TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 23, No. 2, April 2025, pp. 349∼370
ISSN: 1693-6930, DOI: 10.12928/TELKOMNIKA.v23i2.26455

Improving visual perception through technology: a comparative analysis of real-time visual aid systems

Othmane Sebban[1], Ahmed Azough[2], Mohamed Lamrini[1]
[1]Laboratory of Applied Physics, Informatics and Statistics (LPAIS), Faculty of Sciences Dhar El Mahraz, Sidi Mohamed Ben Abdellah University, Fez, Morocco
[2]De Vinci Higher Education, De Vinci Research Center, Paris, France

Article Info

Article history:

Received Jul 7, 2024
Revised Jan 16, 2025
Accepted Jan 23, 2025

Keywords:

Accessibility
Assistive technology
Benchmarking
Deep learning
Point of interest detection
Visually impaired

ABSTRACT

Visually impaired individuals continue to face barriers in accessing reading and listening resources. To address these challenges, we present a comparative analysis of cutting-edge technological solutions designed to assist people with visual impairments by providing relevant feedback and effective support. Our study examines various models leveraging InceptionV3 and V4 architectures, long short-term memory (LSTM) and gated recurrent unit (GRU) decoders, and datasets such as Microsoft Common Objects in Context (MSCOCO) 2017. Additionally, we explore the integration of optical character recognition (OCR), translation tools, and image detection techniques, including scale-invariant feature transform (SIFT), speeded-up robust features (SURF), oriented FAST and rotated BRIEF (ORB), and binary robust invariant scalable keypoints (BRISK). Through this analysis, we highlight the advancements and potential of assistive technologies. To assess these solutions, we have implemented a rigorous benchmarking framework evaluating accuracy, usability, response time, robustness, and generalizability. Furthermore, we investigate mobile integration strategies for real-time practical applications. As part of this effort, we have developed a mobile application incorporating features such as automatic captioning, OCR-based text recognition, translation, and text-to-audio conversion, enhancing the daily experiences of visually impaired users. Our research focuses on system efficiency, user accessibility, and potential improvements, paving the way for future innovations in assistive technology.

Corresponding Author:

Othmane Sebban
Laboratory of Applied Physics, Informatics and Statistics (LPAIS), Faculty of Sciences Dhar El Mahraz
Sidi Mohamed Ben Abdellah University
Fez 30003, Morocco
Email: othmane.sebban@usmba.ac.ma

1. INTRODUCTION

Visually impaired people [1] face many difficulties in their daily activities, including navigating unfamiliar environments, reading text, and interpreting visual information [2]. Although assistive technologies such as Microsoft Seeing AI and Google Lookout have been developed to analyze images in real-time and provide descriptions, these solutions still have significant limitations, including high costs, restricted functionality, and dependence on a stable internet connection. These constraints considerably reduce their effectiveness, particularly in critical situations such as crossing streets or reading important documents in real-time. Despite technological advancements, assistive technologies for the visually impaired struggle to deliver real-time performance because of computational demands and limited adaptability to real-world environments. Their dependence on the internet reduces their offline effectiveness, compromising the immediate and reliable assistance users need for essential tasks such as crossing streets or reading important documents.

To address these limitations, our study proposes a comprehensive benchmarking system designed to evaluate and optimize the performance of assistive technologies for visually impaired users. This system measures the effectiveness of various components, including image captioning, optical character recognition (OCR), real-time translation, and key image element detection [3]. Our goal is to encourage the development of mobile applications [4] that combine both accuracy and speed, ensuring optimal performance in real-time use cases. The mobile application we developed, "SeeAround," integrates these functionalities to provide reliable visual assistance [5]. The general diagram of our system, presented in Figure 1, outlines the various modules and their interactions, illustrating how they work together to offer real-time assistance.

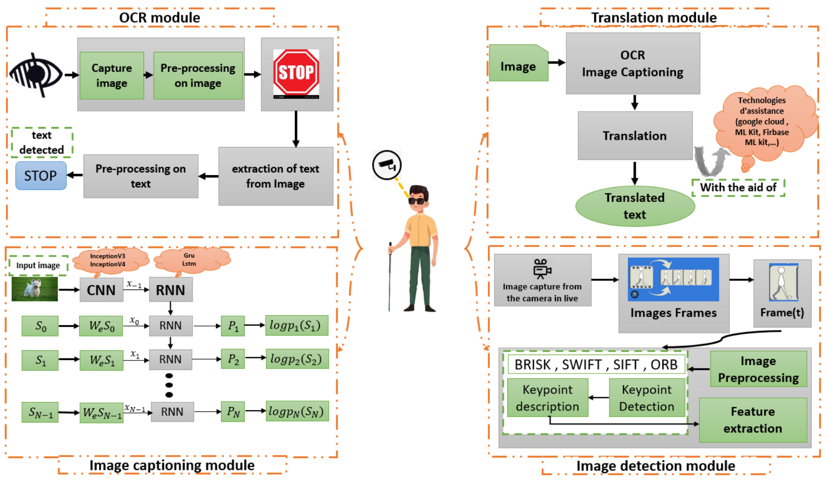


Figure 1. General diagram of the real-time visual assistance system for the visually impaired

For image captioning, we use an encoder-decoder architecture combining convolutional neural networks (CNN) and recurrent neural networks (RNN). Specifically, we employ InceptionV3 [6] and InceptionV4 [7] models, adapted to process images efficiently in real-world contexts for visually impaired people. Additionally, we use the Microsoft Common Objects in Context (MSCOCO) 2017 dataset to train the models with enhanced parameters, optimizing them for mobile environments. Our system also integrates long short-term memory (LSTM) and gated recurrent unit (GRU) decoders to capture temporal sequences more effectively, improving the generation of image captions by modeling long-term relationships between visual and textual elements [8].

The OCR component processes images containing mainly text, even in visually complex environments. Using advanced algorithms, it accurately detects and extracts text, providing contextual information essential for visually impaired users. Furthermore, the real-time translation functionality of our mobile application removes language barriers by supporting a wide range of languages [9]. Recognized text is translated into the chosen language and then converted into speech via our text-to-speech module [10], making it easier to understand image descriptions and textual content extracted by OCR.

An essential aspect of providing relevant visual information is precisely extracting key image elements during camera analysis [11]. Each image has different characteristics such as saturation, brightness, contrast, and camera angle, meaning that uniform processing approaches can prove ineffective. To address this, we have incorporated advanced image detection algorithms, including scale-invariant feature transform (SIFT) [1], speeded-up robust features (SURF) [1], oriented features from accelerated segment test (FAST) and rotated BRIEF (ORB) [12], and binary robust invariant scalable keypoints (BRISK) [12]. These algorithms are renowned for their robustness and accuracy under difficult conditions. Our system detects the limitations of current methods and proposes improvements to ensure optimum performance, particularly in the real-life situations of visually impaired people.

This paper is structured into five main sections, each addressing a key aspect of the study. Section 2 provides a detailed overview of previous research and current technologies designed for visually impaired users. Section 3 focuses on our benchmarking system and the key components used in our application. Section 4 presents the experimental results and their analysis, demonstrating the impact of our technical choices. Finally, section 5 concludes by summarizing our findings, discussing the implications of this work, and suggesting avenues for future research and development in assistive technology.

2. RELATED WORK

Visual impairment, a common disability, presents different levels of severity. Assistive technologies are crucial in providing visual alternatives via various products, devices, software, and systems [4], [6]. Sandhya *et al.* [13] present an application that helps visually impaired people understand their environment using neural networks and natural language processing. The application generates textual descriptions of images captured by a camera and integrates an OCR module to read the text on signs and documents. The descriptions are then converted into audio, providing information in several languages such as Telugu, Hindi, and English. This application requires a system with an integrated GPU. Ganesan *et al.* [14] propose an innovative approach to facilitating access to printed content for the visually impaired, using CNNs and LSTMs to encode and decode information. The system integrates OCR to convert printed text into digital format and then into speech via a text-to-speech application programming interface (API), making the content accessible via voice reading. Bagrecha *et al.* [15] present ”VirtualEye,” an innovative application in assistive technologies for the visually impaired, offering functions such as object and distance detection, recognition of Indian banknotes, and OCR. The system provides voice instructions in English and Hindi, enhancing users' independence and improving their quality of life. Uslu *et al.* [16] aim to generate grammatically correct and semantically relevant captions for visual content via a personalized mobile app, improving accessibility, particularly for the visually impaired. Integrated with the "CaptionEye" Android app, the system enables captions to be generated offline and controlled by voice, offering a user-friendly interface. Çaylı *et al.* [17] present a captioning system designed to provide natural language descriptions of visual scenes, improving accessibility and reducing social isolation for visually impaired people. This research demonstrates the practical application of computer vision and natural language processing to create assistive tools. Despite significant advances, it is crucial to continue developing reliable mobile systems adapted to everyday life to improve users' autonomy and quality of life.

3. METHOD

In this section, we present our solution based on a benchmarking analysis. Subsection 3.1 presents the initial module, detailing the process of benchmarking the modules illustrated in Figure 1 of our system. Subsection 3.2 explores the generation of image captions using encoder-decoder architectures optimized with TensorFlow Lite, integrating multi-GRU and LSTM models for accurate descriptions. Finally, subsection 3.3 compares the performance of four separate systems using various models for image captioning, OCR, translation, and key point extraction.

3.1. Description of the benchmarking process for the evaluation of visual assistance systems

Benchmarking measures a company's performance against market leaders [18], [19] to identify gaps and drive continuous improvement. In this study, realistic tasks adapted to the needs of visually impaired people, such as image caption generation, text recognition, and key point extraction, were designed. By comparing our solutions with industry standards, the aim is to close performance gaps and improve assistive technologies.

3.1.1. Benchmarking criteria and methodology

To improve accessibility and comprehension of multimedia content for visually impaired users, we have integrated automatic subtitling, OCR, text translation, and image recognition modules. These components use advanced machine-learning algorithms to optimize processing accuracy and speed, ensuring fluid, instantaneous interaction. Every component has been designed for a smooth, optimized experience.

3.1.2. Performance evaluation of visual assistance systems

We evaluated each visual assistance system according to four key criteria: accuracy, response time, robustness, and generalizability. Accuracy was measured by comparing the results obtained with expectations for tasks such as image description, OCR, and translation. Response time, expressed in milliseconds, was used to assess system efficiency. Robustness was analyzed under difficult conditions, including low lighting, high noise, and complex backgrounds, to ensure reliability. Finally, generalizability was examined using unpublished images, videos, and documents to judge its suitability for new contexts.

3.1.3. Comparative analysis of benchmark results

We analyzed the results to determine the best-performing systems for each task and usage scenario, emphasizing their strengths and weaknesses. This rigorous comparative analysis identified the most effective real-time visual assistance solutions, providing valuable insights into their capabilities. These findings will help guide the future development of more advanced, efficient, and user-friendly technologies tailored to the needs of visually impaired individuals.

3.2. Optimization of automatic image caption generation

This subsection presents a system for automatically generating image captions, based on an encoder-decoder model. The CNNs InceptionV3 and InceptionV4 are used to extract visual features as encoders. The multilayer decoder, composed of GRU and LSTM, generates the semantic captions, as illustrated in Figure 2. The work mentioned in [6], [7] combines CNN and recurrent networks, but the excessive increase in the number of time steps, due to the length of the legends, led to inferior performance. By reducing this number, we optimized the use of GRU and LSTM, leading to better results.

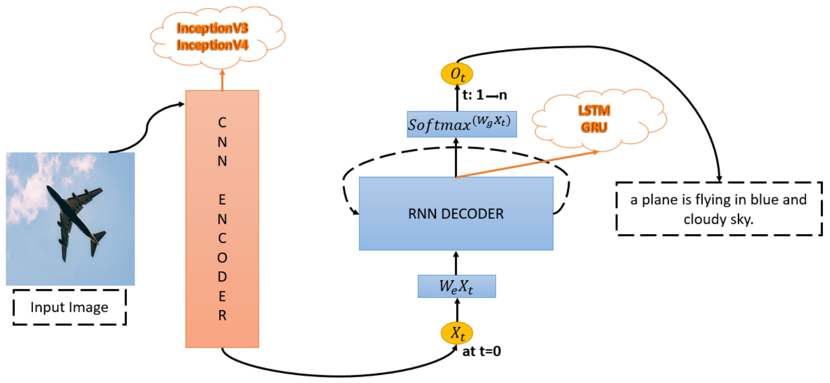


Figure 2. Model architecture for multi-RNN-based automated image captioning

3.2.1. InceptionV3-gated recurrent unit-based multi-layer image caption generator model

The image captioning system consists of two main elements: the encoder and the decoder, each based on a distinct neural architecture. The encoder, based on InceptionV3 [6], extracts key information from the image. This is then passed on to the decoder, which uses a GRU to generate the caption word by word. The proposed general model is illustrated in Figure 3.

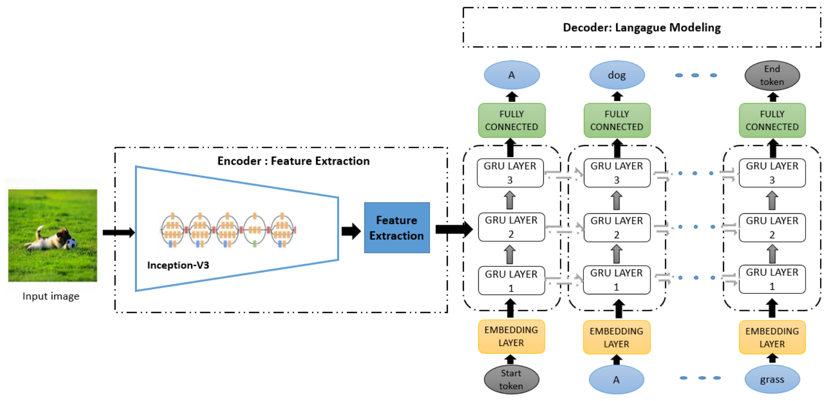


Figure 3. Flowchart of the InceptionV3 multi-layer GRU image caption generator

Encoder-InceptionV3 architecture: the encoder, based on a CNN, is composed of convolution, pooling, and connection layers. Sparse interactions identify key visual elements, while parameter sharing enables the same kernel to be applied for optimized learning. Before training, a 2048-dimensional vector is generated from the images via the mean pooling layer of the Inception-v3 model [6]. During training, a dense layer [20] refines this vector, progressively compressing it into a more compact and discriminative representation for image captioning.

Decoder-GRU implementation: the decoder exploits the extracted features to generate descriptive sentences, relying on RNNs to store part of the input data. However, RNNs face limitations due to gradient fading and explosion problems, compromising their ability to handle long-term dependencies. To overcome these difficulties, we use GRUs, an enhanced version of RNNs with control mechanisms [20] facilitating dependency management. Figure 4 shows a typical GRU architecture with its update, reset, and hidden state gates. The decoder comprises an integration level, a multilayer GRU, and a linear layer. The integration level converts words into vectors suitable for language modeling, while the GRU adjusts the hidden state using its gate mechanisms.

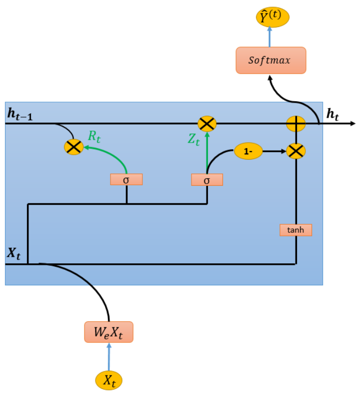


Figure 4. Architecture of GRU

In these (1)-(4) [21], x_t represents the input, and h_t is the hidden state at time t. The weights associated with the reset, update, and new information creation gates are denoted as W_r, W_z, and W_u, respectively. The hyperbolic tangent and sigmoid activation functions are symbolized by $\tanh$ and σ, respectively.

$$r_t = \sigma(W_{xr}x_t + U_{hr}h_{t-1}) \tag{1}$$

$$z_t = \sigma(W_{xz}x_t + U_{hz}h_{t-1}) \tag{2}$$

$$u_t = \tanh(W_{xu}x_t + U_{hu}(r_t \odot h_{t-1})) \tag{3}$$

$$h_t = (1 - z_t)h_{t-1} + z_t u_t \tag{4}$$

3.2.2. InceptionV4-long short-term memory-based multi-layer image caption generator model

The model follows an encoder-decoder approach, where the InceptionV4 [7] acts as an encoder to extract visual features from images. These features are then transmitted to a recurrent neural network equipped with LSTM cells that act as decoders. The latter uses this information to generate sequences of words, thus producing descriptive captions for the images. The general scheme of the model is illustrated in Figure 5.

Encoder-InceptionV4 architecture: we use InceptionV4, a CNN pre-trained by Google, as the encoder in our framework. This model extracts high-level visual features through deep convolutional layers. The InceptionV4-based encoder [7], [22], [23] converts raw images into fixed-length vectors by capturing relevant information from the intermediate pooling layer, just before the final output. This process provides a concise and relevant image representation for subsequent processing.

Decoder-LSTM implementation: the decoder is a deep recurrent neural network with LSTM cells , as shown in Figure 6. In our model, the decoder operates in two phases: learning and inference. During learning, the RNN decoder with LSTM cells aims to maximize the probability of each word in a caption based on the convoluted features of the image and previously generated words [7].

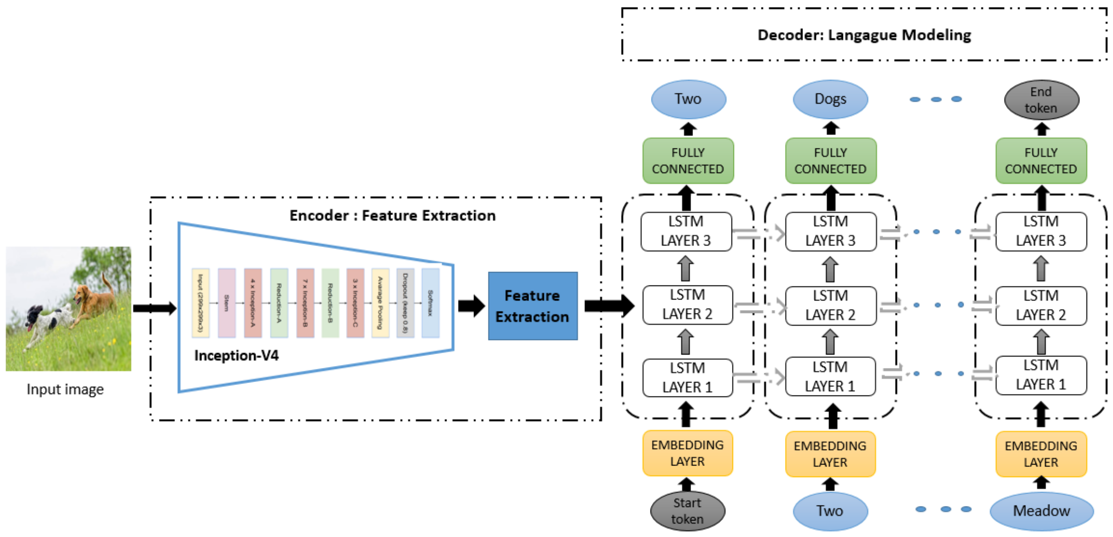


Figure 5. Flowchart of the InceptionV4 multi-layer LSTM image caption generator

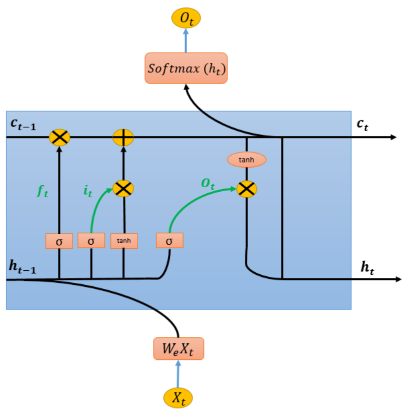


Figure 6. Architecture of LSTM

To learn a sentence of length N, the decoder loops back on itself for N time steps, storing previous information in its cell memory. The C_t memory is modified at each time step by the LSTM gates: the forget gate f_t, the input gate i_t, and the output gate o_t. The LSTM decoder learns the word sequences from the convolved features and the original caption. At step $t = 0$, the hidden state h_t of the decoder is initialized using these image features F. The main idea of the encoder-decoder model is illustrated by (5)-(11):

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{5}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{6}$$

$$\tilde{C}_t = \sigma\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{7}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{8}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{9}$$

$$h_t = o_t * \tanh(C_t) \tag{10}$$

$$O_t = \arg\max(\text{softmax}(h_t)) \tag{11}$$

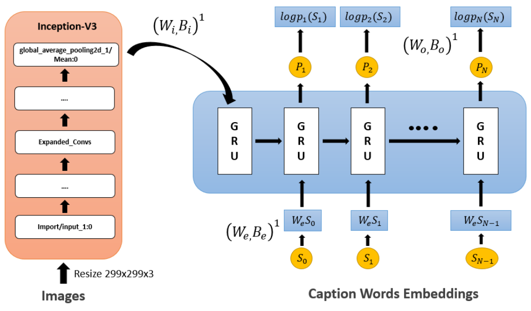
3.2.3. Training process and techniques for model optimization

The transformation of input x_t at time t into output word O_t is guided by equations using learnable weight and bias vectors (W_f, b_f), (W_i, b_i), (W_o, b_o), activated by sigmoid σ and hyperbolic tangent $\tanh$ functions [6], [7]. Each word X_t is converted into fixed-length vectors using a word representation W_e of dimension $V \times W$, where V is the number of words in the vocabulary and W is the length of the embedding learned during training. The decoder's objective is to maximize the probability p of a word's appearance at time t given the cell and hidden states, features F, and previous words $X_{t:0\rightarrow t}$. This is achieved by minimizing the loss function L, which is the cross-entropy of the sampled word probabilities [6], [7].

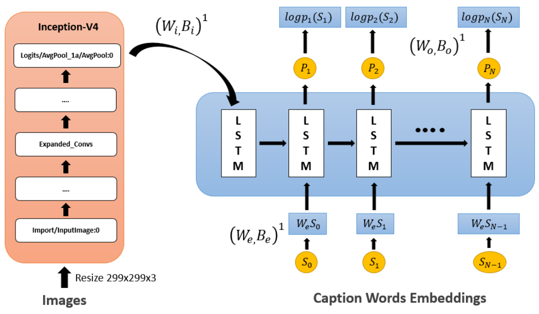
$$Z = \arg\max_{\beta} \left(\sum_{t=0}^{N} \log(p(O_t | X_{t:0\rightarrow t-1}, \phi_t; \beta)) \right) \tag{12}$$

$$L = H(u, v) = \min \left(\sum_{t=0}^{N} -u(X_t) \log(v(O_t)) \right) \tag{13}$$

where $H(u, v)$ is the cross entropy, u and v represent the softmax probability distributions of the ground truth word X_t and the generated word O_t at time t. During inference, the input image is passed through the encoder to obtain the convolved features, which are then sent to the decoder. At time $t = 0$, the decoder samples the start token $O_{t=0} = \langle S \rangle$ from the input features F. For subsequent instants, the decoder samples a new word based on the input features and previously sampled words $O_{t:0\rightarrow t}$ until it encounters an end token $\langle /S \rangle$ at instant $t = N$ [6], [7]. Figure 7 illustrates the backup architecture of the TensorFlow model for the LSTM and GRU encoder-decoders. Figure 7(a) shows our InceptionV3-GRU model, which uses a CNN to extract visual features and GRU units to generate captions. Figure 7(b) shows the architecture of the InceptionV4-LSTM model, where InceptionV4 extracts visual features and an LSTM generates captions.



(a)



(b)

Figure 7. The architecture for training phases of: (a) the InceptionV3-GRU model and (b) the InceptionV4-LSTM model

These diagrams show the architectures of models using InceptionV3 [6] and InceptionV4 [7] as encoders, with decoders based on GRU or LSTM units. The input image is resized and processed by convolutional layers, then features are extracted via global pooling to initialize the hidden state of the decoders. Each word in the legend is then converted to vectors and processed to generate the probability of the next word.

Table 1 summarizes the main parameters used to train the different image caption generation models, such as batch size, number of epochs, and time steps, which influence performance and quality. My new image caption generation model optimizes previous versions [6], [7]. Reducing time steps from 22 to 18 improves performance by reducing computational complexity. Increasing batch size to 148 stabilizes training while limiting captions to 16 words enhances efficiency.

Table 1. Pre-trained model settings for image caption generation

	Embedding size	Caption preprocessing	Error rate	Batch size	Num timesteps	Epochs
InceptionV4-LSTM [7]	256	20 words	2×10^{-3}	100	22	120
InceptionV4-LSTM (our model)	256	16 words	2×10^{-3}	148	18	120
InceptionV3-GRU [6]	256	20 words	2×10^{-3}	128	22	120
InceptionV3-GRU (our model)	256	16 words	2×10^{-3}	148	18	120

3.2.4. Common dataset utilization for enhanced performance

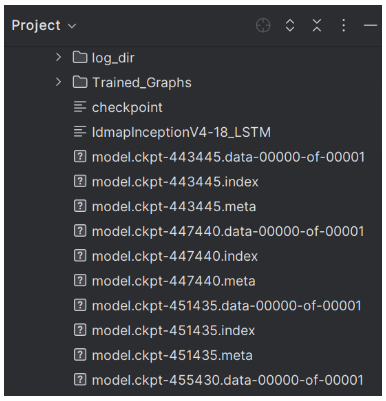
High-quality data is crucial for an effective model. Using diverse datasets helps avoid overfitting and improve performance. We used MSCOCO 2017 [20], which contains annotated images with five human captions. Table 2 compares MSCOCO 2017, Flickr 30k [24] and MSCOCO 2014 [25], highlighting the distribution of training, validation, and test sets, with MSCOCO 2017 offering the largest number of examples for image captioning.

Table 2. Characteristics of datasets used to train image caption generation models

Dataset	Training split (k)	Validation split (k)	Testing split (k)	Total images (k)
Flickr30k (imeca[6])	28	1	1	8
MSCOCO 2014 (cam2caption[7])	83	41	41	144
MSCOCO 2017 (Our model)	118	41	5	164

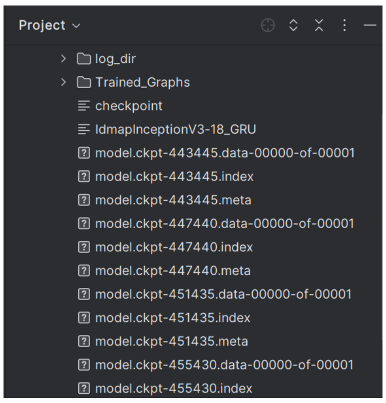
3.2.5. Integration of our pre-trained model in the mobile application

We are optimizing our encoding-decoding model for real-time use in the “SeeAround” mobile application via TensorFlow, exploiting its dataflow graph architecture [7] and processor parallelism to improve efficiency. Graph-based image preprocessing accelerates speed by a factor of six. During training, checkpoints and metadata files are generated regularly. Checkpoints store learned weights, while graph definitions link them, enabling the model to be reconstructed and reused for inference and training. Figure 8 shows the backup architecture for the LSTM and GRU models, with Figure 8(a) illustrating the LSTM model and Figure 8(b) the GRU model.



```
Project
  log_dir
  Trained_Graphs
  checkpoint
  ldmapInceptionV4-18_LSTM
  model.ckpt-443445.data-00000-of-00001
  model.ckpt-443445.index
  model.ckpt-443445.meta
  model.ckpt-447440.data-00000-of-00001
  model.ckpt-447440.index
  model.ckpt-447440.meta
  model.ckpt-451435.data-00000-of-00001
  model.ckpt-451435.index
  model.ckpt-451435.meta
  model.ckpt-455430.data-00000-of-00001
```

(a)

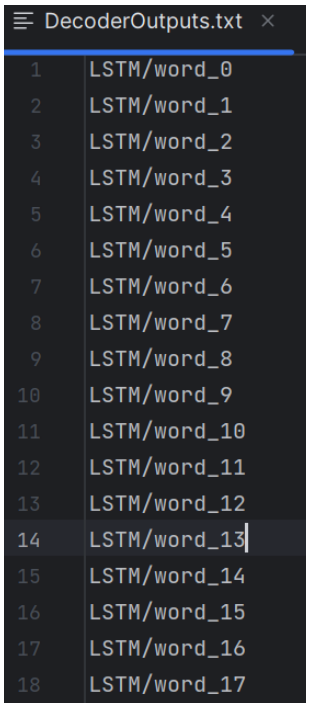


```
Project
  log_dir
  Trained_Graphs
  checkpoint
  ldmapInceptionV3-18_GRU
  model.ckpt-443445.data-00000-of-00001
  model.ckpt-443445.index
  model.ckpt-443445.meta
  model.ckpt-447440.data-00000-of-00001
  model.ckpt-447440.index
  model.ckpt-447440.meta
  model.ckpt-451435.data-00000-of-00001
  model.ckpt-451435.index
  model.ckpt-451435.meta
  model.ckpt-455430.data-00000-of-00001
  model.ckpt-455430.index
```

(b)

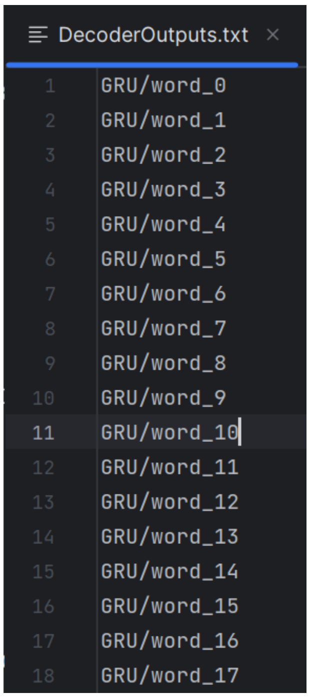
Figure 8. TensorFlow model backup architecture for: (a) the LSTM encoder-decoder and (b) the GRU encoder-decoder

We have combined the pre-processing, encoding, and decoding files into three ProtoBuf files to create an end-to-end model suited to static and real-time requirements. A final ProtoBuf file serves as a black box for subtitle generation. My model uses 18 words instead of 22 [6], [7], improving reliability and speeding up real-time subtitle generation for image streams from the camera. Figure 9 shows the captions generated by the LSTM and GRU decoding models. Figure 9(a) illustrates the captions generated by the LSTM model, while Figure 9(b) shows those generated by the GRU model.



```
DecoderOutputs.txt
LSTM/word_0
LSTM/word_1
LSTM/word_2
LSTM/word_3
LSTM/word_4
LSTM/word_5
LSTM/word_6
LSTM/word_7
LSTM/word_8
LSTM/word_9
LSTM/word_10
LSTM/word_11
LSTM/word_12
LSTM/word_13
LSTM/word_14
LSTM/word_15
LSTM/word_16
LSTM/word_17
```

(a)



```
DecoderOutputs.txt
GRU/word_0
GRU/word_1
GRU/word_2
GRU/word_3
GRU/word_4
GRU/word_5
GRU/word_6
GRU/word_7
GRU/word_8
GRU/word_9
GRU/word_10
GRU/word_11
GRU/word_12
GRU/word_13
GRU/word_14
GRU/word_15
GRU/word_16
GRU/word_17
```

(b)

Figure 9. captions generated by the model: (a) LSTM decoder outputs and (b) GRU decoder outputs

3.3. Detailed descriptions of our visual assistance systems

In this subsection, we present the four systems designed for image caption generation, OCR, machine translation, and key point extraction. Each system is built on specialized templates carefully selected for their efficiency and relevance. As illustrated in Table 3, these choices are guided by performance metrics and task-specific adaptability, ensuring optimal accuracy and reliability.

Table 3. Systems description for benchmarking evaluation

	Image captioning	OCR	Translation	Keyframe extraction
System 1	InceptionV4-LSTM with 22 words [7]	Google Mobile Vision API	Google Cloud Translation API	SIFT
System 2	InceptionV4-LSTM with 18 words (our model)	Firebase Vision Text Detector	Google ML kit	SURF
System 3	InceptionV3-GRU with 22 words [6]	Google Firebase Machine Learning kit	Firebase ML Kit	BRISK
System 4	InceptionV3-GRU with 18 words (our model)	TessTwo-Android	Google Translate API	ORB

3.3.1. Inception-V4 with LSTM and the harmony of advanced vision and language processing

Using the Google Mobile Vision API to recognize text in images: technological advances in information capture and text recognition have led to innovative services such as document analysis and secure access to devices. OCR [26], a technology for detecting and extracting text from scanned images or directly from the camera, can work with or without an internet connection. Google offers Mobile Vision, an open-source tool for creating text recognition and instant translation applications on Android. In this research, OCR is used to assist the visually impaired. Although this technology is effective for document scanning and text analysis, it encounters limitations in applications dependent on a stable Internet connection, particularly in areas with low connectivity. The extracted text data is then processed by a REST API [27], which interacts with a database and displays the information live on the device, as illustrated in Figure 10.

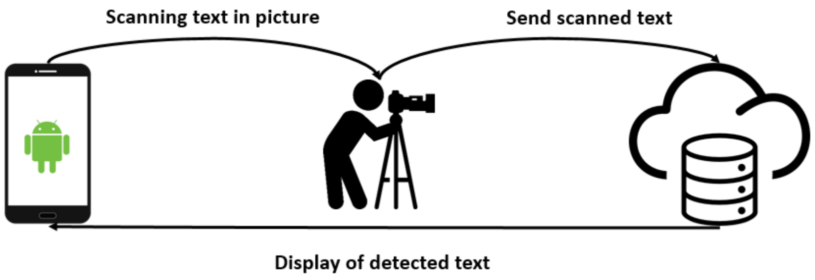


Figure 10. Using the Google Mobile Vision API for the OCR process

Google Cloud Translation to convert recognized text into multiple languages: The Google Cloud Platform offers pre-trained machine learning models for creating applications that interact with their environment [28]. Among these models, the Google Cloud Translation API offers the possibility of converting content between dozens of languages. We used this API to translate information, captioning, and OCR. Google Translate then renders this data in the language chosen by the visually impaired person. Figure 11 illustrates this workflow, from text recognition to captioning and OCR to automated translation, based on cloud technologies.

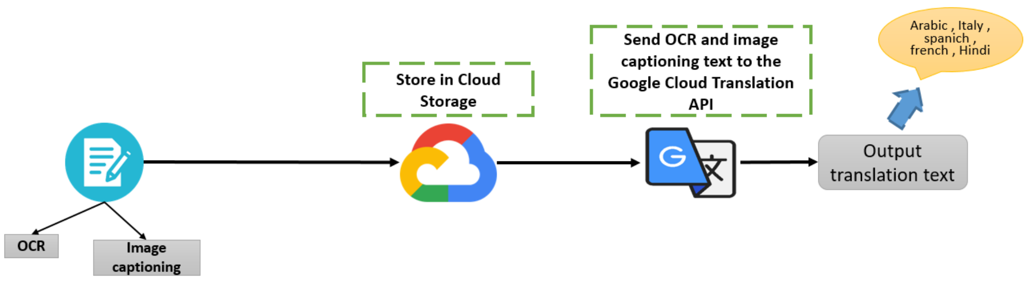

Store in Cloud Storage
Send OCR and image captioning text to the Google Cloud Translation API
Arabic , Italy , spanich , french , Hindi
Output translation text
OCR
Image captioning

Figure 11. Multilingual translation process with the Google Cloud Translation API

Detection keyframe with SIFT: the SIFT descriptor, designed by Lowe [29], is widely used for its efficiency in image processing, particularly for identifying and characterizing points of interest using local gradients. The SIFT process is divided into four main phases, including the application of the Gaussian difference (DoG) method. This involves subtracting images filtered by Gaussian filters applied at different scales. The extrema detected between two adjacent levels are then exploited for further analysis [30], [31]. As (14):

$$D(X, \sigma) = (G(X, k\sigma) - G(X, \sigma)) * I(X) \tag{14}$$

where I is the input image and X is a specific point $X(x, y)$. The variable σ represents the scale, while $G(X, \sigma)$ denotes the Gaussian applied to the point X. Regions of interest based on Gaussian differences (DoG) [31] are identified as extrema in the image plane and along the scale axis of the function $D(x, \sigma)$. To locate these points, the $D(x, \sigma)$ value of each point is compared with its neighbors at the same and different scales. The SIFT algorithm extracts and describes these points of interest for obstacle detection and recognition.

3.3.2. Inception-V4 with LSTM and firebase: text detection optimization

Firebase vision text detector for efficient text detection: Google's Firebase Cloud Storage service enables developers to store and share user content, such as photos, videos, and audio files, in the cloud [32]. Based on Google Cloud Storage, it offers a scalable object storage solution, perfectly integrated with web and mobile applications. In our Android application architecture, we use firebase vision for text detection, integrating all modules [33]. Only recognized documents undergo content extraction via Google Cloud's text recognition API. The application displays the extracted information in the user interface and stores it to avoid redundant processing, thus improving efficiency. The images are sent to the text recognition engine in the cloud, as illustrated in Figure 12.

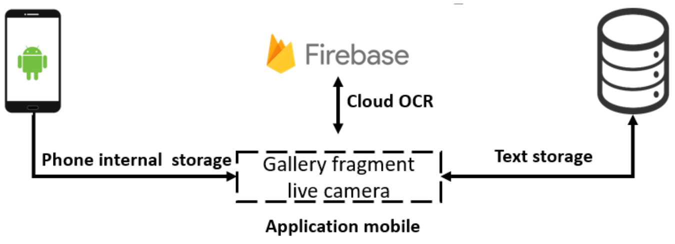


Figure 12. Mobile application architecture for text extraction and storage with firebase OCR

Google ML kit translates detected text: Google offers Firebase, which integrates ML-Kit, an SDK for mobile application development on Android and iOS. ML-Kit provides tools to simplify development, including functionality for translating text into English. The proposed algorithm integrates essential functions such as text detection, text-to-speech, and captioning in languages selected by visually impaired users. The application supports the main international languages [34] and can operate offline. Figure 13 shows the architecture of the algorithm, illustrating the process using Google ML Kit.

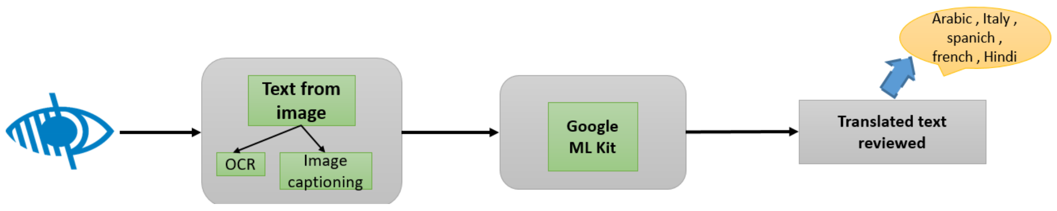


Figure 13. The architecture of the text translation system for the visually impaired using Google ML Kit

Detection keyframe with SURF: SURF performs very well in noisy environments with large variations between groups, remaining reliable even when matching distances are high and variable, especially for the best matches between key points [35]. Considered an improvement on SIFT, this algorithm detects and describes points of interest in images. It is optimized for speed while maintaining robustness to scale and rotation transformations. Bay *et al.* [30] developed the SURF (accelerated robust features) system in 2008 for rapid extraction of visual features. The SURF detector uses the Hessian matrix determinant and integral images to increase its efficiency. The standard descriptor is based on Haar wavelet responses and can be extended from 64 to 128 bins to better handle large perspective changes [35]. SURF consumes fewer computing resources than SIFT while maintaining high performance in noisy environments. In (15) describes the Hessian matrix at points $x = (x, y)$ and scale σ:

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{yx}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \tag{15}$$

where $L_{xx}(x, \sigma)$ is the convolution of the second-order Gaussian derivative with the image I at point x, and similarly for $L_{xy}(x, \sigma)$ and $L_{yy}(x, \sigma)$.

3.3.3. Inception-V3 with gated recurrent unit and prioritization for data optimization

Combined use of Google Firebase ML for recognition: the Firebase ML Kit is a versatile tool for developers, making it easy to integrate AI functions such as text recognition, face detection, and barcode reading into Android and iOS applications without requiring in-depth knowledge of machine learning. Our system will leverage this technology to perform real-time text recognition, seamlessly integrated into our application. Figure 14 illustrates these capabilities, showing how the ML Kit can perform these tasks both locally and in the cloud [36].

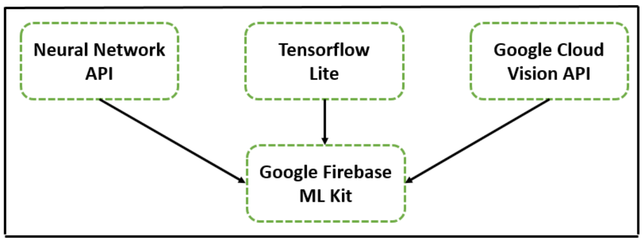


Figure 14. The architecture of the Google Firebase ML Kit integrates various machine learning technologies

Combined use of Google Firebase ML for text translation: Google's translation service, which supports around 108 languages, has simplified translation tasks. We're integrating this service with Firebase ML Kit [37], a powerful API for extracting text from images. Automatic language detection identifies the language of a speech sample. For real-time processing, a stand-alone translator quickly converts text detected in images [38]. The Firebase ML Kit automates this process, making information accessible to visually impaired people in their native language. This integration is particularly beneficial for tourists and people working in multilingual environments, as it overcomes language barriers, enabling personalized access to information, as illustrated in Figure 15.

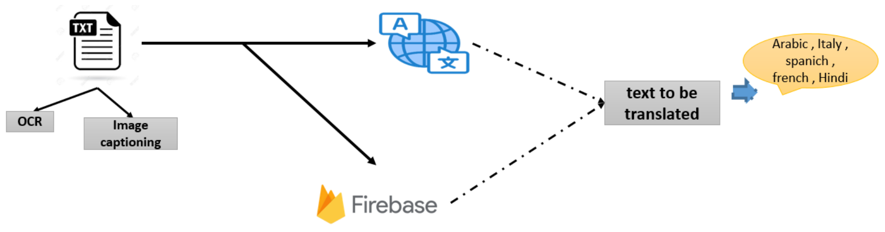


Figure 15. Integration of the Firebase ML Kit for machine translation of text from images

Detection keyframe with BRISK: in 2011, S. Leutenegger and his team developed the BRISK [39] algorithm, which relies on AGAST to detect the best corners using the FAST score. The BRISK descriptor, represented as a binary string, enables rapid comparisons and remains invariant to rotation and brightness while being stable against scale changes and moderate affine transformations [37]. Just as powerful as SIFT, BRISK requires fewer computing resources. This study [40] uses this algorithm to identify characteristic points, while histogram of oriented gradients (HOG) descriptors [41] are used to describe them, facilitating the identification of key images. HOG is particularly useful for object classification and detection, while the minutiae detected by BRISK form the basis for calculating HOG descriptors.

After their detection, the HOGs of the local areas around these points are calculated, forming an initial feature matrix. $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where each vector $\mathbf{x}_i$ represents a BRISK key point extracted from the sampled video images. The BRISK descriptor of a key point k is represented by a binary vector $\mathbf{D}_k$, constructed as (16):

$$D_k = [b_1, b_2, \ldots, b_n] \tag{16}$$

where each b_i is determined by comparing the intensity of the sampled points:

$$b_i = \begin{cases} 1 & \text{si } I(p_{a_i}) > I(p_{b_i}) \\ 0 & \text{sinon} \end{cases} \tag{17}$$

$I(p)$ represents the intensity at point p, and p_{a_i} and p_{b_i} are specific points in the pattern around the key point. BRISK is appreciated for its speed and efficiency, particularly in environments where computing resources are limited.

3.3.4. Inception-V3 with gated recurrent unit and towards enhanced robustness on Android

TessTwo a robust solution for Android is used for OCR: integrating OCR into Android applications requires an external library, as this functionality is not included natively. One commonly used option is Tesseract, in particular its "tess-two" variant, which provides a Java interface for Android [42]. Tesseract's OCR process begins by sending an image via an API, followed by pre-processing using tools like OpenCV to improve quality. In the OCR engine [42], the image is converted to black and white and then segmented to detect individual words. Each word is then analyzed to correct malformed or merged characters before these are recognized using specific linguistic data. With the help of pre-trained datasets and libraries such as Leptonica, Tesseract extracts the textual content, which is then transformed into usable formats such as plain text, making this information usable in various systems, as illustrated in Figure 16.

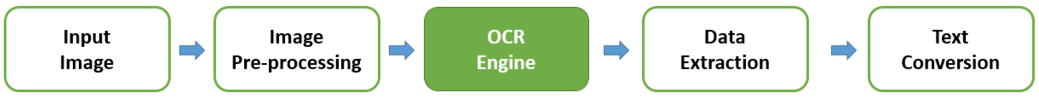


Figure 16. Tesseract "TessTwo-Android" OCR system workflow

The Translate API is used to translate the recognized text: the Google Translate API supports over 100 languages and various media types. Accessible to websites and mobile applications, it facilitates the conversion of texts, sites, documents, and speech from one language to another. Translation can be literal, preserving the structure of the source language, or semantic, favoring a more natural transmission of meaning in the target language [43]. We have integrated this API into our OCR and image captioning system to adapt English texts into native languages such as Arabic, Italian, Spanish, French, and Hindi, making information more accessible to the visually impaired. Using the Google Translate online service simplifies and enhances this process, as illustrated in Figure 17.

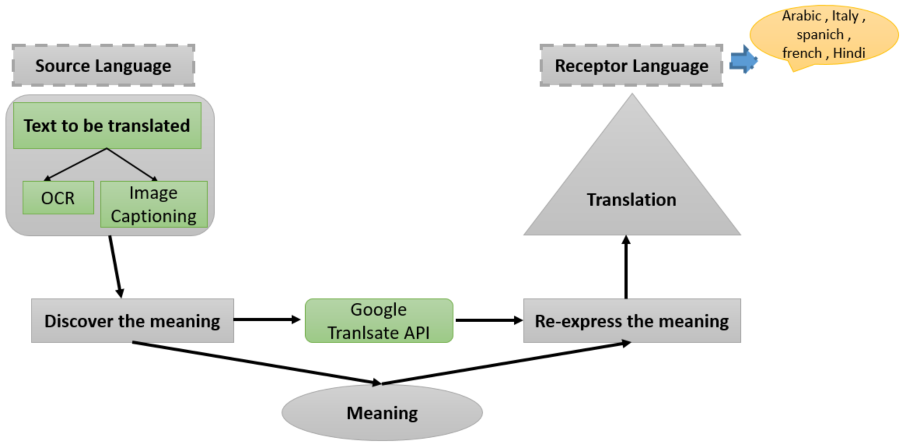


Figure 17. Automatic API translation from recognized text

Detection keyframe with ORB: this algorithm, developed by OpenCV, combines the FAST feature detector with the BRIEF descriptor [44], adding orientation management to ensure rotation invariance. ORB thus improves the quality of keypoint detection and description by using FAST to locate points of interest and then selecting the most relevant using the Harris index. The orientation of each point is calculated relative to the center of the patch to correct for rotation-related problems. Three main equations describe how the ORB method works: [45], [46].

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{18}$$

This equation is used to calculate the spatial order moments of the image, where $I(x,y)$ is the intensity at the point (x,y), and p and q are the orders of the moments along the x and y axes, respectively.

3.3.5. Performance of SeeAround visual assistance systems with key point detection and optical character recognition

We present the performance of our “SeeAround” mobile application, which integrates key point detection and OCR algorithms. Figure 18 compares the performance of four point-of-interest detection algorithms, essential for real-time image analysis in visual assistance systems. Figure 18(a) shows results obtained with the SIFT algorithm, while Figure 18(b) uses SURF. Detection by BRISK is illustrated in Figure 18(c), and finally, the ORB algorithm is shown in Figure 18(d).

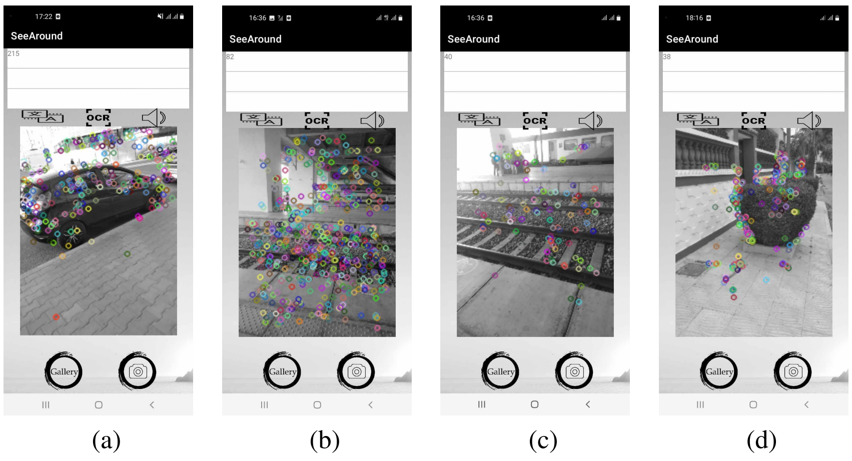


Figure 18. Keyframe extraction process with keypoint detection: (a) SIFT, (b) SURF, (c) BRISK, and (d) ORB

Figure 19 shows OCR results using different methods. Figure 19(a) illustrates the performance of the Google Mobile Vision API, while Figure 19(b) displays that of the Firebase Vision text detector. The results of the Google Firebase Machine Learning Kit are shown in Figure 19(c), and finally, Figure 19(d) shows the performance of TessTwo-Android.

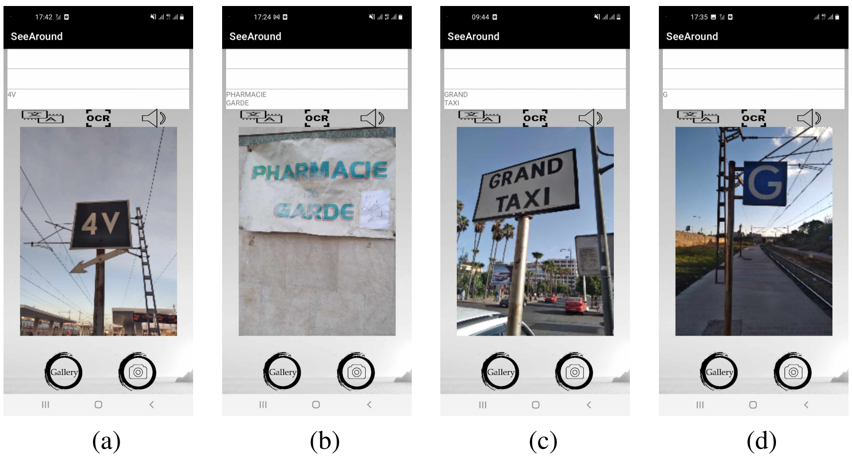


Figure 19. OCR results demonstrating different methods: (a) Google Mobile Vision API, (b) Firebase Vision Text Detector, (c) Google Firebase machine learning Kit, and (d) TessTwo-Android

4. RESULT AND DISCUSSION

The results and discussion section presents the overall results of our system, generated by its various modules. The evaluation criteria are detailed in subsection 4.1, which describes the main modules of our system. Subsection 4.2 presents the results obtained with different pre-trained subtitle generation models, as well as the OCR, translation, and image detection modules, in a summary table. Finally, subsection 4.3 discusses the performance, speed, and robustness of our system.

4.1. The evaluation metrics

To assess the quality of the pre-trained model and modules such as OCR, translation, and image detection, we use several evaluation measures, including the bilingual evaluation understudy (BLEU) score. This widely used measure analyzes the quality of generated sentences by comparing n-gram co-occurrences between generated (candidate) and reference sentences [47]. BLEU-N, a precise indicator for short sentences, evaluates the proportion of n-grams (up to 4-word sequences) shared between a candidate sentence and one or more reference sentences.

$$\text{BLUE} - N(ci, Si) = b(ci, Si) \exp\left(\sum_{n=1}^{N} \omega_n \log P_n(ci, Si)\right) \tag{19}$$

where

$$b(ci, Si) = \begin{cases} 1 & \text{if } lc > ls \\ e^{1-\frac{ls}{lc}} & \text{if } lc \leq ls \end{cases} \text{ is a brief penalty; lc is}$$

The total length of candidate sentences c_i, l_S, is the effective reference length at the corpus level. When multiple references are available for a candidate phrase, the closest reference length is chosen. The value of ω_n is generally kept constant, as mentioned in [47].

$$\text{For all } n; N = 1, 2, 3, 4 : \quad P_n(ci, Si) = \frac{\sum_k \min(h_k(ci), \max(h_k(S_{ij})))_{j \in \mathcal{M}}}{\sum_k h_k(ci)}$$

The ROUGE-L metric is based on the longest common sequence (LCS). Taking the length $l(c_i, s_{ij})$ of the LCS between a pair of sentences, this metric is defined as (20) [48]:

$$ROUGEL(ci, Si) = \frac{(1+\beta^2) Rl Pl}{Rl + \beta^2 Pl} \tag{20}$$

where

$$Rl = \max_j \left(\frac{l(ci; S_{ij})}{|S_{ij}|}\right) \text{ stands for the recall of LCS} Pl = \max_j \left(\frac{l(ci; S_{ij})}{|Ci|}\right)$$

It represents the precision of the LCS, while E is a constant generally adjusted to favor recall. In addition to ROUGE-L, there are other measures such as ROUGE-N and ROUGE-S, as mentioned in [48].

SPICE, developed by Anderson and colleagues [49], uses a semantic representation to encode objects, attributes, and relationships in a description, allowing evaluation at the semantic level. SPICE converts candidate and reference legends into syntactic dependency trees using a dependency parser [50]. Let's assume that c is a candidate and that S designates a set of reference captions. The candidate's scene graph is denoted $G(c) = \langle O(c), E(c), K(c) \rangle$, while that of the reference captions is denoted $G(S)$. The operation $T(\cdot)$ converts a scene graph into a collection of tuples in the form $T(G(c))) \Leftrightarrow O(c) \cup E(c) \cup K(c)$. Consequently, precision and recall can be expressed as (21) and (22):

$$P(c, S) = \frac{|T(G(c)) \otimes T(G(S))|}{|T(G(c))|} \tag{21}$$

$$R(c, S) = \frac{|T(G(c)) \otimes T(G(S))|}{|T(G(S))|} \tag{22}$$

Consequently, the calculation of the SPICE metric can be expressed as (23):

$$SPICE(c,S) = F_1(c,S) = \frac{2 \cdot P(c,S) \cdot R(c,S)}{P(c,S) + R(c,S)} \tag{23}$$

The symbol $\otimes$ is the binary matching operator, which returns matching tuples. Although SPICE provides a better evaluation of semantic information, it neglects grammatical requirements and cannot judge the natural fluency of sentences. Additionally, since the evaluation focuses mainly on noun similarity, it is not suitable for tasks such as machine translation [50].

The performance of the module's OCR and translation processes is evaluated according to their accuracy. For OCR, this corresponds to the ability to correctly recognize characters in digitized text. For translation, the evaluation focuses on the faithfulness of the conversion of words from one language to another while preserving meaning and context. The evaluation criteria for each feature are as (24) and (25):

$$\text{Accuracy for OCR } = \frac{\text{Number. of correctly identified characters}}{\text{Total number. of characters}} \times 100(\%) \tag{24}$$

$$\text{Accuracy for translation } = \frac{\text{Number. of correctly translated words}}{\text{Total number. of words}} \times 100(\%) \tag{25}$$

To evaluate the performance of the BRISK, SIFT, ORB, and SURF algorithms, we use variable values and weights. However, for the sake of simplicity, we focus on two main criteria: processing speed and number of features detected. A simplified version of the equation is thus proposed:

$$\text{Score}_{\text{alg}} = V_{\text{speed}} + N_{\text{features}} + A_{\text{accuracy}} \tag{26}$$

where V_{speed} represents the processing speed of the algorithm, $N_{features}$ indicates the number of features detected by the algorithm, and $A_{accuracy}$ designates the accuracy of feature matching. Match accuracy is calculated as the ratio between the number of correct matches and the total number of matches, as (27):

$$\text{Precision} = \frac{\text{Number of correct matches}}{\text{Total number of correspondences}} \tag{27}$$

4.2. Results

To present the results of our study, we analyzed the evolution of learning rates and losses during training for two distinct models: InceptionV3-GRU and InceptionV4-LSTM. We proposed to reduce the number of trained words from 22 to 18, as previously suggested [6], [7], through the use of more advanced datasets, such as MSCOCO 2017. Figure 20 illustrates this evolution over 120 epochs. Figure 20(a) shows that the learning rate of the InceptionV3-GRU model gradually decreases from 0.002 to 0.0017 over 396,600 iterations, indicating a stabilization of the model. Similarly, Figure 20(b) shows a reduction in the rate for the InceptionV4-LSTM model, from 0.002 to 0.0016 over 479,400 iterations.

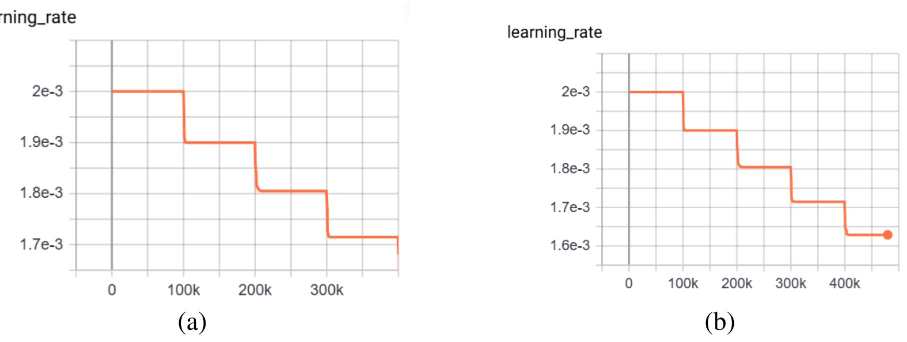


Figure 20. Evolution of learning rates over 120 epochs for: (a) the InceptionV3-GRU model and (b) the InceptionV4-LSTM model

Figure 21 illustrates the evolution of the loss function during model learning. Figure 21(a) shows a steady decrease in the loss for the InceptionV3-GRU model over around 396,600 iterations, indicating a continuous improvement in model performance. Similarly, Figure 21(b) shows a similar reduction in loss for the InceptionV4-LSTM model over around 479,400 iterations, confirming the latter's effectiveness.

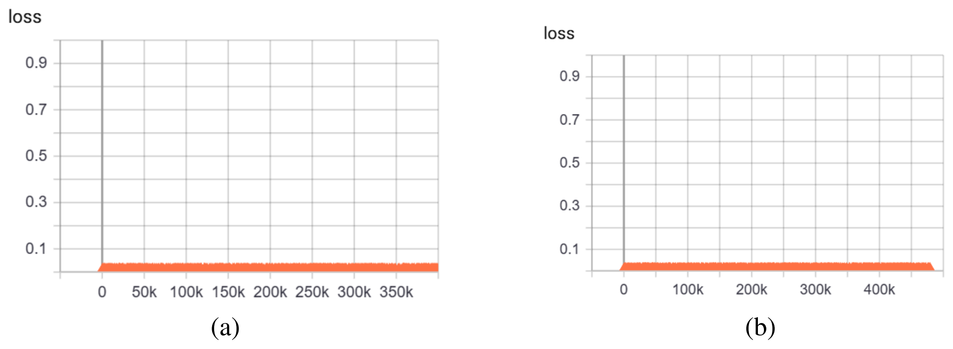


Figure 21. Evolution of learning loss over 120 epochs for: (a) the InceptionV3-GRU model and (b) the InceptionV4-LSTM model

A stable loss can indicate good generalization if it is low enough, but a value of 0.1 might be too high for our application, requiring adjustments. Loss stability was achieved using advanced parameters and the MSCOCO 2017 dataset. The effectiveness of our method was measured with the BLUE, ROUGE-L, and SPICE indicators. The tests carried out with three GRU-LSTM multilayer models using the Inception-V3 and Inception-V4 architectures are presented in Table 4, with the best scores in bold.

Table 4. Comparison of model performance with evaluation metrics

Model	B-1	B-2	B-3	B-4	ROUGE-L	SPICE
InceptionV3-LSTM with 22 words [6]	0.3884	0.1818	0.0769	0.03738	0.2971	0.0228
InceptionV3-GRU with 18 words (our model)	**0.558**	**0.220**	**0.084**	**0.0645**	**0.465**	**0.0396**
InceptionV4-LSTM with 22 words [7]	0.650	0.469	0.362	0.227	0.480	-
InceptionV4-GRU with 18 words (our model)	**0.701**	**0.589**	**0.4851**	**0.3518**	**0.576**	**0.0404**

Table 5 compares the algorithms' scores for speed, number of features detected, and matching accuracy with specific weightings. Two main measures are used to evaluate their performance: the number of features detected ($N_{features}$) and accuracy ($A_{accuracy}$), both normalized to maximum values. For example, for BRISK, ($N_{features}$) is 47, giving a normalized value of 0.198, and its precision of 0.72 is adjusted to 0.96 from the maximum of 0.75.

Table 5. Scores calculated for each image detection algorithm

Algorithm	$V_{\text{speed (seconds)}}$	N_{features}	$A_{\text{accuracy (\%)}}$	Score (%)
BRISK	**0.35**	47	72	57.9
ORB	0.49	51	**75**	60.8
SIFT	2.13	**237**	56.2	**87.5**
SURF	0.99	91	68	64.5

Table 6 shows the final results for each system, including the performance of the various modules: image captioning (see Table 4), OCR, translation, and keyframe extraction (see Table 5). These results are based on the analysis of some 230 images. Using the equation applied in Table 6, the overall accuracy of each system can be calculated as (28):

$$\text{Overall accuracy} = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{28}$$

where n is the total number of modules evaluated and P_i is the percentage accuracy of the ith module.

Table 6. System evaluation for captioning, OCR, translation, and keyframe extraction modules

	Image captioning (%)	OCR (%)	Translation (%)	Keyframe extraction (%)	Overall accuracy (%)
System 1	65.5	92.9	96.7	**87.5**	**85.65**
System 2	**70.1**	**98.7**	**97.1**	64.5	82.6
System 3	38.84	95.6	92	57.9	71.085
System 4	55.8	94.3	95.8	60.8	76.675

4.3. Discussion

The results in subsection 4.1 show that systems based on the InceptionV4-LSTM architecture offer improved performance, particularly in terms of accuracy and speed, for image caption generation. The optimization of recurrent neural networks and the reduction of time steps play a crucial role in these improvements, confirming the importance of adapting these models to real-time environments, particularly for visually impaired users. Of the four systems evaluated, system 1 (InceptionV4-LSTM with 22 words) uses SIFT, offering high accuracy (87.5%) but slower processing (2.13 s) with a BLEU-1 score of 65.5%. System 2 (InceptionV4-LSTM with 18 words) provides a good speed-accuracy balance, ideal for real-time applications, with 64.5% accuracy and a BLEU-1 score of 70.1%. System 3 (InceptionV3-GRU with 22 words) is fast at keyframe extraction (0.35s) but less accurate in captioning, with a BLEU-1 score of 38.84%. System 4 (InceptionV4-LSTM optimized with 18 words) is robust, offering strong performance in captioning (BLEU-1 of 55.8%) and decent keyframe accuracy (60.8%).

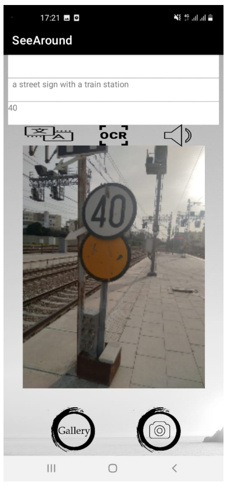
The results obtained show that the choice of system depends on specific needs in terms of speed or accuracy in different visual environments. For example, BLEU scores for system 4 reached 0.701 for BLEU-1 and 0.351 for BLEU-4, demonstrating a notable superiority in caption accuracy over other systems. These performances are in line with the need for a system capable of delivering reliable results in contexts where accuracy is paramount, such as in complex environments.

Unlike previous studies [13]-[17], which focus primarily on text recognition or image captioning, our approach integrates image caption generation, OCR, translation, as well as real-time image detection algorithms into a single mobile application. This combination of technologies offers more comprehensive and immediate assistance, surpassing existing systems in terms of versatility and accuracy, particularly in environments where Internet connectivity is limited. For example, system 1, with a score of 85.65% for all modules, demonstrated its ability to operate effectively in these conditions.

Although SIFT detected the greatest number of features, its slowness limits its use in real-time applications, where immediate feedback is critical for visually impaired users. This demonstrates the inherent trade-off between feature detection robustness and speed, which is a crucial factor in selecting the appropriate algorithm for assistive technologies. ORB offers a good compromise between speed and accuracy, making it more suitable for scenarios where quick responses are paramount, such as in outdoor environments or crowded urban spaces. However, the slight drop in accuracy observed with ORB may affect the reliability of the assistance in recognizing smaller or more detailed objects.

This study makes a significant contribution to the improvement of real-time visual assistance technologies. By combining several advanced modules into a single mobile application, we have not only improved the accuracy of visual descriptions and translations but also provided a more robust and versatile solution for visually impaired people. With an overall accuracy rate of 85.65% for system 1, these results pave the way for new assistive applications with great potential for adoption in everyday environments.

To improve accessibility for the visually impaired, the "SeeAround" solution integrates optical character recognition and real-time image captioning, as illustrated in Figure 22. In Figures 22(a)-(g), the process starts with image capture, followed by text detection and captioning. In Figures 22(b)-(h), the extracted text is translated to help visually impaired users understand the content in their native language. Finally, in Figures 22(c)-(i), translations are presented to the user in spoken form. Translations are presented to the user in audio form. This feature not only enhances user autonomy but also provides real-time assistance in a variety of environments.

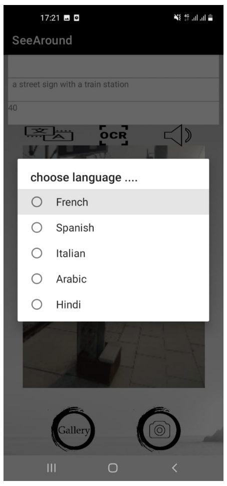

17:21

SeeAround

a street sign with a train station

40

(a)


17:21

SeeAround

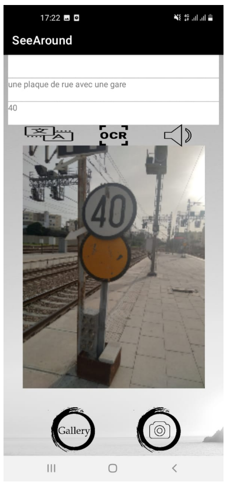
a street sign with a train station

40

choose language ....

- ○ French
- ○ Spanish
- ○ Italian
- ○ Arabic
- ○ Hindi

(b)

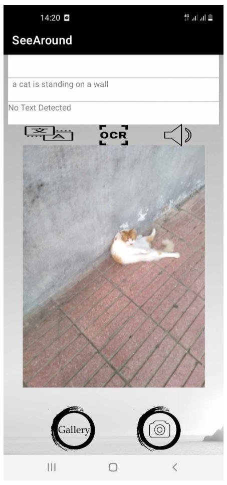

17:22

SeeAround

une plaque de rue avec une gare

40

(c)

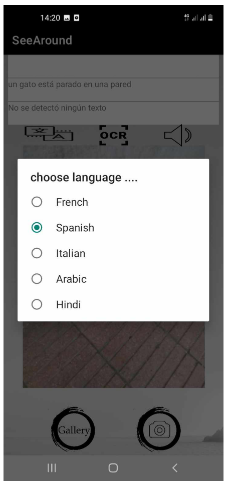

14:20

SeeAround

a cat is standing on a wall

No Text Detected

(d)


14:20

SeeAround

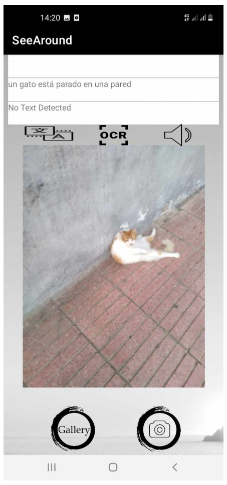
un gato está parado en una pared

No se detectó ningún texto

choose language ....

- ○ French
- [✓ Spanish]
- ○ Italian
- ○ Arabic
- ○ Hindi

(e)

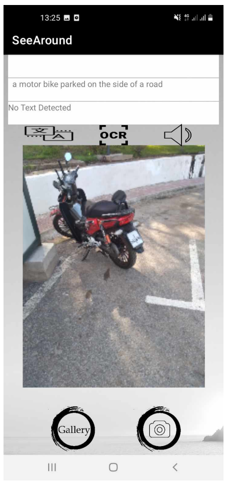

14:20

SeeAround

un gato está parado en una pared

No Text Detected

(f)

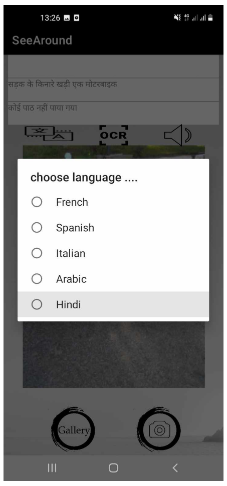

13:25

SeeAround

a motor bike parked on the side of a road

No Text Detected

(g)


13:26

SeeAround

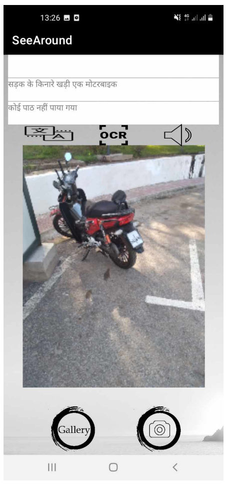
सड़क के किनारे खड़ी एक मोटरबाइक

कोई पाठ नहीं पाया गया

choose language ....

- ○ French
- ○ Spanish
- ○ Italian
- ○ Arabic
- ○ Hindi

(h)


13:26

SeeAround

सड़क के किनारे खड़ी एक मोटरबाइक

कोई पाठ नहीं पाया गया

(i)

Figure 22. The SeeAround process; (a)-(g) first column: OCR and generate caption, (b)-(h) second column: select language for translation, and (c)-(i) third column: OCR and generate caption with audio speech after translation

5. CONCLUSION

In this study, we evaluated real-time visual assistance systems for the visually impaired using high-performance architectures such as InceptionV3 and InceptionV4, as well as text recognition modules and image detection algorithms. The integration of these technologies into mobile applications has significantly improved users' autonomy and quality of life. Our tests revealed remarkable performance in terms of accuracy, speed, and adaptability to various environments. Our "SeeAround" mobile application integrates automatic captioning, OCR, and translation, drawing on services such as Firebase, Google ML Kit, and Google Cloud. The application also uses advanced algorithms such as BRISK, ORB, SIFT, and SURF to accurately identify points of interest and process images quickly, even in complex environments. In conclusion, this study shows that these assistance systems can become more powerful while remaining affordable and easy to access, thus meeting the growing needs of visually impaired people. Our tests on four separate systems demonstrated a significant improvement in subtitle accuracy and a marked reduction in image processing time, making image recognition and analysis faster and more efficient. In the future, improvements will be made to the application to further optimize processing times by integrating hardware acceleration and parallel processing techniques. In addition, further research will be carried out to adapt these systems to other contexts and explore their potential in areas such as inclusive education and workplace assistance. This work paves the way for future innovations in assistive technologies aimed at improving the quality of life of visually impaired people.

REFERENCES

[1] A. Budrionis, D. Plikynas, P. Daniušis, and A. Indrulionis, "Smartphone-based computer vision travelling aids for blind and visually impaired individuals: A systematic review," *Assistive Technology*, vol. 34, no. 2, pp. 178–194, Apr. 2020, doi: 10.1080/10400435.2020.1743381.

[2] B. Kuriakose, R. Shrestha, and F. E. Sandnes, "Tools and technologies for blind and visually impaired navigation support: A review," *IETE Technical Review*, vol. 39, no. 1, pp. 3–18, Sep. 2020, doi: 10.1080/02564602.2020.1819893.

[3] Y. Guo, Y. Chen, Y. Xie, X. Ban, and M. S. Obaidat, "An offline assistance tool for visually impaired people based on image captioning," *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Dec. 2022, doi: 10.1109/bibm55620.2022.9994947.

[4] J. Madake *et al.*, "Integrating scene and text recognition for improved scene caption to assist visually impaired," *Communications in Computer and Information Science*, pp. 32–44, 2023, doi: 10.1007/978-3-031-45124-9_4.

[5] S. P. Manay, S. A. Yaligar, Y. T. S. S. Reddy, and N. J. Saunshimath, "Image Captioning for the Visually Impaired," vol. 789, Nov. 2021, doi: 10.1007/978-981-16-1338-8_43.

[6] R. Keskin, O. T. Moral, V. Kilic, and A. Onan, "Multi-gru based automated image captioning for smartphones," *The Signal Processing and Communications Applications Conference (SIU)*, Jun. 2021, doi: 10.1109/siu53274.2021.9477901.

[7] P. Mathur, A. Gill, A. Yadav, A. Mishra, and N. K. Bansode, "Camera2caption: A real-time image caption generator," *2017 International Conference on Computational Intelligence in Data Science(ICCIDS)*, Jun. 2017, doi: 10.1109/iccids.2017.8272660.

[8] J. Liang, D. Doermann, and H. Li, "Camera-based analysis of text and documents: A survey," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 7, no. 2–3, pp. 84–104, Jul. 2005, doi: 10.1007/s10032-004-0138-z.

[9] R. O. Ogundokun *et al.*, "An Android based language translator application," *Journal of Physics: Conference Series*, vol. 1767, no. 1, p. 012032, Feb. 2021, doi: 10.1088/1742-6596/1767/1/012032.

[10] N. Tyagi, "Image - to - audio captioning for the visually impaired," *International Journal of Science and Research (IJSR)*, vol. 12, no. 10, pp. 1609–1613, Oct. 2023, doi: 10.21275/sr231020171202.

[11] R. S. Kızıltepe, J. Q. Gan, and J. J. Escobar, "A novel keyframe extraction method for video classification using Deep Neural Networks," *Neural Computing and Applications*, vol. 35, no. 34, pp. 24513–24524, Aug. 2021, doi: 10.1007/s00521-021-06322-x.

[12] S. A. Tareen and Z. Saleem, "A comparative analysis of SIFT, surf, Kaze, AKAZE, Orb, and brisk," *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, Mar. 2018, doi: 10.1109/icomet.2018.8346440.

[13] S. Sandhya, M. Manaswini, and T. Aarthika, "Image captioning for assisting the visually impaired," *Intelligent Computing and Communication*, pp. 451–460, 2023, doi: 10.1007/978-981-99-1588-0_38.

[14] J. Ganesan *et al.*, "Deep learning reader for visually impaired," *Electronics*, vol. 11, no. 20, p. 3335, Oct. 2022, doi: 10.3390/electronics11203335.

[15] J. Bagrecha, T. Shah, K. Shah, T. Gandhi, and S. Palwe, "VirtualEye: Android application for the visually impaired," *Recent Trends in Intensive Computing*, Dec. 2021, doi: 10.3233/apc210204.

[16] B. Uslu, Ö. Çaylı, V. Kılıç, and A. Onan, "Resnet based deep gated recurrent unit for image captioning on smartphone," *European Journal of Science and Technology*, Apr. 2022, doi: 10.31590/ejosat.1107035.

[17] Ö. Çaylı, B. Makav, V. Kılıç, and A. Onan, "Mobile application based Automatic Caption Generation for Visually impaired," *Advances in Intelligent Systems and Computing*, pp. 1532–1539, Jul. 2020, doi: 10.1007/978-3-030-51156-2_178.

[18] G. Büyüközkan and J. Maire, "Benchmarking process formalization and a case study," Benchmarking for Quality Management *Technology*, vol. 5, no. 2, pp. 101–125, Jun. 1998, doi: 10.1108/14635779810212356.

[19] L. Hagge and J. Kreutzkamp, "A benchmarking method for information systems," *Journal of Lightwave Technology*, Monterey Bay, CA, USA, 2003, pp. 245-253, doi: 10.1109/ICRE.2003.1232756.

[20] V. Kılıç, "Deep gated recurrent unit for Smartphone-based image captioning," *Sakarya University Journal of Computer and Information Sciences*, vol. 4, no. 2, pp. 181–191, Aug. 2021, doi: 10.35377/saucis.04.02.866409.

[21] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,"

arXiv preprint, Dec. 2014, doi: 10.48550/arXiv.1412.3555.

[22] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, doi: 10.1109/cvpr.2015.7298935.

[23] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-V4, inception-resnet and the impact of residual connections on learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Feb. 2017, doi: 10.1609/aaai.v31i1.11231.

[24] F. Xiao, W. Xue, Y. Shen, and X. Gao, “A new attention-based LSTM for image captioning,” *Neural Processing Letters*, vol. 54, no. 4, pp. 3157–3171, Feb. 2022, doi: 10.1007/s11063-022-10759-z.

[25] M. Kalimuthu, A. Mogadala, M. Mosbach, and D. Klakow, “Fusion models for improved image captioning,” *Pattern Recognition. ICPR International Workshops and Challenges*, pp. 381–395, 2021, doi: 10.1007/978-3-030-68780-9_32.

[26] J. Llerena-Izquierdo, F. Procel-Jupiter, and A. Cunalema-Arana, “Mobile application with cloud-based Computer Vision Capability for University Students’ Library Services,” *Innovation and Research*, pp. 3–15, Nov. 2020, doi: 10.1007/978-3-030-60467-7_1.

[27] F. D. Nurzam and E. T. Luthfi, “Implementation of real-time scanner Java language text with Mobile Vision Android based,” *2018 International Conference on Information and Communications Technology (ICOIACT)*, Mar. 2018, pp. 724-729, doi: 10.1109/icoiact.2018.8350738.

[28] H. H. Wang, “Speech recorder and translator using google cloud speech-to-text and translation,” *Journal of IT in Asia*, vol. 9, no. 1, pp. 11–28, Nov. 2021, doi: 10.33736/jita.2815.2021.

[29] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004, doi: 10.1023/b:visi.0000029664.99615.94.

[30] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (surf),” *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, Jun. 2008, doi: 10.1016/j.cviu.2007.09.014.

[31] H. Jabnoun, F. Benzarti, and H. Amiri, “Object detection and identification for blind people in video scene,” *2015 15th International Conference on Intelligent Systems Design and Applications (ISDA)*, Dec. 2015, pp. 363-367, doi: 10.1109/isda.2015.7489256.

[32] A. V. S. Siripalli, N. Shinde, and Prof. L. Sharma, ”Audio Computing Image to Text Synthesizer - A Cutting-Edge Content Generator Application,” in *International Research Journal of Engineering and Technology (IRJET)*, vol. 10, no. 05, May 2023.

[33] M. Madhuram and A. Parameswaran, ”A Text Extraction Approach towards Document Image Organisation for the Android Platform,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 9, Sep. 2018.

[34] V. Bagal, K. Gaykar, and P. Ahirao, ”Image based Text Translation using Firebase ML Kit,” *Grenze International Journal of Engineering & Technology (GIJET)*, vol. 8, no. 2, pp. 399-404, 2022.

[35] H. Proença, “Performance evaluation of keypoint detection and matching techniques on Grayscale Data,” *Signal, Image and Video Processing*, vol. 9, no. 5, pp. 1009–1019, Aug. 2013, doi: 10.1007/s11760-013-0535-1.

[36] I. Fortuna and Y. Khaeruzzaman, “Implementation of OCR and face recognition on mobile based voting system application in Indonesia,” *IJNMT (International Journal of New Media Technology)*, pp. 20–27, Jul. 2022, doi: 10.31937/ijnmt.v9i1.2658.

[37] D. K. Neoh *et al.*, “PicToText: Text recognition using Firebase Machine Learning Kit,” *2021 2nd International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, Sep. 2021, doi: 10.1109/aidas53897.2021.9574187.

[38] C. R. Konduru, M. U. Raehan, T. Hussain, R. C. Ravindranath, R. K. Singh, and R. Chettri, ”Review of Android Based Portable Sign and Text Recognition System,” *International Journal of All Research Education and Scientific Methods (IJARESM)*, vol. 9, no. 5, pp. 1-20, May 2021.

[39] S. Sivan and G. Darsan, “Computer Vision based assistive technology for blind and visually impaired people,” *Proceedings of the 7th International Conference on Computing Communication and Networking Technologies*, Jul. 2016, doi: 10.1145/2967878.2967923.

[40] R. Jenabzadeh and A. Behrad, “Video summarization using sparse representation of local descriptors,” *Intelligent Decision Technologies*, vol. 13, no. 3, pp. 315–327, Sep. 2019, doi: 10.3233/idt-180112.

[41] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, pp. 886-893, 2005, doi: 10.1109/cvpr.2005.177.

[42] M. G. Marne, P. R. Futane, S. B. Kolekar, A. D. Lakhadive, and S. K. Marathe, “Identification of optimal optical character recognition (OCR) engine for proposed system,” *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Aug. 2018, pp. 1-4, doi: 10.1109/iccubea.2018.8697487.

[43] N. Mishra and C. Patvardhan, “Atma: Android Travel Mate Application,” *International Journal of Computer Applications*, vol. 50, no. 16, pp. 1–8, Jul. 2012, doi: 10.5120/7852-1083.

[44] Muh. R. Latief, N. J. Saleh, and A. Pammu, “The effectiveness of machine translation to improve the system of translating language on cultural context,” *IOP Conference Series: Earth and Environmental Science*, vol. 575, no. 1, p. 012178, Oct. 2020, doi: 10.1088/1755-1315/575/1/012178.

[45] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” *Computer Vision – ECCV* 2010, pp. 778–792, 2010, doi: 10.1007/978-3-642-15561-1_56.

[46] S. Gupta, M. Kumar, and A. Garg, “Improved object recognition results using SIFT and orb feature detector,” *Multimedia Tools and Applications*, vol. 78, no. 23, pp. 34157–34171, Oct. 2019, doi: 10.1007/s11042-019-08232-6.

[47] O. González-Chávez, G. Ruiz, D. Moctezuma, and T. Ramirez-delReal, “Are metrics measuring what they should? an evaluation of Image Captioning Task Metrics,” *Signal Processing: Image Communication*, vol. 120, p. 117071, Jan. 2024, doi: 10.1016/j.image.2023.117071.

[48] M. S. Wajid, H Terashima-Marin, P Najafirad, and M. A. Wajid, “Deep learning and knowledge graph for image/video captioning: A review of datasets,” *Evaluation Metrics, and methods*, vol. 6, no. 1, p. e12785, Aug. 2023, doi: 10.1002/eng2.12785/v1/review2.

[49] P. Anderson, B. Fernando, M. Johnson, and S. Gould, “SPICE: Semantic propositional image caption evaluation,” *Computer Vision – ECCV 2016*, pp. 382–398, 2016, doi: 10.1007/978-3-319-46454-1_24.

[50] G. Luo, L. Cheng, C. Jing, C. Zhao, and G. Song, “A thorough review of models, evaluation metrics, and datasets on image captioning,” *IET Image Processing*, vol. 16, no. 2, pp. 311–332, Nov. 2021, doi: 10.1049/ipr2.12367.

BIOGRAPHIES OF AUTHORS

Othmane Sebban is a software engineer who graduated with honors from Sidi Mohamed Ben Abdellah University in 2020. He specializes in intelligent decision-making systems. He is a Ph.D. student at the same university. He is simultaneously specializing in computer science and deep learning while working as a software engineer at the Morrocan Ministry of the Interior (MI). His main areas of interest in this study are the applications of contemporary AI algorithms to mechanical component design. He can be contacted at email: othmane.sebban@usmba.ac.ma.

Ahmed Azough is an Associate Professor at De Vinci Engineering School in Paris, where he leads the M.Sc. in Computer Science and Data Science and co-directs the De Vinci Immersive Lab. He holds an H.D.R. in Virtual Reality and Photogrammetry (University of Fez, 2021), a Ph.D. in Video Semantic Analysis (Lyon 1 University, 2010), and a Master's in Computer Science (INSA Lyon, 2006). His research focuses on augmented, virtual, and mixed reality applications in tourism, education, and healthcare, as well as multi-camera event detection for video surveillance systems. He previously worked as an R&D engineer at Orange Labs and Genomic Vision and held academic positions at Lyon 1 University and the University of Fez. He can be contacted at email: ahmed.azough@devinci.fr.

Mohamed Lamrini is a Professor of Computer Science at the Faculty of Sciences Dhar El Mehraz, part of Sidi Mohamed Ben Abdellah University (USMBA) in Fez, Morocco. He earned his Ph.D. from Claude Bernard University Lyon 1 in 1993. His research interests include software quality assurance methodologies, artificial Intelligence models, as well as industrial engineering, focusing on statistical methods and tools. He is also a member of the LPAIS Laboratory. He can be contacted at mohamed.lamrini@usmba.ac.ma.