)
35. Simonyan, K.; Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR arXiv:1409.1556 (2014)
36. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9 (2015)
37. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and < 0.5 MB model size. CoRR-arXiv:160207360 [cs] (2016)
38. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C.: MobileNetV2: inverted residuals and linear bottlenecks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
39. He, K.; Zhang, X.; Ren, S.; Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
40. Akilan, T.; Wu, Q.J.; Zhang, H.: Effect of fusing features from multiple DCNN architectures in image classification. IET Image Process. **12**, 1102–1110 (2018)
41. Daneshtalab, S.; Rastiveis, H.: Decision level fusion of orthophoto and LIDAR data using confusion matrix information for LNAD cover classification. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. **42**, 59–64 (2017). https://doi.org/10.5194/isprs-archives-XLII-4-W4-59-2017
42. Du, P.; Liu, S.; Xia, J.; Zhao, Y.: Information fusion techniques for change detection from multi-temporal remote sensing images. Inf. Fusion **14**, 19–27 (2013)
43. Hedeya, M.A.; Eid, A.H.; Abdel-Kader, R.F.: A super-learner ensemble of deep networks for vehicle-type classification. IEEE Access **8**, 98266–98280 (2020)
44. Ji, Y.; Zhang, H.; Jonathan Wu, Q.M.: Salient object detection via multi-scale attention CNN. Neurocomputing **322**, 130–140 (2018)
45. Ju, C.; Bibaut, A.; van der Laan, M.: The relative performance of ensemble methods with deep convolutional neural networks for image classification. J. Appl. Stat. **45**, 2800–2818 (2018)
46. Koh, B.H.D.; Woo, W.L.: Multi-view temporal ensemble for classification of non-stationary signals. IEEE Access **7**, 32482–32491 (2019)
47. Liu, Y.; Guo, Y.; Georgiou, T.; Lew, M.S.: Fusion that matters: convolutional fusion networks for visual recognition. Multimed. Tools Appl. **77**, 29407–29434 (2018)
48. Alhichri, H.: Multitask classification of remote sensing scenes using deep neural networks. In: IGARSS 2018—2018 IEEE International Geoscience and Remote Sensing Symposium, pp. 1195–1198 (2018)
49. Krizhevsky, A.; Sutskever, I.; Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in

Neural Information Processing Systems, vol. 25, pp. 1097–1105. Curran Associates, Inc., Red Hook (2012)
50. Breiman, L.: Stacked regressions. Mach. Learn. **24**, 49–64 (1996)
51. Maas, A.L.; Hannun, A.Y.; Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the 30th International Conference on Machine Learning JMLR: W&CP, Atlanta, Georgia, USA (2013)
52. Ioffe, S.; Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning, Lille, France—July 06–11, 2015, pp. 448–456 (2015)
53. Yager, R.R.: On the dispersion measure of OWA operators. Inf. Sci. **179**, 3908–3919 (2009)
54. Yager, R.R.: On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Trans. Syst. Man Cybern. **18**, 183–190 (1988)
55. Yager, R.R.: On prioritized multiple-criteria aggregation. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **42**, 1297–1305 (2012)
56. Yager, R.R.; Alajlan, N.: A generalized framework for mean aggregation: toward the modeling of cognitive aspects. Inf. Fusion **17**, 65–73 (2014)
57. IDS Imaging Development Systems GmbH. Industrial cameras—IDS Imaging Development Systems GmbH Commercial
58. Kowa LM4NCL 1/2″ 3.5 mm F1.4 Manual Iris C-Mount Lens. RMA Electronics, Inc., Hingham
59. Ganda, D.: A survey on multi label classification. Recent Trends Program. Lang. **5**, 19–23 (2018)
60. Sorower, M.S.: A Literature Survey on Algorithms for Multi-label Learning. Oregon State University, Corvallis (2010)
61. Zhang, M.-L.; Zhou, Z.-H.: A review on multi-label learning algorithms. IEEE Trans. Knowl. Data Eng. **26**, 1819–1837 (2014)
62. Qiu, J.; Wang, J.; Yao, S.; Guo, K.; Li, B.; Zhou, E.; Yu, J.; Tang, T.; Xu, N.; Song, S.; Wang, Y.; Yang, H.: Going deeper with embedded FPGA platform for convolutional neural network. In: Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays Association for Computing Machinery, Monterey, California, USA, pp. 26–35 (2016)
63. Zhang, C.; Li, P.; Sun, G.; Guan, Y.; Xiao, B.; Cong, J.: Optimizing FPGA-based accelerator design for deep convolutional neural networks. In: Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays Association for Computing Machinery, Monterey, California, USA, pp. 161–170 (2015)
64. Suda, N.; Chandra, V.; Dasika, G.; Mohanty, A.; Ma, Y.; Vrudhula, S.; Seo, J.; Cao, Y.: Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks. In: Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays Association for Computing Machinery, Monterey, California, USA, pp. 16–25 (2016)
65. Pano-Azucena, A.D., Tlelo-Cuautle, E., Tan, S.X.-D., Ovilla-Martinez, B., De la Fraga, L.G.: FPGA-based implementation of a multilayer perceptron suitable for chaotic time series prediction. Technologies. **6**(4), 90(2018)
66. Duarte, J.; Han, S.; Harris, P.; Jindariani, S.; Kreinar, E.; Kreis, B.; Ngadiuba, J.; Pierini, M.; Rivera, R.; Tran, N.; Wu, Z.: Fast inference of deep neural networks in FPGAs for particle physics. J. Inst. **13**, P07027 (2018)