International Journal of Electrical and Computer Engineering (IJECE)
Vol. 14, No. 4, August 2024, pp. 4759∼4774
ISSN: 2088-8708, DOI: 10.11591/ijece.v14i4.pp4759-4774

Design a smart platform translating Arabic sign language to English language

Maha Alamri[1], **Sonia Lajmi**[1,2]
[1]Faculty of Computing and Information, Al-Baha University, Al Baha, Saudi Arabia
[2]MIRACL Laboratory, Technopole of Sfax, University of Sfax, Sfax, Tunisia

Article Info

Article history:

Received Feb 9, 2024
Revised Mar 12, 2024
Accepted Apr 2, 2024

Keywords:

Deep learning
Fine-tuning
Real-time translation
Recognition
Sign language

ABSTRACT

Sign language is the only means of communication for deaf and hearing-disabled people in their communities. It uses body language and gestures, such as hand shapes and facial expressions, to convey a message. It is important to note that sign language is specific to the region; that is, Arabic sign language (ArSL) is different from English sign language. Therefore, this research proposes a way to improve the translation of ArSL using a new artificial intelligence (AI) architecture. Specifically, a convolutional neural network (CNN) based on fine-tuning of the SSD-ResNet50 V1 FPN is applied to build a real-time ArSL recognition and translation system with fast and accurate results. The proposed AI architecture can provide translation of sign language in real-time to enhance communication in the deaf community. We achieved an average F-score of 86% and an average accuracy of 94%.

Corresponding Author:

Maha Alamri
Faculty of Computing and Information, Al-Baha University
Al Baha, Saudi Arabia
Email: m.alamri@bu.edu.sa

1. INTRODUCTION

People can communicate by sending messages, expressing their inner sentiments, and exchanging thoughts, either vocally or non-verbally. However, members of the deaf community are unable to speak. Therefore, the development of sign language (SL) was intended to help hearing-impaired individuals express their sentiments [1]. SL is an entirely natural language with its own lexicon and grammatical structures. Several different SLs are used across the world [2]. Therefore, deaf communities from different countries cannot easily understand each other since they do not use the same SL. Among deaf communities worldwide, the use of the Internet for communication purposes is an increasing trend. In fact, the deaf use a variety of internet video-sharing platforms, including vlogs (video blogs) for entertainment, education, and other forms of SL communication [3]. However, it is difficult to access and analyze this enormous collection of signed video footage because there is a lack of associated text-based tagging information, and 'extracting' content directly from videos is still a difficult challenge [4]. In this context, Hamroun *et al.* [5] proposes a multimedia retrieval approach that facilitates the exploitation of the video contents. Moreover, support for the representation and description of SL video content using sign writing systems is also limited [4]. Additionally, the majority of videos are uploaded without subtitles or transcripts, which makes viewing more challenging for individuals with hearing impairments [6].

Therefore, researchers working in the deaf community are looking for automatic means and techniques to assist and serve deaf individuals by automating systems that help them communicate and make their lives simpler [7]. In this context, the development of AI technology, especially machine learning (ML) and deep learning (DL) techniques, is crucial. Indeed, many sign recognition systems have been suggested using ML techniques that record human motions, evaluate them, and then output the SL or text [1].

Automatic translation from or into SL relies on recognition to create written text using an annotated SL corpus or other tools related to grammatical structure, syntactic rules, or synthesis designed for SL. Thus, research on automatic translation is necessary for the growth and development of all SLs [8]. While such studies are challenging, the topic is receiving increasing attention due to its wide range of potential applications [9].

The current research aims to improve communication between deaf and hearing-disabled communities and non-deaf communities by proposing a new AI architecture. This architecture is designed to recognize ArSL in real time and translate it to English text using DL and transfer learning techniques.

– How can AI techniques be used to resolve the problems related to real-time ArSL recognition and detection?
– Is it possible to build an architecture using a CNN based on the learning transfer of the SSD-ResNet50 V1 FPN to translate ArSL to English?
– What is the effectiveness of the proposed algorithm compared with other deep learning algorithms, such as MobileNet?

In this paper, our research contributes to the field of deaf communication using an AI technique. Firstly, we propose an AI technique to resolve the problems related to real-time ArSL recognition and detection. Secondly, build an architecture using a CNN based on the learning transfer of the SSD-ResNet50 V1 FPN to translate ArSL to English. Thirdly, our experimental results demonstrate that our architecture encompasses others and obtains an average accuracy of 94%.

The remainder of the paper is organized as follows: related and existing works are discussed in section 2. The Arabic sign language (ASiL) system architecture is described in section 3. In section 3.1., we detail the construction of the ASiL dataset. The construction of the training model is described in section 3.2. Experimental setup and results are given in section 3.3.1., section 3.3., section 4., and section 4.4. Lastly, section 5. concludes the paper with a summary and perspective of our works.

2. LITERATURE REVIEW

According to a recent report from the World Health Organization [10], more than 430 million individuals worldwide or more than 5% of the world's population have speech or hearing impairments. Further, approximately 700 million people, or one in every ten people, will experience debilitating hearing loss by the year 2050. For deaf and hearing-disabled people, SL is the primary form of communication. It is a native language that is used by 70 million people worldwide and 18 million in the Arabic world alone [8].

SL is a visual language that uses body language and gestures, such as hand shapes, lip patterns, and facial expressions, to transmit messages [11]. There is no single SL that is used by deaf people around the world; rather, SLs differ by country and geographic region [7]. Moreover, there are different SLs even in countries that speak the same language. For example, American SL (ASL) and British SL (BSL) are separate SLs [12]. Arabic countries are no different, and there are different SLs for Jordan, Egypt, Yemen, Saudi Arabia, and Oman [13]. Even though these languages coexist in nations with similar cultures and share some signs, most are different. Consequently, the League of Arab States (LAS) and the Arab League Educational, Cultural, and Scientific Organization (ALECSO) suggested a new SL named ArSL in 1999 to standardize various languages into one [14]. A dictionary of roughly 3200 signs is available for this language, and it was released in two parts [15], [16]. ArSL is now used mainly in the Arab Gulf countries, and it is the main language used by Arabic news media [14].

Additionally, the syntax of SL differs significantly from that of spoken language, including the lexicon, word order, and language structure [9]. Sentences are arranged in ArSL so that the subject (S) precedes the verb (V) and the object (O). In contrast to Arabic, which has several structures, including SVO, VOS, VSO, SOV, and OSV, ArSL primarily uses SVO and SOV structures. Additionally, prepositions are absent from ArSL when used indirectly; for example صلى الرجل في المسجد is translated in ArSL as رجل مسجد صلى [14]. As a result, SL is different from spoken language and independent of it. Hence, there is a need for SL recognition (SLR) in addition to machine translation between spoken language and SL to enhance two-way communication, which would positively impact communication between individuals who use SL and those who might not understand it [17] as well as between deaf communities that use different SLs.

In recent years, several studies have been on SLR [1], which has been defined as a system that can recognize signs and provide a set of words as output [14]. SLR encompasses data collection, pre-processing, and feature extraction. There are two methods for recognizing SL: sensor-based and image-based methods [1]. Sensor-based methods require the user to wear sensor-equipped instrument 'gloves' to detect hand gestures. In contrast, image-based methods do not require the user to wear any equipment. Compared to sensor-based systems, the image-based method offers users greater flexibility [13], [17].

Recent research has focussed on the performance of image-based methods for identifying ArSL [18], [19]. These systems were created utilizing a variety of ML techniques, including elastic graph matching [20], support vector machines (SVMs) [21], and artificial neural networks [22]. Numerous research domains, including image identification and classification, have benefitted from DL methods [13].

Nagi *et al.* [23] employed a CNN along with colour segmentation and morphological image processing. When applied to 6,000 sign images derived from six gestures, their model had a 96% accuracy rate. Moreover, Tang *et al.* [24] used a CNN with a deep belief network (DBN). Thirty-six distinct hand postures were used to train the DBN and CNN models. The accuracy of the DBN model was 98.12%, which was greater than the accuracy of the CNN model. Sruthi and Lijiya [25] employed a CNN architecture to identify Indian SL alphabets, achieving an accuracy of 98.64%.

To reduce the amount of time and dataset samples required to train the CNN model, some researchers have employed transfer learning techniques. Saleh and Issa [26] utilized fine tuning to improve performance in classifying 32 hand gestures from the ArSL dataset using pre-trained VGG-16 and ResNet152 networks. The testing accuracy for VGG16 and ResNet152 were 99.4% and 99.6%, respectively. [2] employed MobileNet-LSTM with transfer learning and achieved accuracy of 99.7% and 72.4%, respectively.

ElAlfi *et al.* [27] presented a knowledge-based approach for translating Arabic text into ArSL. The process of translating an Arabic word into its corresponding sign follows specific rules. Five experts utilized the proposed system to examine 101 sentences, and the F1 score was 97.7%. The approach addresses the word-level distinctions between Arabic and ArSL but overlooks sentence-level variations in syntax and structure. Al-Rikabi and Hafner [28] translated Arabic text sentences into ArSL using syntax transformation.

Researchers have created ArSL datasets. For example, Latif *et al.* [29] created a dataset called ArSL2018, consisting of 54,049 images of 32 ArSL signs and alphabets collected from 40 participants in different age groups. Sidig *et al.* [11] constructed another database for ArSL named KArSL, including 502 signs performed by three professional signers, each of which is repeated 50 times by each signer. Bencherif *et al.* [30] created a video-based sign database for ArSL. The signs include the Arabic alphabet, numbers, and some daily use signs, including 80 static and dynamic signs, with each sign repeated five times by 40 signers.

Luqman and El-Alfy [2] created an ArSL database consisting of 6748 videos of 50 signs performed by four signers. Moreover, Batnasan *et al.* [31] collected and annotated a letter dataset (ArSL21L: Arabic sign language letter dataset benchmarking and an educational avatar for metaverse applications) consisting of 14202 images of 32 letter signs with various backgrounds performed by 50 people. The KSU-ArSL dataset consists of 80 Arabic alphabets, numbers, and common words recorded by 40 healthy subjects using three cameras, with each sign repeated five times in five separate sessions on the same day. As a result, there are 200 video samples per class and 16,000 total samples [32].

As observed from the literature, research on ArSL recognition and translation is in the early stages [2]. The absence of language resources, such as corpora and linguistic studies of ArSL, could be partly affecting the development of ArSL recognition and translation [33], [11]. The use of DL models could help to resolve the problem of SL detection. To the best of our knowledge, previous research on DL models has only reported accuracy and not response times. However, the latter is an important factor in real-time systems. Moreover, few works have focused on the detection and translation of ArSL to Arabic text in real-time, and no study has attempted to translate ArSL to English language in real-time.

3. METHOD

An overview of the ASiL platform is presented in Figure 1. It consists of two fundamental steps: data pre-processing and training. Data pre-processing is a fundamental step for detection systems based on DL algorithms. We constructed an Arabic sign language dataset. A part of this dataset is presented in section 3.1.. The result of this step is a set of images for training and testing and an XML document describing each image. The input for the training is a record file which containing all the images for training and the corresponding annotations and a label map file which contains the several classes and their meanings (a class refers to an English term). We train the SSD-ResNet50 V1 FPN [34] model with our new ASiL dataset. Real-time detection can then be performed, with the name of the sign presented in English. Using the common objects in context (COCO) [35] dataset, we pre-trained several detection models, including the SSD-ResNet50 V1 FPN and SSD MobileNet V1 FPN 640x640 from the TensorFlow 2 detection model zoo [36].

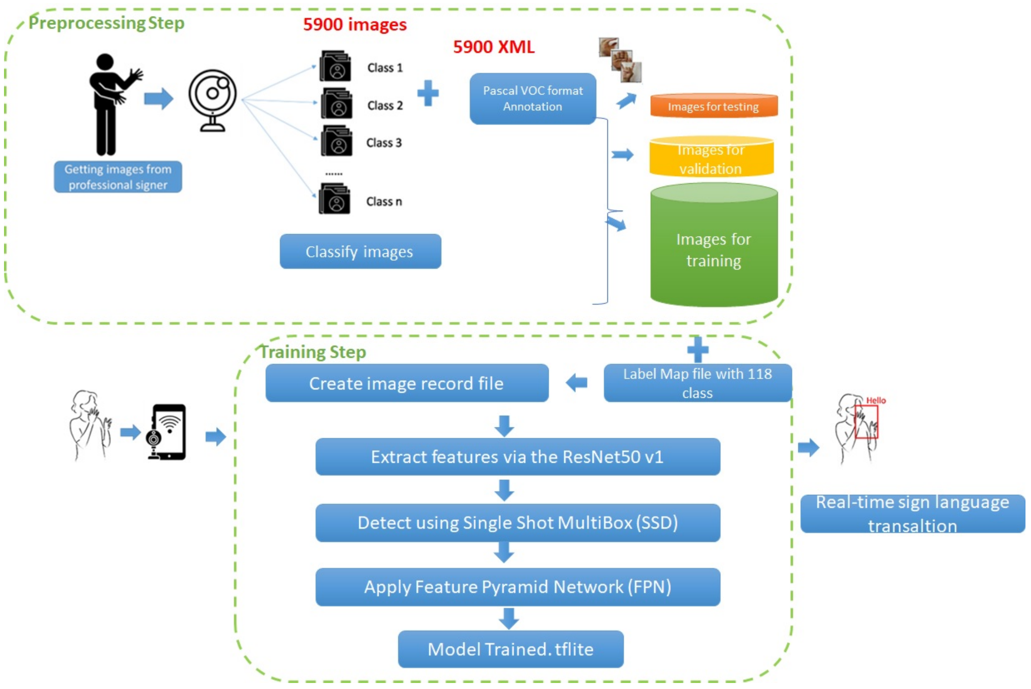

Preprocessing Step
5900 images
5900 XML
Class 1
Class 2
Class 3
Class n
Getting images from professional signer
Pascal VOC format Annotation
Images for testing
Images for validation
Images for training
Classify images
Training Step
Create image record file
Label Map file with 118 class
Extract features via the ResNet50 v1
Detect using Single Shot MultiBox (SSD)
Apply Feature Pyramid Network (FPN)
Model Trained. tflite
Hello
Real-time sign language transaltion

Figure 1. An overview of the ASiL Arabic sign language platform

3.1. Construction of the ASiL dataset

As mentioned above, this step is fundamental for the training of the DL model. In fact, a new database is needed for SL because we cannot use a standard object detection dataset like COCO. This dataset [37] is used by default in the object detection API of the TensorFlow library. We need to recognize specific objects that only have meaning for deaf-mute Arabic people. For this work, we constructed only 118 classes, corresponding to 118 static hand signs. For each class, we made 50 images, which means 5,900 images in total. First, we record a set of images and classes to enrich the existing database. After the recording step, the obtained images must be labeled because they were initially recorded without annotation. The LabelImg [38] an annotation tool is used to annotate the bounding boxes defining the hands responsible for SL in each image. This tool can be easily installed on GitHub. Figure 2 illustrates the use of the LabelImg tool to annotate the collected images. With this tool, an XML file is generated for each image. An example of an image annotation XML file is shown in Figure 3. Each XML file stores the location of the object, the coordinates of the area, as well as the annotation (the name of the gesture).

After this annotation step, the database is divided into two parts: one for testing and another for training, and the ratio between the training and test datasets is 7:3. That means 70% of the dataset is reserved for training and 30% for testing. Then, a labelMap file in *.pbtxt* format must be provided. This document lists all the classes used in ArSL, that is, all the meanings of the gestures made by the professional signatories. This document provides auxiliary information for the following annotation and detection steps. Images, annotations, and labelMaps are transcribed into TFRecords to prepare for training with the TensorFlow framework. After the recording step, the obtained images must be labeled because they were initially recorded without annotation.

3.2. Construction of fine-tuning based training classification

Our training model is based on the TensorFlow object detection API [39]. This API contains several deep-learning algorithms capable of detecting objects in images. The set of the configuration file *pipeline.conf* is needed to parameterize the number of classes, and the batch size. The pipeline contains all the layer configuration and testing. The objective of training based on fine-tuning learning is to start not from zero but from a pre-trained model, which can considerably improve the accuracy of the system. In this context, we use the pyCOCOtools package (COCO's Python API tools) as a starting point for the training (Checkpoint 0). As mentioned above, editing *pipeline.conf* is crucial to accommodate the new classes and the new data. To facilitate the construction of the new system, each step of the training process is recorded in a file in the form of checkpoints. Following training, the Tensorflow model generated Graphdef and Checkpoint graphs. These graphs are converted to TensorFlow Lite (tflite) format, after which an interpreter is constructed. The interpreter executes the model using a collection of operators. Tensorflow is used to manipulate data, create models, train, and forecast. However, deep learning needs a large amount of computing power. Training for embedded and mobile devices is possible, but it will take a considerable amount of time. To address this issue, TensorFlow will be utilized for training and Tensorflow Lite for inference.

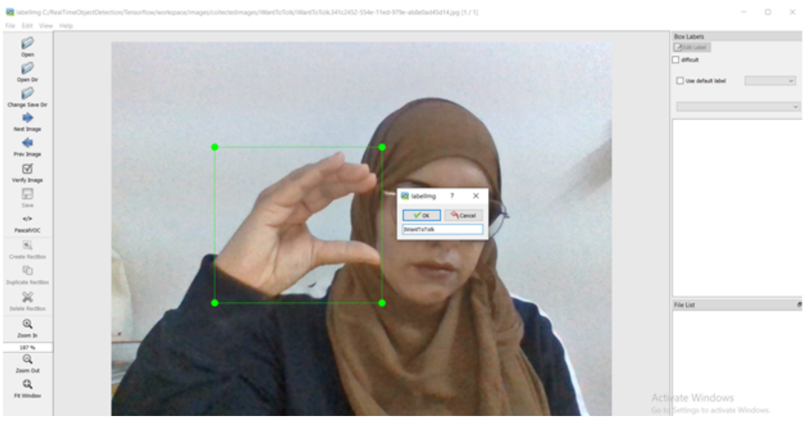
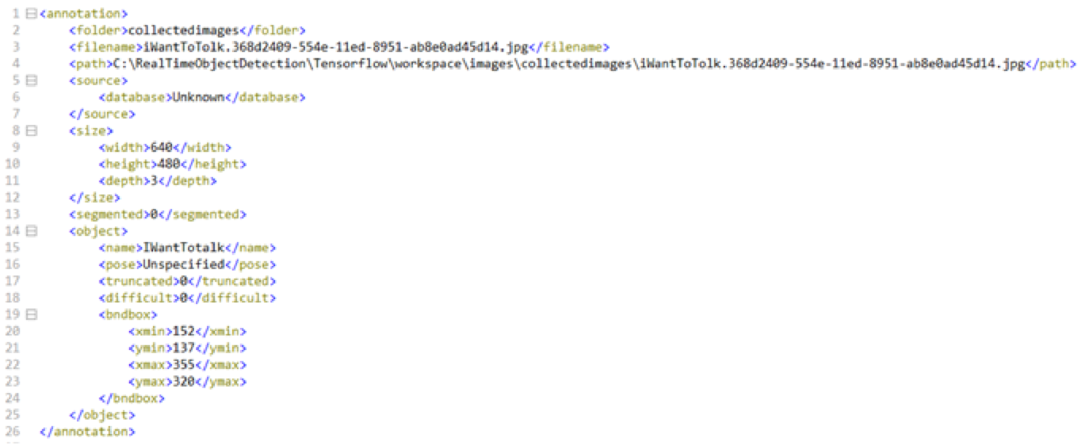


Figure 2. Using LabelImg to annotate the captured image



```
<annotation>
    <folder>collectedimages</folder>
    <filename>iWantToTolk.368d2409-554e-11ed-8951-ab8e0ad45d14.jpg</filename>
    <path>C:\RealTimeObjectDetection\Tensorflow\workspace\images\collectedimages\iWantToTolk.368d2409-554e-11ed-8951-ab8e0ad45d14.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>640</width>
        <height>480</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>IWantTotalk</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>152</xmin>
            <ymin>137</ymin>
            <xmax>355</xmax>
            <ymax>320</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 3. An example of an XML document describing an image and the boundary box with the name of the object

The multi-level features that the SSD-ResNet50 architecture receives as inputs are produced by the feature pyramid network (FPN). An object detector receives the extracted feature map layers from the FPN, which functions as an extractor. Any hand sign gesture object that the model localizes will have an object boundary box drawn around it at each location. The training results are converted into TensorFlow Lite *(.tflite)* format for mobile usage. The following steps are detailed in the next sections:

- Extracting characteristics via the ResNet50 V1 network.
- Apply feature pyramid network (FPN).
- To detect the object, use the ResNet50 network's output as the SSD network's input.
- Converting to TensorFlow lite format.
- Making an Android app to implement the TensorFlow Lite model.

3.2.1. Extracting characteristics via the ResNet50 V1 network

For feature extraction, we utilize the ResNet50V1 network [34]. The ResNet50 V1 network makes use of a CNN. The construction block of the 50-layer ResNet is designed as a bottleneck. A bottleneck residual block decreases the number of parameters and matrix multiplications by using 1x1 convolutions, often known as a bottleneck. This allows for significantly quicker training for each layer. It employs a stack of three layers rather than two. Figure 4 illustrates the deep residual learning architecture that serves as the foundation for the ResNet50 model. The network can hold onto its prior learning thanks to the residual blocks. If there is nothing new to learn, the network will add what it has already learned by using an identity mapping function where the output equals the input to retain what it has learned.

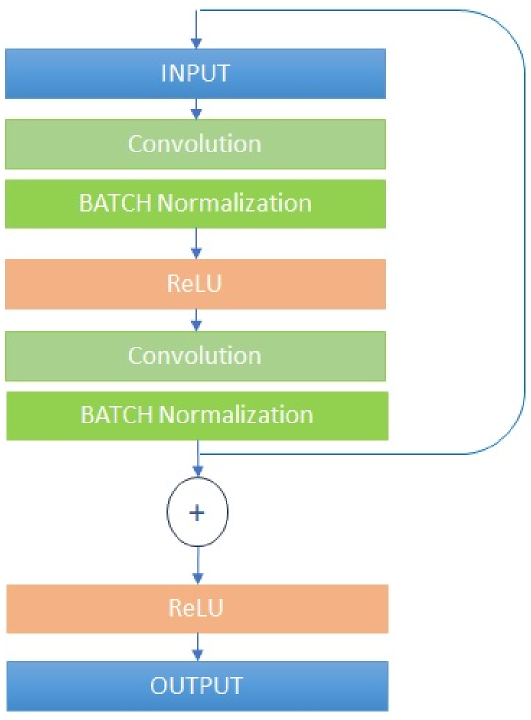


Figure 4. Residual ResNet50 v1 block

3.2.2. Apply feature pyramid network (FPN)

Feature pyramids are an essential component of recognition algorithms that recognize things of various sizes. These pyramids are size-invariant. An object's scale change is compensated by altering its level in the pyramid, allowing models to identify items at a wide variety of scales by scanning through pyramid levels and positions. The primary benefit of highlighting each level of an image pyramid is that it results in a multi-scale feature representation in which all levels are semantically strong, even high-resolution levels. Nonetheless, including each level increases inference time, consumes a lot of memory, and causes inconsistency between train and test-time inference. The architecture, which is known as the FPN [40], which is seen in Figure 5, uses lateral connections and a top-down-sampling approach to integrate high-resolution, semantically weak features with low-resolution, semantically strong features to increase the accuracy of small object detection.

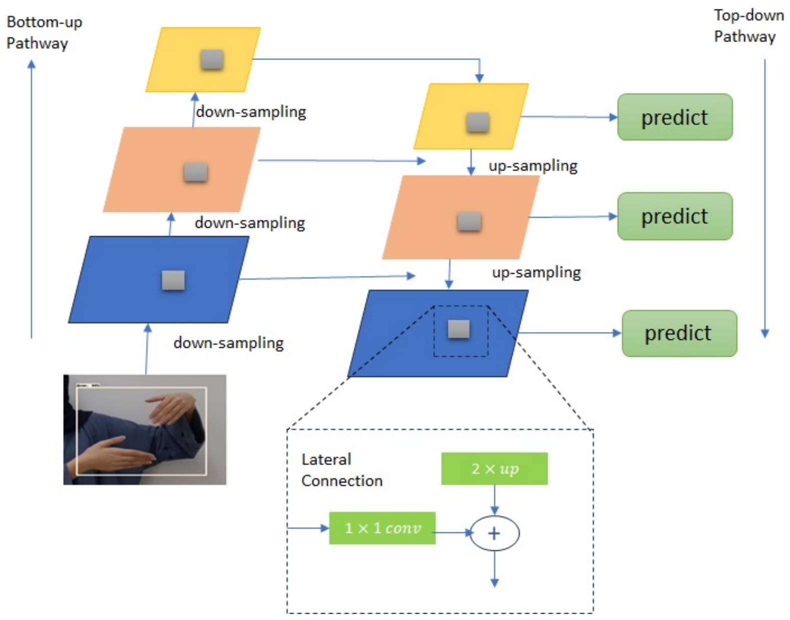


Figure 5. FPN block demonstrating the lateral connection fused by addition and the top-down manner

3.2.3. Detecting objects with SSD

A feed-forward convolutional network is the basis of the single shot multibox detector (SSD) [41], a method that generates a fixed-size set of bounding boxes and scores for the presence of object class instances in those boxes. A non-maximum suppression step then generates the final detection. Bounding boxes that have been carefully chosen based on their sizes, aspect ratios, and placements throughout the image are represented by default boxes. Predicting category scores and box offsets is the fundamental function of SSD, which aims to identify which default box will be used to apply convolutional filters to the feature maps.

A widely used architecture for high-quality image categorization serves as the foundation for the initial network layers. The model's usage of multi-scale convolutional bounding box outputs connected to many feature maps at the network's top is one of its main characteristics. We can efficiently model the space of potential box forms using this representation. The assignment of ground truth information to certain outputs within the defined set of detector outputs is the primary distinction between training SSD and training a conventional detector that employs region suggestions. SSD uses several additional layers, such as feature map extraction, in addition to the VGG-16 base network. Therefore, we propose to use the ResNet50 V1 network for feature map extraction instead of the VGG16 base network.

To recognize Arabic sign items, the SSD adds extra auxiliary bits after ResNet50 V1. The SSD model creates a vector that represents the chance of occurrence of *c+1* item, where *c* is the number of levels and a background layer denotes the lack of an object. A vector consisting of four components (x, y, h, and w) represents the item's location in the frame.

After each training phase, we compute the loss function until it is minimized, at which point we make adjustments to make it more like the real object. In this context, we employ the smooth L1-loss, which can be understood as a combination of L1-loss and L2-loss, for the localization loss. When the argument's absolute value is large, it behaves as an L1-loss; when the argument's absolute value is approaching zero, it behaves as an L2-loss. Smooth L1-loss combines the advantages of L1-loss (steady gradients for large values of x_1) and L_2-loss (less oscillations during updates when x is small). The equation is presented by (1). The information is then sent to the SSD network, which figures out the coordinates, the loss function value, and the chance that the item will show up after images are fed into ResNet50 V1 and the FNP network to extract features.

$$L_{1;smooth} = \begin{cases} |x| & if\,|x| > \alpha; \\ \frac{1}{|\alpha|} & if\,|x| \leq \alpha \end{cases} \quad (1)$$

α is a hyper-parameter is taken as 1. $\frac{1}{|\alpha|}$ appears near x^2 to make it continuous.

3.2.4. Converting to Tensorflow lite (flite)

A slim version of TensorFlow designed for embedded and mobile devices is called TensorFlow Lite. It makes machine learning model execution on mobile devices possible. The process of the model is shown in Figure 1. The model file format, a graph processing interpreter, a set of helpful kernels, and an interface for the hardware acceleration layer are all included in TensorFlow Lite. This effort solely piques our curiosity because of the model's file format. TensorFlow Lite is the name given to this special model file format. It depends less on the hardware configuration and is extremely light.

3.3. Parameters selection

In this section, we describe the parameters selected for the realization of our system. The first part is devoted to the choice of the training and testing environment. Then we discuss the used loss functions and compare them with other loss functions to argue our choice. Finally, we present and discuss our choice of learning rate. Indeed, the learning rate is a hyperparameter that controls the extent to which the model weights are adjusted based on the displayed error after each step.

3.3.1. Selection of training and testing environment

The Python language is used to implement our ASiL platform. We set up two kinds of environments: i) one for training and another for ii) collecting data and real-time testing. textcolorblue Indeed, for the training environment, we use Google Colab. It is a Jupyter notebook in a cloud environment. It has a portable operating system interface (POSIX) x64 with 16 GB of RAM and a GPU (NVIDIA-SMI 460.32.03). The GPU is very important to optimize the execution time in the training phase. The version of the CUDA Toolkit is 11.2.152 with CuDNN 8.1.1.33-1. The version related to Tensorflow is 2.9.2, and the Tensorboard version is 2.9.1. For the testing, it is done in Anaconda/Jupyter Notebook 6.4.12 on a laptop with Windows 10 Professional x64 OS and a classic CPU Intel (R) Core (TM) i5-8250U @ 1.6 GHz 1.8 GHz with 8 GB of RAM, with the same version of TensorFlow and OpenCV 4.6.0. It is important to use the camera. The variation is important because we want to simulate a real environment. This environment looks like a small device should not have very sophisticated performance like the presence of a NIVIDIA GPU.

3.3.2. Loss function

The objective of object detection, a popular deep learning challenge, is to find (localize) and categorize items in an image. A loss function that gauges how effectively the model predicts object locations and classes must be defined in order to train an object detection model. The classification loss and the localization loss are the two types of object detection loss functions. The former is used to train a head that classifies objects in order to identify their type, whereas the latter is used to train a different head that locates objects by regressing a rectangular box. To prevent the overfitting issue, regularization is used. In the following, the classification loss, localization loss, regularization loss, and total loss results will be discussed. The performance indicates that our model achieves outstanding performance in ArSL detection and recognition tasks. We use TensorBoard to visualize parameter changes during the model training process.

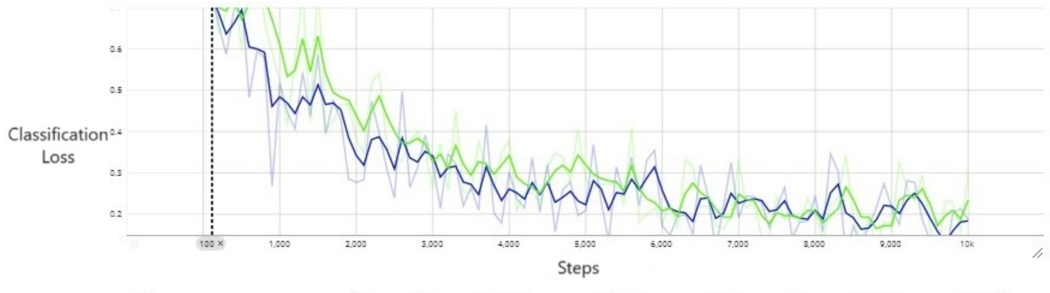
3.3.3. Classification loss

This work makes use of focal loss. In order to ensure that predictions on difficult cases get better over time rather than getting unduly confident with easy ones, focused loss concentrates on the examples that the model gets incorrect rather than the ones that it can predict with confidence. Focal loss uses a technique known as downsizing to accomplish this. By decreasing the impact of simple examples on the loss function, a strategy known as "down weighting" directs greater focus toward difficult examples. By incorporating a modulating element into the cross-entropy loss, this method can be put into practice. The equation (2) details the focal loss function.

$$FocalLoss = -\sum_{i=1}^{i=n}(i-p_i)^{\gamma}log_b(p_i) \tag{2}$$

where the focusing parameter, γ, needs to be adjusted by cross-validation.

Figure 6 represents the classification loss for the ASiL model. We represent the classification loss performance during the training of both versions of ASiL. We obtained approximately the same result after 10,000 steps and a loss of less than 0.2.

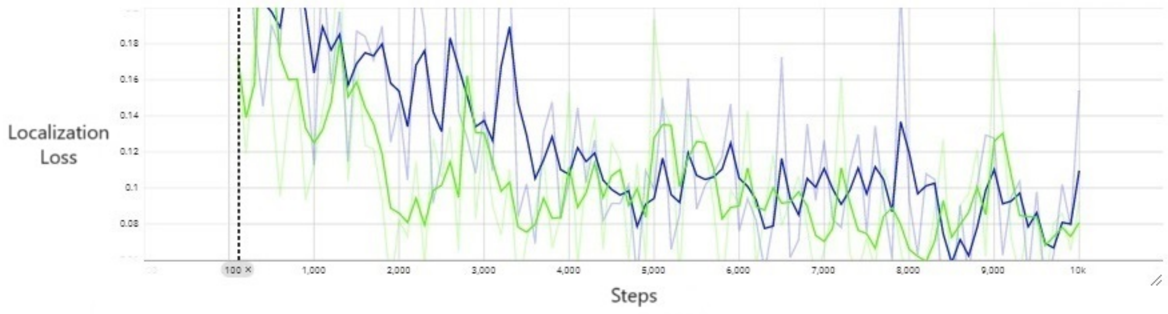


	Run↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
●	ASiL_MoBNet/train	0.165	0.8042	0.8042	0.2341	↓0.5701	↓-71%	100	10,000
●	ASiL_ssd_restNet50/train	0.1313	0.7325	0.7325	0.1838	↓0.5488	↓-75%	100	10,000

Figure 6. Classification loss for the ASiL based on MobileNet and ResNet50

– Localization loss

The localization function used is the (1) explained in section 3.2.3.. Figure 7 represents the localization loss for the ASiL model. We represent the localization loss performance during the training of both versions of ASiL. We obtain approximately the same result after 10,000 steps.



	Run↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
●	ASiL_MoBNet/train	0.0592	0.2884	0.1725	0.0808	↓0.0917	↓-53%	100	10,000
●	ASiL_ssd_restNet50/train	0.0585	0.3999	0.3999	0.1096	↓0.2903	↓-73%	100	10,000

Figure 7. Localization loss for the ASiL based on MobileNet and ResNet50

– Regularization loss

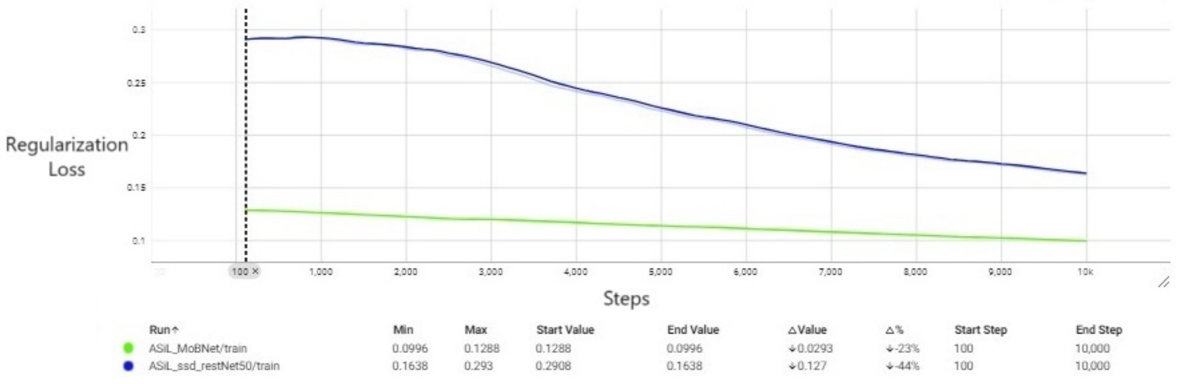
One strategy to discourage the model's complexity is regularization. By punishing the loss function, it achieves this. This assists in resolving the overfitting issue. Regularization loss in TensorFlow is just the total of the L_1 or L_2 penalty terms computed using the model's weight matrices. The total loss that the model is attempting to minimize is then formed by adding the regularization loss to the initial loss function.

$$L_1 Regularization = \lambda \sum_{j=0}^{m} |w_j| \tag{3}$$

$$L_2 Regularization = \lambda \sum_{j=0}^{m} w_2^j \tag{4}$$

L_1 and L_2 regularization are two popular regularization methods that are used to deal with overfitting and feature selection. L1 regularization, sometimes referred to as lasso regression, consists of appending the coefficient's *absolute value of magnitude* as a penalty term to the loss function. L_2 regularization, sometimes referred to as ridge regression, consists of appending the coefficient's *squared magnitude* as a penalty term to the loss function. λ=0.0004. Experimentation is what determines this value.

Figure 8 represents the regularization loss for the ASiL model. We represent the regularization loss performance during the training of both versions of ASiL. The performances of ASiL based on ResNet50 decreased very fast 44% compared to 23% for MobileNet. The loss regularization obtained for ResNet is less than 0.0996 compared to MobileNet's 0.1638.

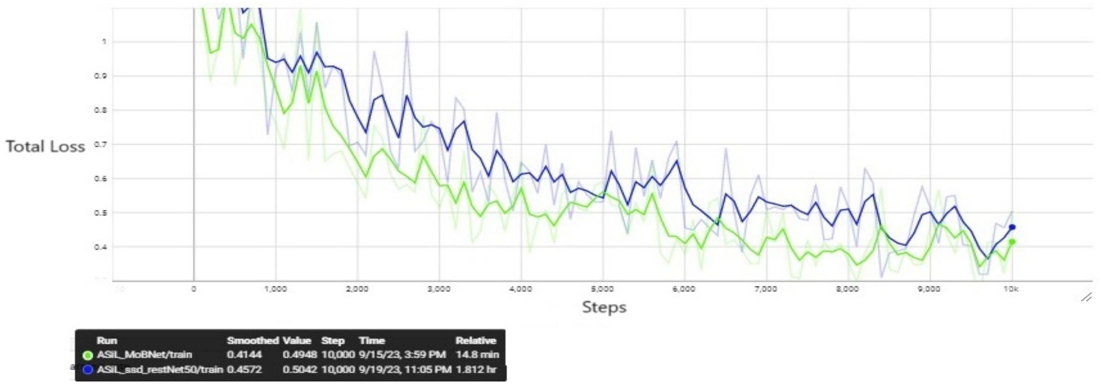


	Run↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
🟢	ASiL_MoBNet/train	0.0996	0.1288	0.1288	0.0996	↓0.0293	↓-23%	100	10,000
🔵	ASiL_ssd_restNet50/train	0.1638	0.293	0.2908	0.1638	↓0.127	↓-44%	100	10,000

Figure 8. Regularization loss

– Total Loss

The total loss is the sum of the classification loss, the localization loss, and the regularization loss. We obtained an acceptable value of about 0.4 after training 10,000 steps. Figure 9 shows the Total loss during 10,000 steps.



	Run	Smoothed	Value	Step	Time	Relative
🟢	ASIL_MoBNet/train	0.4144	0.4948	10,000	9/15/23, 3:59 PM	14.8 min
🔵	ASIL_ssd_restNet50/train	0.4572	0.5042	10,000	9/19/23, 11:05 PM	1.812 hr

Figure 9. Total loss

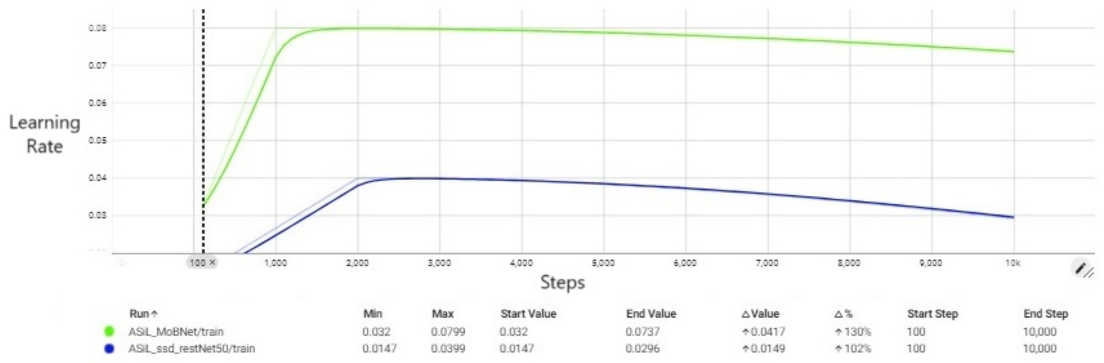
3.3.4. Learning rate

One hyperparameter that regulates how much the model is altered in response to the predicted error after each update of the model weights is the learning rate. Selecting the learning rate is difficult since a value that is too high could lead to learning an unstable training process or a sub-optimal set of weights too quickly, while a value that is too low could result in a lengthy training process that could become stuck. When setting up a deep learning model, the learning rate can be the most crucial hyperparameter. Consequently, developing an intuition regarding the dynamics of the learning rate on model behavior and understanding how to look into how the learning rate affects model performance is crucial. We employ the momentum optimizer and the cosine annealing learning rate schedule in our model. The impact of cosine annealing is that it begins with a large learning rate and then reduces it quickly to a minimum value before dramatically increasing it once more. When the learning rate is reset, it functions as if the learning process were to restart itself, and using the same good weights as the starting point is known as a *warm restart* as opposed to a *cold restart*, which may utilize a fresh set of small random integers .

$$\eta_t = \eta_i^{min} + \frac{1}{2}(\eta_i^{max} - \eta_i^{min})(1 + cos(\frac{\tau_{cur}}{\tau_i}\pi)) \tag{5}$$

where η_i^{min} and η_i^{max} are ranges for the learning rate and τ_{cur} account for how many epochs have been performed since the last restart.

Figure 10 illustrates our comparison of the two versions of the ASiL learning model's performance in terms of learning rate. The first one is ASiL based on MobileNet, and the second one is the ASiL model based on ResNet50. The version based on ResNet50 has a better learning rate.



	Run ↑	Min	Max	Start Value	End Value	△Value	△%	Start Step	End Step
●	ASiL_MoBNet/train	0.032	0.0799	0.032	0.0737	↑0.0417	↑130%	100	10,000
●	ASiL_ssd_restNet50/train	0.0147	0.0399	0.0147	0.0296	↑0.0149	↑102%	100	10,000

Figure 10. Learning rate

4. RESULTS AND DISCUSSION

The real-time sign was sent into the trained model as input after the model had been trained and tested during the training phase. On COCO, an object detector's performance is measured using 12 metrics; more than 10 indicators are utilized to assess target detection models. The most often utilized ones are f-score, recall, accuracy, and precision. Thus, we assess our model using those metrics. Equations (6), (7), (8), and (9) demonstrate how they are all ranked as true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{6}$$

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

$$F\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{9}$$

The bounding box and score for every sign-hand object are recorded in the created log file. The output frame with the designated Arabic sign, the evaluation results, and the log file were obtained as the last step.

4.1. Precision/recall performance of each sign gesture

We illustrate the system's performance in precision in Figure 11 and recall in Figure 12. The corresponding equations for precision and recall are given by (7) and (8), respectively. There are just 9% of signs that are less than 80% accurate. Our model's average recall is 85%, and its average precision is 91%. Most signs have a precision near 1, as shown in Figure 11. However, during the training process, our model was unable to identify eight signs from 118 signs gestures: bed, October, wall, west, whistle, chalk, employee, and engagement. This issue can be attributed to their pronounced similarity to other visual indicators within the dataset.

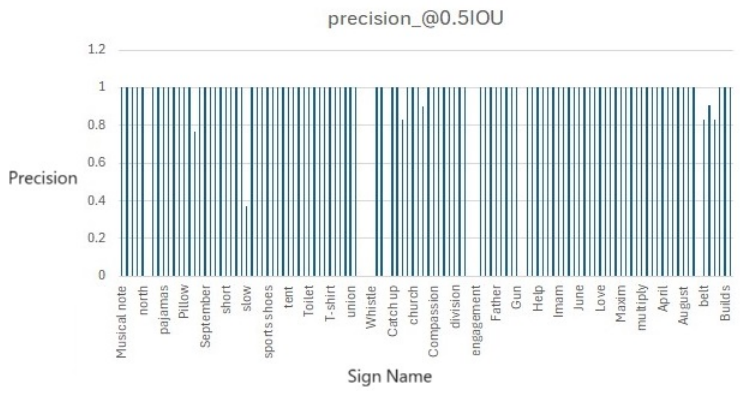


Figure 11. Precision of ASiL model

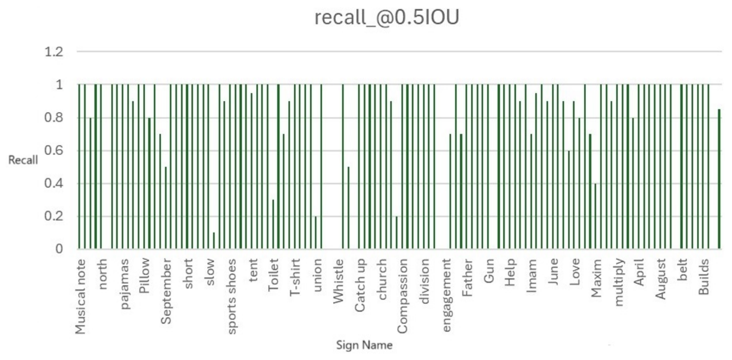


Figure 12. Recall of ASiL model

4.2. F-score performance of each sign gesture

The F-score is a measure that takes into consideration precision and recall at the same time and is defined by (9). Figure 13 demonstrates that the F-score is consistently high, approaching 1, for the majority of signs. However, eight signs highlighted in the previous section exhibit an F-score of 0, because they are not recognized by the model. An average F-score of 86.4% is measured for our model based on ResNet50.

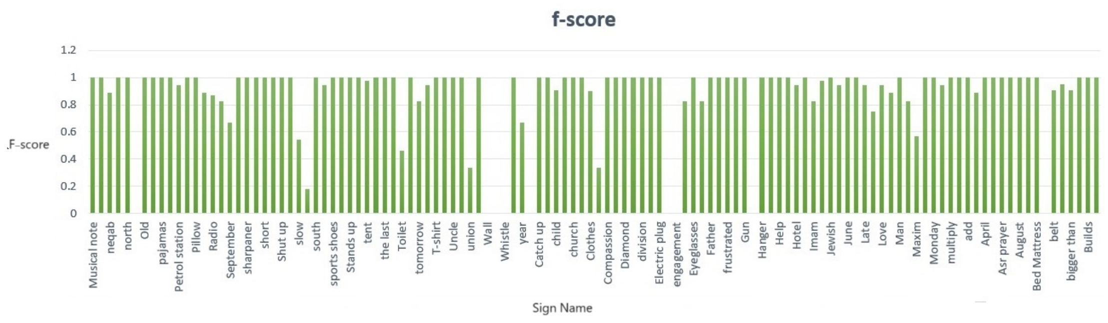


Figure 13. F-score associated with the ASiL model

4.3. Accuracy performance of each sign gesture

Figure 14 details the gesture name used as the class name and the accuracy found after testing the system. Only 14% of signs (i.e., 17 signs) do not have an accuracy different from 1. Among them, only 6.77% (i.e., 8 signs) have an accuracy of less than 0.8. The bed, October, Wall, West, Whistle, chalk, Employee, and engagement signs were not detected and recognized by our model. The system has an average accuracy of 94%.

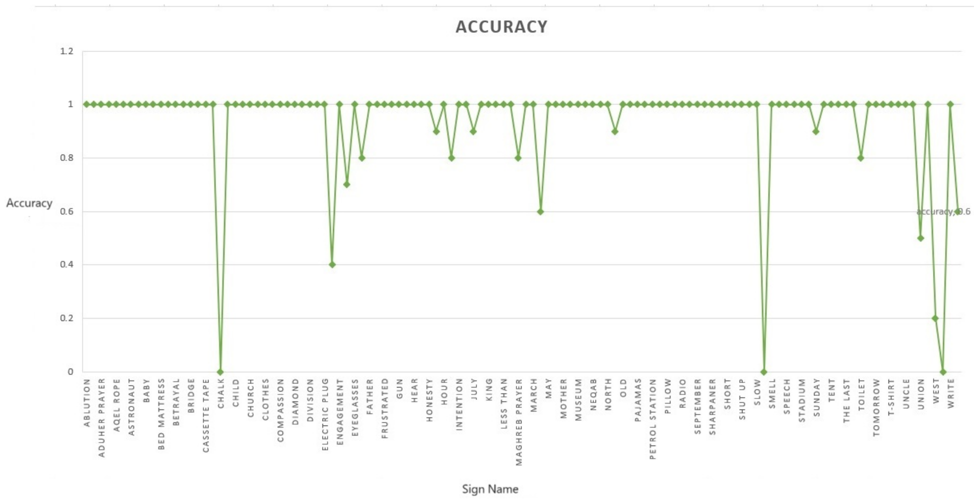


Figure 14. Accuracy for each sign using ASiL model training

4.4. Comparison with other model performances

The final step of the evaluation is the comparison of the other most relevant systems in the literature. Our ability to compare with other current systems in the literature is limited due to the absence of a comparable dataset. In fact, we have created our dataset using the widely used Arabic sign language in the Arab world. Due to this rationale, we have incorporated an alternative version of ASiL that relies on the MobileNet model of Fine-tuning. MobileNet is renowned for its rapidity and efficacy in detecting objects. Figure 15 shows the precision values of our system compared to the ASiL MobileNet version precision values for each sign. We see that most of the precision values are equal to 1. If we compare them to the precision values corresponding to the ASiL MobileNet version, we can notice a significant improvement. It can be seen that most precision values exceed 0.85, which means that the improvement encompasses all concepts. The average precision for ASiL ResNet50 is 91% compared with ASiL MobileNet's is 33%.

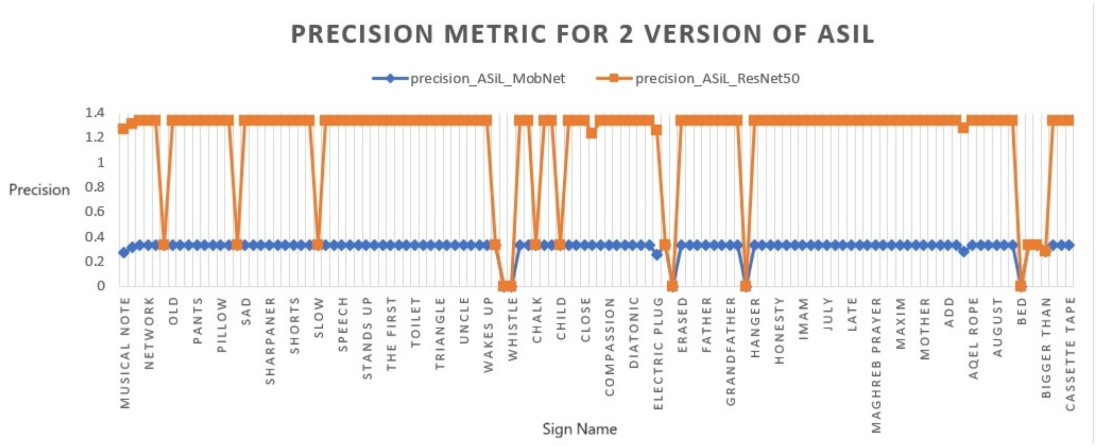


Figure 15. Precision metric for ASiL based on ResNet50 model compared with precision of ASiL based on MobileNet

Likewise, we notice a significant improvement from an F-score point of view, as shown in Figure 16. As seen, the curve of our system is near to 1 compared with the curve corresponding to the MobileNet model. Indeed, the values corresponding to MobileNet are less than 0.6. This value is not acceptable.

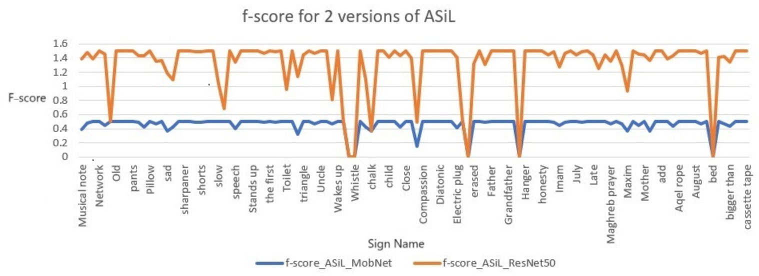


Figure 16. F-score metric for ASiL based on ResNet50 model compared with F-score of ASiL based on MobileNet

Figure 17 shows a comparison of the recall values of the two versions of ASiL (based on ResNet50 and based on MobileNet). Both versions have similar recall results. textcolorblue For ASiL ResNet50 it has a high precision value of 91% and a high recall value of 84.9%, which is very acceptable. As for ASiL MobileNet, it has a low precision value of 33% and a high recall value of 87%, i.e., almost all of the correct answers, but the low precision (for example, 33%) will provide many more incorrect answers as a response. of those correct: it will therefore be difficult to use.

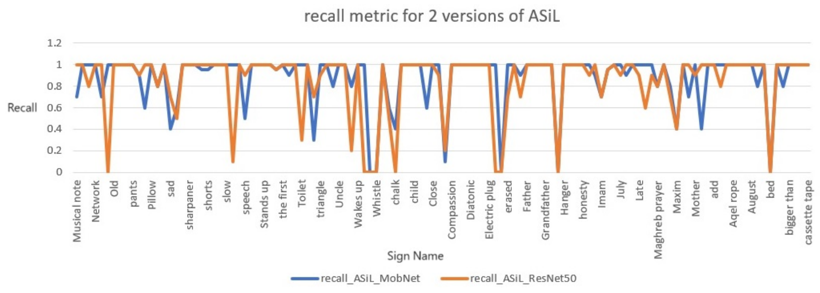


Figure 17. Recall metric for ASiL based on ResNet50 model compared with recall of ASiL based on MobileNet

5. CONCLUSION

This paper introduces a comprehensive approach to addressing the communication gap between deaf individuals who use ArSL and non-deaf individuals. We proposed an AI architecture for the detection and translation of Arabic sign language into English in real-time. Our ASiL platform is based on the ResNet50 model and compared with the MobileNet model.

Through a series of experiments, we demonstrated the system's accuracy using different evaluation methods for training and testing, achieving an impressive average F-score of 86.4% and an accuracy of 94%. Those metrics and others are discussed and presented for each sign. The ASiL platform is a step in the direction of social inclusion rather than only a technological solution. By bridging the language barrier between sign languages and the spoken language, we aim to make it easier for deaf individuals to integrate into society. Therefore, our model is lightweight, which makes it easy to deploy on a smartphone by incorporating it into the tflite format and deploying it in mobile applications on any OS (Android and iOS), which makes communication between deaf-mute and non-deaf more appropriate. In fact, the tflite file containing the trained model measures 50 MB.

Although our system exhibits excellent performance, it nevertheless has significant limitations. Our future plans involve creating a video-based dataset that focuses on Arabic hand sign language. In addition, we want to utilize the recurrent neural network (RNN) on this video dataset and analyze the outcomes. Furthermore, the analysis reveals that out of the total 118 signs, our system fails to detect eight of them. To address this issue, we want to enhance their recognition by improving the preprocessing phases for those signs. Additionally, we have intentions to create a smartphone application. Currently, the initial model is being converted into TensorFlow light format in order to simplify its integration with a mobile application. This application's development will involve validating the technique by testing its effectiveness with actual deaf-mute users. Additionally, it will assess the convenience of use and user-friendliness for the end user.

ACKNOWLEDGEMENT

This research received grant no. (171/2023) from the Arab Observatory for Translation (an affiliate of ALECSO), which is supported by the Literature, Publishing & Translation Commission in Saudi Arabia.

REFERENCES

[1] T.-W. Chong and B.-G. Lee, "American sign language recognition using leap motion controller with machine learning approach," *Sensors*, vol. 18, no. 10, Oct. 2018, doi: 10.3390/s18103554.

[2] H. Luqman and E.-S. M. El-Alfy, "Towards hybrid multimodal manual and non-manual arabic sign language recognition: mArSL database and pilot study," *Electronics*, vol. 10, no. 14, Jul. 2021, doi: 10.3390/electronics10141739.

[3] C. D. D. Monteiro, C. M. Mathew, R. Gutierrez-Osuna, and F. Shipman, "Detecting and identifying sign languages through visual features," in *2016 IEEE International Symposium on Multimedia (ISM)*, Dec. 2016, pp. 287–290, doi: 10.1109/ISM.2016.0063.

[4] M. Borg and K. P. Camilleri, "Towards a transcription system of sign language video resources via motion trajectory factorisation," in *Proceedings of the 2017 ACM Symposium on Document Engineering*, Aug. 2017, pp. 163–172, doi: 10.1145/3103010.3103020.

[5] M. Hamroun, S. Lajmi, M. Jallouli, and A. Souid, "Efficient text-based query based on multi-level and deep-semantic multimedia indexing and retrieval," *Multimedia Tools and Applications*, Nov. 2023, doi: 10.1007/s11042-023-17256-y.

[6] H. Bitar, G. Amoudi, R. Alsulami, And S. Alahmadi, "Building and evaluating an Android mobile App for people with hearing disabilities in Saudi Arabia to provide a real-time video transcript: a design science research study," *Revista Română de Informatică și Automatică*, vol. 31, no. 3, pp. 109–122, 2021, doi: 10.33436/v31i3y202109.

[7] M. Ahmed *et al.*, "Arabic sign language translator," *Journal of Computer Science*, vol. 15, no. 10, pp. 1522–1537, 2019, doi: 10.3844/jcssp.2019.1522.1537.

[8] M. Ahmed *et al.*, "Towards the design of automatic translation system from Arabic sign language to Arabic text," in *2017 International Conference on Inventive Computing and Informatics (ICICI)*, Nov. 2017, pp. 325–330, doi: 10.1109/ICICI.2017.8365365.

[9] D. M. Madhiarasan and P. P. P. Roy, "A comprehensive review of sign language recognition: different types, modalities, and datasets," *arXiv:2204.03328*, Apr. 2022.

[10] World Health Organization, "Deafness and hearing loss," *World Health Organization*, 2023. https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss (accessed Mar. 21, 2023).

[11] A. I. Sidig, H. Luqman, S. Mahmoud, and M. Mohandes, "KArSL: Arabic sign language database," in *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 20, no. 1, pp. 1–19, Jan. 2021, doi: 10.1145/3423420.

[12] G. M. Bin Makhashen, H. A. Luqman, and E.-S. M. El-Alfy, "Using Gabor filter bank with downsampling and SVM for visual sign language alphabet recognition," in *2nd Smart Cities Symposium (SCS 2019)*, 2019, vol. 2019, doi: 10.1049/cp.2019.0188.

[13] A. Alani and G. Cosma, "ArSL-CNN: A convolutional neural network for arabic sign language gesture recognition," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 2, pp. 1096–1107, 2021, doi: 10.11591/ijeecs.v22.i2.pp1096-1107.

[14] H. Luqman and S. A. Mahmoud, "A machine translation system from Arabic sign language to Arabic," *Universal Access in the Information Society*, vol. 19, no. 4, pp. 891–904, 2020, doi: 10.1007/s10209-019-00695-6.

[15] Arab League of Educational, Cultural and Scientific Organisation, "Arabic sign dictionary for the deaf 1," Arab Organization of Sign Language Interpreters, 2021. https://selaa.org/ar/node/204 (accessed Mar. 21, 2023).

[16] Arab League of Educational, Cultural and Scientific Organisation, "Arabic sign dictionary for the deaf 2," Arab Organization of Sign Language Interpreters, 2007. https://selaa.org/node/215 (accessed Mar. 21, 2023).

[17] M. Mustafa, "A study on Arabic sign language recognition for differently abled using advanced machine learning classifiers," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 4101–4115, 2021, doi: 10.1007/s12652-020-01790-w.

[18] M. Mohandes, S. Aliyu, and M. Deriche, "Arabic sign language recognition using the leap motion controller," in *2014 IEEE 23/superscriptrd International Symposium on Industrial Electronics (ISIE)*, Jun. 2014, pp. 960–965, doi: 10.1109/ISIE.2014.6864742.

[19] R. Alzohairi, R. Alghonaim, W. Alshehri, and S. Aloqeely, "Image based Arabic sign language recognition system," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 3, pp. 185–194, 2018, doi: 10.14569/IJACSA.2018.090327.

[20] J. Triesch and C. Von Der Malsburg, "A system for person-independent hand posture recognition against complex backgrounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 12, pp. 1449–1453, 2001, doi: 10.1109/34.977568.

[21] S. K. Yewale and P. K. Bharne, "Hand gesture recognition using different algorithms based on artificial neural network," in *2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, Apr. 2011, pp. 287–292, doi: 10.1109/ETNCC.2011.6255906.

[22] M. HafizurRahman and J. Afrin, "Hand gesture recognition using multiclass support vector machine," *International Journal of Computer Applications*, vol. 74, no. 1, pp. 39–43, 2013, doi: 10.5120/12852-9367.

[23] J. Nagi *et al.*, "Max-pooling convolutional neural networks for vision-based hand gesture recognition," in *2011 IEEE International Conference on Signal and Image Processing Applications*, ICSIPA 2011, 2011, pp. 342–347, doi: 10.1109/ICSIPA.2011.6144164.

[24] Tang, K. Lu, Y. Wang, J. Huang, and H. Li, "A real-time hand posture recognition system using deep neural networks," *ACM Transactions on Intelligent Systems and Technology*, vol. 6, no. 2, pp. 1–23, May 2015, doi: 10.1145/2735952.

[25] S. C. J. and L. A, "SigNet: a deep learning based indian sign language recognition system," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, Apr. 2019, pp. 596–0600, doi: 10.1109/ICCSP.2019.8698006.

[26] Y. Saleh and G. F. Issa, "Arabic sign language recognition through deep neural networks fine-tuning," *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 16, no. 5, May 2020, doi: 10.3991/ijoe.v16i05.13087.

[27] E. E. ElAlfi, M. M. R. El Basuony, and S. M. El Atawy, “Intelligent Arabic text to Arabic sign language translation for easy deaf communication,” *International Journal of Computer Applications*, vol. 92, no. 8, pp. 22–29, Apr. 2014, doi: 10.5120/16030-5126.

[28] S. Al-Rikabi and V. Hafner, “A humanoid robot as a translator from text to sign language,” in *5th Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC 2011)*, Poland, pp. 375–379, 2011.

[29] G. Latif, N. Mohammad, J. Alghazo, R. AlKhalaf, and R. AlKhalaf, “ArASL: Arabic alphabets sign language dataset,” *Data in Brief*, vol. 23, Apr. 2019, doi: 10.1016/j.dib.2019.103777.

[30] M. A. Bencherif *et al.*, “Arabic sign language recognition system using 2D hands and body skeleton data,” *IEEE Access*, vol. 9, pp. 59612–59627, 2021, doi: 10.1109/ACCESS.2021.3069714.

[31] G. Batnasan, M. Gochoo, M.-E. Otgonbold, F. Alnajjar, and T. K. Shih, “ArSL21L: Arabic sign language letter dataset benchmarking and an educational avatar for metaverse applications,” in *2022 IEEE Global Engineering Education Conference (EDUCON)*, Mar. 2022, pp. 1814–1821, doi: 10.1109/EDUCON52537.2022.9766497.

[32] Center of Smart Robotics Research, “KSU-ArSL, Arabic Sign language,” *IEEE Dataport*, 2022, doi: 10.21227/8axv-ma58.

[33] M. and D. Almohimeed, Abdulaziz, Wald, “Arabic text to Arabic sign language translation system for the deaf and hearing-impaired community,” in *EMNLP The Second Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*, Edinburgh, UK, Scotland, 2011, pp. 101–109.

[34] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[35] T.-Y. Lin *et al.*, “Microsoft COCO: common objects in context,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3686–3693.

[36] J. Huang *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 3296–3297, doi: 10.1109/CVPR.2017.351.

[37] T.-Y. Lin *et al.*, “Microsoft COCO: common objects in context,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3686–3693, May 2014.

[38] Tzutalin, “Labelimg,” github.com, 2015. https://github.com/tzutalin/labelImg (accessed Aug. 19, 2023).

[39] TensorFlow, “Object detection,” tensorflow.org, 2023. https://www.tensorflow.org/hub/tutorials/ (accessed Sep. 21, 2023)

[40] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 936–944, doi: 10.1109/CVPR.2017.106.

[41] W. Liu *et al.*, “SSD: single shot multibox detector,” in Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, Vol. 9905., Springer, Cham, 2016, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2.

BIOGRAPHIES OF AUTHORS

Maha Alamri obtained her Ph.D. degree from Bangor University, UK, in 2019. She is an assistant professor in artificial intelligence and serves as the vice dean for the female section in the Faculty of Computing and Information at Al-Baha University, Saudi Arabia. Her research interests include natural language processing and machine learning. She can be contacted at email: m.alamri@bu.edu.sa.

Sonia Lajmi obtained her Ph.D. degree in computer science in 2011 as a co-degree between the National Institute of Applied Science (INSA) of Lyon and the Faculty of Economic and Management Science (FSEG) of Sfax. She began her career as a teacher-researcher at the University of Sfax in 2007. She comes from a background in information systems, with a master’s degree in information systems and multimedia obtained at the University of Sfax in 2005. Since October 2015, Sonia has held a full-time position as a research assistant professor at Al-Baha University in KSA, as assistant head of the IT department. Sonia has also held a position as a full-time assistant professor at the University of Sfax, Tunisia, since October 2011. Her research interests are in data science. Much of her work has focused on improving the annotation of multimedia documents, mainly through the application of data mining, ontology, statistics, and performance evaluation. Since 2023, she has been an ambassador for women in data science in the Albaha region of KSA. She can be contacted at email: sonia.lajmi@isims.usf.tn.