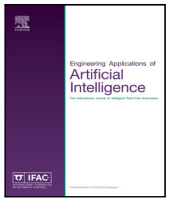
Engineering Applications of Artificial Intelligence 114 (2022) 105188

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Efficient gesture recognition for the assistance of visually impaired people using multi-head neural networks

Samer Alashhab [a], Antonio Javier Gallego [a,*], Miguel Ángel Lozano [b]

[a] Department of Software and Computing Systems, University of Alicante, Carretera San Vicente del Raspeig s/n, Alicante, 03690, Spain
[b] Department of Computer Science and AI, University of Alicante, Carretera San Vicente del Raspeig s/n, Alicante, 03690, Spain

ARTICLE INFO

Keywords:
Multi-head architectures
Hand gesture detection
Visual impairments
Deep Neural Networks

ABSTRACT

Existing research for the assistance of visually impaired people mainly focus on solving a single task (such as reading a text or detecting an obstacle), hence forcing the user to switch applications to perform other actions. This paper proposes an interactive system for mobile devices controlled by hand gestures that allow the user to control the device and use several assistance tools by making simple static and dynamic hand gestures (e.g., pointing a finger at an object will show a description of it). The system is based on a multi-head neural network, which initially detects and classifies the gestures, and subsequently, depending on the gesture detected, performs a second stage that carries out the corresponding action. This architecture optimizes the resources required to perform different tasks, it takes advantage of the information obtained from an initial backbone to perform different processes in a second stage. To train and evaluate the system, a dataset with about 40k images was manually compiled and labeled including different types of hand gestures, backgrounds (indoors and outdoors), lighting conditions, etc. This dataset contains synthetic gestures (whose objective is to pre-train the system to improve the results) and real images captured using different mobile phones. The comparison made with nearly 50 state-of-the-art methods shows competitive results as regards the different actions performed by the system, such as the accuracy of classification and localization of gestures, or the generation of descriptions for objects and scenes.

1. Introduction

Gestures are an important part of our communication. They are a form of non-verbal exchange of information that have aroused great interest as regards the design of Human–Computer Interaction (HCI) systems, as they allow users to express themselves naturally and intuitively in different contexts (Mitra and Acharya, 2007). In some scenarios, a single gesture may be more effective than words (e.g., a *pinch* gesture makes it easier to express the desired zoom level, than explaining it with words).

Hand gesture recognition methods have a significant number of applications, such as controlling unmanned aerial vehicles (UAVs) (Ma et al., 2017), interacting with autonomous vehicles (Holzbock et al., 2022), recognizing sign language (Pigou et al., 2015), or manipulating objects in virtual reality environments (Lin et al., 2017a) or in 3D design tools (Wang and Bao, 2007). In the case of applications such as object manipulation, it is necessary to track the pose of hand and fingers, whereas other applications have to classify the gesture into certain categories, which is the case of sign language recognition. Both dynamic and static gestures are used in these latter applications, depending on whether or not they change over time, respectively (Prakash and Gautam, 2019).

One of the contexts in which hand gestures play a prominent role is the field of assistive technologies for people with visual impairments, in which a good user interaction design is of vital importance (Manduchi and Coughlan, 2012). Some devices and applications in this field could greatly benefit from an agile, natural and intuitive interaction system that employs hand gestures. Examples of these devices are OrCam MyEye,[1] which reads text and identifies objects in the scene, or the eyewear object recognition device proposed by Pintado et al. (2019), which assists people with visual impairments in a market setting. There are also mobile applications such as SuperVision for Cardboard,[2] which turns a smartphone and a Google Cardboard device into low-cost electronic glasses. However, these systems are limited to a very specific action, requiring the user to press or switch the application to perform another task. A hand gesture-based interface could, therefore, play a key role in improving these technologies.

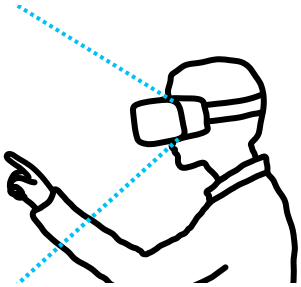
Our goal is to develop a gesture recognition method on which to build an interactive low-cost system for mobile devices controlled by

* Corresponding author.
E-mail addresses: salashhab@ua.es (S. Alashhab), jgallego@dlsi.ua.es (A.J. Gallego), malozano@ua.es (M.Á. Lozano).

[1] https://www.orcam.com.
[2] http://supervisioncardboard.com.

https://doi.org/10.1016/j.engappai.2022.105188
Received 11 May 2022; Received in revised form 24 June 2022; Accepted 4 July 2022
Available online xxxx

(a) Low cost Google Cardboard. This image was taken from [11].



(b) Scheme to illustrate the user interface of the proposed system.

Fig. 1. Google Cardboard and scheme of the proposed user interface.

hand gestures (see Fig. 1(a)), with the objective of helping people with visual impairments. In our proposal, gesture recognition is performed from an egocentric point of view (Tekin et al., 2019), as the method is intended for applications based on RGB cameras located at the point from which the user views the object (see Fig. 1(b)). In this respect, gestures could be used to interact with the application, triggering actions such as identifying the object the user is pointing to, describing the scene when the user makes a static loupe gesture, or zooming in and out when a dynamic pinch gesture is employed.

The first step towards this idea was initially developed in Alashhab et al. (2019), focusing on the classification task and considering a limited set of gestures. In this paper, a completely different and novel approach is proposed to perform multiple tasks simultaneously (including, in addition to classification, actions such as localization and captioning, among others). Also, both the dataset and the experimentation carried out are considerably extended.

In summary, the main contributions made in this work are:

- A novel multi-task architecture that results in a much more efficient and effective model than the use of different separate networks. The proposal is based on a multi-head neural network that integrates the recognition of dynamic and static gestures, object localization and image description functions in the same architecture. Each head of this network is dedicated to a function associated with a given gesture and is executed only if that gesture is detected by the backbone.
- An exhaustive experimentation of all the parts and actions of the proposed architecture. This experimentation shows that the proposal, in addition to obtaining good results for the different actions carried out, also runs in almost real time on mobile devices.
- Our contributions also include the development of a dataset with both real and synthetic images that are annotated at different levels, including gesture category, gesture and fingertip bounding boxes, and object and scene descriptions.

The rest of the paper is organized as follows: Section 2 shows a review of the state of the art regarding hand gesture, object and image recognition, while the proposed application interface is described in Section 3, the datasets used to train and evaluate our model are detailed in Section 4, and the proposed approach is introduced in Section 5. A comprehensive set of experiments is then shown in Section 6. Finally, our conclusions and future work are addressed in Section 8.

2. Related work

Hand gesture recognition approaches include both methods based on dedicated hardware (or other props) and computer vision-based methods (Sonkusare et al., 2015). The first group contains solutions based on gloves equipped with sensors (Mazumdar et al., 2013) and on gloves marked with colors (Wang and Popović, 2009; Lamberti and Camastra, 2011). In this second approach, each color is used to identify different parts of the hand, which can be detected and tracked by a camera, and no additional hardware is required. In addition to gloves, there are other kinds of sensors whose purpose is also to recognize hand gestures, such as wearable devices that monitor muscle activity on the basis of surface electromyography (Moin et al., 2021) or ultrasonic Doppler sensors (Raj et al., 2012). These solutions based on hardware and sensors have the main advantage of being very precise. However, they have the main limitation of requiring additional equipment for their use, which can be expensive, cumbersome and uncomfortable for the user.

On the other hand, the second group of vision-based hand gesture recognition approaches only needs a camera and a system to process the image, for which very compact and cheap devices can currently be found. This second group contains two main categories: appearance-based and 3D model-based methods. The first contains several techniques based on segmenting the hand by color (Pun et al., 2011) (i.e., from RGB images). In many of these approaches, the average radius of the hand is calculated, and the blobs outside that radius are considered to be spread fingers (Perimal et al., 2018). The methods in this group have the advantage that they do not need a database of gestures for training (Rajesh et al., 2012), but this comes with the limitation that they can recognize only gestures that consist of folded or spread fingers. In Prakash et al. (2017), the position of the spread fingertips is detected from the vertices of the convex hull of the hand. Other methods rely on depth to segment the hand (Kim and Lee, 2016) (i.e., from RGB-D images). There are also similar solutions that rely on the curve of the hand to identify spread fingers (Ren et al., 2011) and their fingertips (Lai et al., 2016).

The group of appearance-based approaches also includes other solutions that use RGB-D images. In Dinh et al. (2014), depth is used to first segment the hand silhouette and remove the background, after which a trained Random Forest (RF) classifier is applied in order to recognize the different parts of the hand in the RGB-D image. Another method (Bamwenda and Özerdem, 2019) for static gesture recognition obtains a depth-based histogram of oriented gradient features and applies Artificial Neural Networks (ANN) and Support Vector Machines (SVM) for classification. In Molina et al. (2017), motion patterns are recognized from sequences of RGB-D images so as to identify dynamic gestures. In order to improve both efficiency and performance when the input is a sequence of images, in Tang et al. (2019) key frames are extracted to reduce the number of samples that must be processed. Features are obtained from appearance and motion between consecutive key frames, and a Bag of Features (BoF) approach is applied to classify hand gestures.

With regard to the general recognition of gestures from RGB and RGB-D images, the methods that have been shown to be most effective are those based on Deep Neural Networks (DNN). Previous solutions were usually based on the extraction of specific descriptors – commonly called handcrafted features – which led to non-generic models that

failed with small changes in input conditions, such as variations in lighting, colors, etc. Oppositely, DNN-based solutions select by themselves the most suitable features for the task at hand – feature learning –, hence showing superior generalization capabilities (especially when coupled with other regularization methods). Most of these approaches use Convolutional Neural Networks (CNN), which have obtained excellent results for image recognition (Schmidhuber, 2015). Architectures used for this purpose are those such as AlexNet (Krizhevsky et al., 2012), GoogleNet (Szegedy et al., 2015), DenseNet (Huang et al., 2017), ResNet (He et al., 2016), Xception (Chollet, 2016), and lightweight architectures such as MobileNet (Howard et al., 2017), SqueezeNet (Iandola et al., 2016) and EfficientNet (Tan and Le, 2019), on which several works obtaining excellent results have been based (Alashhab et al., 2019). With regard to gesture classification, there are also CNN-based approaches such as (Lin et al., 2014a), which embodies a previous image calibration step. CNNs have also been applied in order to recognize sign language in a single frame (Bheda and Radpour, 2017) or a sequence of frames (Pigou et al., 2015) (dynamic gestures). In Molchanov et al. (2015), the CNN takes both intensity and depth video sequences as input for the recognition of dynamic gestures with the objective of designing touchless interfaces in cars. In addition to CNNs, other deep learning-based approaches are also used to segment hands by depth, as is the case of SegNet, a deep convolutional encoder–decoder architecture proposed to detect fingertips (Nguyen et al., 2019). Dynamic gesture recognition in video sequences has also been addressed by means of Bidirectional Long Short-Term Memory (BiLSTM) (Xie et al., 2017) and Temporal Segment Networks (TSN) (Benitez-Garcia et al., 2021) in order to capture temporal information.

Most approaches based on 3D models use the skeleton of the hand. However, there are also volumetric solutions (Ge et al., 2019). The skeleton can be obtained using specific hardware, such as instrumented gloves, although there are also devices based on depth cameras that provide a built-in joint tracker, such as Microsoft Kinect (Xi et al., 2018) or Leap Motion Controller,[3] which has been applied to, for example, hand function rehabilitation (Xiao et al., 2021b). There are also libraries, such as OpenPose hand detection[4] (Cao et al., 2019; Simon et al., 2017) or MediaPipe Hands[5] (Zhang et al., 2020a), that make it possible to obtain the skeleton in 2D or 3D from an RGB image. There are also some methods that recover the 3D skeleton of a hand from RGB-D images (Ge et al., 2019) or from a sequence of raw RGB images (Tekin et al., 2019). This 3D data information of the hand skeleton (joint coordinates) has been used as a basis for different types of classifiers with which to recognize gestures, such as Hidden Markov Models (HMM) and Dynamic Time Warping (DTW) in Raheja et al. (2015) or CNNs in Devineau et al. (2018). A comparison (De Smedt et al., 2017) between an SVM classifier using skeleton data and a CNN classifier using RGB-D images shows that the former approach provides superior results as regards hand gesture recognition. However, the use of libraries such as OpenPose or MediaPipe to obtain skeleton data from RGB images has certain limitations, since it is not possible to recognize accurately skeleton points for several poses (Amaliya et al., 2021).

As a summary, Table 1 shows a comparison of all the related works reviewed detailing whether the method is based on sensors, hardware or vision, the type of data used (2D or 3D), the approach followed in the solution, if it proposes an interface based on gestures and the actions it supports. It also indicates whether the method is intended to be integrated into mobile devices and/or to help people with visual difficulties. In this classification, the solution proposed in this work covers gaps that other solutions do not fill, since there are no solutions aimed at helping people with visual impairment that allow them to use multiple assistance tools in the same application in an intuitive and natural way.

[3] https://www.ultraleap.com/product/leap-motion-controller/.

[4] https://github.com/CMU-Perceptual-Computing-Lab/openpose.

[5] https://mediapipe.dev.

Our work focuses on the development of a low-cost general gesture recognition solution that could be integrated into most of the current smartphones equipped with RGB cameras. For this, a vision-based system is proposed, as this will avoid the use of additional specialized equipment. We rely on DNNs that take an RGB image as input, since, as previously indicated, this type of approximation is the one that currently obtains the best results and, furthermore, this type of sensor is available in the vast majority of mobile devices. A possible disadvantage of this approach is that, unlike hardware-based solutions, it can be affected by changes to the image, such as lighting or perspective. To solve the latter, in the proposed interface the images will always be taken from an egocentric point of view. To improve the robustness and generability of the generated model, in addition to applying regularization mechanisms, we will use a highly varied training image dataset (described in Section 4).

Besides gesture recognition, our purpose is to integrate the functions required to identify objects, describe the scene or zoom in and out into the same network. In order to do this efficiently, we have followed an approach similar to that of Köpüklü et al. (2019), in which a two-stage network architecture is implemented to build a real-time gesture recognizer: the first stage is dedicated to the detection of the gesture, and the second one to the classification, which is executed only if the presence of a gesture is detected in the image.

In our case, we propose a multi-head architecture with a backbone dedicated to the classification of gestures, and a set of specialized heads for each gesture, which will be triggered only if the corresponding gesture is detected. Multi-head architectures have been successfully used in different types of applications, such as human activity recognition using the information provided by different sensors (Zhang et al., 2020c), or for time series classification, with applications in cybersecurity, health care, remote sensing or also for human activity recognition (Xiao et al., 2021a). In this proposal, the multi-head architecture is used to perform multiple actions from a common input image. This differentiates us from the rest of the state-of-the-art proposals for assisting people with visual impairments, which focus on solving a single task (see Table 1), hence forcing the user to switch applications to perform other tasks. Therefore, the development of an interface that provides different assistance tools managed through an intuitive and natural interface is of great relevance.

Depending on the gesture detected, a given action is, therefore, performed using a specialized head: object recognition, image description and zoom in/out. We considered various state-of-the-art DNN architectures for the object recognition head, such as You Only Look Once (YOLO) (Redmon and Farhadi, 2018), Faster R-CNN (FRCNN) (Ren et al., 2015), and RetinaNet (Lin et al., 2017b). These models are able to identify multiple objects in the image and their bounding boxes. We also compared a modified version of the Filter Selection (FS) (Alashhab et al., 2019) approach as an alternative to these object recognition methods, in which a set of filters from the backbone is selected in order to calculate the location of the gesture in the image. This method has the advantage of not adding extra modules to the original architecture. With regard to the head employed to obtain the description of the image, it can be addressed using image captioning methods (Hossain et al., 2018). For this, we have also considered different models (You et al., 2016; Tanti et al., 2018), which usually combine a CNN in order to extract features from the image, and a Recurrent Neural Network (RNN) to generate the description. Finally, with regard to zooming in and out with the pinch gesture, we propose our own architecture as a specialized head for this task.

3. Application interface

One of the areas in which improvements could be made to the applications aimed at helping people with visual impairments, such as *Supervision for Cardboard* (Supervision, 2021), is the interface. An important enhancement to its usability would be that users could use

Table 1
Comparative summary of related works detailing whether the method is based on sensors, hardware or vision, the type of data used (2D or 3D), the approach followed in the solution, whether it proposes an interface based on gestures and the actions it supports. It is also indicated if the method is intended to be integrated into mobile devices and/or to help people with visual impairments.

						Actions						
	Hardware or sensor-based	Vision-based	Data type (2D/3D)	Approach	Gesture-based	Classify	Detect	Object description	Captioning	Dynamic gestures	Assist people with visual impairments	Mobile
Proposed solution		✓	2D	CNN	✓	✓	✓	✓	✓	✓	✓	✓
Previous solution (Alashhab et al., 2019)		✓	2D	CNN	✓	✓	✓				✓	✓
Lin et al. (2014a), Bheda and Radpour (2017), Xiao et al. (2021a) and Benitez-Garcia et al. (2021)		✓	2D	CNN	✓	✓						
Xie et al. (2017)		✓	2D	CNN	✓	✓						✓
Nguyen et al. (2019)		✓	2D	CNN	✓		✓					
Ma et al. (2017) and Zhang et al. (2020c)	✓	✓	2D	CNN		✓						
OrCam MyEye	✓	✓	2D	CNN				✓	✓		✓	
Pintado et al. (2019)	✓	✓	2D	CNN				✓			✓	
You et al. (2016) and Tanti et al. (2018)		✓	2D	CNN					✓			
Mazumdar et al. (2013), Lamberti and Camastra (2011) and Raj et al. (2012)	✓		2D	Hardware	✓	✓						
SuperVision for cardboard		–	2D	Software							✓	✓
Pun et al. (2011) and Prakash et al. (2017)		✓	2D	Appearance	✓	✓	✓					
Perimal et al. (2018), Rajesh et al. (2012) and Wang and Bao (2007)		✓	2D	Appearance	✓	✓						
Cao et al. (2019)		✓	2D	3D model		✓						
Xiao et al. (2021b) and Raheja et al. (2015)	✓		3D	3D model	✓	✓						
Devineau et al. (2018)		✓	3D	3D model	✓	✓						
Amaliya et al. (2021)	✓	✓	3D	3D model	✓		✓					
Zhang et al. (2020a), Tekin et al. (2019), Simon et al. (2017) and Ge et al. (2019)		✓	3D	3D model	✓		✓					
Xi et al. (2018)	✓		3D	3D model			✓					
Bamwenda and Özerdem (2019)	✓	✓	3D	Appearance	✓	✓						
Molina et al. (2017)	✓	✓	3D	Appearance	✓	✓				✓		
Kim and Lee (2016) and Dinh et al. (2014)	✓		3D	Appearance	✓	✓						
Ren et al. (2011) and Lai et al. (2016)	✓		3D	Appearance	✓	✓	✓					
Tang et al. (2019)		✓	3D	Appearance	✓					✓		
Pigou et al. (2015)	✓	✓	3D	CNN	✓	✓						
Holzbock et al. (2022)		✓	3D	CNN	✓	✓				✓		
Molchanov et al. (2015)		✓	3D	CNN	✓	✓						
Köpüklü et al. (2019)		✓	3D	CNN	✓	✓	✓					
Moin et al. (2021)	✓	✓	3D	Hardware	✓	✓						
Wang and Popović (2009) and Lin et al. (2017a)	✓		3D	Hardware	✓		✓					

the application while moving or performing other tasks, without having to touch the mobile screen.

To this end, this paper proposes an interactive system for mobile devices controlled by hand gestures. Users could install their mobile phones on Virtual Reality Glasses (VRG) or on a low cost Google Cardboard (see Fig. 1(a)), and view the environment directly through the mobile screen. The proposed system would allow them to interact with the device using different hand gestures and would use augmented reality to display the result of the actions on the screen (see Fig. 1(b)).

A set of four simple gestures is proposed as a user interface to interact with the system: point, drag, loupe and pinch (see Figs. 2(a)–2(d)). There are three static gestures (point, drag and loupe) with which to execute specific actions, and a dynamic gesture (pinch) with which to zoom-in and zoom-out the image. A better description of the four gestures proposed is provided below:

- *Point*: Static gesture formed by extending the index finger and flexing the remaining fingers into the palm. This gesture allows users to point to the objects of which they wish to obtain a description.
- *Drag*: Static gesture formed by pointing with both the index and the middle fingers. This gesture allows users to freeze the image while simultaneously performing a panning movement of the scene following their fingertips. This is useful in combination with the zoom gesture.
- *Loupe*: Static gesture formed by joining the thumb and the index finger to form the shape of a circle, and leaving the remaining fingers extended. This gesture shows more information about the scene and the objects that appear in it.
- *Pinch*: Dynamic gesture formed by moving the thumb and the index finger towards each other or away from each other, in order to perform a zoom-in or a zoom-out operation, respectively. It is equivalent to the pinch gesture used on touch screens. This dynamic gesture allows the zoom level to be controlled with the movement of the fingers.

This small set of gestures has been selected in order to allow an intuitive and easy interaction with the system. However, our proposal is designed in a generic manner (as will be shown in Section 5), signifying that it would be easy to add new gestures so as to expand its functionality, if necessary.

It is important to note that, in addition to the proposed gestures, the system has to identify whether or not there is a gesture present in the image, and it has to differentiate these gestures from any other possible gestures, both static and dynamic (such as the thumb-up and wave gestures shown in Fig. 2).

4. Datasets

Three datasets were created in order to train and evaluate the proposed model[6]: a dataset containing real images of hand gestures, a synthetic dataset used to pre-train the system, and a dataset containing descriptions of scenes used by the captioning actions.

The samples of the dataset with real images were extracted from videos obtained with mobile phones. To ensure a varied corpus, different phone cameras were used to record indoor and outdoor scenes, with

[6] These datasets are freely available for the scientific community on demand at https://www.dlsi.ua.es/~jgallego/datasets/gestures

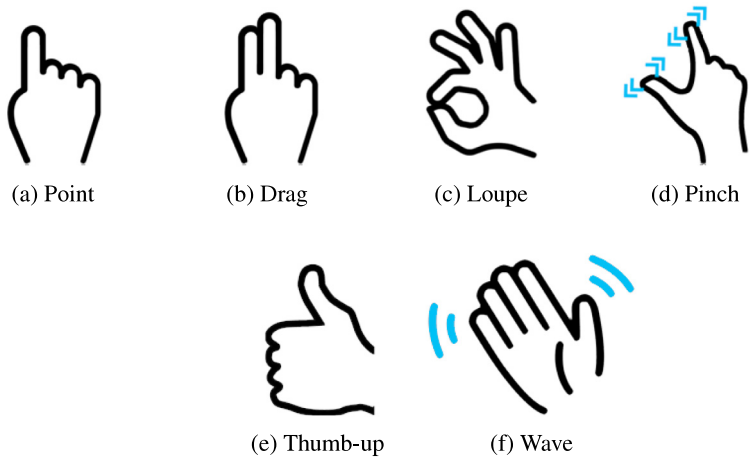


Fig. 2. Graphical description of the proposed hand gestures. The gestures used to interact with the interface are Point, Drag, Loupe and Pinch. The system will consider Thumb-up and Wave as *other gestures*. Point, Drag, Loupe, and Thumb-up are static gestures, while Pinch and Wave are dynamic gestures.

varied backgrounds and under different lighting conditions. The videos were generated by different people, thus including images of different hands (both left and right), with different finger lengths, hand sizes, and skin colors. In addition to images of the four proposed gestures, samples of backgrounds without gestures and of gestures other than those proposed were also extracted from these videos. The background images were used to train the system to discriminate between the presence or absence of hands, and the "*other-gestures*" images were employed to assess whether the system was capable of differentiating them from the proposed gestures. Note that in the case of the dynamic pinch gesture, an average of 10 consecutive frames was extracted for each gesture in order to train and evaluate the methods used to detect the movement of the fingers and their position. A total of 13,559 frames were extracted from the original videos, trying to balance the number of samples selected for each class (the number of samples per class can be found in the "real samples" column of Table 2). Fig. 3 shows some examples of the images included in this dataset.

In order to improve the results obtained and help the training process, a synthetic dataset was created using a modified version[7] of the LibHand tool (Šarić, 2011), an open-source library for the rendering of human hand poses. We modified this library so as to enable the definition of gestures through a set of rules with the ranges of movement allowed for the finger joints, thus enabling variations of these gestures to be generated randomly within these ranges. This tool was used in order to automatically generate and label a dataset with a total of 13,200 images (2200 of each gesture), with random variations in position, in the shape of the gestures, in the color of the skin, including random blur to emulate the motion effect, and with different backgrounds (using random images from Flickr8k Hodosh et al., 2013 and Visual Object Classes (VOC) Everingham et al., 2015 datasets). Fig. 4 shows some examples of the gestures generated.

These two datasets were labeled, indicating both the category and the position of the gesture. For the position, the coordinates were annotated using bounding boxes for (1) the position of the hand within the image, (2) the position of the fingertips, and (3) the coordinates and labels of the objects pointed to. The synthetic dataset labeling was automatically generated by our modification of the LibHand tool. However, the real image dataset had to be manually labeled. To facilitate this process, short videos were recorded with the same gesture displaced on a background. In this way, it was possible to directly assign the same category to all the extracted frames and, in addition, it also facilitated the labeling of the position by only having to displace the coordinates.

As explained in the previous section, the loupe gesture triggers the action of displaying a description of the scene that appears in the image. In order to train and evaluate systems capable of generating these descriptions, it was necessary to use an additional dataset of images with the corresponding associated descriptions. For this, we used the Flickr8k dataset (Hodosh et al., 2013), which contains 8000 images manually selected from the Flickr website with five descriptions of each image. Besides, we added a subset of 4000 images from our dataset of real images, which also included 5 descriptions per image (this allowed the proposed system to adapt to our type of data, i.e., images of gestures taken with mobile phones). This subset includes both loupe and point gesture images (2000 for each). The point gesture was included in order to increase the variability, and also make it possible to use the captioning head for these gestures (which could be appropriate for some applications).

The original resolution used for videos and images was 1920×1080 pixels. However, after conducting a series of initial performance and accuracy experiments at different resolutions, and also motivated by the restrictions of some of the methods evaluated, we decided to scale the images to a spacial resolution of 224×224 pixels. Table 2 shows a summary of the three datasets considered in this work, including the number of samples per class in each dataset.

5. Method

The input received by the system is a sequence of frames captured with a mobile phone camera. The proposed approach processes each of these frames in order to first classify the gesture that appears in the image and then perform an action based on the gesture detected. This is done using an architecture divided into two stages (see Fig. 5): (1) an initial backbone processes the image in order to extract a set of representative features, and (2) these features are then used to classify the gesture and perform an action by means of the head specialized in the gesture detected.

In the second step, the common features extracted by the backbone are first processed using the "Classify" head shown in Fig. 5, which yields an L-dimensional one-hot vector, where L is the number of possible gestures. The other specialized heads are activated or deactivated through the use of a switch-type layer that queries the value set to one in this one-hot vector.

The main advantage of this architecture is that it can carry out multiple processes with a reduced number of parameters and, therefore, with fewer hardware requirements. To achieve this, the same initial features are used for all the actions to be carried out. Moreover, in the second stage of the method, only one of the specialized heads is activated depending on the gesture detected, which also improves performance.

[7] https://github.com/malozano/libhand.

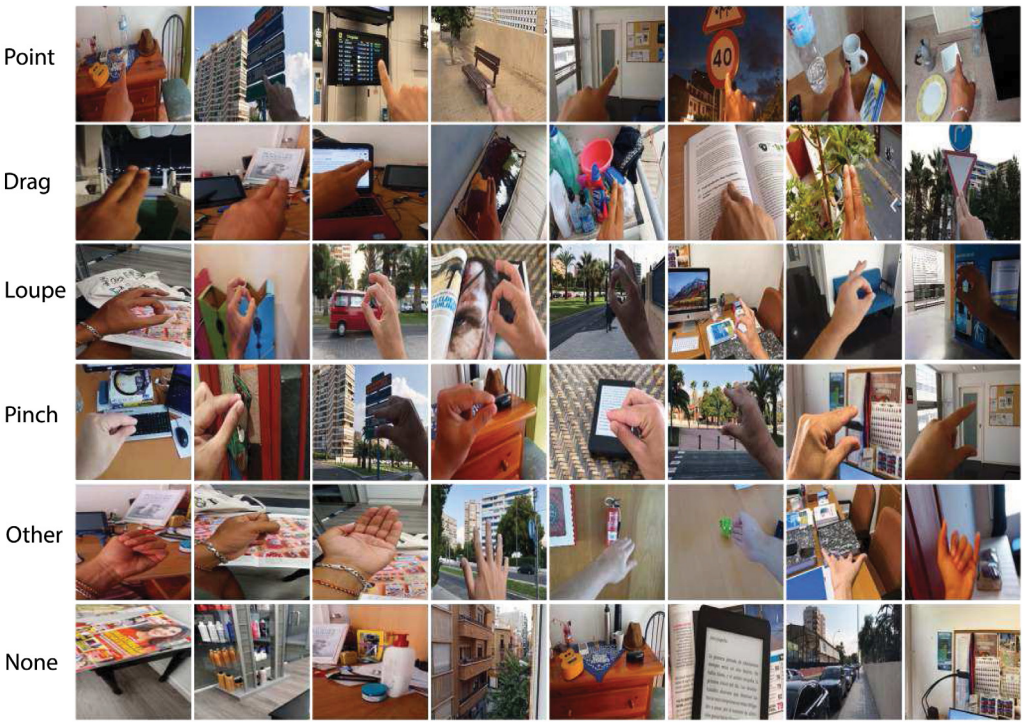


Fig. 3. Some samples of the dataset with real images. The first four rows show the different gestures proposed in order to interact with the interface. The last two rows include some examples of the "*other gestures*" and the "*no gestures*" classes.

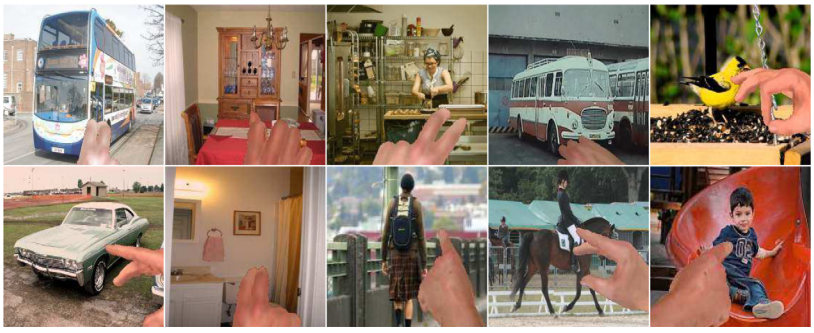


Fig. 4. Some examples of the images generated for the synthetic dataset.

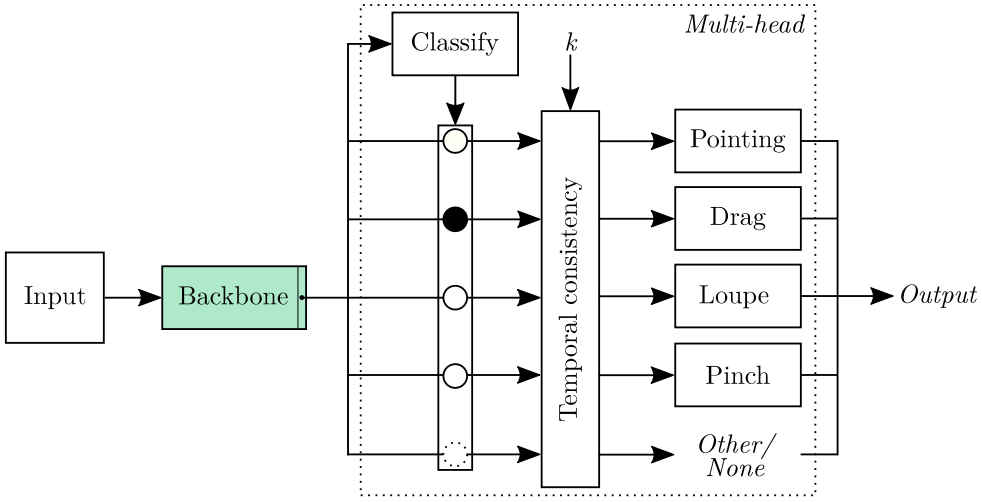


Fig. 5. Scheme of the proposed two-step multi-head network. In this architecture, an initial backbone processes the input image to extract a set of representative features, which are then used to classify the gesture and to perform an action by means of the head specialized in the gesture detected. In addition, a temporal consistency module is added to improve the results and the stability of the system.

Table 2
Summary of the datasets, including class name, number of samples per class, and a short description of the content of each class.

Gesture	# Synthetic samples	# Real samples	# Captioning samples	Description
Point	2200	2088	2000	Images of pointing gestures.
Drag	2200	2143	–	Images including drag gestures.
Loupe	2200	2147	2000	Samples including loupe gestures.
Pinch	2200	2066	–	Sequences of dynamic pinch gestures.
Other	2200	2121	–	Images of gestures other than the four defined.
None	2200	2994	8000	Samples in which no hand appears.
Total	13,200	13,559	12,000	Grand total: 38,759

The stability of the system has been improved by establishing a time margin of k frames with which to validate the detection of a gesture. That is, the system processes the frames of the input video and returns a response for each one. However, in order to ensure a consistent response for consecutive frames, at least k frames must maintain the same response to consider the prediction valid. In the tests carried out, it was sufficient to establish this time margin as 3 frames, as will be shown in the experimentation section.

As Fig. 5 shows, this temporal consistency applies to all gestures, including the "Other" and "None" categories, which are joined and considered as a single negative class and do not perform any specialized processing, but simply do not return a response. In this way, when the (positive) gestures considered are no longer detected during k frames, no response will, therefore, be provided.

Algorithm 1 shows the formalization of the proposed system. It receives as input the video stream, as well as the backbone and heads of the system. It first creates a queue Q^k with k elements to store the system response buffer (used for time consistency). For each frame read from the input video, the method processes it through the backbone and extracts the common descriptor f (line 5). The classification head is then used to determine the gesture type c (line 6) and enqueues this response in the buffer Q^k (line 7). If the last k gestures match (line 8, time consistency criterion), it processes the descriptor f using the corresponding head and returns the new response p obtained. If the last k gestures do not match, the same response p obtained previously is returned.

As can be seen, the method does not perform any loop to process a given frame, only checks and forward steps through the different parts of the network, which only involves matrix operations. Therefore, the proposed system has a linear computational complexity that only depends on the constant number of parameters of the architecture.

The following subsections provide detailed descriptions of the different parts of this approach.

Algorithm 1: Algorithmic formalization of the proposed method.

```
   Input : Video, Backbone, H^{classify,FS,loupe,pinch}
   Output: p
1  Q^k ← Queue^k(∅)
2  p ← ∅
3  while I ← read_frame(Video) do
4      f ← Backbone(I)                          ▷ Section 5.1
5      c ← H^{classify}(f)                      ▷ Section 5.2
6      Q^k ← Enqueue^k(c)
7      if Q^k = ∪^k{c} then
8          switch c do
9              case 'point' or 'drag' do
10                 p ← H^{FS}(f)                ▷ Section 5.3
11             end case
12             case 'loupe' do
13                 p ← H^{loupe}(f)             ▷ Section 5.4
14             end case
15             case 'pinch' do
16                 p ← H^{pinch}(f)             ▷ Section 5.5
17             end case
18             case 'other' or 'none' do
19                 p ← ∅
20             end case
21         end switch
22     end if
23 end while
```

5.1. Backbone

The first step in the method processes each input frame using a backbone to obtain a common feature vector that is then used to carry out the remaining actions. This part of the method is the most important, since the efficiency and efficacy of the proposed solution depends on its result. It was for this reason that a total of 18 different approaches were compared, including network architectures such as MobileNet (Howard et al., 2017), EfficientNet (Tan and Le, 2019), Xception (Chollet, 2016), and SqueezeNet (Iandola et al., 2016). The results obtained will be analyzed in detail in Section 6.2. However, we anticipate that the approach that obtained the best results (when considering the balance between precision and execution time) was Darknet-53 (Redmon and Farhadi, 2018).

Darknet-53 is the backbone used by YOLO v3. It is made up of 53 convolutional layers combined with Batch Normalization layers for regularization, Leaky ReLU activation functions, and residuals or shortcut connections (the complete architecture can be found in Redmon and Farhadi (2018), see Table 1). This network is more efficient and obtains better results than its previous versions or other similar architectures. The last convolutional layer of this backbone – which has 1024 filters of size 3×3 – is connected with a Global Average Pooling (GAP) operation that eventually links to the other layers of the module or the head in question (see Fig. 5). Therefore, this layer returns a 1024-dimensional vector, which represents the common descriptor used by all the specialized heads described below.

5.2. Classification head

With regard to the classification head, only a single dense layer is added to the backbone. This layer comprises L neurons with the SoftMax activation function to classify the L classes in our dataset. This represents adding only $1024L + L$ parameters to the architecture, 6150 for our case with $L = 6$ classes.

As indicated previously, an L-dimensional one-hot vector is obtained as a result of this classification, in which the gesture detected is marked as 1 and the others as 0. This result is used to activate only the head corresponding to the gesture detected.

Also note that the proposed methodology has the additional advantage of allowing new gestures to be easily added. This can be done by adding the new category to the classification head and then fine-tuning the backbone in order to detect the new gesture. The training process will be explained in detail in Section 5.6.

5.3. Pointing and drag gestures

The actions corresponding to these two gestures share a common first part: the detection of the tip of the extended fingers. Up to 8

possible approaches were compared for this process, including object detection methods such as YOLO (Redmon and Farhadi, 2018), RetinaNet (Lin et al., 2017b), and Faster R-CNN (Ren et al., 2015) (as will be seen in Section 6.3). It was eventually determined that the approach that obtained the best results as regards both precision and efficiency was an approximation based on the Filter Selection (FS) method (Alashhab et al., 2019).

5.3.1. Filter selection

FS is a weakly-supervised object detection method that selects the set of filters from a categorical CNN that maximizes the detection precision of the classes considered. That is, this method does not add extra layers, but directly takes advantage of the filters already learned for the classification task and reuses them to obtain the location of the objects for each class. It is, therefore, also more efficient, as it does not require extra parameters. This solution, which was initially proposed for other types of tasks, has been modified to support multiple classes and to speed up the detection process.

FS analyzes all the filters learned by the categorical network (denoted as $\mathcal{F}$) in order to then select a subset $\mathcal{F}^c \subseteq \mathcal{F}$, which will be used to determine the location of the class c. This is done by calculating the Intersection over Union (IoU) between the ground-truth and the predictions $P_f^{(i)}$ for the filter f and the image i of the set of images I^c with samples of the searched class c. Only those filters whose average IoU is greater than a threshold α are selected from this result. The subset of filters $\mathcal{F}^c$ is formally calculated as follows:

$$\mathcal{F}^c = \left\{ f \in \mathcal{F} \mid \frac{1}{|I^c|} \sum_{i=1}^{|I^c|} IoU(P_f^{(i)}, B_c^{(i)}) > \alpha \right\} \tag{1}$$

where $B_c^{(i)}$ are the ground-truth localizations for the image i and class c, and $|I^c|$ represents the cardinality of the set I^c. The prediction set $P_f^{(i)}$ for an input image i and a filter f, is computed as follows:

$$P_f^{(i)} = Blobs((r(A_f^{(i)}) > \beta) \oplus s) \tag{2}$$

where $A_f^{(i)}$ represents the activation map (also known as the feature map) obtained for the filter f and the input image i. Unlike that which occurs in the original method, our approach does not perform a backpropagation pass, but rather uses the activation maps directly, thus significantly speeding up the entire process. The activations obtained are then rescaled to range $[0, 1]$ using the function r and are thresholded using β to obtain a binary matrix $\mathbb{R}^{(w \times h)} \rightarrow [0, 1]^{(w \times h)}$ that is the same size as the input image, where w and h are the width and height, respectively. A morphological dilation operation (denoted by $\oplus$) is applied using a structuring element s. Since the noise is removed by the thresholding operation, the objective of this dilation is to close small gaps and slightly increase the size of the detections. Finally, the function *Blobs* calculates the groups of connected pixels, returning a list of bounding boxes for the blobs detected.

In Eq. (1), in order to calculate the IoU of the predictions obtained for an input image i, each predicted bounding box from the set $P_f^{(i)}$ is mapped onto the ground truth bounding box $B_c^{(i)}$ with which it had a maximum IoU overlap (considering that both P_f and B_c may contain many bounding boxes):

$$IoU(P_f^{(i)}, B_c^{(i)}) = \frac{\text{area}(P_f^{(i)} \cap B_c^{(i)})}{\text{area}(P_f^{(i)} \cup B_c^{(i)})} \tag{3}$$

Once this stage has been completed, the subset of filters $\mathcal{F}_c$ for each target class c is stored to be used in the inference stage for unseen images. In our case, this selection process is carried out on the categorical network described in the previous section (i.e., Darknet-53 + classification head), initialized with the pre-trained weights obtained with the ILSVRC dataset (Krizhevsky et al., 2012), a generic purpose database for object classification, and fine-tuned in order to classify the classes of our dataset (this training process will be explained in Section 5.6). The influence of the different parameters of the proposed method on this and on the other network architectures considered will be evaluated in Section 6.3.

5.3.2. Fingertip detection

In the inference stage, an input sample is forwarded through the trained model (the backbone in Fig. 5), and if a pointing or drag gesture is detected, the activation maps of the network are used to obtain its localization. This is done in a similar way to Eq. (2), but by performing the average of the activations obtained from the selected subset of filters $\mathcal{F}^c$. The function $FS(i, c)$ calculates the localization of targets using the pre-calculated subset of filters $\mathcal{F}^c$, as follows:

$$FS(i, c) = Blobs\left(\left(\left(\frac{1}{|\mathcal{F}^c|} \sum_{f \in \mathcal{F}^c} r(A_f^{(i)})\right) > \beta\right) \oplus s\right) \tag{4}$$

With regard to the drag gesture, the system needs to know only the position of the tip of the spread fingers, since, for the action to be carried out, this information is sufficient to calculate the movement made by the fingers between consecutive frames. However, in the case of the pointing gesture, it is also necessary to identify and describe the closest object to the fingertip. Depending on the final application, this could be done by means of the captioning generation method detailed in the following section or by applying FS to the categories of the objects to be identified by our system. This would, therefore, allow the system to indicate the class of the object pointed to by consulting the annotation of the ILSVRC dataset (Krizhevsky et al., 2012).

5.4. Loupe gesture

The objective of the loupe gesture is to obtain a textual description of a scene. A specialized head based on the caption generation model proposed by Tanti et al. (2018), which is known as *merge-model*, was used for this action. This multimodal architecture performs a late fusion of information. As the authors of the original model indicate, results suggest that the visual and linguistic modalities for caption generation need not be jointly encoded by the RNN, as this yields large memory-intensive models with few tangible advantages in performance; the multimodal integration should rather be delayed to a subsequent stage. Late fusion, therefore, makes it possible to use specialized architectures for each modality, such as, in our case, a backbone with which to process the images and an RNN for the text.

In our implementation (see Fig. 6), the features obtained by the backbone for the input image are processed through the use of a fully connected (FC) layer with 256 neurons. The output of this layer is then combined with that obtained from the recurrent part of the network used for the linguistic information. This other part of the network is made up of an embedding layer followed by an LSTM layer with 256 neurons. Once the combined features have been obtained, two FC layers, each of which contains 256 neurons, are used to obtain the final result. The ReLU activation function is used in all the layers, with the exception of the last layer, which uses a Softmax activation function to determine the next word predicted by the architecture. During inference, the start token "*startseq*" is passed, generating one word, after which the model is recursively called, using the words generated as input, until the end token "*endseq*" is obtained or the maximum description length is reached.

Finally, it should also be noted that a post-processing step is eventually carried out on the sentences generated, since the network sometimes generates texts that begin with "A hand/finger is pointing to..." or "A hand and...". A set of basic rules have, therefore, been defined that modify the phrase in order to remove these texts and thus generate a sentence that refers only to the scene.

5.5. Pinch gesture

The purpose of this gesture is to control the zoom level. When it is detected, the image freezes and the user can increase and decrease the zoom level with the movement of the fingers. The specialized head shown in Fig. 7 is used to control this action. This head receives two inputs, one containing the features extracted by the backbone for the

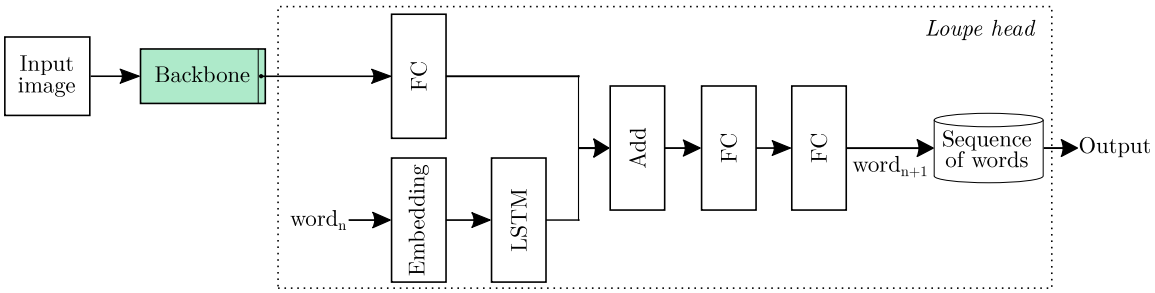


Fig. 6. Image captioning model implemented by the head that processes the loupe gesture. In experimentation, this part of the architecture will be denoted as "Darknet-53 + captioning head".

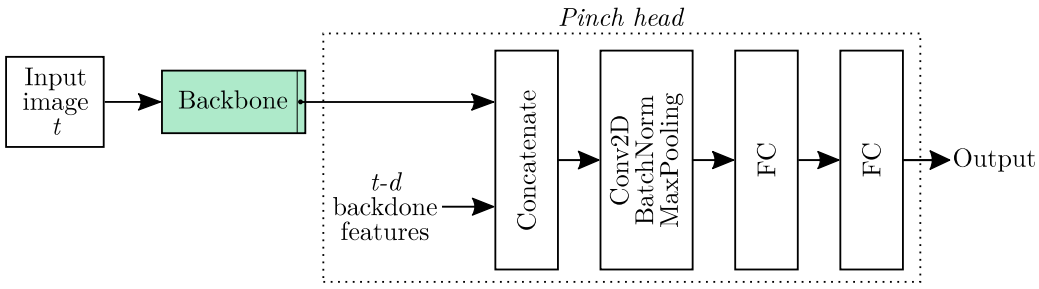


Fig. 7. Scheme of the architecture proposed for the head that processes the pinch gesture. In experimentation, this part of the architecture will be denoted as "Darknet-53 + pinch head".

current frame t, and other containing the features obtained for the frame $t - d$. In other words, the frame obtained d previous frames is used rather than the immediately previous one. In our case, after a series of preliminary experiments, we established $d = 5$ in order to have a notable difference between the frames compared that facilitates the detection of movement. The history of stored frames is managed internally by the specialized head itself, for which it is sufficient to store a buffer with the last d elements.

The architecture of this head is very simple (see Fig. 7), which results in an efficient and effective model. The features of the two inputs (current frame and frame $t - d$) are concatenated and processed by a convolution layer with 64 filters of size 3×3. A Batch Normalization layer is then added, followed by a Max Pooling operation, which is used to reduce dimensionality. The result obtained is connected with an FC layer (containing 32 neurons and a ReLU activation function) through the use of a flatten operation and, finally, with another FC layer with Softmax activation in order to determine whether zoom-in, zoom-out, or no zoom is being performed.

5.6. Training stage

Rather than training the entire architecture in one stage using a combined loss function, a two-phase training process is proposed. In the first phase, the backbone is trained for the classification task, while in the second phase, the weights obtained for the backbone and the classification head are frozen and the remaining heads are trained.

The backbone is initialized using the pre-trained weights obtained with the ILSVRC dataset (Krizhevsky et al., 2012). Weight initialization is a common practice that makes it possible to obtain better results in less training time (Yosinski et al., 2014). A fine-tuning process is then applied to the entire backbone connected with the classification head (Chatfield et al., 2014a), i.e., no layers were frozen for this training, but the entire network was fine-tuned starting from the ILSVRC initialization. For this, the categorical *cross-entropy* loss function between each output activation and its expected activation was used to calculate the error. The network parameters were tuned by means of back-propagation using stochastic gradient descent (Bottou, 2010) and considering the adaptive learning rate proposed by Zeiler (2012) (with an initial value of 0.001 and a decay rate of 0.05).

Training was performed for a maximum of 200 epochs with a mini-batch size of 32 samples for each of the datasets described in Section 4: the synthetic dataset is first employed, after which the obtained weights are fine-tuned using the corpus of real images. Also note that the *early stopping* technique was used to avoid overfitting by stopping training when the loss did not decrease during 10 epochs.

The remaining heads were trained by freezing the backbone and the classification head weights, so only the layers of each of these modules were adjusted. With regard to the pointing and drag heads, the process described in Section 5.3 was carried out using the filters learned by the backbone. In the case of the loupe head, a fine-tuning process (during 200 epochs with a batch size of 32) was performed using the dataset composed of images and textual descriptions (see Section 4).

With regard to the pinch head, sequences labeled as zoom-in, zoom-out or static gestures were used. This module was also trained during 200 epochs with a batch size of 32 but using a special type of data augmentation. In this case, the same transformation was applied to the two frames, which included variations in the speed of the opening and closing gestures (varying the value of d by ± 1 frames) and variations in the static gestures (using the same frame or comparing it with the previous and subsequent frame).

The rest of the details about the training carried out for the experimentation are included in the following section.

6. Experiments

In this section, the different parts of the proposed method are evaluated, starting with the performance of the backbone and the classification head, and continuing with an analysis of the results obtained by each of the specialized heads. In all cases, the results are compared with those of other state-of-the-art methods. It is important to note that all the models were trained and evaluated under the same conditions. The following section details the specific configuration followed for the training of the different methods.

6.1. Experimental setup

All experiments were carried out using the Python programming language (v. 3.7) with the TensorFlow (v. 2.1) and Keras (v. 2.3) libraries. The machine used consists of an Intel(R) Core(TM) i7-8700 CPU @ 3.20 GHz with 16 GB RAM, a NVIDIA GeForce RTX 2070 with 6 GB GDDR6 Graphics Processing Unit (GPU) with the cuDNN library.

In all of the experiments, we used an n-fold cross validation (with $n = 5$), which yields a better Monte-Carlo estimation than when solely performing the tests with a single random partition (Kohavi, 1995). The datasets were consequently divided into n mutually exclusive folds, being the different classes equally represented in each of the partitions. For each fold, we used one of the partitions for test (20% of the samples) and the rest for training (80%). Besides, a validation subset with 10% of the training samples was used for the adjustment of the hyperparameters and to stop training when there was no improvement. The training and testing processes were repeated $n = 5$ times, using the different partitions of the dataset, and finally the average result was calculated.

As for the proposed method, all the compared networks were trained for 200 epochs, with a batch size of 32, and stopping the training process if the loss did not decrease during 10 epochs. In the same way, backbones were initialized with the ILSVRC dataset and fine-tuned for the corpus proposed in this work.

Data augmentation was used to artificially increase the size of the training set by randomly applying different types of transformations to the original training samples. This technique usually improves the performance and helps reduce overfitting (Krizhevsky et al., 2012; Chatfield et al., 2014b). In our case, 10 augmented images were generated for each image in the training set. The transformations applied were randomly selected from the following set of possible transformations: horizontal flips (allowing the system to work regardless of the hand used), horizontal and vertical shifts ([−10, 10]% of the image size), zoom ([−10, 10]% of the original image size), and rotations (in the range [-5°, 5°]).

6.2. Evaluation of gesture classification

In order to assess the performance of the gesture classification methods, three evaluation metrics widely used for this kind of tasks were chosen, Precision, Recall, and F_1. These are binary metrics to measure the result of a single class, so for multi-class problems, the one-vs-all strategy is used, subsequently calculating the average of the results. Taking one class as positive and the rest as negative, these metrics can respectively be defined as:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

$$F_1 = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

where TP, FP, and FN denote the number of true positives, false positives, and false negatives, respectively. Since the experiments were conducted as a multi-class problem, we report the results in terms of macro-F_1 for a global evaluation, which is calculated as the average of the F_1 obtained for each class.

In this part of the method (which detects the initial gesture), we considered three different approaches: (1) categorical CNN architectures with which to classify the gesture in the input image, (2) the use of object detection networks to detect and classify the gesture, and (3) the use of a hand tracking approach based on the MediaPipe library (Zhang et al., 2020b), which detects both the hand and the position of the fingers.

Categorical CNN architectures. With regard to the first type of approximation, 13 representative state-of-the-art CNN topologies (listed in Table 3) were considered. The reader is referred to the cited works for the implementation details. These topologies were used as backbones, whose final layers were removed and replaced with the classification header described in Section 5.2. Table 3 shows the results obtained for the classification task using the synthetically generated dataset (see Section 4). Please recall that all the networks were initialized with the pre-trained weights obtained with the ILSVRC dataset and then trained and evaluated for this synthetic dataset. The training was carried out under the same conditions in all cases (described in Sections 5.6 and 6.1), for 200 epochs, with a batch of 32 and using data augmentation. As will be noted, quite good values are achieved in all cases. The best results are those of Darknet-53 followed by the Xception model, and the worst are those obtained by the VGG architectures. This may be due to the learning capabilities of each model, since in general, and as will be analyzed later, the results improve as the number of network parameters increases. The results reported in this table are the average of the results obtained for the classification of the different classes considered. Additionally, Appendix provides the individual confusion matrices obtained for a detailed inspection of the results.

Table 3
Comparison of the results obtained in terms of Precision, Recall and F_1 with the 13 state-of-the-art CNN topologies considered for the classification of the synthetic dataset. These figures represent the average of the individual classification results obtained for the different classes. The best results obtained per metric are marked in bold type, while the second-best are underlined.

Model	Precision	Recall	F_1
SqueezeNet (Iandola et al., 2016)	87.25	86.13	86.69
ResNet-50 (He et al., 2016)	95.74	94.75	95.24
VGG 16 (Simonyan and Zisserman, 2014)	85.12	67.37	75.21
VGG 19 (Simonyan and Zisserman, 2014)	84.77	65.40	73.84
Inception v3 (Szegedy et al., 2015)	96.60	96.25	96.42
MobileNet v1 (Howard et al., 2017)	97.85	97.83	97.84
MobileNet v2 (Sandler et al., 2018)	97.24	97.12	97.18
MobileNet v3 (Howard et al., 2019)	94.83	92.99	93.90
EfficientNet-B0 (Tan and Le, 2019)	97.16	97.05	97.10
EfficientNet-B1 (Tan and Le, 2019)	97.64	97.41	97.52
DenseNet121 (Huang et al., 2017)	96.51	96.03	96.27
Xception (Chollet, 2016)	<u>98.33</u>	<u>98.30</u>	<u>98.31</u>
Darknet-53 (Redmon and Farhadi, 2018)	**99.76**	**99.75**	**99.75**

The weights learned using the synthetic dataset were used to initialize the networks before training with the real image dataset. Table 4 shows the results obtained for this second step of the training process. This table also includes a comparison with the result that would be obtained from training without this initialization, that is, initializing with ILSVRC and then training directly with the real dataset. As will be observed, the best results are again obtained with the Darknet-53 architecture followed by Xception. Note that this initialization helps improve the results by an average of more than 3%, and if these results are analyzed individually, by up to almost 10% in the case of EfficientNet-B0 and 5.5% for Darknet-53. As before, Appendix provides the individual confusion matrices obtained for a detailed inspection of the results.

Object detection networks. The second approach evaluated for the classification of the initial gesture was the use of object detection networks, for which four alternatives were compared: Faster R-CNN (FRCNN) (Ren et al., 2015), RetinaNet (Lin et al., 2017b), YOLO v3 (Redmon and Farhadi, 2018), and SelAE (Gallego et al., 2018). Table 5 shows the results obtained when employing these methods, and compares them with the two best results obtained previously by the classification networks (see Table 4). The object detection methods return the position of the gestures in the image (the labeling used for this process is described in Section 4). Since these networks can return multiple predictions, the bounding box predicted with the highest confidence was selected. In order to evaluate the result obtained, it is necessary to differentiate between whether or not the ground truth

Table 4
Results obtained in terms of Precision, Recall and F_1 for the categorical classification of the real image dataset after initializing the 13 state-of-the-art CNN topologies considered using the weights learned with the synthetic dataset. These figures represent the average of the individual classification results obtained for the different classes. This table also includes the result obtained when not applying this initialization, that is, starting only with the weights learned from ILSVRC. The best results obtained per metric are marked in bold type, while the second-best are underlined.

Model	No initialization			Synthetic initialization		
	Precision	Recall	F_1	Precision	Recall	F_1
SqueezeNet (Iandola et al., 2016)	69.13	67.08	68.09	71.23	67.95	69.55
ResNet-50 (He et al., 2016)	80.26	78.98	79.61	81.78	80.07	80.92
VGG16 (Simonyan and Zisserman, 2014)	76.72	73.58	75.12	78.45	76.29	77.35
VGG19 (Simonyan and Zisserman, 2014)	79.19	78.33	78.76	81.20	79.90	80.54
Inception v3 (Szegedy et al., 2015)	75.68	73.33	74.49	78.62	75.50	77.03
MobileNet v1 (Howard et al., 2017)	82.72	82.75	82.73	83.96	83.04	83.50
MobileNet v2 (Sandler et al., 2018)	83.44	82.42	82.93	84.68	86.60	84.12
MobileNet v3 (Howard et al., 2019)	84.13	79.86	81.94	85.24	83.27	84.24
EfficientNet-B0 (Tan and Le, 2019)	67.29	61.09	64.04	78.50	69.86	73.93
EfficientNet-B1 (Tan and Le, 2019)	75.15	69.50	72.21	78.59	75.14	76.86
DenseNet121 (Huang et al., 2017)	82.88	79.29	81.05	85.90	81.43	83.61
Xception (Chollet, 2016)	87.01	86.25	86.63	91.42	90.21	90.81
Darknet-53 (Redmon and Farhadi, 2018)	**89.80**	**87.64**	**88.71**	**95.31**	**93.14**	**94.21**
Average	79.49	76.93	78.18	82.68	79.95	81.28

Table 5
Summary of the results obtained in terms of Precision, Recall and F_1 by the different approaches considered for the gesture classification task. These figures represent the average of the individual classification results obtained for the different classes. The best result for each metric is marked in bold type, and the second best result is underlined.

Approach	Model	Precision	Recall	F_1
Categorical	Xception (Chollet, 2016)	91.42	90.21	90.81
	Darknet-53 (Redmon and Farhadi, 2018)	**95.31**	**93.14**	**94.21**
Object detection	FRCNN (Ren et al., 2015)	78.71	70.34	74.29
	RetinaNet (Lin et al., 2017b)	86.25	83.29	84.74
	SelAE (Gallego et al., 2018)	88.17	86.21	87.18
	YOLO v3 (Redmon and Farhadi, 2018)	94.30	93.13	93.71
Hand tracking	MediaPipe (Zhang et al., 2020b)	76.81	77.94	77.37

contains a gesture. When it does, the prediction is considered TP if its IoU with the ground truth is greater than 0.5, FP when it is less than 0.5, or FN in the case of not returning any prediction. The opposite applies for the gesture "None", which is considered to be TP when there is no prediction and FP when there is.

Table 5 shows that, of all the object detection approaches, YOLO v3 obtains the best result, followed by SelAE and RetinaNet. However, if we compare them with the methods specifically trained for categorical classification, Darknet-53 (the backbone used by YOLO v3 itself) still obtains the best score.

Hand tracking approach. This table also includes the result obtained using the third approach: the detection of gestures based on the hand tracking method provided by MediaPipe Hands (Zhang et al., 2020b). This method detects 21 3D keypoints corresponding to the joints of the fingers of a hand from a single RGB image. We evaluated different approaches to classify gestures using this information, such as kNN (k-Nearest Neighbor) or SVM either directly on the 21 keypoints or by accumulating the joint angles of each finger. The latter (using the sum of angles and SVM) was that which obtained the best results, and was, therefore, the one that was finally included in the comparison. However, as will be observed in the table, the results of this method are not competitive if we compare them with those of the other approximations (with the exception of FRCNN). These worse results are owing to the fact that, in many cases, MediaPipe does not detect the hand correctly. These results also coincide with a recent work (Amaliya et al., 2021) in which the performance of this method is compared with other approaches for the recognition of sign language.

Fig. 8 shows some samples of the detections made by this method. The first row shows correct detections and the second row shows the cases in which it has not been able to detect the hand. As will be noted, these failures occur when the hand is partially occluded (i.e., only the fingers or part of the hand can be seen), which is quite common in the proposed application owing to the position of the camera.

6.2.1. Significance tests

To further extend the previous comparison and derive strong conclusions out of them, we now perform a statistical analysis of the results obtained. For that, we resort to the non-parametric Wilcoxon signed-rank test (Demsar, 2006) and carry out a pairwise comparison of the different classifiers in terms of the performance considering the F_1 figures for each fold, label and classifier. Fig. 9 shows the results of this test, considering all the possible combinations of the previously compared methods, both the uninitialized categorical CNN architectures and those initialized with synthetic data, as well as the object detection networks and the hand tracking approach. The yellow and green colors in this figure respectively indicate that the method in the row significantly improves that of the column when considering the statistical significance levels of 90% and 95%.

In general, this figure shows that the initialization with the synthetic dataset produces a significant improvement in the results, since the test is passed 22 times more for this case. Regarding the individual results, it can be easily observed that the methods with the worst results are those that appear in the columns with more green or yellow circles, since they represent the approaches that are exceeded more times. Among these are SqueezeNet, EfficientNet, FRCNN and MediaPipe. The methods that significantly stand out are Xception, Darknet-53, RetinaNet, SelAE and YOLO v3, corresponding to the rows that pass the test more times. Specifically, the network selected for our proposal (Darknet-53) manages to outperform all others. It can be seen that when this method is not initialized, it does not improve Xception or YOLO v3, but after this initialization it does manage to overcome them with a significance level of 95%.

6.2.2. Multi-objective optimization problem

Another important aspect to consider is the efficiency of the method selected. Since the more parameters models have, the slower the performance and the greater the storage space required, it is necessary

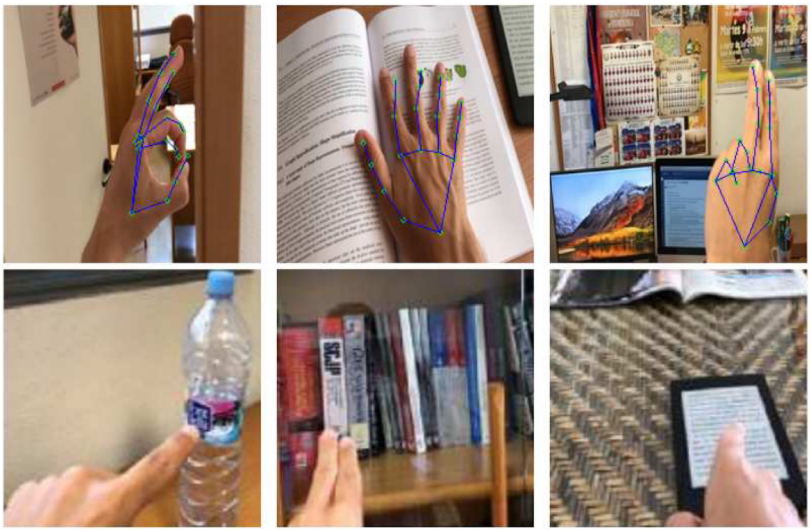


Fig. 8. Some examples of the detections made by MediaPipe. The first row shows correct detections and the second row shows the cases in which the method failed to detect the hand.

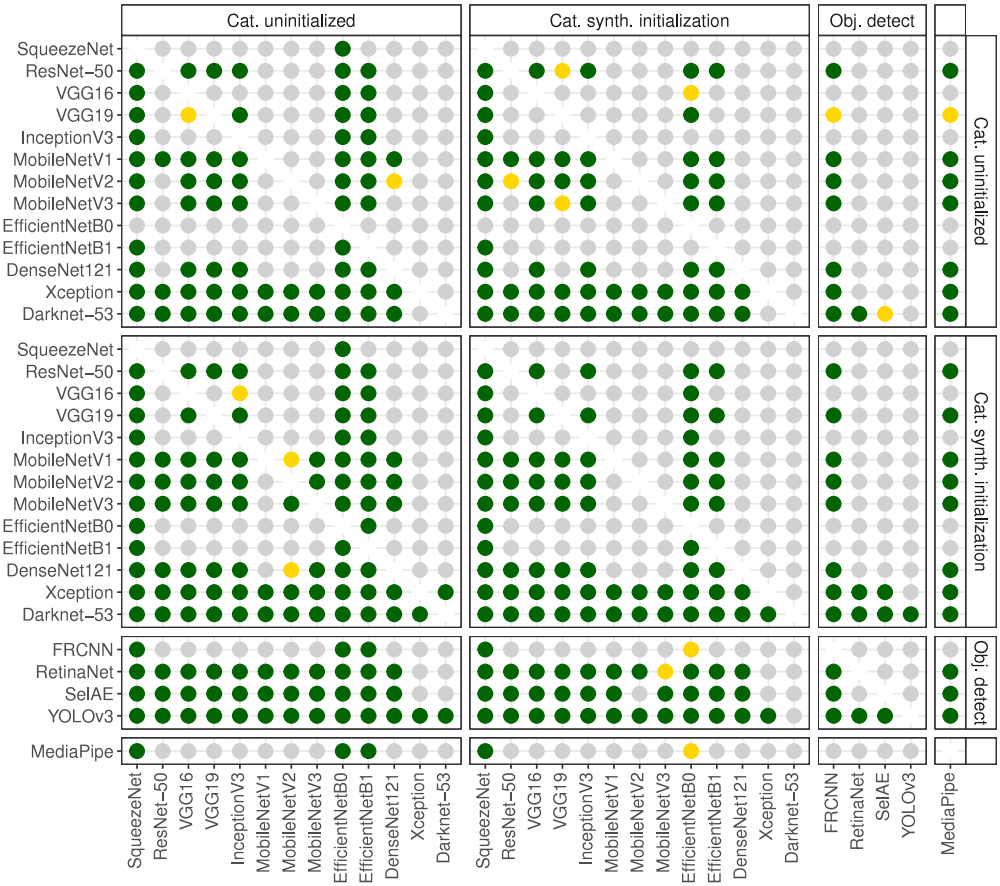


Fig. 9. Wilcoxon signed-rank test of the pairwise comparison in terms of the F_1 score of the considered classification algorithms. Yellow and green colors respectively indicate that the method in the row significantly improves that of the column when considering the statistical significance value of 90% and 95%.

to reach a trade-off between the efficiency of the network and its accuracy. However, these criteria are, quite often, contradictory, since an improvement to one of them usually entails a worsening of the other. From this point of view, the selection of the best model can be seen as a Multi-objective Optimization Problem (MOP) in which two functions are meant to be optimized simultaneously.

The most common means employed to deal with problems of this nature is that of resorting to the concept of *non-dominance*: one solution is said to dominate another if, and only if, it is better or equal in each objective function, and at least strictly better in one of them. The best solutions (there may be more than one) are, therefore, those that are non-dominated. In the MOP framework, the strategies within this set define the so-called Pareto frontier and can be considered the best without having to define any order among them (Miettinen, 1999). This will allow us to detect previously evaluated approaches that reach a trade-off between efficacy and efficiency.

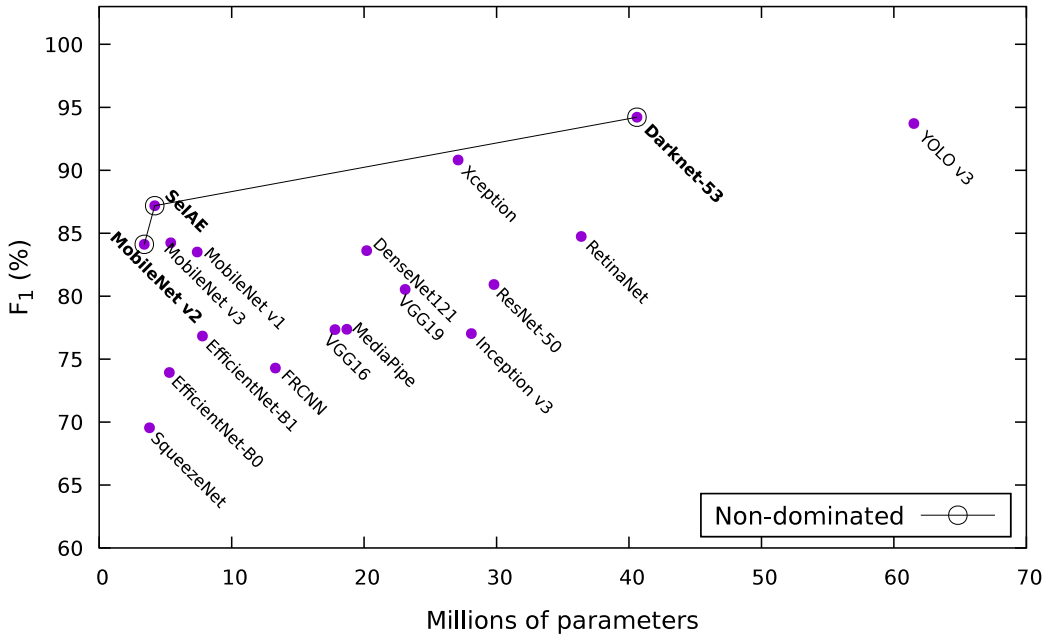


Fig. 10. Analysis of F_1 and efficiency as a Multi-objective Optimization Problem (MOP). Non-dominated elements are highlighted.

Assuming this MOP scenario, Fig. 10 shows the results obtained by the different methods evaluated in the previous section, in which each point is a 2-dimensional value defined by its F_1 (using the result previously obtained in Tables 4 and 5) and the number of parameters of the corresponding topology. As can be seen, the best results for both criteria (i.e., the *non-dominated* elements) are obtained by MobileNet v2 (with the lowest number of parameters), Darknet-53 (with the best F_1), and SelAE (as an intermediate solution). In general, it is observed that the F_1 improves as the number of parameters increases, which also determines a greater learning capacity. This trend is not true for less effective architectures, such as VGG, or for object detection methods, such as FRCNN, RetinaNet, or even YOLO v3, possibly because in these cases they are trained to solve a much more complex task. In the case of MediaPipe, in addition to this motivation, it presents the previously mentioned problem that the proposed user interface is not suitable for this library.

As argued above, it is essential to obtain high precision in the first step of the proposed method, and it was for this reason that DarkNet-53 was selected for the backbone, since it obtains an F_1 that is 7.03% higher than the next non-dominated result (SelAE). In addition, this architecture performs well in current mobile devices. According to the tests conducted on a Samsung A51 and a Huawei P30 lite, an average response time of between 3 and 4 FPS was obtained.

6.2.3. Temporal consistency

Another important part of the proposed architecture that had to be evaluated was the temporal consistency module (see Fig. 5) and the effect of the value selected for the k parameter. This parameter makes it possible to control the number of frames with the same response that must elapse for the response to be valid. Fig. 11 shows a graph in which this value is studied in the range $[1, 4]$ for the classification task and using the Darknet-53 backbone. As can be seen, the result improves when it is set to 2 or 3 frames, since this prevents frames with isolated errors from occurring. However, if this value is increased, the result starts to worsen, since it has to wait many frames in order to change the response, and it consequently makes mistakes in all the transitions between gestures. The decision was, therefore, made to set the value of k at 2, since at most it generates an erroneous frame between transitions, and in return it improves the F_1 from 94.21% to 97.03%.

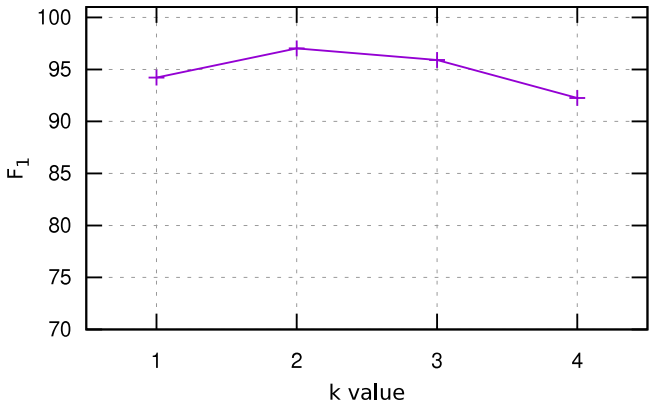


Fig. 11. Evaluation of the k parameter in the temporal consistency for the classification task using the Darknet-53 backbone.

6.3. Evaluation of pointing and drag gestures

Once the main configuration of the architecture had been established, the results obtained by each of the specialized heads were analyzed. This section focuses on the heads used for the pointing and drag gestures, starting with an analysis of the methodology proposed for these actions (see Section 5.3), which is then compared with other architectures. In all cases, the dataset with fingertip labeling for the point and drag gestures was used (see Section 4).

The results were also evaluated using the F_1 metric (Eq. (7)), but in this case we considered the objects (i.e., the fingertips) whose location was correctly detected. This was done by calculating the bounding box of the predicted objects (P), which was then matched with the bounding box of the ground truth (B) with which it had a higher IoU (using Eq. (3)). A predicted bounding box P was considered to be correctly detected if $\text{IoU}(P, B) \geq \lambda$. We established $\lambda = 0.5$, a threshold value commonly used in this type of tasks, and calculated the metric F_1 considering the correct detections to be TP (i.e., when their IoU was greater than λ), the wrong detections to be FP (i.e., when a P did not overlap with any B with a IoU greater than λ), and those cases in which a ground truth object was not detected were considered to be FN. Note

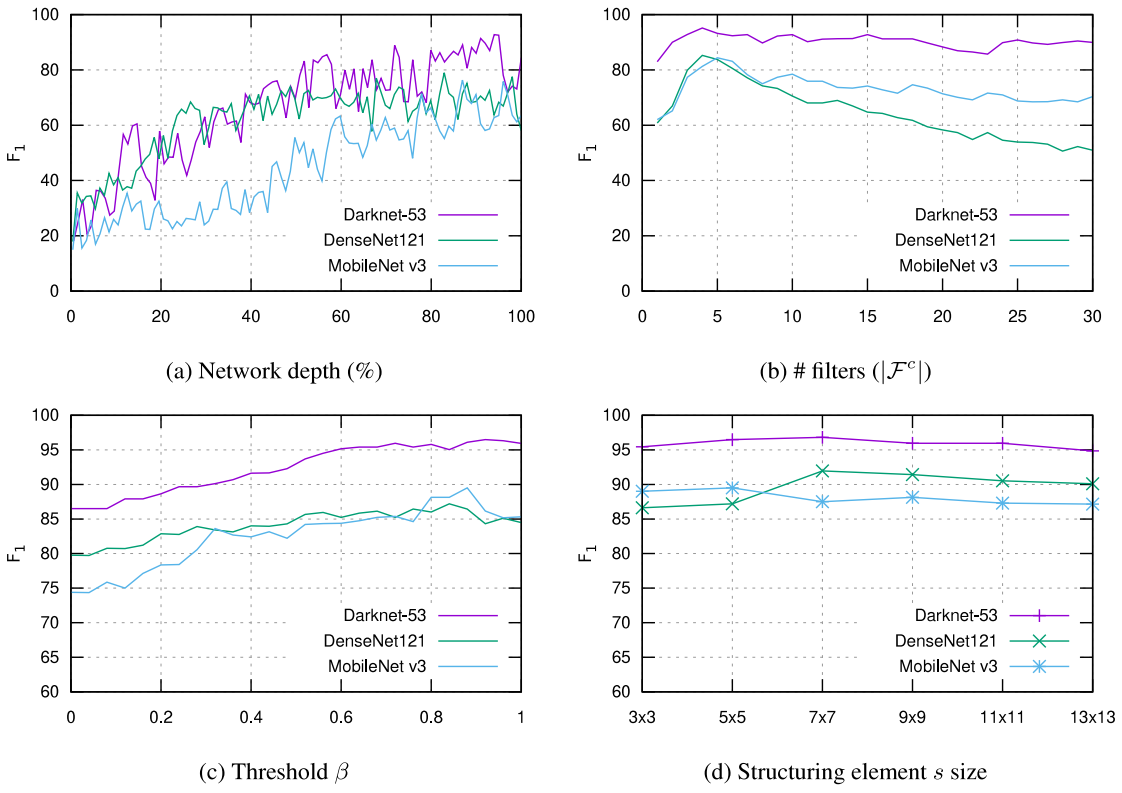


Fig. 12. Localization results (F_1 %) obtained by varying (a) the layer of the network from which the filters are selected (given in percentages with respect to 100% of the total network depth), (b) the number of filters in the set $|\mathcal{F}^c|$, (c) the threshold β, and (d) the structuring element s size.

that if multiple detections of the same object were predicted, only the first was counted as positive while the others were counted as negative.

First, an evaluation of the parameters of the proposed methodology (see Section 5.3) will be carried out, starting with the selected layer (variable l in Eq. (1)). This will be done by setting the remaining parameters to an initial configuration ($|\mathcal{F}^c| = 3$, $\beta = 0.6$, and a square structuring element of size $s = 5 \times 5$), and by varying only the selected layer l. Three representative network architectures from those previously evaluated (Darknet-53, DenseNet121 and MobileNet v3) will be considered in this analysis, although this adjustment was made for all networks, as will be shown later.

Fig. 12(a) shows the influence of the CNN layer selected in order to predict the localization. This was done by computing the result obtained with all the layers from the network models considered, while the remaining parameters were set to the aforementioned values. Since each network has a different number of layers, in this figure we represent the result as a function of the layer depth, where 100% of the depth signifies the last layer of the network. As will be observed, the results in the first part (more or less up to 50% of depth) were not good. However, as expected, better localization results were obtained in the last part of the networks (from 50% of depth), from which higher-level features are generally learned. The "*conv2d_49*" layer was, therefore, eventually selected for Darknet-53, "*conv5_block6_2_conv*" for DenseNet121, and "*re_lu_35*" for MobileNet v3 (the full network architectures can be consulted in the corresponding papers).

Another important variable that had to be analyzed was the number of filters selected in order to obtain the localization, that is, the size of the set $|\mathcal{F}^c|$ in Eq. (4), which can be adjusted by modifying the threshold value α. In this experiment, we used the best layers previously selected: $\beta = 0.6$ and $s = 5 \times 5$. Fig. 12(b) shows the results obtained by varying the size of this set. As will be noted, a maximum is obtained when using between 3 and 6 filters, and the best results are obtained with 4 filters for Darknet-53 and DenseNet121, and with 5 filters for MobileNet v3.

Another parameter that had to be analyzed was the value of the threshold β (see Eqs. (2) and (4)). As before, we set the remaining parameter values to the best ones found and varied only this parameter in the range $[0, 1]$. Fig. 12(c) shows that better results are obtained with higher values for this threshold, i.e., when selecting only those pixels with the highest activations. The specific values selected for each network are: $\beta = 0.92$ for Darknet-53, $\beta = 0.84$ for DenseNet121, and $\beta = 0.88$ for MobileNet v3.

Finally, we also analyzed the influence of the size of the structuring element s (see Eqs. (2) and (4)) that is used for the dilation of the result obtained from the activation of the filters before calculating the bounding box with the position of the detected objects. The influence of this parameter was assessed by varying the size of the structuring element between 3×3 and 13×13, and setting the remaining parameters to the best ones found in the previous experiments. Fig. 12(d) shows the result of this analysis. As can be seen, the result remains fairly stable when varying this parameter, and improves only slightly with a kernel size of 7×7 for Darknet-53 and DenseNet121, and 5×5 for MobileNet v3.

Having analyzed the different parameters of the proposed method and determined the best configuration, the results obtained are now compared with those of other state-of-the-art methods, including the use of FS on the rest of categorical networks and the four object detection networks evaluated previously (FRCNN Ren et al., 2015, RetinaNet Lin et al., 2017b, YOLO v3 Redmon and Farhadi, 2018, and SelAE Gallego et al., 2018). These last four proposals will be denoted as "single-head", since they are task-specific approaches that, unlike the multi-head ones, do not have to solve other tasks. For this comparison, we show the average value of the IoU obtained, along with the Average Precision (AP), given that these metrics are widely used to evaluate object detection methods, as occurs in the PASCAL VOC challenge. The most recent PASCAL challenge AP metric has been used (by interpolating all the points rather than using a fixed set of uniformly-spaced recall values) (Everingham et al., 2015). This

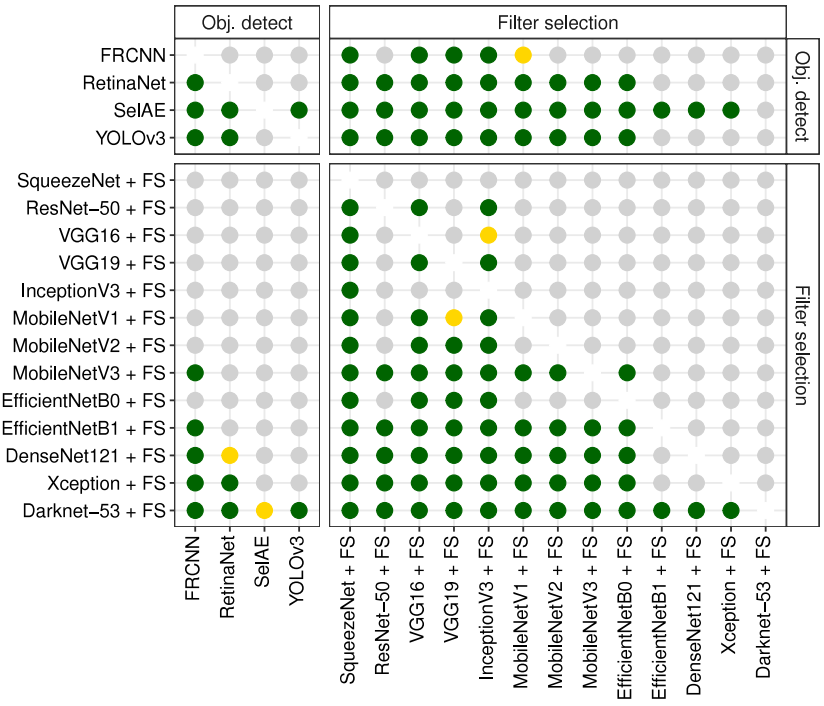


Fig. 13. Wilcoxon signed-rank test of the pairwise comparison in terms of mAP of the considered object location algorithms. Yellow and green colors respectively indicate that the method in the row significantly improves that of the column when considering the statistical significance value of 90% and 95%.

Table 6
Comparison of the results obtained using the proposed approach (Darknet-53 + FS) and other state-of-the-art solutions, including object detection methods, and the result obtained after applying FS to the other categorical networks considered previously. The table differentiates the methods that are specific to the task (single-head) from those trained following the proposed multi-head architecture. The best results for each metric are marked in bold type, while the second-best are underlined.

Approach	Method	Avg(Iou)	mAP
Object detection (Single-head)	FRCNN	77.95	85.46
	RetinaNet	83.35	90.08
	YOLO v3	<u>84.71</u>	<u>95.63</u>
	SelAE	83.38	91.21
Filter selection (Multi-head)	SqueezeNet + FS	65.25	70.12
	ResNet-50 + FS	74.74	83.99
	VGG16 + FS	71.07	79.46
	VGG19 + FS	71.91	81.77
	Inception v3 + FS	58.89	78.30
	MobileNet v1 + FS	71.13	84.28
	MobileNet v2 + FS	74.51	85.61
	MobileNet v3 + FS	75.40	88.93
	EfficientNetB0 + FS	77.40	85.25
	EfficientNetB1 + FS	79.57	91.25
	DenseNet121 + FS	80.26	91.20
	Xception + FS	83.66	92.14
	Darknet-53 + FS	**85.77**	**96.32**

metric calculates the mean value in the recall interval $[0, 1]$, which is equivalent to the area under the curve (AUC) of the Precision–Recall curve (PRC). The mAP is calculated by averaging the AP obtained for each class.

Table 6 shows the results of this comparison. As can be seen, the method proposed in order to carry out this task (Darknet-53 + FS) is that which obtains the best results, followed by YOLO v3 (the architecture to which the backbone employed belongs). In general, the object detection approaches obtain quite good results. However, in addition to slightly improving the result, FS is a more efficient solution since it does not add any processing layer to the network, it simply takes advantage of the activations of the filters already calculated by the backbone in order to perform this detection.

As in the previous experiment, to rigorously validate these results, we have performed a statistical analysis using Wilcoxon's non-parametric signed-rank test and pairwise comparing the results of all the methods evaluated in terms of mAP. Fig. 13 shows the results of this test, in which the proposal improves the rest of the methods with a significance of 95%, with the exception of SelAE, whose result is also exceeded but reducing the significance to 90%.

Fig. 14 shows an example of the filters obtained using the proposed method Darknet-53+FS for an input image. This figure also shows the process of adding up the result until the final prediction is attained. The first row of this image shows the input frame and the process carried out for the first filter, while the second, third and fourth rows show, in addition to the process performed on the filter, the result of the incremental sum with the previous filters.

6.4. Loupe gesture evaluation

The descriptions generated by the head dedicated to the loupe gesture were evaluated by employing the widely adopted Bi-Lingual Evaluation Understudy (BLEU) metric (Papineni et al., 2002). It is generally used to assess the quality of machine-generated sentences by comparing them with reference sentences in problems related to language generation, image captioning, text summarizing, or speech recognition, among others. The output of this metric is in the range $[0, 1]$, where values closer to 1 represent more similar texts. A score of 1 indicates that the sentences are the same. However, it is not necessary to attain this value for the text to be correct. The use of n-grams of a length of between 1 and 4 were considered for the calculation of this metric. This length refers to the number of words in a row that have to match, signifying that the length of 4 (denoted as BLEU-4) would be the most challenging. Also note that for this experiment, in addition to the images with the loupe gesture, the pointing gesture and the images without gesture were also evaluated. These last two cases were added to consider further evaluation samples and also to assess them in case they also have to generate a captioning in the final application.

Table 7 shows the results of this evaluation, in which the proposed method (Darknet-53 + captioning) is compared to the original *merge-model* approach (Tanti et al., 2018) (on which the specialized head of

Table 7
Comparison of the results obtained (in terms of BLEU) for the image captioning task. The table differentiates the methods that are specific to the task (single-head) from those trained following the proposed multi-head architecture with the captioning head.

Approach	Method	BLEU-4	BLEU-3	BLEU-2	BLEU-1
Single-head	Merge-model (Tanti et al., 2018)	0.1518	0.1864	0.3151	0.4987
	MobileNet v3 + captioning	0.1803	0.2921	0.3477	0.5614
	DenseNet121 + captioning	0.1997	0.3117	0.3904	0.5882
	Darknet-53 + captioning	0.2163	0.3236	0.4408	0.6135
Multi-head	MobileNet v3 + captioning	0.1749	0.2803	0.3295	0.5401
	DenseNet121 + captioning	0.1931	0.3027	0.3777	0.5718
	Darknet-53 + captioning	**0.2088**	**0.3181**	**0.4390**	**0.6027**

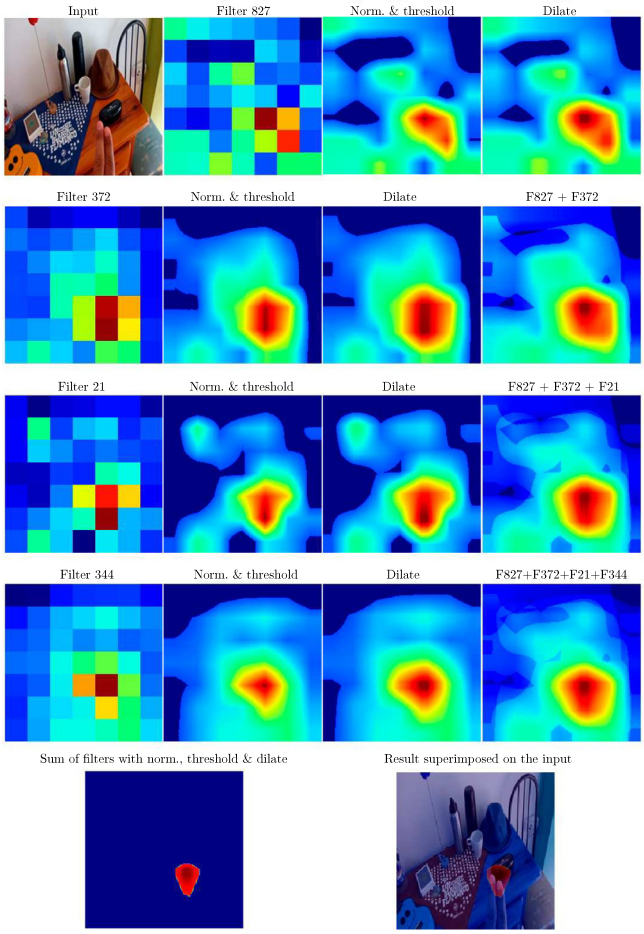


Fig. 14. Example of the process performed to calculate the location of the fingertips using the proposed approach (Darknet-53 + FS). The first image in the first row is the input. The intermediate results obtained for each of the four filters selected are shown, along with their incremental sum with the previous filters. The last row shows the final result, including an overlay of the predicted localization with the input image. A higher activation value is indicated in dark red.

our proposal is based) and with the result that would be obtained when exchanging Darknet-53 for DenseNet121 or MobileNet v3. Note that the table differentiates multi-head methods from those that are specifically designed and trained for the task, i.e., the architecture composed of only the backbone and the layers used by the captioning head. In this latter case, the network is initialized with the weights pre-trained for ILSVRC and then trained (without freezing the backbone) with the dataset created for this task.

As will be observed in the table, the best result is obtained using the single-head approach that combines Darknet-53 with the proposed captioning head. However, this result is only slightly better than that obtained when considering the whole proposed multi-head architecture (result of the last row). Also, the statistical tests included in Fig. 15(a) indicate that the improvement obtained by the single-head approach versus the multi-head approach is not significant. This small improvement does not, therefore, justify the use of an independent network to carry out this task, since this would suppose a considerable increase in the resources required.

The poor result obtained by the merge-model method is perhaps due to the fact that it considers a simpler backbone (VGG19). As a reference, the results reported by Tanti et al. (2018) for the Flikr8k dataset are 0.191, 0.287, 0.424, 0.611 for BLEU-4, BLEU-3, BLEU-2, and BLEU-1, respectively.

Fig. 16 shows ten examples of the captions generated for our dataset. The first two columns of these examples include the cases in which the texts generated mention the hand or the fingers and the result obtained after post-processing them. As will be noted, the method generates descriptions that correctly detail the scenes and the objects that appear in them.

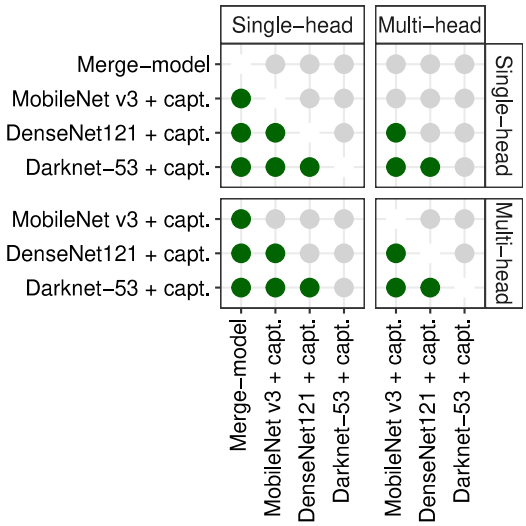
6.5. Evaluation of pinch gesture

Finally, the pinch gesture was evaluated. As indicated in the methodology, this is a dynamic gesture that may change in order to indicate whether the user wishes to zoom in, zoom out, or maintain the current zoom level. As in the previous sections, the Precision, Recall and F_1 metrics were employed to assess the performance of the detection of these actions.

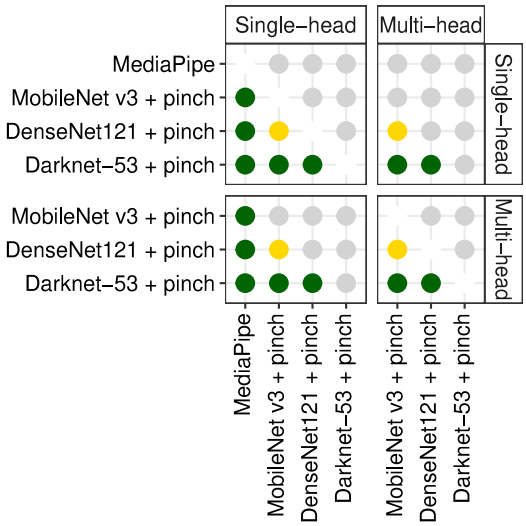
For this evaluation, the proposed approach was compared with both single-head and multi-head solutions. For the single-head case, the use of MediaPipe and the pinch head combined with different backbones was considered. These were initialized with the weights obtained for ILSVRC and fine-tuned for the dataset prepared for this purpose. In the case of MediaPipe, the distance between the thumb and index fingers was calculated in order to determine which gesture was being performed: zooming in when the distance increases, zooming out when the distance decreases, and maintaining the zoom level when the distance remains stable (allowing a small threshold of ± 3 px of variation). For the multi-head case, in addition to Darknet-53, DenseNet121 and MobileNet v3 were also evaluated.

Table 8 shows the results of this comparison. As will be observed, MediaPipe obtains the lowest scores since, as previously argued, this method is quite dependent on the visibility of the hand. The proposed approach (multi-head Darknet-53 + pinch) achieves an F_1 of 90.64%, which is only 0.97% less than the result obtained by the same architecture but trained following a single-head approach. This experiment shows that the proposed training process does not entail a notable deterioration in the result, but does in return allow the achievement of an efficient system for the simultaneous processing of different actions.

As before, we performed statistical tests to rigorously validate these results. As can be seen in Fig. 15(b), the improvement obtained with the single-head solution is not significant with respect to the multi-head proposal. Therefore, this small difference would not justify the use of a parallel network to process this gesture, since this would imply a reduction in efficiency and an increase in the resources required.



(a) Wilcoxon test in terms of BLEU-4 for the loupe head algorithms.



(b) Wilcoxon test in terms of F_1 for the pinch head algorithms.

Fig. 15. Wilcoxon signed-rank test of the pairwise comparison of the considered algorithms for the loupe head and the pinch head. Yellow and green colors respectively indicate that the method in the row significantly improves that of the column when considering the statistical significance value of 90% and 95%.

Fig. 16. Some evaluation examples of the captioning model using the proposed hand gesture dataset. The first two columns show those cases in which the description mentions the hand or the finger, and the result obtained after the post-processing step.

Table 8
Comparison of the results obtained for the pinch gesture in terms of Precision, Recall and F_1. These figures represent the average of the individual classification results obtained for the different classes. The table separates the methods specifically designed for this task (referred to as "single-head") from those trained following the proposed multi-head architecture.

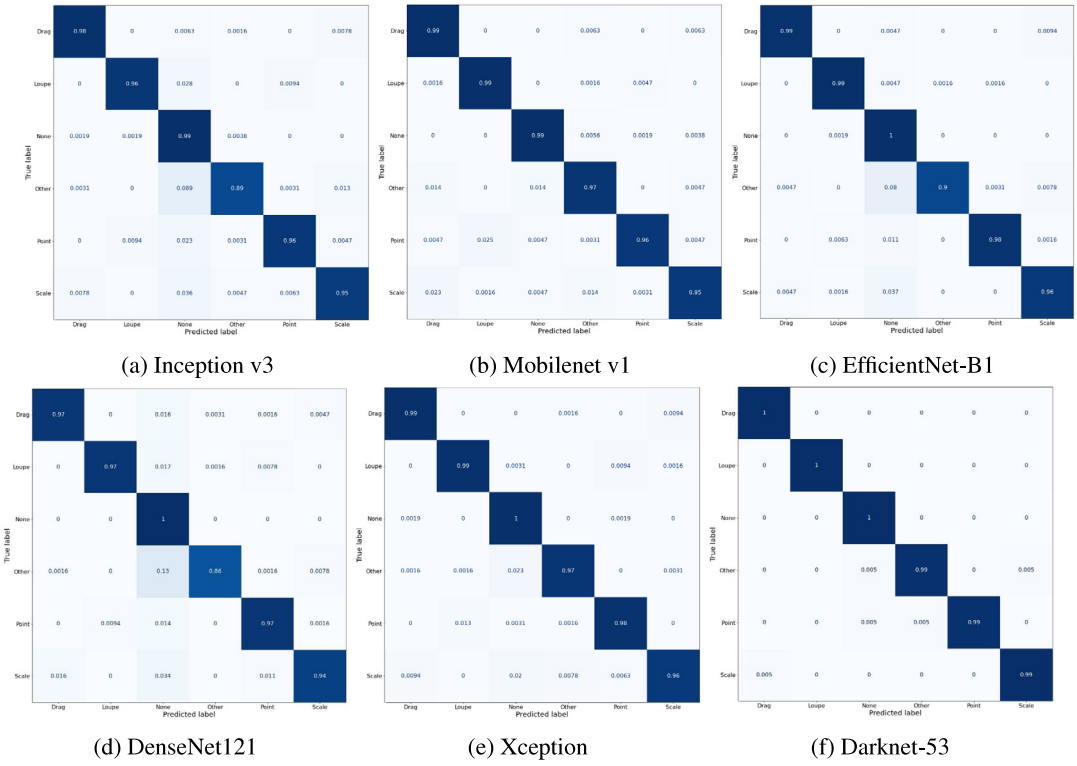
Approach	Method	Precision	Recall	F_1
Single-head	MediaPipe	63.50	63.57	63.53
	MobileNet v3 + pinch head	83.73	82.19	82.95
	DenseNet121 + pinch head	85.12	85.75	85.43
	Darknet-53 + pinch head	**93.27**	**90.01**	**91.61**
Multi-head	MobileNet v3 + pinch head	82.83	81.92	82.37
	DenseNet121 + pinch head	84.87	84.11	84.49
	Darknet-53 + pinch head	92.06	89.27	90.64

7. Discussion and limitations

Although exhaustive experimentation has been carried out with very good results, the current system has some limitations that should be resolved before putting it into practice. For this purpose, two key aspects should be studied: the usability of the interface and the generality and/or robustness of the model.

To evaluate the usability of the user interface, a set of user tests could be carried out with the participation of people with visual impairments. The objective of this study will be to analyze the proposed interface, the set of gestures considered and the actions performed for each gesture. The proposed architecture allows gestures and actions to be easily added or modified, and the proposed study will, therefore, help adjust the platform to design a more usable interface.

To improve the generalization capacity and the robustness of the system, a key aspect would be to review the database used. Although a very complete corpus has been considered for this work – with about 40k very varied images – to put the system into practice and guarantee



(a) Inception v3 (b) Mobilenet v1 (c) EfficientNet-B1

(d) DenseNet121 (e) Xception (f) Darknet-53

Fig. A.17. Results in terms of a normalized confusion matrix for a selection of 6 of the state-of-the-art CNN topologies compared in Table 3 for the classification of the synthetic dataset.

its proper functioning, it would be necessary to expand this dataset. For this, we would start by considerably increasing the synthetic database, since this would not have a labeling cost, and a greater amount of backgrounds and variations in the generated hands and gestures could be added. For the dataset with real images, the usability tests carried out could be used to record real use cases. This would add much more variability by including the hands of more people with different sizes and skin colors, new backgrounds, etc. In addition, this process could be applied more times if errors are still detected in the system. For example, errors could be reported along with the recording and, in this way, the system could be retrained to fix them.

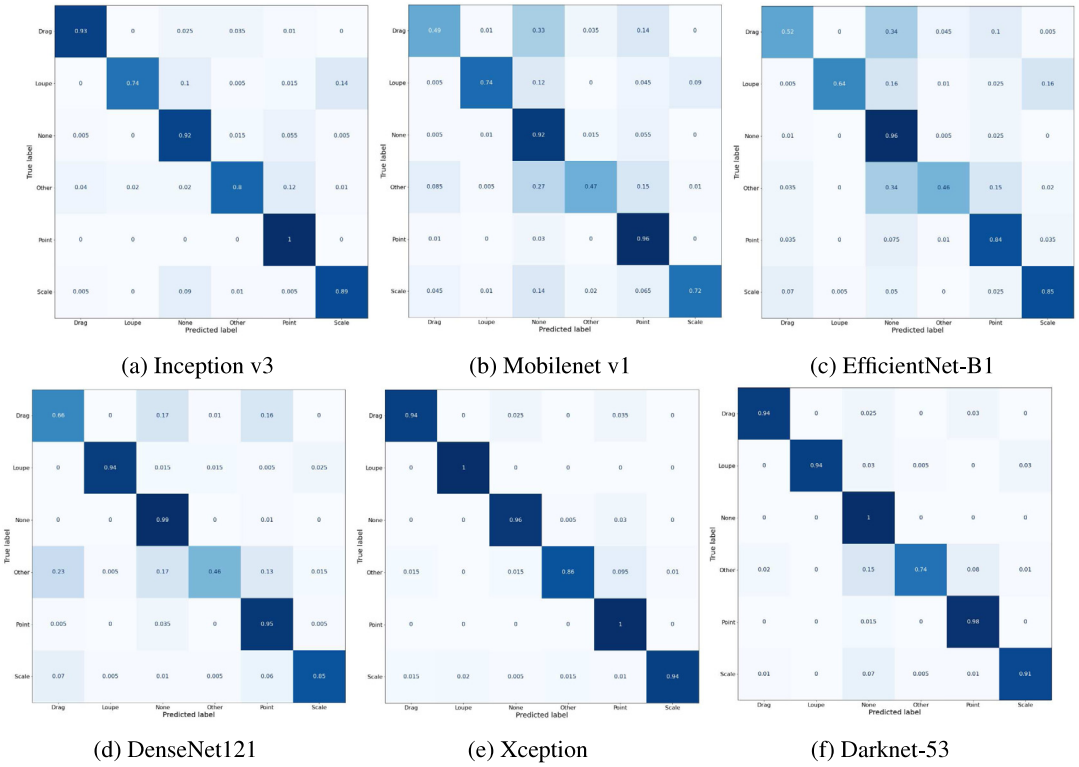
Another possible avenue to develop a more robust and generic system would be to improve the proposed methodology itself. For this, a key aspect is the descriptor used to represent the scene, since the rest of the processes depend on it. For this, instead of relying solely on the output of the backbone, several descriptors of this network could be combined or even a series of layers with a specialized loss could be added so that other features are extracted, following an approach similar to the one proposed by Sitaula et al. (2021b,a), where descriptors representing background and foreground, or objects and context are combined. Furthermore, this solution would also be efficient since, by using the same backbone and even adding a parallel branch, it would not affect the performance of the system.

The inclusion of regularization mechanisms or the use of other proposals for the specialized heads could also be studied. The heads that have the most room for improvement are those used for the loupe and pinch gestures, since the others, as validated by statistical tests, obtain a significant improvement compared to the rest of the state-of-the-art methods. For the loupe gesture, it would be key to collect a much larger database of descriptions, considering, for example, the Microsoft Common Objects in Context (COCO) dataset (Lin et al., 2014b), which contains 328 K images. For dynamic gestures, such as the pinch gesture, recursive layers including attention could be integrated to extract time-series features as proposed in Al-qaness et al. (2022).

8. Conclusions

This paper proposes an interactive system for mobile devices controlled by hand gestures, whose objective is to assist people with visual impairments. This system allows users to interact with the device using simple static and dynamic hand gestures, each of which triggers a different action, such as describing the scene or the object pointed to, zooming, etc. The method also optimizes the resources required to perform different tasks, signifying that the system can be embedded in mobile devices. This has been done by employing an efficient multi-head neural network that uses the same features extracted by a common backbone to perform the different actions, which are, moreover, activated only if its corresponding gesture is detected. Therefore, the proposed system allows performing multiple actions in the same application. This is very helpful for visually impaired people, as they do not have to switch applications to perform another action, but can access multiple assistance tools through an intuitive and natural command interface, such as hand gestures. This differentiates our proposal from the rest of the state-of-the-art approaches that focus on a single type of assistance task.

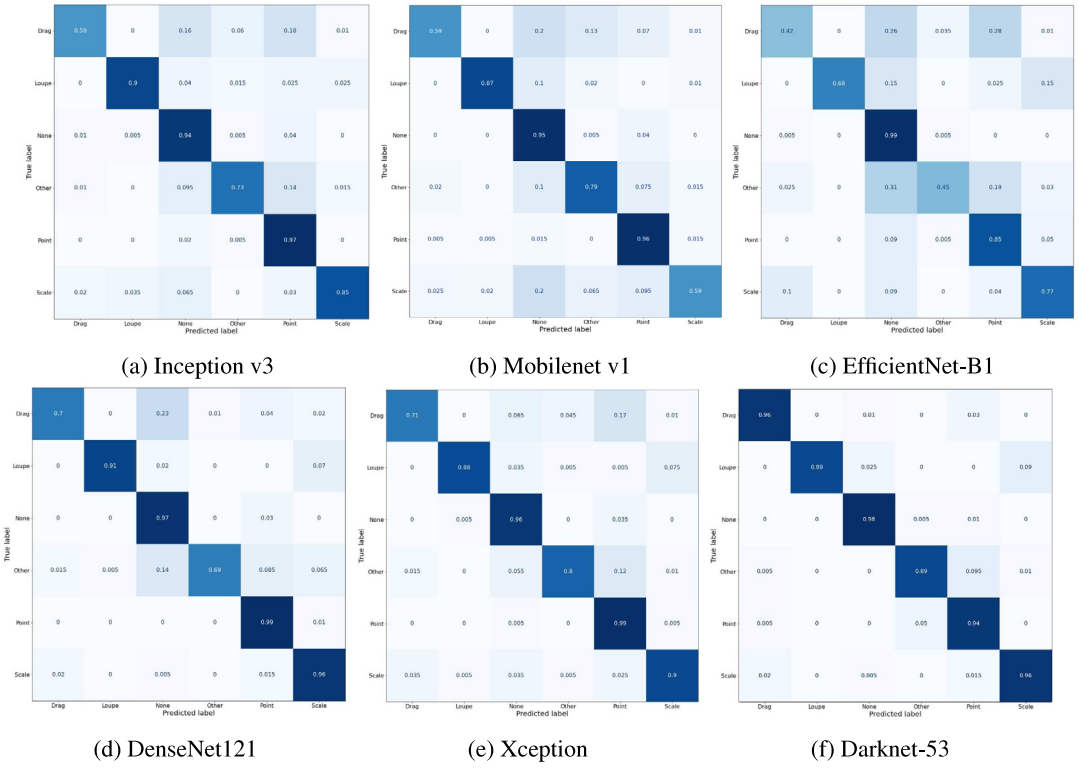
Three different datasets with a total of about 40k images were created to train and evaluate the proposed methodology. The samples were labeled at different levels: category, position of the hands and fingertips, position and category of the objects pointed to, and description of the scenes. The experimentation carried out in each of the steps of the proposed method both attained good results and showed the efficiency of the architecture, resulting in the approximation that obtained the highest precision when adjusting the trade-off between performance and accuracy. The proposed temporal consistency module has proven to improve results by almost 3% thanks to a simple criterion of continuity in the predictions. When comparing the results of each of the specialized heads with those of other state-of-the-art approaches, including specific options for those same tasks, the best results (or almost the best) are in all cases attained by these specialized heads, thus demonstrating the effectiveness of the proposed architecture, even



(a) Inception v3 (b) Mobilenet v1 (c) EfficientNet-B1

(d) DenseNet121 (e) Xception (f) Darknet-53

Fig. A.18. Results in terms of a normalized confusion matrix for a selection of 6 of the state-of-the-art CNN topologies compared in Table 4 (columns "No initialization") for the classification of the real dataset when not applying the proposed initialization based on the weights obtained with the synthetic dataset.



(a) Inception v3 (b) Mobilenet v1 (c) EfficientNet-B1

(d) DenseNet121 (e) Xception (f) Darknet-53

Fig. A.19. Results in terms of a normalized confusion matrix for a selection of 6 of the state-of-the-art CNN topologies compared in Table 4 (columns "Synthetic initialization") for the classification of the real dataset when applying the initialization based on the weights obtained with the synthetic dataset.

when compared to specific approaches. Moreover, the architecture has shown a good performance in current mobile devices, with an average response time of between 3 and 4 FPS obtained in tests conducted on a Samsung A51 and a Huawei P30 lite.

As future work, it is intended to solve the limitations of the system and put it into practice. The first action will be to carry out a set of user tests with the participation of visually impaired people to evaluate the system and its usability. Once the final interface and the set of gestures have been determined, it is intended to improve the generality and robustness of the system, expanding the variability of the scenes and objects considered in the datasets used for the initial classification and for the detection of objects and the captioning actions. A last point to address is that of studying how to improve the proposed architecture, including the intermediate descriptor and the specialized heads.

CRediT authorship contribution statement

Samer Alashhab: Data curation, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Antonio Javier Gallego:** Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Miguel Ángel Lozano:** Project administration, Resources, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Confusion matrices

This appendix includes the confusion matrices obtained for the gesture classification experiments shown in Section 6.2. Specifically, the confusion matrices have been calculated for a representative selection of 6 of the 17 state-of-the-art methods compared. Figs. A.17–A.19 show the confusion matrices corresponding respectively to the results of the Tables 3, 4 (columns "No initialization") and 4 (columns "Synthetic initialization"). In each confusion matrix, the predicted label is represented on the horizontal axis and the ground truth on the vertical axis.

References

Al-qaness, M.A.A., Dahou, A., Elaziz, M.A., Helmi, A.M., 2022. Multi-ResAtt: Multilevel residual network with attention for human activity recognition using wearable sensors. IEEE Trans. Ind. Inf. 1. http://dx.doi.org/10.1109/TII.2022.3165875.

Alashhab, S., Gallego, A.-J., Lozano, M.A., 2019. Hand gesture detection with convolutional neural networks. In: De La Prieta, F., Omatu, S., Fernández-Caballero, A. (Eds.), Distributed Computing and Artificial Intelligence, 15th International Conference. Springer International Publishing, Cham, pp. 45–52.

Alashhab, S., Gallego, A., Pertusa, A., Gil, P., 2019. Precise ship location with CNN filter selection from optical aerial images. IEEE Access 7, 96567–96582. http://dx.doi.org/10.1109/ACCESS.2019.2929080.

Amaliya, S., Handayani, A.N., Akbar, M.I., Herwanto, H.W., Fukuda, O., Kurniawan, W.C., 2021. Study on hand keypoint framework for sign language recognition. In: 2021 7th International Conference on Electrical, Electronics and Information Engineering (ICEEIE). pp. 446–451. http://dx.doi.org/10.1109/ICEEIE52663.2021.9616851.

Bamwenda, J., Özerdem, M., 2019. Recognition of static hand gesture with using ANN and SVM. Dicle Univ. J. Eng..

Benitez-Garcia, G., Prudente-Tixteco, L., Castro-Madrid, L.C., Toscano-Medina, R., Olivares-Mercado, J., Sanchez-Perez, G., Villalba, L.J.G., 2021. Improving real-time hand gesture recognition with semantic segmentation. Sensors 21 (2), http://dx.doi.org/10.3390/s21020356, URL https://www.mdpi.com/1424-8220/21/2/356.

Bheda, V., Radpour, D., 2017. Using deep convolutional networks for gesture recognition in American sign language. CoRR, abs/1710.06836 arXiv:1710.06836, URL http://arxiv.org/abs/1710.06836.

Bottou, L., 2010. Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010. Springer, pp. 177–186.

Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., Sheikh, Y.A., 2019. Openpose: Realtime multi-person 2D pose estimation using part affinity fields. IEEE Trans. Pattern Anal. Mach. Intell..

Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A., 2014a. Return of the devil in the details: Delving deep into convolutional nets. CoRR, abs/1405.3531.

Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A., 2014b. Return of the devil in the details: Delving deep into convolutional nets. In: British Machine Vision Conference. pp. 1–11. http://dx.doi.org/10.5244/C.28.6.

Chollet, F., 2016. Xception: Deep learning with depthwise separable convolutions. CoRR, abs/1610.02357.

De Smedt, Q., Wannous, H., Vandeborre, J.-P., Guerry, J., Saux, B.L., Filliat, D., 2017. 3d hand gesture recognition using a depth and skeletal dataset: Shrec'17 track. In: Proceedings of the Workshop on 3D Object Retrieval, pp. 33–38.

Demsar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.

Devineau, G., Moutarde, F., Xi, W., Yang, J., 2018. Deep learning for hand gesture recognition on skeletal data. In: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018). IEEE, pp. 106–113.

Dinh, D.-L., Kim, J.T., Kim, T.-S., 2014. Hand gesture recognition and interface via a depth imaging sensor for smart home appliances. Energy Procedia 62, 576–582.

Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A., 2015. The pascal visual object classes challenge: A retrospective. Int. J. Comput. Vis. 111 (1), 98–136.

Gallego, A.-J., Gil, P., Pertusa, A., Fisher, R.B., 2018. Segmentation of oil spills on side-looking airborne radar imagery with autoencoders. Sensors 18 (3), http://dx.doi.org/10.3390/s18030797, URL http://www.mdpi.com/1424-8220/18/3/797.

Ge, L., Ren, Z., Li, Y., Xue, Z., Wang, Y., Cai, J., Yuan, J., 2019. 3d hand shape and pose estimation from a single rgb image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10833–10842.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.

Hodosh, M., Young, P., Hockenmaier, J., 2013. Framing image description as a ranking task: Data, models and evaluation metrics. J. Artificial Intelligence Res. 47, 853–899.

Holzbock, A., Tsaregorodtsev, A., Dawoud, Y., Dietmayer, K., Belagiannis, V., 2022. A spatio-temporal multilayer perceptron for gesture recognition. In: 33rd IEEE Intelligent Vehicles Symposium (IV22).

Hossain, M.Z., Sohel, F., Shiratuddin, M.F., Laga, H., 2018. A comprehensive survey of deep learning for image captioning. CoRR, abs/1810.04020 arXiv:1810.04020 URL http://arxiv.org/abs/1810.04020.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H., 2019. Searching for MobileNetV3. arXiv:1905.02244.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2261–2269. http://dx.doi.org/10.1109/CVPR.2017.243.

Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360.

Kim, M.-S., Lee, C.H., 2016. Hand gesture recognition for kinect v2 sensor in the near distance where depth data are not provided. Int. J. Softw. Eng. Its Appl 10 (12), 407–418.

Kohavi, R., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings IJCAI, Vol. 2. In: IJCAI'95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 1137–1143, URL http://dl.acm.org/citation.cfm?id=1643031.1643047.

Köpüklü, O., Gunduz, A., Kose, N., Rigoll, G., 2019. Real-time hand gesture detection and classification using convolutional neural networks. In: 2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019). IEEE, pp. 1–8.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. 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. Curran Associates, Inc..

Lai, Z., Yao, Z., Wang, C., Liang, H., Chen, H., Xia, W., 2016. Fingertips detection and hand gesture recognition based on discrete curve evolution with a kinect sensor. In: 2016 Visual Communications and Image Processing (VCIP). IEEE, pp. 1–4.

Lamberti, L., Camastra, F., 2011. Real-time hand gesture recognition using a color glove. In: International Conference on Image Analysis and Processing. Springer, pp. 365–373.

Lin, W., Du, L., Harris-Adamson, C., Barr, A., Rempel, D., 2017a. Design of hand gestures for manipulating objects in virtual reality. In: Kurosu, M. (Ed.), Human-Computer Interaction. User Interface Design, Development and Multimodality. Springer International Publishing, Cham, pp. 584–592.

Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P., 2017b. Focal loss for dense object detection. CoRR, abs/1708.02002 arXiv:1708.02002 URL http://arxiv.org/abs/1708.02002.

Lin, H.I., Hsu, M.H., Chen, W.K., 2014a. Human hand gesture recognition using a convolution neural network. In: 2014 IEEE International Conference on Automation Science and Engineering (CASE). pp. 1038–1043. http://dx.doi.org/10.1109/CoASE.2014.6899454.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014b. Microsoft coco: Common objects in context. In: European Conference on Computer Vision. Springer, pp. 740–755.

Ma, Y., Liu, Y., Jin, R., Yuan, X., Sekha, R., Wilson, S., Vaidyanathan, R., 2017. Hand gesture recognition with convolutional neural networks for the multimodal UAV control. In: 2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS). pp. 198–203. http://dx.doi.org/10.1109/RED-UAS.2017.8101666.

Manduchi, R., Coughlan, J., 2012. (Computer) vision without sight. Commun. ACM 55 (1), 96–104.

Mazumdar, D., Talukdar, A.K., Sarma, K.K., 2013. Gloved and free hand tracking based hand gesture recognition. In: 2013 1st International Conference on Emerging Trends and Applications in Computer Science. pp. 197–202. http://dx.doi.org/10.1109/ICETACS.2013.6691422.

Miettinen, K., 1999. Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston.

Mitra, S., Acharya, T., 2007. Gesture recognition: A survey. IEEE Trans. Syst. Man Cybern. C 37 (3), 311–324. http://dx.doi.org/10.1109/TSMCC.2007.893280.

Moin, A., Zhou, A., Rahimi, A., Menon, A., Benatti, S., Alexandrov, G., Tamakloe, S., Ting, J., Yamamoto, N., Khan, Y., et al., 2021. A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition. Nat. Electron. 4 (1), 54–63.

Molchanov, P., Gupta, S., Kim, K., Kautz, J., 2015. Hand gesture recognition with 3D convolutional neural networks. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 1–7. http://dx.doi.org/10.1109/CVPRW.2015.7301342.

Molina, J., Pajuelo, J.A., Martínez, J.M., 2017. Real-time motion-based hand gestures recognition from time-of-flight video. J. Signal Process. Syst. 86 (1), 17–25.

Nguyen, D.H., Do, T.N., Na, I.-S., Kim, S.-H., 2019. Hand segmentation and fingertip tracking from depth camera images using deep convolutional neural network and multi-task segnet. arXiv preprint arXiv:1901.03465.

Papineni, K., Roukos, S., Ward, T., Zhu, W.-J., 2002. BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, pp. 311–318.

Perimal, M., Basah, S., Safar, M., Yazid, H., 2018. Hand-gesture recognition-algorithm based on finger counting. J. Telecommun. Electron. Comput. Eng. 10 (1–13), 19–24.

Pigou, L., Dieleman, S., Kindermans, P.-J., Schrauwen, B., 2015. Sign language recognition using convolutional neural networks. In: Agapito, L., Bronstein, M.M., Rother, C. (Eds.), Computer Vision - ECCV 2014 Workshops. Springer International Publishing, Cham, pp. 572–578.

Pintado, D., Sanchez, V., Adarve, E., Mata, M., Gogebakan, Z., Cabuk, B., Chiu, C., Zhan, J., Gewali, L., Oh, P., 2019. Deep learning based shopping assistant for the visually impaired. In: 2019 IEEE International Conference on Consumer Electronics (ICCE). pp. 1–6. http://dx.doi.org/10.1109/ICCE.2019.8662011.

Prakash, R.M., Deepa, T., Gunasundari, T., Kasthuri, N., 2017. Gesture recognition and finger tip detection for human computer interaction. In: 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). IEEE, pp. 1–4.

Prakash, J., Gautam, U.K., 2019. Hand gesture recognition. Int. J. Recent Technol. Eng. 7 (6).

Pun, C.-M., Zhu, H.-M., Feng, W., 2011. Real-time hand gesture recognition using motion tracking. Int. J. Comput. Intell. Syst. 4 (2), 277–286.

Raheja, J., Minhas, M., Prashanth, D., Shah, T., Chaudhary, A., 2015. Robust gesture recognition using kinect: A comparison between DTW and HMM. Optik 126 (11–12), 1098–1104.

Raj, B., Kalgaonkar, K., Harrison, C., Dietz, P., 2012. Ultrasonic Doppler sensing in HCI. IEEE Pervasive Comput. 11 (2), 24–29. http://dx.doi.org/10.1109/MPRV.2012.17.

Rajesh, R.J., Nagarjunan, D., Arunachalam, R., Aarthi, R., 2012. Distance transform based hand gestures recognition for PowerPoint presentation navigation. Adv. Comput. 3 (3), 41.

Redmon, J., Farhadi, A., 2018. YOLOv3: An incremental improvement. CoRR, arXiv:1804.02767 URL http://arxiv.org/abs/1804.02767.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NIPS).

Ren, Z., Meng, J., Yuan, J., 2011. Depth camera based hand gesture recognition and its applications in human-computer-interaction. In: 2011 8th International Conference on Information, Communications & Signal Processing. IEEE, pp. 1–5.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520.

Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural Netw. 61, 85–117.

Simon, T., Joo, H., Matthews, I., Sheikh, Y., 2017. Hand keypoint detection in single images using multiview bootstrapping. In: CVPR.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. pp. 1–10. http://dx.doi.org/10.1016/j.infsof.2008.09.005, arXiv:1409.1556, URL http://arxiv.org/abs/1409.1556,

Sitaula, C., Aryal, S., Xiang, Y., Basnet, A., Lu, X., 2021a. Content and context features for scene image representation. Knowl.-Based Syst. 232, 107470. http://dx.doi.org/10.1016/j.knosys.2021.107470, URL https://www.sciencedirect.com/science/article/pii/S0950705121007322.

Sitaula, C., Xiang, Y., Aryal, S., Lu, X., 2021b. Scene image representation by foreground, background and hybrid features. Expert Syst. Appl. 182, 115285. http://dx.doi.org/10.1016/j.eswa.2021.115285.

Sonkusare, J.S., Chopade, N.B., Sor, R., Tade, S.L., 2015. A review on hand gesture recognition system. In: 2015 International Conference on Computing Communication Control and Automation. IEEE, pp. 790–794.

2021. Supervision for google cardboard. Advanced magnifier for the visually impaired based on google cardboard. Accessed: 6/4/2021, http://supervisioncardboard.com.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2015. Rethinking the inception architecture for computer vision. CoRR, abs/1512.00567 URL http://arxiv.org/abs/1512.00567.

Tan, M., Le, Q., 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (Eds.), Proceedings of the 36th International Conference on Machine Learning, Vol. 97. In: Proceedings of Machine Learning Research, PMLR, pp. 6105–6114.

Tang, H., Liu, H., Xiao, W., Sebe, N., 2019. Fast and robust dynamic hand gesture recognition via key frames extraction and feature fusion. arXiv:1901.04622.

Tanti, M., Gatt, A., Camilleri, K.P., 2018. Where to put the image in an image caption generator. Nat. Lang. Eng. 24 (3), 467–489.

Tekin, B., Bogo, F., Pollefeys, M., 2019. H+ o: Unified egocentric recognition of 3d hand-object poses and interactions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4511–4520.

Šarić, M., 2011. LibHand: A library for hand articulation. Accessed: 6/4/2021, http://www.libhand.org/.

Wang, X., Bao, H., 2007. Gesture recognition based on adaptive genetic algorithm. J. Comput.-Aided Des. Comput. Graph. 19 (8), 1056–1062.

Wang, R.Y., Popović, J., 2009. Real-time hand-tracking with a color glove. ACM Trans. Graph. 28 (3), 1–8.

Xi, C., Chen, J., Zhao, C., Pei, Q., Liu, L., 2018. Real-time Hand Tracking Using Kinect. In: Proceedings of the 2nd International Conference on Digital Signal Processing, pp. 37–42.

Xiao, Z., Xu, X., Zhang, H., Szczerbicki, E., 2021a. A new multi-process collaborative architecture for time series classification. Knowl.-Based Syst. 220, 106934. http://dx.doi.org/10.1016/j.knosys.2021.106934, URL https://www.sciencedirect.com/science/article/pii/S0950705121001970.

Xiao, Z., Zhao, Y., Li, N., Zhou, S., Xu, H., et al., 2021b. Research on key technologies of hand function rehabilitation training evaluation system based on leap motion. J. Comput. Commun. 9 (01), 19.

Xie, C., Li, C., Zhang, B., Chen, C., Han, J., 2017. Deep Fisher discriminant learning for mobile hand gesture recognition. arXiv:1707.03692.

Yosinski, J., Clune, J., Bengio, Y., Lipson, H., 2014. How transferable are features in deep neural networks? In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems (NIPS). pp. 3320–3328.

You, Q., Jin, H., Wang, Z., Fang, C., Luo, J., 2016. Image captioning with semantic attention. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4651–4659.

Zeiler, M.D., 2012. ADADELTA: an adaptive learning rate method. CoRR, abs/1212.5701 arXiv:1212.5701 URL http://arxiv.org/abs/1212.5701.

Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., Grundmann, M., 2020a. MediaPipe hands: On-device real-time hand tracking. arXiv:2006.10214.

Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., Grundmann, M., 2020b. Mediapipe hands: On-device real-time hand tracking. arXiv preprint arXiv:2006.10214.

Zhang, H., Xiao, Z., Wang, J., Li, F., Szczerbicki, E., 2020c. A novel IoT-perceptive human activity recognition (HAR) approach using multihead convolutional attention. IEEE Internet Things J. 7 (2), 1072–1080. http://dx.doi.org/10.1109/JIOT.2019.2949715.