applied sciences

MDPI

Article

Vision Transformers and Transfer Learning Approaches for Arabic Sign Language Recognition

Nojood M. Alharthi and Salha M. Alzahrani *

Department of Computer Science, College of Computers and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia; s44280612@students.tu.edu.sa
* Correspondence: s.zahrani@tu.edu.sa

Abstract: Sign languages are complex, but there are ongoing research efforts in engineering and data science to recognize, understand, and utilize them in real-time applications. Arabic sign language recognition (ArSL) has been examined and applied using various traditional and intelligent methods. However, there have been limited attempts to enhance this process by utilizing pretrained models and large-sized vision transformers designed for image classification tasks. This study aimed to create robust transfer learning models trained on a dataset of 54,049 images depicting 32 alphabets from an ArSL dataset. The goal was to accurately classify these images into their corresponding Arabic alphabets. This study included two methodological parts. The first one was the transfer learning approach, wherein we utilized various pretrained models namely MobileNet, Xception, Inception, InceptionResNet, DenseNet, and BiT, and two vision transformers namely ViT, and Swin. We evaluated different variants from base-sized to large-sized pretrained models and vision transformers with weights initialized from the ImageNet dataset or otherwise randomly. The second part was the deep learning approach using convolutional neural networks (CNNs), wherein several CNN architectures were trained from scratch to be compared with the transfer learning approach. The proposed methods were evaluated using the accuracy, AUC, precision, recall, F1 and loss metrics. The transfer learning approach consistently performed well on the ArSL dataset and outperformed other CNN models. ResNet and InceptionResNet obtained a comparably high performance of 98%. By combining the concepts of transformer-based architecture and pretraining, ViT and Swin leveraged the strengths of both architectures and reduced the number of parameters required for training, making them more efficient and stable than other models and existing studies for ArSL classification. This demonstrates the effectiveness and robustness of using transfer learning with vision transformers for sign language recognition for other low-resourced languages.

Keywords: Arabic sign language; transfer learning; VGG; ResNet; MobileNet; Xception; Inception; DenseNet; InceptionResNet; ViT; Swin; BiT

check for updates

Citation: Alharthi, N.M.; Alzahrani, S.M. Vision Transformers and Transfer Learning Approaches for Arabic Sign Language Recognition. *Appl. Sci.* **2023**, *13*, 11625. https://doi.org/10.3390/app132111625

Received: 18 September 2023
Revised: 17 October 2023
Accepted: 23 October 2023
Published: 24 October 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Hearing loss refers to the inability to hear at the same level as individuals with normal hearing, who have thresholds of 20 decibels or above in both ears. It can range from mild to profound and can occur in one or both ears. Profound hearing loss is typically observed in deaf individuals, who have minimal or no hearing ability. Various factors can contribute to hearing loss, including congenital or early-onset hearing loss during childhood, chronic infections of the middle ear, exposure to loud noises, age-related changes in hearing, and the use of certain drugs that can cause damage to the inner ear [1]. The World Health Organization estimates that more than 1.5 billion people (nearly 20% of the global population) suffer from hearing loss, and this number could rise to over 2.5 billion by 2050 (https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss, accessed on 17 September 2023). According to the General Authority for Statistics in Saudi Arabia, there are approximately 289,355 deaf people in the Kingdom of Saudi Arabia,

according to the latest update for 2017 (https://www.stats.gov.sa/ar/904, accessed on 17 September 2023). Sign language (SL) is a visual form of communication used by deaf and hearing-impaired individuals. It involves the use of hand movements and facial expressions to convey meaning, following specific grammar rules. Each SL has a wide variety of signs, with slight variations in hand shape, motion, and position. Additionally, SL incorporates non-manual features and facial expressions, including eyebrow movements, to enhance communication. Like spoken language, SL develops naturally because of groups of people interacting with one another. Regions and cultures additionally play an essential role in their development [2]. Understanding SL outside the deaf community is uncommon, which makes communication between deaf individuals and ordinary individuals difficult. Some deaf children are born to ordinary parents, and consequently, communication gaps exist within the family. Moreover, there is no standardized form among SLs, which makes learning a SL challenging [3].

There are two main types of SLs, namely, alphabet and ideographic [4]. First is the SL alphabet (SLA), wherein each word is spelled letter by letter, used by deaf individuals to participate in the traditional educational and pedagogical process. Second is ideographic SL (ISL), which expresses each meaningful word by a specific hand gesture; for instance, the word "father" is represented by a sign. ISL shares four main components: hand movements, facial expressions, lip movements, and body movements [4,5]. One difficulty to mention here is that people from different areas might use various gestures to represent the same word just like different dialects in natural languages. There are two main types of SL recognition systems. Device-based systems involve wearable tools, such as gloves to track and interpret gestures [6]. Vision-based systems, on the other hand, utilize techniques for processing and analyzing images and videos of SL's speakers using artificial intelligence (AI). Images and videos are captured by cameras, and there is no need to be equipped with devices or sensors. Vision-based SL recognition systems are in their developmental stages and have varying degrees of success [6].

Arabic SL (ArSL) is a natural language that is used by the deaf and hearing-impaired people in the Arab world and has a large community. ArSL differs from an Arab country to another based on the local dialects and cultures. ArSL recognition is a challenging task because Arabic contains hundreds of words which could be very similar in their hand poses. The gestures are not only played with hands but also include other different nonverbal communication means such as facial expressions and body movements [7]. Different research works have investigated the use of sensors and device-based techniques for ArSL [5,8,9]. These studies have developed a speech recognition engine to convert common Arabic words into ArSL and vice versa. Some studies were designed to measure sign language motion using data gloves [8,9]. To the best of our knowledge, few research works have investigated ArSL vision-based methods which rely on AI techniques including image processing, computer vision, machine learning [3], and deep learning [7,10–12]. Therefore, the contributions of this work can be highlighted in the following points:

- This study explores the use of transfer learning approach using pretrained deep learning models such as VGG, ResNet, MobileNet, Inception, Xception, DenseNet, and InceptionResNet for ArSL classification using a large-size dataset. Additionally, a state-of-the-art pretrained model, BiT (Big Transfer), by Google Research, was evaluated for ArSL classification.
- This study investigates the use of vision transformers namely ViT and Swin, with the transfer learning approach for ArSL classification. Since the literature review studies have shown successful vision-based transformer models applied to other sign languages, these vision transformers are also investigated and evaluated for ArS classification.
- Finally, this research work investigates several deep learning convolutional neural network (CNN) architectures trained from scratch for ArSL classification and compares their results with the proposed transfer learning models for ArSL classification. An

ablation study was conducted to compare different architectural models of CNN for the classification of Arabic sign language alphabets.

The rest of the paper is organized as follows. Section 2 gives an extensive review of SL recognition methods and resources, moving from machine learning to deep learning methods, and transfer learning-based methods for SL classification. Section 3 introduces the methodology framework. Section 4 presents the experimental setup, datasets for ArSL and evaluation metrics. Section 5 presents our experimental results using different architectures of deep learning CNN models, pretrained models, and vision transformers, and finally, in Section 6, we give the conclusion and future remarks to be continued.

2. Related Works

During the past two decades, many research works have been published for automatic SL recognition in different languages and countries [13–16], most of them used device-based sensors such as gloves and wearable devices [17], and Leap Motion Controller (LMC) [7,15,18–22]. As device-based methods are expensive and need the person to be equipped with sensors and due to the recent advances in AI, the current trend and efforts in SL research has been directed to the use of vision-based techniques combined with machine learning and deep learning [23–26]. In this section, extensive literature review studies in SL classification field are discussed from 2020 until present. Prior to these years, review papers by [18,27,28] have investigated SL research works in the gap between 1998 to 2021, which provide a systematic analysis of research employed in SL recognition-related studies. Spanning over a period of two decades, these reviews cover various stages involved in the process, such as image acquisition, image segmentation, feature extraction, and classification algorithms. They also highlight the accomplishments of different researchers in achieving high recognition accuracy. Additionally, the reviews discuss several limitations and obstacles faced by vision-based approaches, including a lack of available datasets, the complexity of certain techniques, the nature of signs themselves, the complexity of backgrounds, and illumination conditions in images and videos.

2.1. Overview of SL Classification Research Based on Methods

For the recognition and classification of hand gestures, several machine learning methods have been employed such as principle component analysis (PCA) [4], support vector machine (SVM) [4,28–30], K-nearest neighbor (KNN) [3,28], and linear discriminant analysis (LDA) [4,31]. Recently, deep learning methods have been employed widely in the literature and we focus on them in this paper. The following methods are commonly used for SL recognition: multilayer perceptron (MLP) [28], convolutional neural network (CNN) [23,30,32–36], recurrent neural network (RNN) [37,38], long short-term memory (LSTM) [32,34], and transfer learning-based methods [14]. Table 1 shows the SL classification research studies using various feature extraction techniques coupled with different machine learning and deep learning methods. As can be seen in the table, different research works have employed CNN for SL classification [23,30,32,34,36]. In ref. [34], researchers developed a real-time sign language recognition system using CNN and a convex hull algorithm to segment the hand region from input images, employing YCbCr skin color segmentation. The CNN model they proposed consisted of an input layer, two convolution layers, pooling layers, a flattening layer, and two dense layers. When tested with real-time data, the system achieved an accuracy of 98.05%. A 3D CNN model was trained from British SL and achieved a 98% accuracy, outperforming state-of-the-art results on British SL [36]. A real-time system was created in ref. [32] to teach SL to beginners, which utilized a technique called skin-cooler modeling to separate the hand region from the background, and CNN was used to classify images. The system achieved a 72.3% accuracy on signed sentences and 89.5% on isolated sign words. In ref. [30], CNN was employed for hand gesture recognition using a modified structure from AlexNet and VGG16 models for feature extraction and SVM for classification of SL numbers and letters. Their two-level models achieved a 99.82% accuracy, better than the state-of-the-art models. Similarly, in ref. [23], CNN and VGG19 were applied for SL classification using single-handed isolated

signs, which achieved 99% with VGG19 and 97% with CNN. RNN is a type of neural network that handles sequential data of different lengths. Unlike other neural networks, RNNs are not limited to fixed-size inputs, allowing for the processing of sequences with varying lengths or images of various sizes [28]. To address the issue of long-term dependencies, LSTM is a specific type of RNN. Regular RNNs struggle with predicting words stored in long-term memory and rely on recent information for predictions. However, LSTM networks have a built-in ability to retain information for extended periods. This is achieved through a memory cell as a container for holding information over time, making them highly suitable for tasks such as language translation, speech recognition, and time series forecasting [28]. Research works introduced RNN for SL classification [37,38], to recognize static signs which attained a 95% accuracy in ref. [37], and to improve a real-time sign recognition system for American sign language, which obtained 99% in ref. [38]. In ref. [32], RNN and LSTM were used to classify SL based on breaking continuous signs into smaller parts and analyzing them with neural networks. This approach eliminated the need to train for different combinations of subunits. The models were tested on 942 signed sentences containing 35 different sign words. The average accuracy achieved was 72% for signed sentences and 89.5% for isolated words.

Table 1. Summary of intelligent vision-based methods using various feature extraction techniques.

Method	SL	Feature Extraction/Characterization	Accuracy	Ref.
SVM	Indian	HOG based on Taguchi and TOPSIS	92% for ISL and 99.2% for ASL	[29]
	American	Used images captured in complex backgrounds	99.80%	[4]
KNN	Arabic	Classified 28-letter Arabic alphabet using 9240 images.	90.25%	[3]
LDA	American	Used five classes with 50 input images.	62%	[31]
MLP	American	Used static ASL gesture	96.96%	[28]
CNN	Italian	Used multiangle images, frontal images only.	99% with VGG19, 97% with CNN	[23]
	American	CNN based on AlexNet and VGG16 for hand gestures	99.82%	[30]
	British	3D CNN for 26 BSL alphabets classification	98.00%	[36]
	Turkish	Employed feature extraction module designed with parallel convolutions with different dilation rates	Best baseline model achieved a 62.02% accuracy.	[34]
	Indian	Used a 2D CNN for feature extraction, where twelve 3D features at time t were processed	72.3% on signed sentences and 89.5% on isolated sign words.	[32]
		3D CNN for videos for eight emergency situations	82%	[2]
	Arabic	Each captured sign was represented by five modalities: color, depth, joint points, face, and face-HD	MobileNet-LSTM 99.7% for signer dependent and 72.4% for signer-independent	[33]
		3D CNN was used to recognize 25 gestures	98% for observed data and 85% for new data	[7]
		3D CNN skeleton network and a 2D point convolution network using 80 static and dynamic signs that were repeated five times by 40 signers	In dependent mode: 98.39% for dynamic, 88.89% for static signs. In independent mode: 96.69% for dynamic, 86.34% for static.	[35]

Table 1. *Cont.*

Method	SL	Feature Extraction/Characterization	Accuracy	Ref.
RNN	American	Sphere radius, angles between fingers, and distance between finger positions using 2600 samples	99.44%	[38]
	Arabic	Recurrent neural networks (RNN)	95.11%.	[37]
LSTM	Indian	942 signed sentences using 35 different sign words	72.3% for signed sentences and 89.5% for isolated sign words	[32]
		Recurrent neural network with long short-term memory (RNN-LSTM)	98%	[2]
Transfer Learning	American	VGG-19	98.44%	[14]
	Arabic	VGG-16 and ResNet152 with ArSL classification	99%	[39,40]
	Indian	VGG-16 and YOLO (You Only Look Once) v5, for advanced object detection'algorithm	99.60%	[2]

In transfer learning, the insights, features, or representations learned from the source task are transferred and utilized to enhance the learning process or performance on the target task. The network's features of the pretrained model are reused, the output of the last layer is removed and replaced by a new layer for a different set of classes of the new task. Transfer learning was used for SL classification in ref. [14], wherein a YCbCr segmentation technique using skin color as a basis and the local binary pattern were utilized to accurately segment shapes and capture texture features and local shape information. The VGG-19 model was adjusted to obtain features, which were then combined with manually crafted features using a serial fusion technique. This approach achieved an accuracy of 98.44%. In ref. [2], a combination of three models including 3D CNN, mixed RNN-LSTM, and YOLO for advanced object recognition in a video dataset created for eight different emergency situations. The frames were extracted and used to evaluate these models, which attained 82% for 3D-CNN, 98% for RNN-LSTM, and 99.6% for YOLO models.

2.2. Overview of SL Classification Research Based on Languages

Many types of SLs have been used in previous studies based on countries and spoken languages. Languages used in SL research are numerous including American [16,22,41–46], Mexican [47], Arabic [3,7,10,11,39,40,48,49], Algerian [50], Pakistani [51], Indian [52–54], Bangla [55], Chinese [56,57], Indonesian [58], Italian [23], Peruvian [59], Turkish [34], and Urdu [24]. Based on [60], which analyzed SL classification research studies published from 2014 to 2021, another breakdown of literature works was presented according to the local variation in the SL they referred to. Most of the works were dedicated to the American SL, but Arabic was also included. Chinese SL was the second after American SL. Table 2 shows a summary of research works in SL based on languages presented in our work in the years between 2020 until the present. According to the number of studies in this very near range of years, American SL is still the most dominant in SL research, and other SLs borrowed many principles from it.

2.3. Overview of SL Classification Research Based on Datasets

Many studies in the literature have utilized public datasets, with a focus on American SL due to its accessibility. Other SLs often require researchers to create their own datasets, which they then publish for other researchers to use. In our review of the previous literature, we found that ArSL itself was diverse due to variations between cities and dialects. One significant challenge for SL research is the limited data availability, particularly for

ArSL [33]. Datasets, also called databases in earlier research on SL, can be obtained through public or private repositories [16,23,47,59]. Datasets of SL can be divided into three main categories: finger spelling, isolated signs, and continuous signs. Finger spelling datasets are based primarily on the shape and direction of the finger. Most of the numbers and alphabets for SL are static and use only fingers. Isolated datasets are equivalent to spoken words and can be static (i.e., images) or dynamic (i.e., videos). Continuous SL datasets are formed from videos of signers representing words and sentences in the language. The evaluation of SL datasets should consider several factors including size, background conditions, and sign representation. One of the factors that determine the variability of an SL database is the number of signers, which is important for evaluating the generalization of recognition systems. Signer-independent recognition systems are evaluated on signers other than those involved in the system training by increasing the number of signers. The number of samples per sign is another factor whereby having several samples per sign with some variations per sample is important for training deep learning models. Samples in SL datasets can be grayscale or RGB format. Any researcher interested in this field can obtain an SL dataset from online repositories such as Kaggle or by building a dataset [3,33–35,51,63,64]. Table 3 summarizes various datasets in terms of isolated vs. continuous, static vs. dynamic, language, recognition model, lexicon size, acquisition mode, signs, signers, single vs. double handed, and background conditions. Existing datasets available for ArSL were explored from the point of view of isolated and continuous signing. To the best of our knowledge, there were only two datasets available for ArSL, where the second was developed from the first by Latif et al. [12]. The ArSL alphabets dataset consisted of 54,049 images compiled by more than 40 volunteers for 32 signs and alphabets. More details are given in Section 4.

Table 2. Summary of SL research reviewed in this work from 2020 to 2023, listed alphabetically according to their acronyms.

Sign Language	Acronym	No of studies	Studies [2020–2023]
Arabic	ArSL	7	[3,7,10,11,39,40,48]
Algerian	AlSL	1	[50]
American	ASL	11	[16,22,38,41–46,61,62]
Bangla	BdSL	1	[55]
British	BSL	1	[36]
Chinese	CSL	2	[26,28,56,57]
Indian	ISL	2	[52–54]
Indonesian	ISINDO	1	[58]
Italian	LIS	1	[23]
Mexican	MSL	1	[47]
Pakistani	PKSL	1	[51]
Peruvian	PSL	1	[59]
Turkish	TSL	1	[28]
Urdu	USL	1	[24]

2.4. Overview of SL Classification Research Based on Arabic Language

ArSL classification deals with the identification of alphabets (SLA) and ideographic (ISL), whereby the latter is based on the diversity of Arabic dialects in countries and their cultures. In 2001, the Arab Federation of the Deaf designated ArSL as the official language for individuals with speech and hearing impairments in Arab countries. Despite Arabic being widely spoken, ArSL is still undergoing development. One major challenge faced by ArSL is "Diglossia," where regional dialects are spoken instead of the written language in each nation. In this regard, different spoken dialects made different ArSLs [18], but still,

few studies have addressed this problem. The focus of existing works has been given to alphabetic ArSL classification [3,10,12]. The datasets used in ArSL research works were mostly collected by the researchers themselves. In ref. [7], a feature extractor with deep behavior was used to deal with the minor details of ArSL. A 3D CNN was used to recognize 25 gestures from the ArSL dictionary. The system achieved a 98% accuracy for observed data and an 85% average accuracy for new data. The results could be improved as more data from more different signers were included. In ref. [11], a vision-based system by applying CNN for recognizing ArSL letters and translating them into Arabic speech was proposed. The proposed system automatically detected hand sign letters and spoke out the result in Arabic with a deep learning model which achieved a 90% accuracy. Their results assured that using deep learning such as CNN was highly dependable and encouraging. In ref. [35], a 3D CNN skeleton network and 2D CNN models were used for ArSL classification using 80 static and dynamic signs. The dataset was created by repeating each sign five times by 40 signers. Their models attained accuracies of 98.39% for dynamic, 88.89% for static signs in the dependent mode, and 96.69% for dynamic, 86.34% for static in the independent mode: When mixing both dependent and independent modes, the results achieved were 89.62% for signer dependent and 88.09% for signer independent. Additional recent research works presented a model utilizing fine-tuning with deep learning for the specific task of recognizing ArSL [39,40], which initiated the research direction of using transfer learning for ArSL and also created the datasets used in this study. CNN models reducing the size of the dataset required for training and at the same time, reaching a higher accuracy, were investigated [40]. Models like VGG-16 and ResNet152 were adjusted using the ArSL dataset. To address the class size imbalance, random undersampling was utilized, reducing the image count from 54,049 to 25,600. Moreover, the study in ref. [39] presented a modified ResNet18 model and reported the best accuracy score compared to the previous models, which was 1.87% higher than the existing works using ResNet. A recent study [49] explored continuous ArSL classification using video data and CNN-LSTM-SelfMLP models with MobileNetV2 and ResNet18 backbones for feature extraction. Their study achieved an 87.7% average accuracy.

Table 3. Summary of the key parameters of sign language recognition studies.

Ref.	Background Conditions	Hands	Signers	Signs	Acquisition Mode	Lexicon Size	Method	Dataset			
[23]	-	-	11	22 signed letters	-	11,008	CNN	LIS	Static	Public	Isolated
[59]	-	Double-handed	-	24 static gestures	HTC Vive, Vive Hand Tracking SDK 0.8.2.	35,400	MLP, SVM KNN, RF	PSL			
[47]	-	-	150	570 words and 30 phrases	Kinect sensor	90,000	MLP	MSL			
[16]	Nonuniform background	Single-handed	-	26 letters and 10 digits	-	-	LMC	ASL	Static + dynamic		
[55]	White backgrounds	Double-handed	-	50 sets of 36 Bangla basic sign characters	Camera	1800-character images	CNN	BdSL			
[3]	Uniform background + nonuniform background	Single-handed	10	28-letter Arabic alphabet	Integrated laptop webcam, smart Huawei mobile camera	9240 images	KNN classifier	ArSL		Self-built	
[34]	Nonuniform background	Single + double	43	226 signs	Microsoft Kinect v2	38,336	CNN	TSL			

Table 3. *Cont.*

Ref.	Background Conditions	Hands	Signers	Signs	Acquisition Mode	Lexicon Size	Method	Dataset			
[63]	Nonuniform background	Single + double	4	20 meaningful words of Arabic signs	Microsoft Kinect v2 camera	7350 RGB videos, 7350 depth videos	single deep networks	ArSL			
[49]	Uniform background	Both	4	Manual, nonmanual gestures for signs	Kinect V2 acquisition system	6667 videos	MobileNetV2, ResNet18, CNN-LSTM-SelfMLP	ArSL			
[33]	Nonuniform background	Single + double	4	50 signs	Kinect V2 sensors	6748 video samples	CNN	CSL	dynamic	Self -built	Continuous
[64]	Nonuniform background	Single + double	50	10,000 utterances each with 10–30 characters	Microsoft Kinect 2.0	50,000 videos	CNN	CSL			
[50]	22 background simple to cluttered		41	65 sign classes		6238 videos	SVM, VGG16-GRU, I3D, I3D-GRU-Attention	AlSL			
[51]	Uniform background	Single-handed	10	36 static alphabet signs + 3 dynamic signs	48MP smart phone camera	5120 images, 353 videos	Bag of words + SVM	PKSL	Static dynamic	Self-built	Mixed
[65]	Uniform background	Single-handed	40	80 signs	Kinect V1, V2, Sony cam	16,000 × 3	Concatenation of 3D and 2D CNN	ArSL	Static	Public	

3. Methodology

3.1. General Framework

The framework in this study was developed using vision-based approaches instead of sensor-based approaches. Several proposed models were designed, developed, and evaluated to obtain the optimum possible results with ArSL recognition and classification. In particular, this work aims to investigate the following:

Problem: Few research works have investigated ArSL vision-based methods which rely on AI techniques. The use of transfer learning with recently built large-sized deep learning models and vision transformers remains a relatively unexplored area of research with ArSL.

Solution: To address the abovementioned problem, this work utilized two methodological parts for the classification of Arabic signs to alphabets. First was the transfer learning approach using pretrained deep learning models including VGG, ResNet, MobileNet, Xception, Inception, InceptionResNet, DenseNet, and BiT, and vision transformers including ViT and Swin. Second was the deep learning approach using CNN architectures, namely, CNN-1, CNN-2, and CNN-3 with data augmentation and batch normalization techniques. The aim of the latter was to be compared with the pretrained models and vision transformers for the ArSL classification task. The general framework of the methodology is shown in Figure 1. The input dataset at a glance was split into training and test splits to ensure that our study did not suffer from data leakage and inflation of accuracy results problems [66]. Different pretrained models using transfer learning were used to classify images in the training and test splits. Each model has its own hyperparameters that we initialized, then, fine-tuning was performed on each model separately using an added

dense layer and one classification layer. On the other hand, three CNN architectures with some variants were proposed with a set of layers which include convolutions, pooling, and other fully connected layers, in order to produce an output of one of a number of possible classes. We proposed three different CNN architectures as can be seen in the figure and trained them from scratch on the dataset. The first two CNNs differed in the number of convolution layers, and the third CNN used image augmentation, batch normalization, and dropout techniques to train the models.

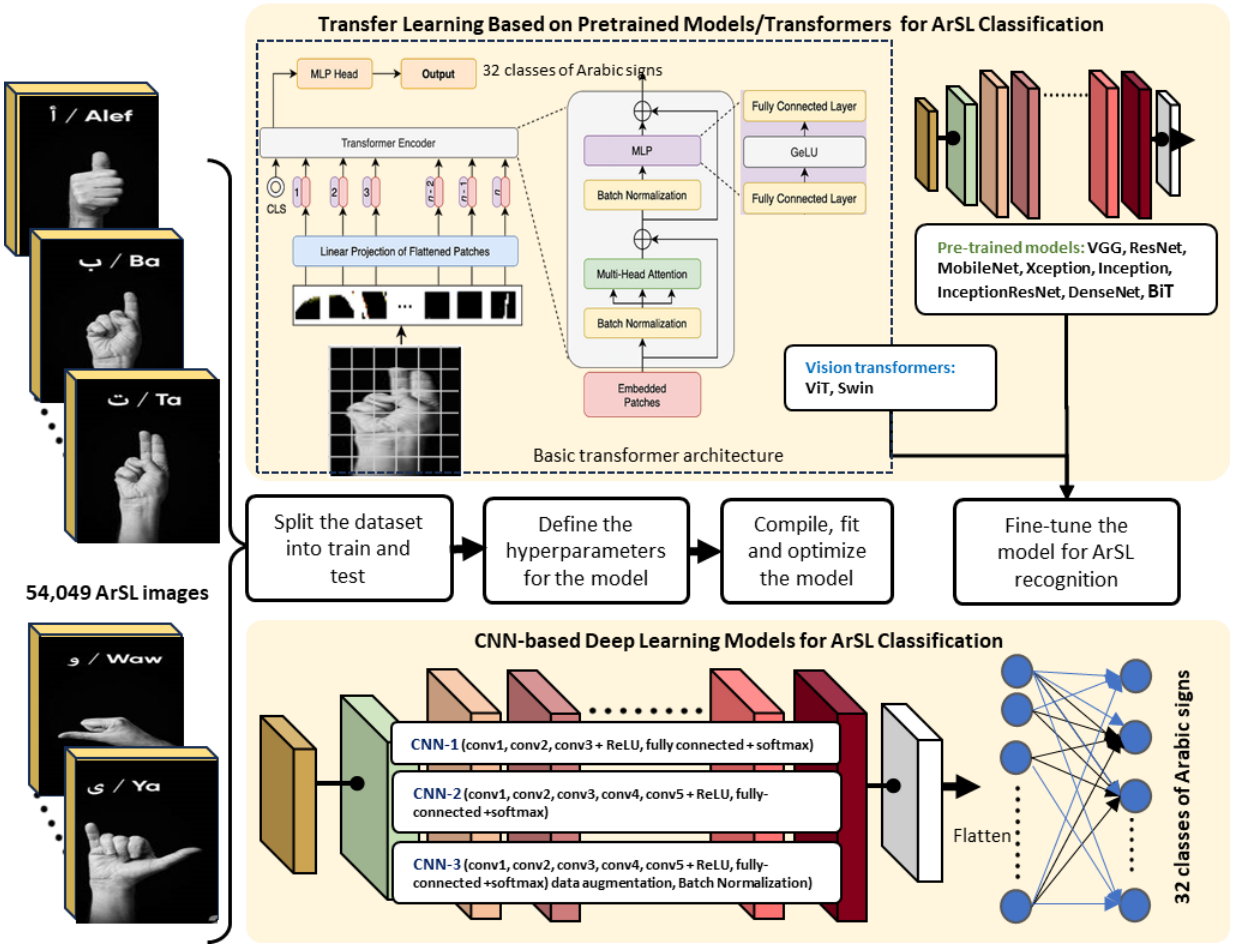


Figure 1. The general framework of the methodology proposed for this study.

3.2. Transfer Learning for ArSL Classification

Image classification is the process of learning features from images and categorizing them according to these features. Throughout this section, different pretrained small- and large-size deep learning models for image classification were utilized. By fine-tuning these models, they learned an inner representation of images that could then be used to extract features useful for downstream tasks. In our research, an ArSL dataset of labeled images were used to train a standard classifier by placing a linear layer on top of the pretrained models. We utilized the transfer learning approach in our experiments, following a series of stages: preprocessing the data, dividing it into segments, fine-tuning the hyperparameters, determining the loss function and optimizer, defining the model architecture, and training it. Once trained, we validated and tested the results. Several deep learning pretrained models were utilized in this study as follows.

VGG was introduced by the Oxford Vision Group and achieved second place in the ImageNet challenge [67]. The VGG family of networks remains popular today and is often used as a benchmark for comparing newer architectures. VGG improved upon previous designs by replacing a single 5×5 layer with two stacked 3×3 layers, or a single

7×7 layer with three stacked 3×3 layers. This approach has several benefits, as using stacked layers reduces the number of weights and operations compared to a single layer with a large filter size. Moreover, stacking multiple layers enhances the discriminative capabilities of the decision function. We utilized in this study two variants: VGG16, as one of the baselines in [39], and VGG19.

ResNet, short for residual neural network, is a deep learning architecture that was introduced by researchers at Microsoft [68]. It has become one of the most popular and influential network architectures in the field of computer vision. The main idea behind ResNet is the use of residual connections, which aim to overcome the degradation problem that arises when deep neural networks are trained. The degradation problem refers to the observation that as the network depth increases, the network's accuracy starts to saturate and then rapidly degrade. To address this problem, ResNet introduces "skip connections" that allow the network to bypass some layers and learn residual mappings. These skip connections connect layers that are not adjacent to each other, allowing the network to skip over certain layers and reuse the learned features from previous layers. By doing so, ResNet ensures that the gradient signals during the training process can flow more easily, preventing the degradation problem. The main building block in ResNet is called a residual block. It consists of two or more convolutional layers, usually with a size of 3×3, and a skip connection that adds the input of the block to the output of the block. This skip connection enables the network to learn the residual mapping, which is the difference between the input and the output of the block. ResNet-152, as another baseline in [40], and ResNet-50, with the number indicating the total number of layers in the network, were used in this study.

MobileNet [69] and MobileNetV2 [70] were also utilized in this study. MobileNet is a convolutional neural network architecture designed to be lightweight and efficient for mobile and embedded devices with limited computational resources. MobileNet achieves this efficiency by using a combination of depthwise separable convolutions and a technique called linear bottlenecks. Depthwise separable convolutions split the standard convolution operation into two separate operations: a depthwise convolution and a pointwise convolution. This reduces the number of parameters and computations required, resulting in a lighter model. Both variants were applied to the ArSL classification dataset used in our study.

Inception [71] and InceptionResNet [72] were fine-tuned for ArSL in this work. The Inception architecture, also known as GoogleNet, is characterized by its use of an Inception module designed to capture multiscale features in an efficient manner. It achieves this by using multiple convolutional filters of different sizes (1×1, 3×3, and 5×5) in parallel and concatenating their outputs. By doing so, the Inception module can capture features at different scales without significantly increasing the number of parameters.

InceptionResNet, on the other hand, combines the Inception architecture with residual connections to allow the network to directly propagate information from one layer to a deeper layer. This helps to alleviate the problem of vanishing gradients and enables the network to learn more easily, even when it becomes very deep. The combination of the Inception module and residual connections in InceptionResNet results in a powerful and efficient architecture, has demonstrated state-of-the-art performance on various benchmark datasets, and has been widely adopted in both research and industry applications. We utilized two variants called InceptionV3 and InceptionResNetV2 for ArSL classification.

Xception, proposed in 2016, stands for "Extreme Inception", as it is an extension of Google's Inception architecture [73]. The main goal of Xception is to improve the efficiency and of deep learning models. It achieves this by replacing traditional convolutional layers with a modified version called depthwise separable convolutions which separate the spatial filtering and channel filtering processes. It applies spatial filtering on each input channel separately, followed by a 1×1 convolution that combines the filtered outputs. This reduces the number of parameters and allows for better learning capabilities. We utilized Xception in our study and evaluated it for ArSL classification.

DenseNet, developed in [74], has a primary building block called the dense block, which consists of a series of layers that are connected to each other in a dense manner. Within a dense block, each layer receives the feature maps of all preceding layers as input, resulting in highly connected feature maps. This encourages feature reuse and enables the network to capture more diverse and abstract features. To ensure that the network does not become excessively large and computationally expensive, DenseNet incorporates a transition layer between dense blocks. The transition layer consists of a convolutional layer followed by a pooling operation, which reduces the number of feature maps, compresses information, and controls the growth of the network. We utilized one variant called DenseNet169.

BiT (Big Transfer by Google Research) [75] was additionally utilized in this study as one of the large-scale pretrained models. It was designed to transfer knowledge from pretraining tasks to downstream tasks more effectively. The BiT model follows a two-step process: pretraining and transfer learning. During pretraining, the model is trained on large datasets containing diverse and unlabeled images. The goal is to learn general visual representations that capture various concepts and patterns in the images. This helps the model develop a strong understanding of the visual world. After pretraining, the knowledge learned by the BiT model is transferred to specific downstream tasks. Transfer learning involves fine-tuning the pretrained model on a smaller labeled dataset that is specific to the target task. By starting with a pretrained model, the transfer learning process benefits from the rich visual representations learned during pretraining. This often leads to improved performance and faster convergence on the target task compared to training a model from scratch. BiT offers a range of model variations, each with a different size and capacity. In this study, BiT helps accelerate transfer learning by fine-tuning it on our ArSL dataset. This approach enables the model to leverage the knowledge gained during pretraining, leading to improved performance and efficiency for the classification of sign language alphabets. This study utilized two variants of BiT architectures namely BiT-m-r50x1 and BiT-m-r50x3 for ArSL recognition.

Additionally, two pretrained vision transformers were utilized for ArSL classification in this study as follows. ViT, introduced in ref. [76], is a vision transformer which treat an image as a sequence of patches, similar to how words are treated in NLP tasks. The image is divided into small patches, and each patch is flattened into a 1D vector. These patch embeddings serve as the input to the transformer model. The transformer architecture consists of multiple layers, including multihead self-attention and feed-forward neural networks. The self-attention mechanism allows the model to attend to different patches and capture the relationships between them. It helps the model understand the context and dependencies between different parts of the image. ViT also incorporates positional embeddings, which provide information about the spatial location of each patch. Two variants of ViT namely ViT_b16 and ViT_132 were used in our work.

Swin transformer was also utilized in this study. According to ref. [77], Swin was built on the Transformer architecture, which consists of multiple layers of self-attention mechanisms to process and understand sequential data. Swin has shifted windows that work by dividing the input image into smaller patches or windows and shifting them during the self-attention process. This approach allows Swin to efficiently process large images without relying on computationally expensive operations, such as sliding window mechanisms or convolutional operations. By shifting the patches, Swin extends the receptive field to capture global dependencies and enables the model to have a better understanding of the visual context. We employed in this work the variant named SwinV2Tiny256 for ArSL classification.

3.3. CNN for ArSL Classification: An Ablation Study

To further validate the performance of pretrained models and vision transformers, we conducted an ablation study on CNN deep learning architectures. The purpose of the study was to investigate and understand the individual contributions and importance of

different components of the CNN models. The study involved systematically removing or disabling specific features, layers, or components of the model and evaluating its impact on the model's overall performance. We targeted three questions in this study: Which layers or modules of the model are essential for achieving a high performance? What is the relative importance or contribution of convolution filters and size to the model? How does the model's performance change when using batch normalization and data augmentation? To this aim, three CNN architectures were utilized. CNN-1 had three convolution layers, three max-pooling layers, one fully connected (dense) layer, and one output classification (softmax) layer. CNN-2 network had five convolution layers, five max-pooling layers, one fully connected (dense) layer, and an output classification (softmax) layer. The convolution layers in both CNN-1 and CNN-2 used filters, which were applied to the entire input data. Each filter was characterized by a set of adjustable weights. The max-pooling layers were used to select the neuron with the highest activation value within each local receptive field, forwarding only that value. The output from the last convolutional layer was flattened. The fully connected layer, also known as the dense layer, was responsible for receiving the flattened input and pasted their weights to the final classification (softmax) layer. This was necessary because the fully connected layer expected a one-dimensional input, while the output of the convolutional layer was three-dimensional. The final layer acted as a classification layer, utilizing the softmax function. This is typically employed in the output layer of a classifier to generate probabilities for each input's class identification. The CNN-3 network was the same as CNN-2 but also applied batch normalization, dropout, and data augmentation. Batch normalization provides a method to process data in the hidden layers of a network, similar to the standard score. It ensures that the outputs of the hidden layer are normalized within each minibatch, maintaining a mean activation value close to 0 and a standard deviation close to 1. This technique can be applied to both convolutional and fully connected layers. When the training data are limited, the network may become prone to overfitting. To address this issue, data augmentation was used to artificially increase the size of the training set. We used common image augmentations including random rotation with 0.2 degrees and random horizontal and vertical flipping. The proposed CNN architectures for ArSL classification are shown in Table 4. ArSL images with three RGB color channels were resized to 224 × 224 pixels before fed as input to the CNN models. In all of our CNN models, we used the leaky ReLU (rectified linear unit) activation function as follows:

$$LeakyReLU(x) = \begin{cases} x \ \ if \ x > 0, \\ \alpha \times x \ \ if \ x \leq 0 \end{cases}$$

where x represents the input to the activation function, and α is a small constant that determines the slope of the function when x is less than or equal to 0. Typically, the value of α is set to a small positive number, such as 0.01, to ensure a small gradient when the input is negative. This helps to address the issue of dead neurons in the ReLU activation function, where neurons can become inactive and stop learning if their inputs are always negative. Leaky ReLU allows for a small gradient and thus enables the learning process even for negative values. By conducting this ablation study into our methodology, we gained insights into which components were critical for the model's performance and which were less important or redundant. This helped understand the underlying behavior of the model, leading to better model design choices for ArSL classification, as shown in Section 5.

Table 4. Architectural details of CNN models for ArSL classification.

Model		Input Shape	No Convolution Layers	Conv. Filters	Conv. Size	Max Pool	Trainable Params	≈Time (s)
CNN-1	CNN-1V1	224 × 224	conv1, conv2, conv3 + leaky ReLU, fully connected layer + softmax	224, 112, 56	3 × 3	2 × 2	1,693,640	5646
	CNN-1V2	224 × 224		64, 64, 64	3 × 3	2 × 2	1,681,312	1499

Table 4. *Cont.*

Model		Input Shape	No Convolution Layers	Conv. Filters	Conv. Size	Max Pool	Trainable Params	≈Time (s)
CNN-2	CNN-2V1	224 × 224	conv1, conv2, conv3, conv4, conv5 + leaky ReLU, fully connected layer + softmax	224, 112, 56, 28, 14	3 × 3	2 × 2	328,346	4262
	CNN-2V2	224 × 224		64, 64, 64, 64, 64	3 × 3	2 × 2	249,888	1128
CNN-3	CNN-3V1	224 × 224	conv1, conv2, conv3, conv4, conv5 + leaky ReLU, fully connected layer + softmax, data augmentation	64, 64, 64, 64, 64	3 × 3	2 × 2	250,144	1793
	CNN-3V2	224 × 224		224, 112, 56, 28, 14	3 × 3	2 × 2	353,612	4306
	CNN-3V3	224 × 224	+1 batch normalization, 1 dropout	256, 126, 64, 62, 16	5 × 5, 3 × 3	2 × 2	430,798	3852
	CNN-3V4	224 × 224	+1 batch normalization, 1 dropout (30 epochs)	256, 126, 64, 62, 16	3 × 3	2 × 2	431,146	11,322

4. Experimental Setup

4.1. Dataset

To evaluate the proposed models, the ArSL dataset presented in ref. [12] was utilized in this work. The dataset contains 54,049 images distributed around 32 classes of Arabic signs. The dimensions of the images are 64 × 64, and many variations of images are presented using different lighting and backgrounds. A sample from the dataset were randomly generated as can be seen in Figure 2. One observation noted was that the number of images per each class was not balanced. Table 5 shows the classification of the Arabic alphabet signs, with labels, the English transliteration of each alphabet, and the number of images, which is also visualized in Figure 3. It has been claimed in ref. [12] that this dataset is sufficient for the training and classification of ArSL alphabets. We did not make any attempts to identify and eliminate the noisy features, as we think their inclusion would enhance the model's resilience and ability to apply them to various situations.



Figure 2. Arabic Sign Language dataset samples.

Table 5. The classification of the Arabic alphabet signs, with labels and number of images.

No	Letter in Arabic	Transliteration	No. of Images	No	Letter in Arabic	Transliteration	No. of Images
1	أ	aleff	1672	17	ظ	dha	1723
2	ب	bb	1791	18	ع	ain	2114
3	ت	taa	1838	19	غ	ghain	1977
4	ث	thaa	1766	20	ف	fa	1955
5	ج	jeem	1552	21	ق	gaaf	1705
6	ح	haa	1526	22	ك	kaaf	1774
7	خ	khaa	1607	23	ل	laam	1832
8	د	dal	1634	24	م	meem	1765
9	ذ	thal	1582	25	ن	nun	1819
10	ر	ra	1659	26	هـ	ha	1592
11	ز	zay	1374	27	و	waw	1371
12	س	seen	1638	28	ئ	ya	1722
13	ش	sheen	1507	29	ة	toot	1791
14	ص	saad	1895	30	ال	al	1343
15	ض	dhad	1670	31	لا	la	1746
16	ط	ta	1816	32	ي	yaa	1293

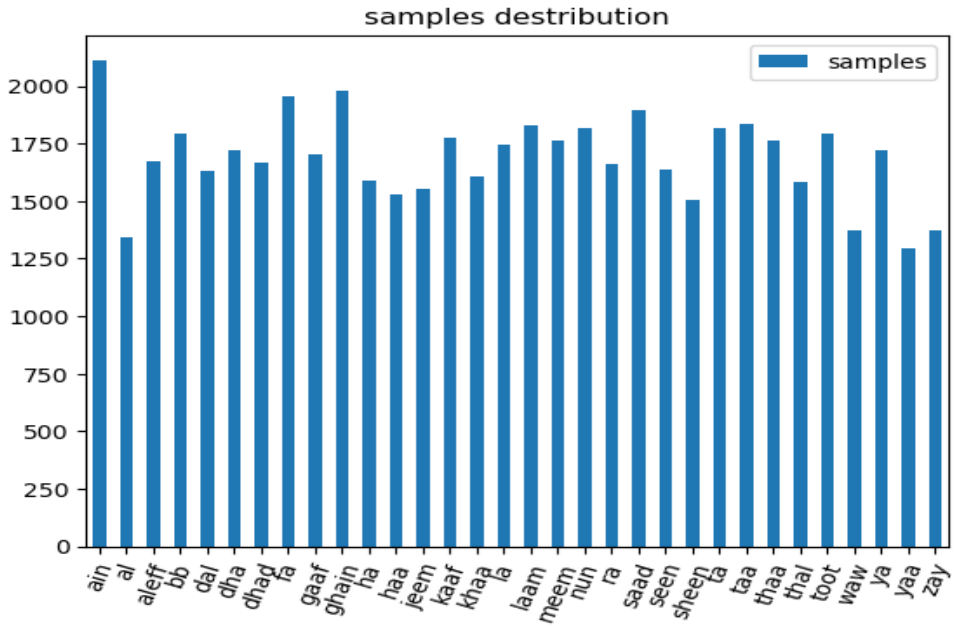


Figure 3. Samples distribution of the Arabic Sign Language dataset.

4.2. Baselines

This work was compared to four baselines from existing work reported in the studies in refs. [39,40]. In these two studies, VGG16, and ResNet152 were fine-tuned for ArSL recognition and classification using our dataset. We also tested VGG19 and ResNet 50 vari-

ants using our dataset to establish comparisons between the performance of the proposed pretrained models and vision transformers and that of the baselines.

4.3. Resources and Tools

The resources utilized in this research included three machines used for our initial experiments, each with Intel (R) Core (TM) i7-10700T 3.2 GHz processors and 16 GB of RAM. For our final experiments, we used a Tesla A100 offered by Google Collab Pro for accelerated deep learning tasks. The A100 GPU is a powerful and high-performance GPU that excels in deep learning workloads by offering NVIDIA Ampere architecture, 6912 CUDA Cores, 432 Tensor Cores and 40 GB of high-bandwidth memory (HBM2). Model training and testing were implemented using TensorFlow 2.8.0, which uses Keras as a high-level API. Complementary libraries for training the proposed models, including Scikit-Learn, Matplotlib, Pandas, NumPy, os, seaborn and glop, were used.

4.4. Evaluation

The effectiveness of the proposed ArSL classification was evaluated based on four major outcomes [32]: true positives (tp), false positives (fp), true negatives (tn), and false negatives (fn). The accuracy determines the ability to differentiate ArSL signs correctly, which was computed by the following formula:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

In deep learning, the loss metric is a measure of how well the model is performing in terms of its ability to make accurate predictions. It represents the discrepancy between the predicted output of the model and the actual output or target value. The loss metric is commonly used to quantify the error or cost associated with the model's predictions. The goal is to minimize this error by adjusting the model's parameters during the training process. The loss measure in this study was computed by the following formula:

$$Loss\ function = \frac{1}{N} \sum \left(y_{pred} - y_{true} \right)^2$$

where y_{pred} is the predicted output of the model, and y_{true} the ground-truth output or target value. We also used AUC, precision, recall, and F1-score to evaluate our models. The AUC (area under the receiver operating characteristic curve) measures the overall performance of a binary classification model across various classification thresholds. It quantifies the model's ability to distinguish between positive and negative samples.

The AUC represents the area under the curve of the receiver operating characteristic (ROC) curve, which plots the true positive rate (recall) against the false positive rate ($1 -$ specificity) at various classification thresholds. Precision measures the proportion of true positive predictions out of all positive predictions made by the model. It indicates how well the model correctly identifies positive samples.

$$Precision = tp \ / \ (tp + fp)$$

Recall, also known as sensitivity or true positive rate, measures the proportion of true positive predictions out of all actual positive samples. It indicates how well the model captures the positive samples.

$$Recall = tp \ / \ (tp + fn)$$

The F1 score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance by considering both precision and recall simultaneously.

$$F1\ Score = 2 \times (Precision \times Recall) \ / \ (Precision + Recall)$$

The F1 score ranges between 0 and 1, with 1 indicating perfect precision and recall, and 0 indicating poor performance.

5. Experimental Results

5.1. Experimental Results from the CNN Approach

CNN-1, CNN-2, and CNN-3 utilized different convolutional and max-pooling layers with various sizes and number of filters. Following the convolution and pooling layers, two fully connected layers were utilized to flatten the output. Eight variants of CNN models were tested wherein the number of filters were changed in the convolution layers. To obtain initial results from these models, the models were trained in one cycle (i.e., one epoch) on 48645 images for training and 5404 files for validation. Then, we used 10 epochs to show the performance of the suggested architectures. The model which showed robust performance was trained for more epochs (30 epochs and 100 epochs).

Figure 4 illustrates the performance of the proposed CNN architectures in terms of accuracy and loss over epochs. Meanwhile, Table 6 presents the results of our ablation study, showcasing metrics such as loss, accuracy, precision, recall, AUC, and F1-score. After noting the unstable performances of CNN-1 and CNN-2 models, we observed that the performance became more consistent with CNN-3 when we implemented augmentation and batch normalization techniques. To further improve the CNN-3 model, we increased the number of training epochs and closely monitored training progress, as illustrated in Figure 5. The results indicated that the model became more resilient and displayed a gradual improvement in accuracy, accompanied by a decrease in loss values. Specifically, after 100 epochs, the CNN-3 model achieved an accuracy of 0.6323. Consequently, we decided to compare its performance with transfer learning methods, which is discussed in Section 5.2.

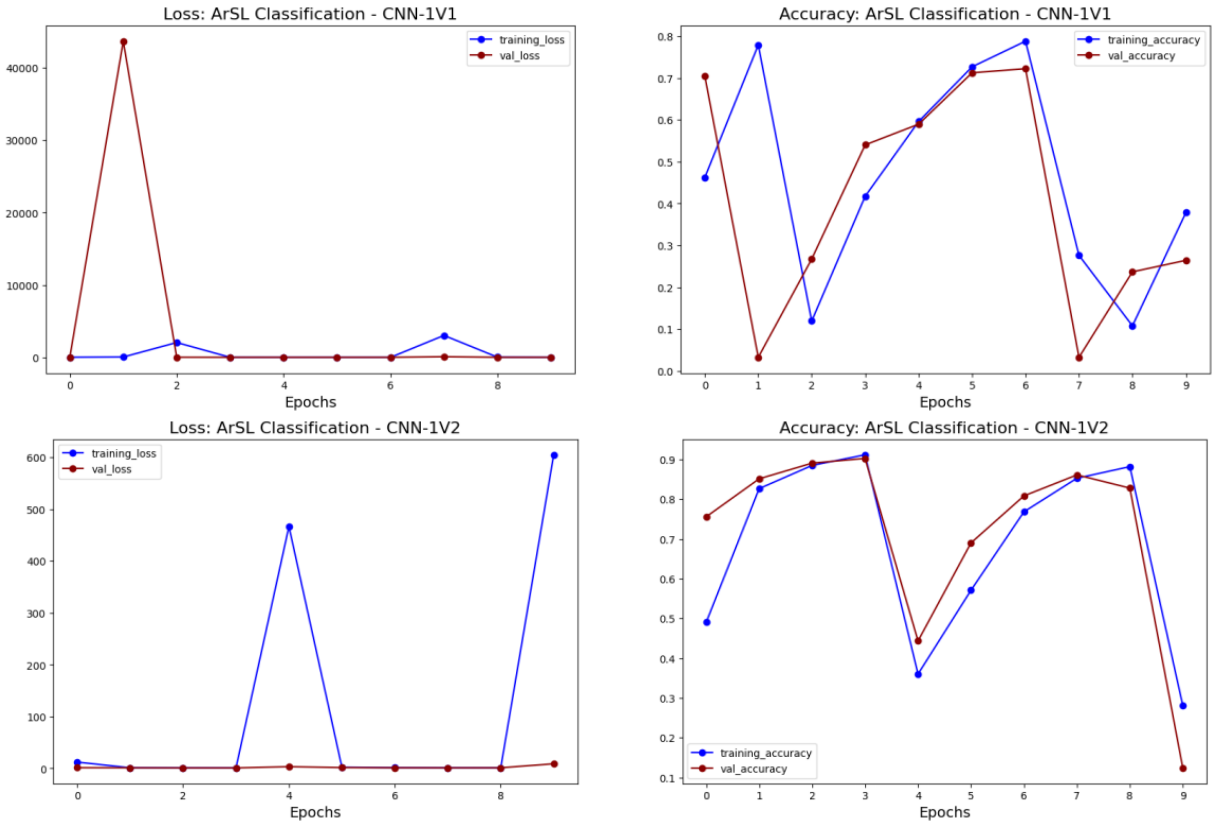


Figure 4. *Cont.*

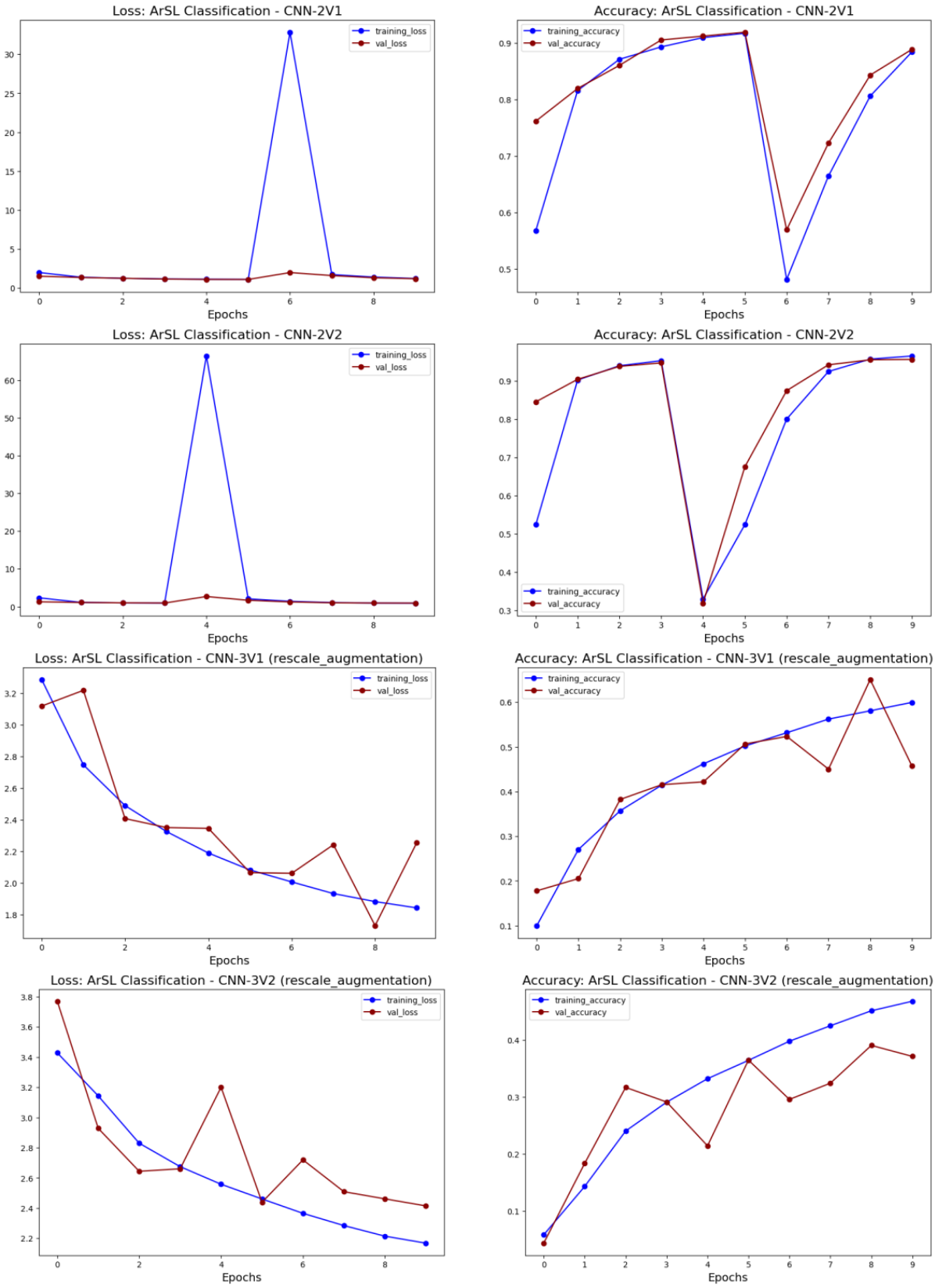


Figure 4. *Cont.*

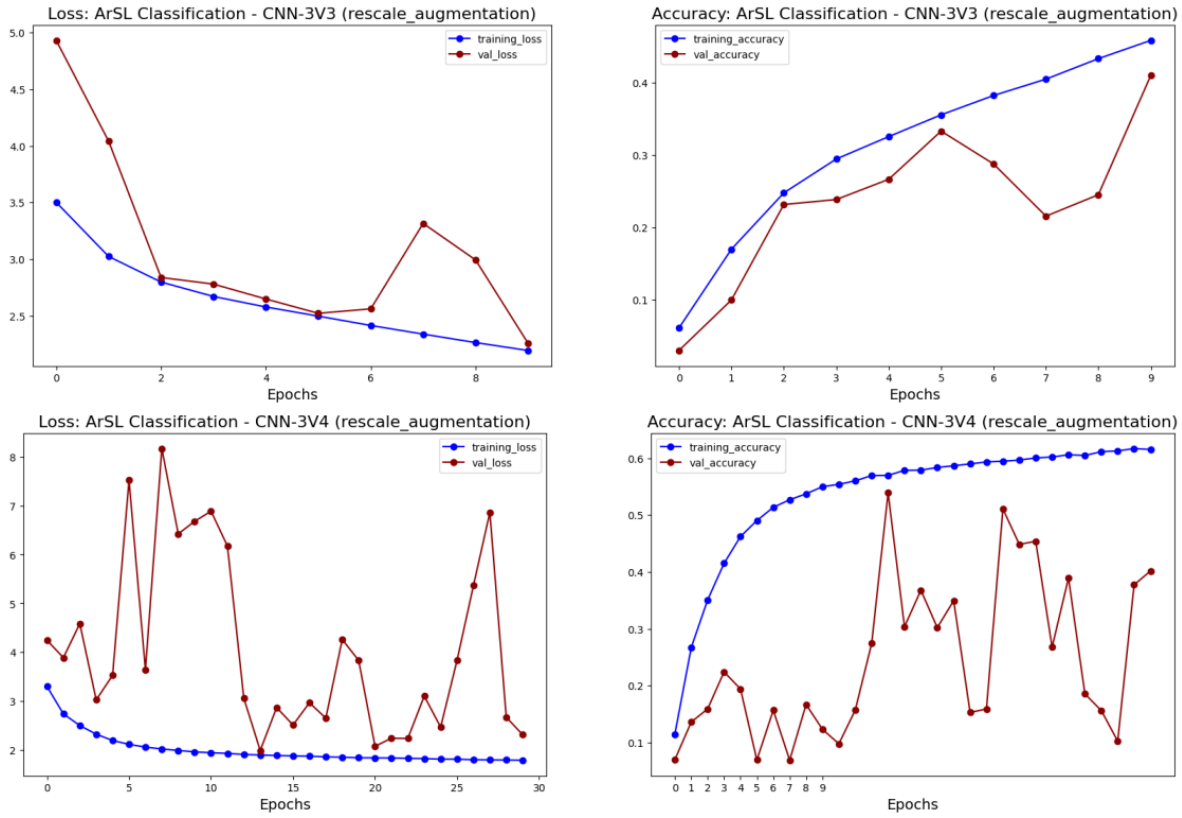


Figure 4. Performance of various CNN model architectures used for ArSL classification.

Table 6. Ablation study results from deep learning CNN models trained for ArSL classification.

	Model	One-Cycle		After Training/Fine-Tuning					
		Accuracy	Loss	Recall	Precision	AUC	Accuracy	Loss	F1
CNN-1	CNN-1V1 (10 epochs)	0.7048	1.6866	0.1995	0.3775	0.7975	0.2644	3.6616	0.2612
	CNN-1V2 (10 epochs)	0.7557	1.5515	0.1208	0.1277	0.6039	0.1244	9.0458	0.1158
CNN-2	CNN-2V1 (10 epochs)	0.7618	1.4914	0.7578	0.9745	0.9936	0.8890	1.1705	0.8487
	CNN-2V2 (10 epochs)	0.8447	1.2781	0.8929	0.9938	0.9982	0.9556	0.9266	0.9402
CNN-3	CNN-3V1 (10 epochs)	0.1778	3.1198	0.2759	0.6394	0.9375	0.4574	2.2542	0.3398
	CNN-3V2 (10 epochs)	0.0433	3.7693	0.1356	0.6222	0.9163	0.3708	2.4147	0.1962
	CNN-3V3 (10 epochs)	0.0298	4.9315	0.1514	0.7491	0.9308	0.4099	2.2581	0.2104
	CNN-3V4 (10 epochs)	0.0696	4.2459	0.2097	0.6267	0.9258	0.4016	2.3162	0.2682
	CNN-3 best (30 epochs)	0.0744	3.3914	0.3932	0.8294	0.9748	0.6377	1.7238	0.5033
	CNN-3 best (100 epochs)	0.0653	4.1342	0.3819	0.8582	0.9734	0.6323	1.7599	0.4842

5.2. Experimental Results from the Transfer Learning Approach

Table 7 shows the experimental results from transfer learning for ArSL classification using 15 variants from the following models: VGG, ResNet, MobileNet, Xception, Inception, InceptionResNet, DenseNet, BiT, ViT, and Swin. In these transformers, the top layer of the model, which is responsible for classification was removed and our own classification layer was added on top. The proposed pretrained models, except for MobileNet, were initialized with pretrained weights from the ImageNet dataset, which contains millions of images and thousands of object categories. MobileNet models were initialized with random weights instead of pretrained weights. VGG19 and ResNet were used to provide powerful base

models for extracting rich features from input images that can be used to train a classifier. As can be seen in the left-half from the table below, using one-cycle of fine-tuning (i.e., one epoch) showed promising results with some models such as ResNet152, while it showed poor performance with large-sized models such as ViT132. The results after one cycle of fine-tuning the models indicated that the pretrained models and vision transformers required additional training to achieve competitive performance.

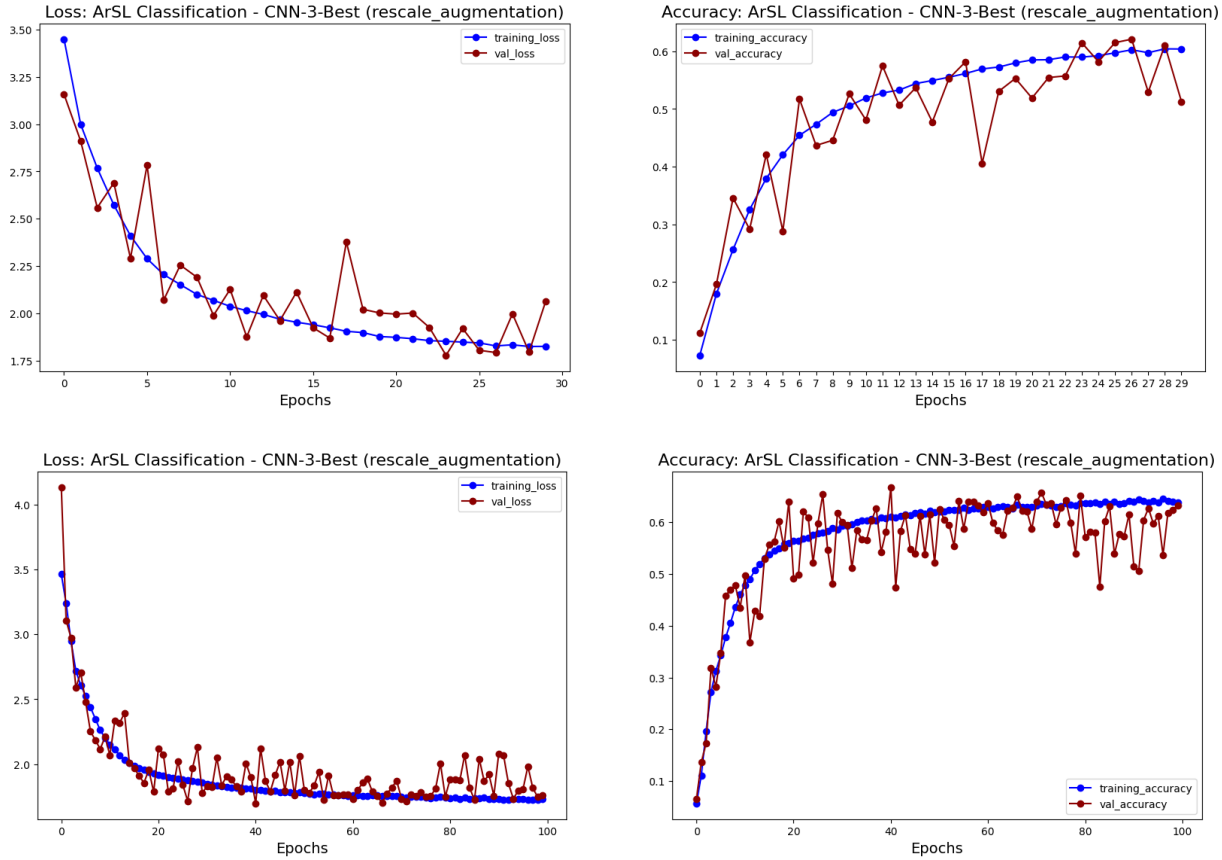


Figure 5. Performance of the CNN-3's best results using a different number of epochs.

To further validate the performance of our models, we also fine-tuned them for several epochs and reported their accuracy and loss results. The right-half of Table 7 shows the experimental results after implementing and fine-tuning the transformers for 10 epochs on 32 classes of signs, 48,645 images for training, and 5404 files for validation. Additionally, we used several figures to monitor the performance of the pretrained models and the vision transformers and that of the baseline models. Figure 6 shows the visualization of pretrained model performance values used for ArSL classification including MobileNet, MobileNetV2, Xception, InceptionV3, InceptionResNetV2, and DenseNet169. We noticed that the pretrained models MobileNet, MobileNetV2, and Xception showed robust learning, while InceptionV3, InceptionResNetV2, and DenseNet169 showed instable performance. Based on the results, after fine-tuning the transformers for ArSL classification, the pretrained models model achieved superior accuracy results in the range between 0.88 and 0.96.

The InceptionResNetV2 model attained the highest accuracy of 0.9817 followed by MobileNet variants which attained an accuracy of 0.96. DenseNet169 achieved the lowest accuracy of 0.5587. Notably, the deep learning CNN-based architecture which was trained from scratch achieved an accuracy of 0.6323, which was low compared to using transfer learning with pretrained models. However, the CNN-3 experimentally chosen structure, the randomly initialed weights, and the long-time of training could change these results.

Hence, we believe that the effectiveness of using transfer learning is highly promising for ArSL recognition.

Table 7. Experimental results from transfer learning for ArSL classification using the baseline models and the proposed models.

Method	Model	Variants	One-Cycle		After Training/Fine-Tuning					
			Accuracy	Loss	Recall	Precision	AUC	Accuracy	Loss	F1
Deep Learning	CNN-3	-	0.0653	4.1342	0.3819	0.8582	0.9734	0.6323	1.7599	0.4842
Transfer Learning	MobileNet	MobileNet	0.9467	0.9608	0.9145	0.9912	0.9992	0.9641	0.8794	0.9500
		MobileNetV2	0.9467	0.9624	0.9169	0.9898	0.9993	0.9665	0.8851	0.9510
	Xception	Xception	0.0365	3.4626	0.8731	0.9778	0.9967	0.9408	0.8821	0.9205
	Inception	InceptionV3	0.8542	1.1860	0.7661	0.9530	0.9953	0.8851	1.0381	0.8420
		InceptionResNetV2	0.7913	7.9896	0.9700	0.9900	0.9995	0.9817	0.7415	0.9800
	DenseNet	DenseNet169	0.6743	1.6610	0.2889	0.8711	0.9276	0.5587	2.0299	0.3888
	BiT	BiT-m-r50x1	0.8690	0.4054	0.9128	0.6505	0.9582	0.9508	0.1901	0.7749
		BiT-m-r50x3	0.8784	0.4404	0.9510	0.4009	0.9570	0.9004	0.5046	0.6185
	ViT	ViT_b16	0.8919	3.4436	0.8414	0.9519	0.9961	0.8919	0.9930	0.8925
		ViT_132	0.0883	3.2602	0.8457	0.9483	0.9934	0.8868	1.0146	0.8928
	Swin	SwinV2Tiny256	0.8769	1.0149	0.9713	0.9820	0.9993	0.9765	0.7284	0.9765
Baselines	VGG	VGG16 [39]	0.8220	15.205	0.8621	0.8629	0.9413	0.8625	16.906	0.8641
		VGG19	0.8312	15.084	0.8558	0.8566	0.9380	0.8558	17.744	0.8558
	ResNet	ResNet50	0.9589	0.8674	0.9719	0.9904	0.9991	0.9824	0.7254	0.9808
		ResNet152 [40]	0.9702	0.8146	0.7948	0.9660	0.9933	0.9110	1.0331	0.8360

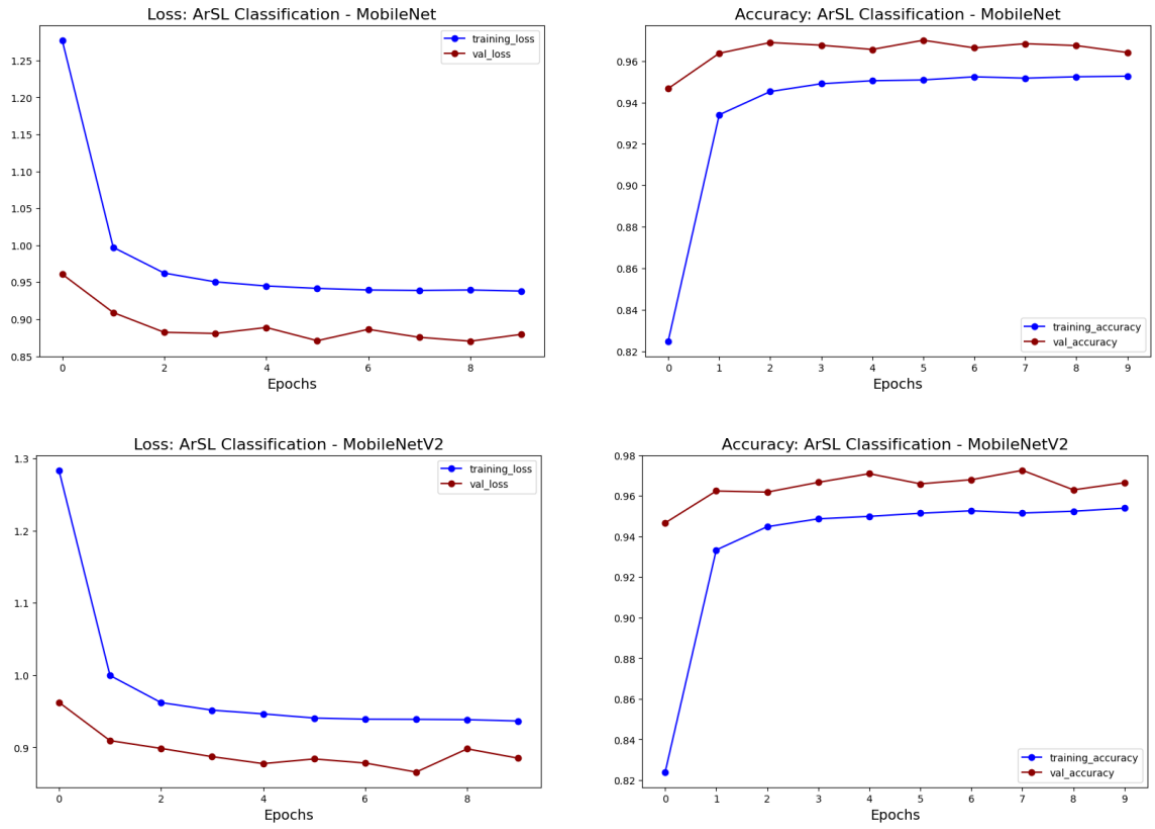


Figure 6. *Cont.*

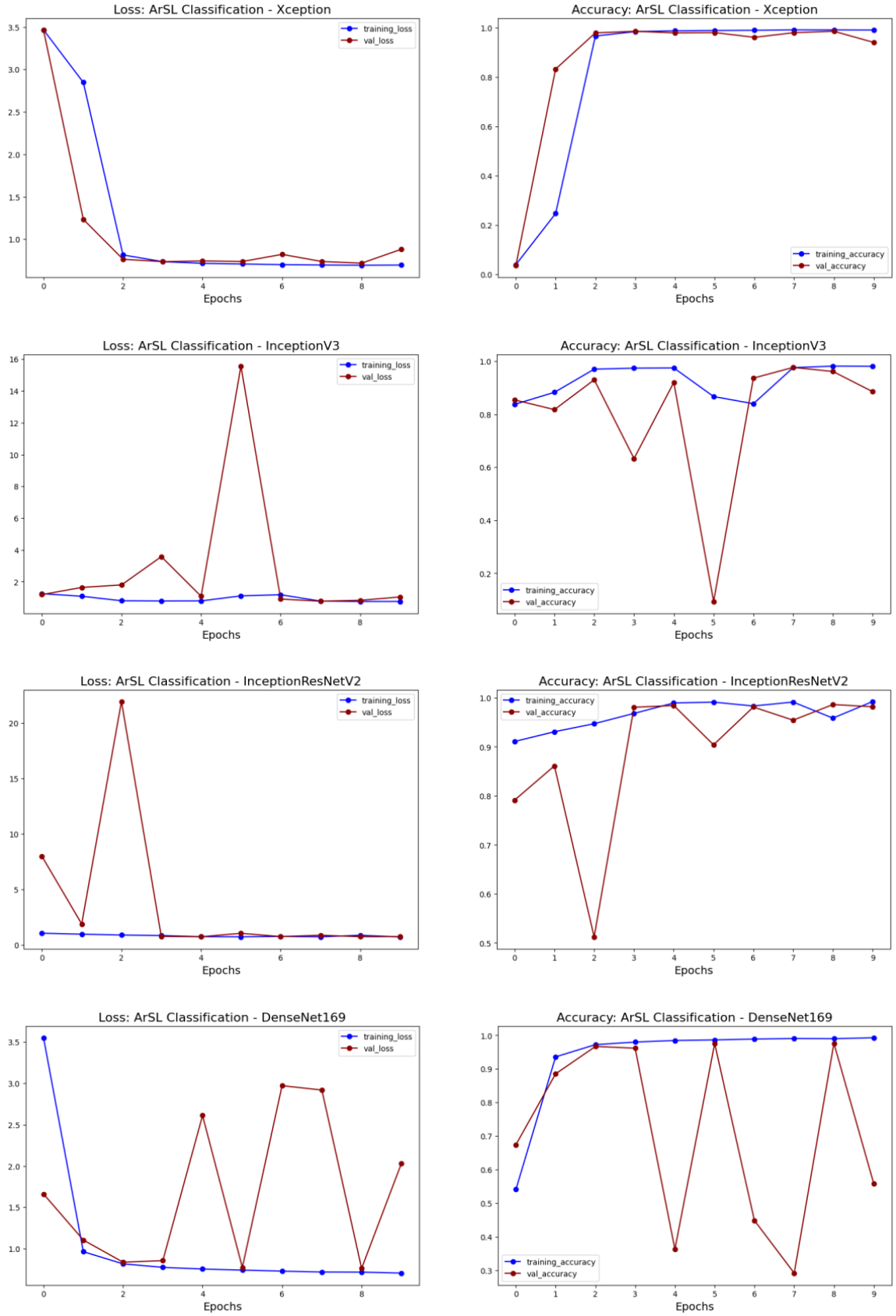


Figure 6. Performance of pretrained deep learning models used for ArSL classification.

Additionally, our findings highlight the importance of selecting appropriate model architectures and hyperparameters to achieve the best possible performance for ArSL tasks. Figure 7 shows the performance of the BiT (Big Transfer) model used for ArSL classification. As can be seen, different variants of BiT may result in different performance values. The accuracy of BiT was above 90%. As another example, different variants of the vision transformers achieved different results as shown in Figure 8. ViT and Swin transformer models achieved great performance with stable learning over time. They achieved an 88–97% accuracy after fine-tuning for classification.

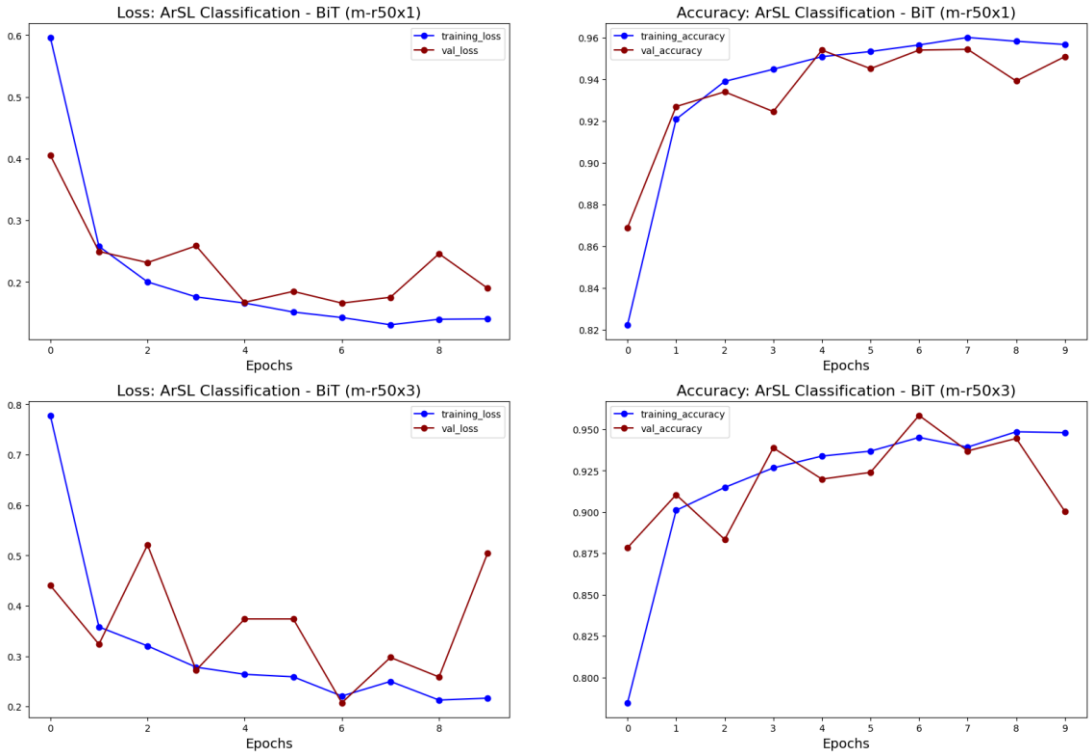


Figure 7. Performance of BiT (Big Transfer) model used for ArSL classification.

As a comparison with other baseline works in ArSL in refs. [39,40], we retrained the VGG16 VGG19, ResNet50, and ResNet152, which attained an 85–98% of accuracy when applied to our dataset. Our findings highlight that these baselines initiated the research direction of using pretrained models and transfer learning for ArSL classification and recognition, and our results from 11 variants of other state-of-the-art models confirmed it as a successful direction. Figures 9 and 10 illustrates the performance of the VGG and ResNet baselines. The results reveal that pretrained CNN architectures can exhibit rapid learning, but they are also susceptible to sudden drops in performance. We found that ResNet and InceptionResNet achieved a higher accuracy compared to the other models. This can be justified by looking at the InceptionResNet architecture that combines the advantages of both the Inception and ResNet models. By combining the concepts of both Inception and ResNet, it leverages the strengths of both architectures to achieve better results in ArSL image classification. Moreover, InceptionResNet incorporates residual connections, allowing for the reuse of features across different layers. This helps to combat the vanishing gradient problem and enables a more efficient training of deep models.

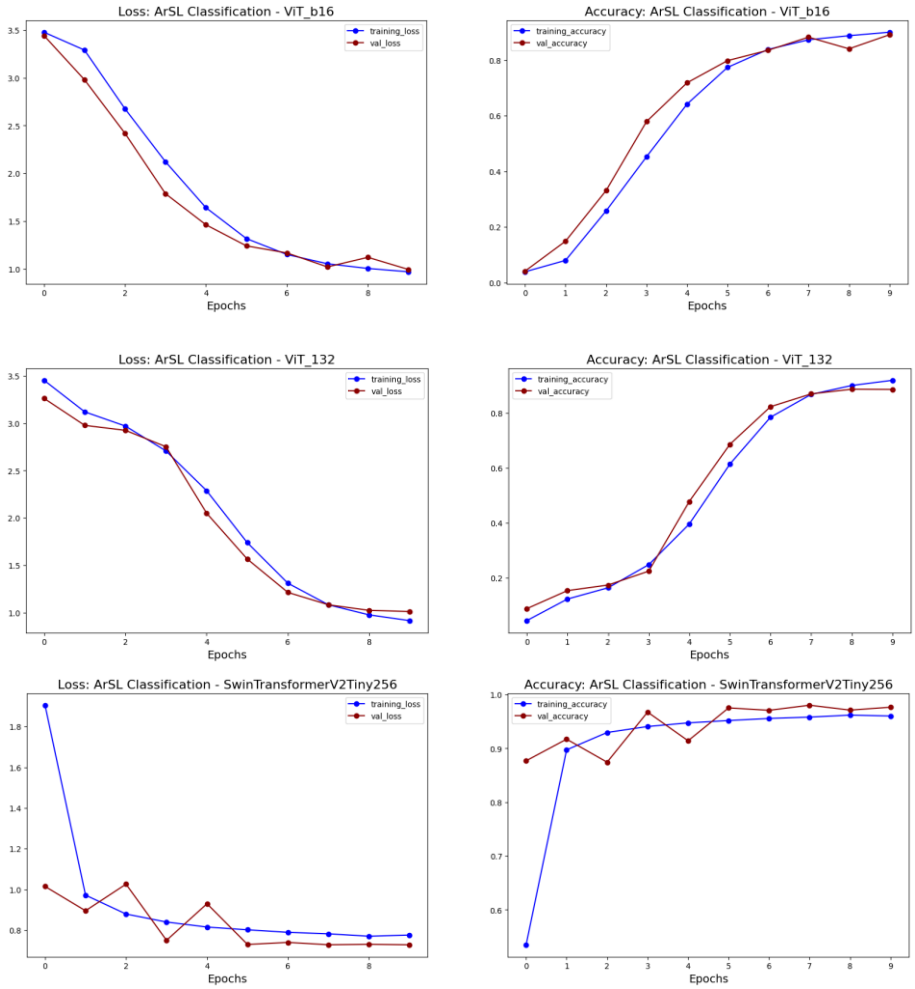


Figure 8. Performance of ViT and Swin vision transformers used for ArSL classification.

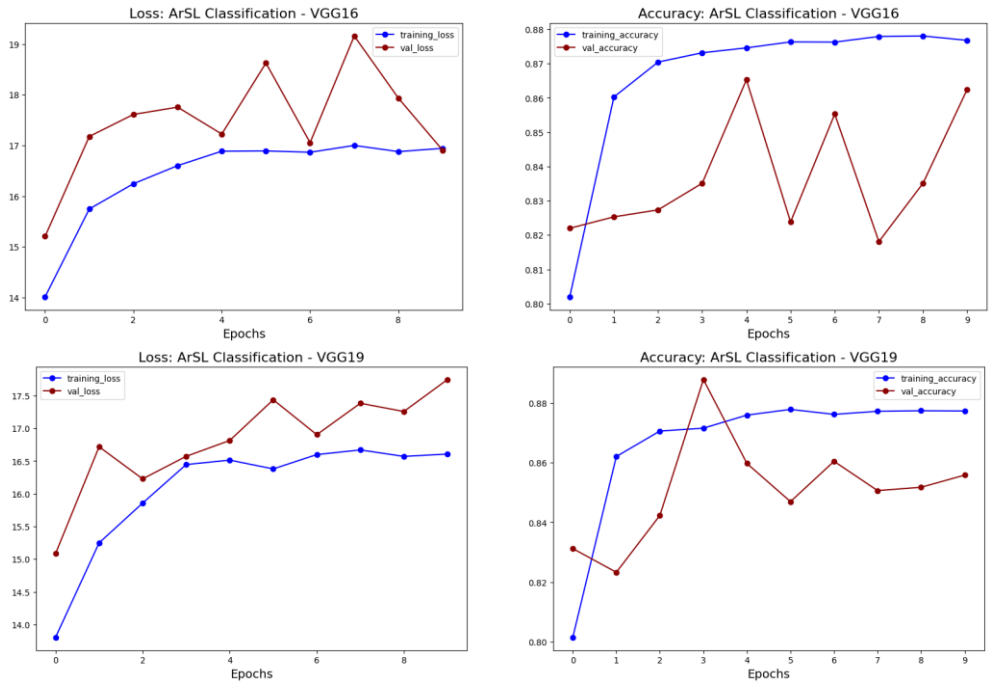


Figure 9. Performance of baseline VGG16 [39] and its variant VGG19.

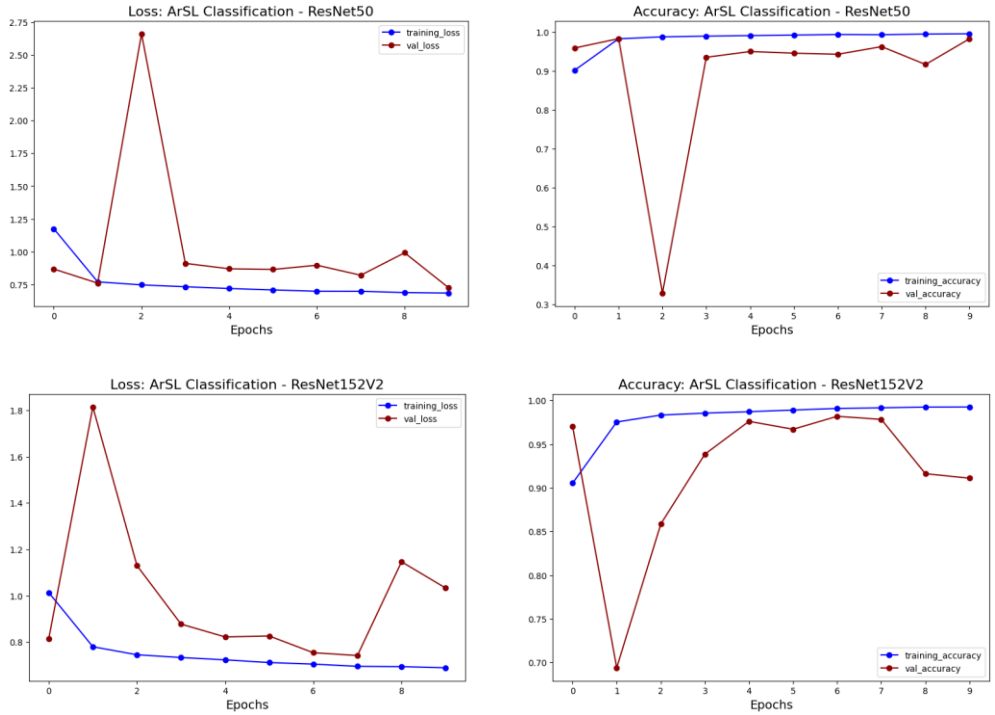


Figure 10. Performance of baseline ResNet152 [40] and its variant ResNet50.

Our experimental study focused on examining the performance of various pretrained models and vision transformers on an isolated ArSL dataset. The results indicated that certain pretrained models, such as MobileNet, Xception, and Inception, tended to overfit the data, as depicted in Figure 6. On the other hand, models such as our CNNs, ViT, and the Big Transfer models exhibited superior learning performance without any signs of overfitting, as demonstrated in Figures 7 and 8.

6. Conclusions and Future Works

This research paper investigated various methods for recognizing Arabic Sign Language (ArSL) using transfer learning. The study utilized pretrained models, originally designed for image classification tasks like VGG, ResNet, MobileNet, Xception, Inception, InceptionResNet, and DenseNet. A pretrained Google Research model named BiT, Big Transfer, was also evaluated on our dataset. Additionally, the paper explored state-of-the-art vision transformers, specifically ViT and Swin, for the task at hand. Different deep learning CNN-based architectures were also employed and compared against the pretrained models and vision transformers. Experimental results revealed that the transfer learning approach, using both pretrained models and vision transformers, achieved a higher accuracy compared to traditional CNN-based deep learning models. Although the pretrained models performed better in terms of accuracy, the vision transformers exhibited more consistent learning. The findings suggest that using recently designed pretrained models such as BiT and InceptionResNet, as well as vision transformers like ViT and Swin, can enhance accuracy, parameter efficiency, feature reuse, scalability, and overall performance in ArSL classification tasks. These advantages make these models a promising choice for future ArSL classification tasks involving words and sentences in image and video datasets. In particular, transfer learning can be a valuable tool for image classification tasks in other sign languages with limited resources and data, outperforming traditional CNN deep learning models in many scenarios. Some limitations of this work are as follows. The dataset was not representative enough to capture the complexity and variability of ArSL words and sentences as a whole. A larger and more diverse dataset could potentially provide a more comprehensive evaluation. Furthermore, practical implementation challenges such as computational resources, model deployment, real-time performance, and

user usability could potentially pose challenges that need to be addressed for real-world applications.

We propose several areas of future research that can enhance the progress of ArSL recognition and sign language understanding. These suggestions can also offer valuable guidance for the creation of robust and efficient models for other sign languages. Some of the recommended future works are as follows:

- Address the imbalance in datasets: Imbalanced datasets can affect the performance of deep learning models. Future work can focus on handling the class imbalance present in ArSL datasets to ensure a fair representation and improve the accuracy of minority classes.
- Extend the research to video-based ArSL recognition: The paper focused on ArSL recognition in images, but future work can expand the research to video-based ArSL recognition. This would involve considering temporal information and exploring techniques such as 3D convolutional networks or temporal transformers to capture motion and dynamic features in sign language videos.
- Investigate hybrid approaches: Explore hybrid approaches that combine both pretrained models and vision transformers. Investigate ways to leverage the strengths of both architectures to achieve an even higher accuracy and stability in ArSL recognition tasks. This could involve combining features from pretrained models with the attention mechanisms of vision transformers.
- Further investigate the fine-tuning strategies and optimization techniques for transfer learning with pretrained models and vision transformers. Explore different hyperparameter settings, learning rate schedules, and regularization methods to improve the performance and stability of the models.
- Explore techniques for data augmentation and generation specifically tailored for ArSL recognition. This can help overcome the challenge of limited data and enhance the performance and generalization of the models. Techniques such as generative adversarial networks (GANs) or domain adaptation can be investigated.
- Investigate cross-language transfer learning: Explore how pretrained models and vision transformers trained on one sign language can be used to improve the performance of models on another sign language with limited resources. This can help address the challenges faced by sign languages with limited data availability.
- Finally, continue benchmarking and comparing different architectures, pretrained models, and vision transformers on ArSL recognition tasks. Evaluate their performance on larger and more diverse datasets to gain a deeper understanding of their capabilities and limitations. This can help identify the most effective models for specific ArSL recognition scenarios.

Author Contributions: Conceptualization, S.M.A.; methodology, N.M.A. and S.M.A.; software, N.M.A. and S.M.A.; validation, S.M.A.; resources, N.M.A.; data curation, S.M.A.; writing—original draft, N.M.A.; writing—review and editing, S.M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: We have made our code available on GitHub, at https://github.com/SalhaAlzahrani/ArSL_Classification (accessed on: 16 October 2023).

Acknowledgments: The researchers would like to acknowledge the Deanship of Scientific Research, Taif University for funding this work.

Conflicts of Interest: The authors declare no conflict of interests.

References

1. May, J.J. Occupational hearing loss. *Am. J. Ind. Med.* **2000**, *37*, 112–120. [CrossRef]
2. Areeb, Q.M.; Maryam; Nadeem, M.; Alroobaea, R.; Anwer, F. Helping Hearing-Impaired in Emergency Situations: A Deep Learning-Based Approach. *IEEE Access* **2022**, *10*, 8502–8517. [CrossRef]
3. Tharwat, G.; Ahmed, A.M.; Bouallegue, B. Arabic Sign Language Recognition System for Alphabets Using Machine Learning Techniques. *J. Electr. Comput. Eng.* **2021**, *2021*, 2995851. [CrossRef]
4. Pan, T.Y.; Lo, L.Y.; Yeh, C.W.; Li, J.W.; Liu, H.T.; Hu, M.C. Real-Time Sign Language Recognition in Complex Background Scene Based on a Hierarchical Clustering Classification Method. In Proceedings of the IEEE Second International Conference on Multimedia Big Data (BigMM), Taipei, Taiwan, 20–22 April 2016; pp. 64–67.
5. Mohammed, R.M.; Kadhem, S.M. A review on Arabic sign language translator systems. *J. Phys. Conf. Ser.* **2021**, *1818*, 012033. [CrossRef]
6. Al-Obodi, A.H.; Al-Hanine, A.M.; Al-Harbi, K.N.; Al-Dawas, M.S.; Al-Shargabi, A.A. A Saudi Sign Language recognition system based on convolutional neural networks. *Build. Serv. Eng. Res. Technol.* **2020**, *13*, 3328–3334. [CrossRef]
7. ElBadawy, M.; Elons, A.S.; Shedeed, H.A.; Tolba, M.F. Arabic sign language recognition with 3D convolutional neural networks. In Proceedings of the 8th International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 5–7 December 2017; pp. 66–71.
8. Baktash, A.Q.; Mohammed, S.L.; Jameel, H.F. Multi-sign language glove based hand talking system. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1105*, 012078. [CrossRef]
9. Sadek, M.I.; Mikhael, M.N.; Mansour, H.A. A new approach for designing a smart glove for Arabic Sign Language Recognition system based on the statistical analysis of the Sign Language. In Proceedings of the 34th National Radio Science Conference (NRSC), Port Said, Egypt, 13–16 March 2017; pp. 380–388.
10. Alsaadi, Z.; Alshamani, E.; Alrehaili, M.; Alrashdi, A.A.D.; Albelwi, S.; Elfaki, A.O. A real time Arabic sign language alphabets (ArSLA) recognition model using deep learning architecture. *Computers* **2022**, *11*, 78. [CrossRef]
11. Kamruzzaman, M.M.; Zhang, Y. Arabic Sign Language Recognition and Generating Arabic Speech Using Convolutional Neural Network. *Wirel. Commun. Mob. Comput.* **2020**, *2020*, 3685614. [CrossRef]
12. Latif, G.; Mohammad, N.; Alghazo, J.; AlKhalaf, R.; AlKhalaf, R. ArASL: Arabic Alphabets Sign Language Dataset. *Data Brief* **2019**, *23*, 103777. [CrossRef]
13. Areeb, Q.M.; Nadeem, M. Deep Learning Based Hand Gesture Recognition for Emergency Situation: A Study on Indian Sign Language. In Proceedings of the International Conference on Data Analytics for Business and Industry (ICDABI), Online, 25–26 October 2021; pp. 33–36.
14. Rajan, R.G.; Leo, M.J. American Sign Language Alphabets Recognition using Hand Crafted and Deep Learning Features. In Proceedings of the International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 26–28 February 2020; pp. 430–434.
15. Aich, D.; Zubair, A.A.; Hasan, K.M.Z.; Nath, A.D.; Hasan, Z. A Deep Learning Approach for Recognizing Bengali Character Sign Langauage. In Proceedings of the 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 1–3 July 2020; pp. 1–5.
16. Chong, T.W.; Lee, B.G. American Sign Language Recognition Using Leap Motion Controller with Machine Learning Approach. *Sensors* **2018**, *18*, 3554. [CrossRef]
17. Rosero-Montalvo, P.D.; Godoy-Trujillo, P.; Flores-Bosmediano, E.; Carrascal-García, J.; Otero-Potosi, S.; Benitez-Pereira, H.; Peluffo-Ordóñez, D.H. Sign Language Recognition Based on Intelligent Glove Using Machine Learning Techniques. In Proceedings of the IEEE Third Ecuador Technical Chapters Meeting (ETCM), Cuenca, Ecuador, 15–19 October 2018; pp. 1–5.
18. Mustafa, M. A study on Arabic sign language recognition for differently abled using advanced machine learning classifiers. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 4101–4115. [CrossRef]
19. Chaikaew, A. An Applied Holistic Landmark with Deep Learning for Thai Sign Language Recognition. In Proceedings of the 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), Phuket, Thailand, 5–8 July 2022; pp. 1046–1049.
20. Bhadra, R.; Kar, S. Sign Language Detection from Hand Gesture Images using Deep Multi-layered Convolution Neural Network. In Proceedings of the IEEE Second International Conference on Control, Measurement and Instrumentation (CMI), Kolkata, India, 8–10 January 2021; pp. 196–200.
21. Htet, S.M.; Aye, B.; Hein, M.M. Myanmar Sign Language Classification using Deep Learning. In Proceedings of the International Conference on Advanced Information Technologies (ICAIT), Yangon, Myanmar, 4–5 November 2020; pp. 200–205.
22. Kasapbaşi, A.; Elbushra, A.E.A.; Al-Hardanee, O.; Yilmaz, A. DeepASLR: A CNN based human computer interface for American Sign Language recognition for hearing-impaired individuals. *Comput. Methods Programs Biomed. Update* **2022**, *2*, 100048. [CrossRef]
23. Schmalz, V.J. Real-time Italian Sign Language Recognition with Deep Learning. In Proceedings of the AIxIA Italian Association for Artificial Intelligence, Milan, Italy, 12 March 2021; pp. 45–57.
24. Zahid, H.; Rashid, M.; Hussain, S.; Azim, F.; Syed, S.A.; Saad, A. Recognition of Urdu sign language: A systematic review of the machine learning classification. *PeerJ. Comput. Sci.* **2022**, *8*, e883. [CrossRef] [PubMed]
25. Tolentino, L.K.; Serfa Juan, R.; Thio-ac, A.; Pamahoy, M.; Forteza, J.; Garcia, X. Static Sign Language Recognition Using Deep Learning. *Int. J. Mach. Learn. Comput.* **2019**, *9*, 821–827. [CrossRef]

26. De Coster, M.; Van Herreweghe, M.; Dambre, J. Sign Language Recognition with Transformer Networks. In Proceedings of the Twelfth Language Resources and Evaluation Conference, Marseille, France, 11–16 May 2020; pp. 6018–6024.
27. Attar, R.K.; Goyal, V.; Goyal, L. State of the Art of Automation in Sign Language: A Systematic Review. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2023**, *22*, 94. [CrossRef]
28. Adeyanju, I.A.; Bello, O.O.; Adegboye, M.A. Machine learning methods for sign language recognition: A critical review and analysis. *Intell. Syst. Appl.* **2021**, *12*, 200056. [CrossRef]
29. Joshi, G.; Singh, S.; Vig, R. Taguchi-TOPSIS based HOG parameter selection for complex background sign language recognition. *J. Vis. Commun. Image Represent.* **2020**, *71*, 102834. [CrossRef]
30. Barbhuiya, A.A.; Karsh, R.K.; Jain, R. CNN based feature extraction and classification for sign language. *Multimed. Tools Appl.* **2021**, *80*, 3051–3069. [CrossRef]
31. Suriya, M.; Sathyapriya, N.; Srinithi, M.; Yesodha, V. Survey on real time sign language recognition system: An LDA approach. In Proceedings of the International Conference on Exploration and Innovations in Engineering and Technology, ICEIET, Wuhan, China, 26–27 March 2016; pp. 219–225.
32. Mittal, A.; Kumar, P.; Roy, P.P.; Balasubramanian, R.; Chaudhuri, B.B. A Modified LSTM Model for Continuous Sign Language Recognition Using Leap Motion. *IEEE Sens. J.* **2019**, *19*, 7056–7063. [CrossRef]
33. Luqman, H.; El-Alfy, E.-S.M. Towards Hybrid Multimodal Manual and Non-Manual Arabic Sign Language Recognition: mArSL Database and Pilot Study. *Electronics* **2021**, *10*, 1739. [CrossRef]
34. Sincan, O.M.; Keles, H.Y. AUTSL: A Large Scale Multi-Modal Turkish Sign Language Dataset and Baseline Methods. *IEEE Access* **2020**, *8*, 181340–181355. [CrossRef]
35. Bencherif, M.A.; Algabri, M.; Mekhtiche, M.A.; Faisal, M.; Alsulaiman, M.; Mathkour, H.; Al-Hammadi, M.; Ghaleb, H. Arabic Sign Language Recognition System Using 2D Hands and Body Skeleton Data. *IEEE Access* **2021**, *9*, 59612–59627. [CrossRef]
36. Kumar, K. DEAF-BSL: Deep lEArning Framework for British Sign Language recognition. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2022**, *21*, 101. [CrossRef]
37. Maraqa, M.; Abu-Zaiter, R. Recognition of Arabic Sign Language (ArSL) using recurrent neural networks. In Proceedings of the 1st International Conference on the Applications of Digital Information and Web Technologies (ICADIWT), Ostrava, Czech Republic, 4–6 August 2008; pp. 478–481.
38. Lee, C.K.M.; Ng, K.K.H.; Chen, C.-H.; Lau, H.C.W.; Chung, S.Y.; Tsoi, T. American sign language recognition and training method with recurrent neural network. *Expert Syst. Appl.* **2021**, *167*, 114403. [CrossRef]
39. Al-Barham, M.; Sa'Aleek, A.A.; Al-Odat, M.; Hamad, G.; Al-Yaman, M.; Elnagar, A. Arabic Sign Language Recognition Using Deep Learning Models. In Proceedings of the 13th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 21–23 June 2022; pp. 226–231.
40. Saleh, Y.; Issa, G.F. Arabic Sign Language Recognition through Deep Neural Networks Fine-Tuning. *Int. J. Online Biomed. Eng. Ijoe* **2020**, *16*, 71–83. [CrossRef]
41. Aly, W.; Aly, S.; Almotairi, S. User-Independent American Sign Language Alphabet Recognition Based on Depth Image and PCANet Features. *IEEE Access* **2019**, *7*, 123138–123150. [CrossRef]
42. Abdullahi, S.B.; Chamnongthai, K. American Sign Language Words Recognition Using Spatio-Temporal Prosodic and Angle Features: A Sequential Learning Approach. *IEEE Access* **2022**, *10*, 15911–15923. [CrossRef]
43. Wu, J.; Sun, L.; Jafari, R. A Wearable System for Recognizing American Sign Language in Real-Time Using IMU and Surface EMG Sensors. *IEEE J. Biomed. Health Inform.* **2016**, *20*, 1281–1290. [CrossRef]
44. Lee, B.G.; Lee, S.M. Smart Wearable Hand Device for Sign Language Interpretation System with Sensors Fusion. *IEEE Sens. J.* **2018**, *18*, 1224–1232. [CrossRef]
45. Li, L.; Jiang, S.; Shull, P.B.; Gu, G. SkinGest: Artificial skin for gesture recognition via filmy stretchable strain sensors. *Adv. Robot.* **2018**, *32*, 1112–1121. [CrossRef]
46. Al Khalissi, R.; Khamess, M. *A Real-Time American Sign Language Recognition System Using Convolutional Neural Network for Real Datasets*; ResearchGate: Berlin, Germany, 2020; Volume 9.
47. Trujillo-Romero, F.; García-Bautista, G. Mexican Sign Language Corpus: Towards an automatic translator. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2023**, *22*, 212. [CrossRef]
48. Al-Shamayleh, A.S.; Ahmad, R.; Jomhari, N.; Abushariah, M.A.M. Automatic Arabic sign language recognition: A review, taxonomy, open challenges, research roadmap and future directions. *Malays. J. Comput. Sci.* **2020**, *33*, 306–343. [CrossRef]
49. Podder, K.K.; Ezeddin, M.; Chowdhury, M.E.H.; Sumon, M.S.I.; Tahir, A.M.; Ayari, M.A.; Dutta, P.; Khandakar, A.; Mahbub, Z.B.; Kadir, M.A. Signer-Independent Arabic Sign Language Recognition System Using Deep Learning Model. *Sensors* **2023**, *23*, 7156. [CrossRef] [PubMed]
50. Khellas, K.; Seghir, R. Alabib-65: A Realistic Dataset for Algerian Sign Language Recognition. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2023**, *22*, 178. [CrossRef]
51. Mirza, M.S.; Munaf, S.M.; Azim, F.; Ali, S.; Khan, S.J. Vision-based Pakistani sign language recognition using bag-of-words and support vector machines. *Sci. Rep.* **2022**, *12*, 21325. [CrossRef] [PubMed]
52. Adithya, V.; Vinod, P.R.; Gopalakrishnan, U. Artificial neural network based method for Indian sign language recognition. In Proceedings of the IEEE Conference on Information & Communication Technologies, Tamil Nadu, India, 11–12 April 2013; pp. 1080–1085.

53. Dhivyasri, S.; KB, K.H.; Akash, M.; Sona, M.; Divyapriya, S.; Krishnaveni, V. An Efficient Approach for Interpretation of Indian Sign Language using Machine Learning. In Proceedings of the 3rd International Conference on Signal Processing and Communication (ICPSC), Coimbatore, India, 13–14 May 2021; pp. 130–133.
54. Kumar, P.; Kaur, S. Sign Language Generation System Based on Indian Sign Language Grammar. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2020**, *19*, 54. [CrossRef]
55. Islam, M.S.; Mousumi, S.S.S.; Jessan, N.A.; Rabby, A.S.A.; Hossain, S.A. Ishara-Lipi: The First Complete MultipurposeOpen Access Dataset of Isolated Characters for Bangla Sign Language. In Proceedings of the International Conference on Bangla Speech and Language Processing (ICBSLP), Sylhet, Bangladesh, 21–22 September 2018; pp. 1–4.
56. Kamal, S.M.; Chen, Y.; Li, S.; Shi, X.; Zheng, J. Technical Approaches to Chinese Sign Language Processing: A Review. *IEEE Access* **2019**, *7*, 96926–96935. [CrossRef]
57. Jiang, X.; Satapathy, S.; Yang, L.; Wang, S.-H.; Zhang, Y. A Survey on Artificial Intelligence in Chinese Sign Language Recognition. *Arab. J. Sci. Eng.* **2020**, *45*, 9859–9894. [CrossRef]
58. Daniels, S.; Suciati, N.; Fathichah, C. Indonesian Sign Language Recognition using YOLO Method. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1077*, 012029. [CrossRef]
59. Nureña-Jara, R.; Ramos-Carrión, C.; Shiguihara-Juárez, P. Data collection of 3D spatial features of gestures from static peruvian sign language alphabet for sign language recognition. In Proceedings of the IEEE Engineering International Research Conference (EIRCON), Lima, Peru, 21–23 October 2020; pp. 1–4.
60. Al-Qurishi, M.; Khalid, T.; Souissi, R. Deep Learning for Sign Language Recognition: Current Techniques, Benchmarks, and Open Issues. *IEEE Access* **2021**, *9*, 126917–126951. [CrossRef]
61. Sharma, S.; Kumar, K. ASL-3DCNN: American sign language recognition technique using 3-D convolutional neural networks. *Multimed. Tools Appl.* **2021**, *80*, 26319–26331. [CrossRef]
62. Jain, V.; Jain, A.; Chauhan, A.; Kotla, S.S.; Gautam, A. American Sign Language recognition using Support Vector Machine and Convolutional Neural Network. *Int. J. Inf. Technol.* **2021**, *13*, 1193–1200. [CrossRef]
63. Abdallah, M.; Hemayed, E. Dynamic Hand Gesture Recognition of Arabic Sign Language using Hand Motion Trajectory Features. *Glob. J. Comput. Sci. Technol. Graph. Vis.* **2013**, *13*, 26–33.
64. Yuan, T.; Sah, S.; Ananthanarayana, T.; Zhang, C.; Bhat, A.; Gandhi, S.; Ptucha, R. Large Scale Sign Language Interpretation. In Proceedings of the 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019), Lille, France, 14–18 May 2019; pp. 1–5.
65. Singh, D.K. 3D-CNN based Dynamic Gesture Recognition for Indian Sign Language Modeling. *Procedia Comput. Sci.* **2021**, *189*, 76–83. [CrossRef]
66. Tampu, I.E.; Eklund, A.; Haj-Hosseini, N. Inflation of test accuracy due to data leakage in deep learning-based classification of OCT images. *Sci. Data* **2022**, *9*, 580. [CrossRef]
67. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556. [CrossRef]
68. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385. [CrossRef]
69. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861. [CrossRef]
70. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arXiv* **2018**, arXiv:1801.04381. [CrossRef]
71. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. *arXiv* **2015**, arXiv:1512.00567. [CrossRef]
72. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv* **2016**, arXiv:1602.07261. [CrossRef]
73. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv* **2016**, arXiv:1610.02357. [CrossRef]
74. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. *arXiv* **2016**, arXiv:1608.06993. [CrossRef]
75. Kolesnikov, A.; Beyer, L.; Zhai, X.; Puigcerver, J.; Yung, J.; Gelly, S.; Houlsby, N. Big Transfer (BiT): General Visual Representation Learning. *arXiv* **2020**, arXiv:1912.11370. [CrossRef]
76. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16 × 16 Words: Transformers for Image Recognition at Scale. *arXiv* **2020**, arXiv:2010.11929. [CrossRef]
77. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *arXiv* **2021**, arXiv:2103.14030. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.