

سلطان سنان

Journal_Paper.docx

 Midocean University

Document Details

Submission ID

trn:oid::2945:404667188

Submission Date

Aug 4, 2026, 5:25 PM GMT+3

Download Date

Aug 4, 2026, 5:59 PM GMT+3

File Name

Journal_Paper.docx

File Size

650.3 KB

24 Pages

10,452 Words

58,252 Characters

11% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

-  **102 Not Cited or Quoted 10%**
Matches with neither in-text citation nor quotation marks
-  **16 Missing Quotations 1%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 6%  Internet sources
- 8%  Publications
- 9%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- 102 Not Cited or Quoted 10%**
Matches with neither in-text citation nor quotation marks
- 16 Missing Quotations 1%**
Matches that are still very similar to source material
- 0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 6% Internet sources
- 8% Publications
- 9% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

- Internet**
www.mdpi.com <1%
- Internet**
public-pages-files-2025.frontiersin.org <1%
- Internet**
www.preprints.org <1%
- Internet**
ouci.dntb.gov.ua <1%
- Internet**
libweb.kpfu.ru <1%
- Submitted works**
Sultan Qaboos University on 2024-05-17 <1%
- Submitted works**
University of Hull on 2023-09-11 <1%
- Internet**
mdpi-res.com <1%
- Internet**
www.techscience.com <1%
- Publication**
David Olatunde Olukanni, Oluseyi Olanrewaju Ajayi, Joshua Olusegun Okeniyi, Sh... <1%

11	Internet	city-test.eprints-hosting.org	<1%
12	Submitted works	Midocean University on 2026-05-02	<1%
13	Internet	dokumen.pub	<1%
14	Internet	arxiv.org	<1%
15	Publication	Mina Nagy Gaber Sorial, Rodaina Abdelsalam, Mayar Ali, Hesham Hassan. "Enhan...	<1%
16	Publication	Poonam Nandal, Tapas Kumar, Meeta Singh, Mamta Dahiya. "Progressive Compu...	<1%
17	Internet	dirros.openscience.si	<1%
18	Publication	Md Atiqur Rahman Jishan, Md Mahmudul Alom Sifat, Khan Mahabi Hoque Nayna, ...	<1%
19	Internet	www.i-csrs.org	<1%
20	Publication	"ICT Systems and Sustainability", Springer Science and Business Media LLC, 2026	<1%
21	Publication	Lecture Notes in Electrical Engineering, 2016.	<1%
22	Submitted works	October University for Modern Sciences and Arts (MSA) on 2023-06-10	<1%
23	Internet	multiple.chat	<1%
24	Internet	www.hindawi.com	<1%

25	Publication	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dharendra Kumar Shukla. "Intelli...	<1%
26	Submitted works	Midocean University on 2026-05-02	<1%
27	Publication	Nahlah Algethami, Raghad Farhud, Manal Alghamdi, Huda Almutairi, Maha Soran...	<1%
28	Submitted works	Technological University Dublin on 2023-07-26	<1%
29	Submitted works	University of Applied Sciences Berlin on 2019-02-12	<1%
30	Submitted works	University of San Diego on 2026-08-04	<1%
31	Publication	V. Sekhar, P. Ajay Kumar Reddy, K. Logesh, T. Raghunadha Reddy, Dara Raju. "Ne...	<1%
32	Internet	github.com	<1%
33	Submitted works	Midocean University on 2026-05-02	<1%
34	Publication	Pushpa Choudhary, Sambit Satpathy, Arvind Dagur, Dharendra Kumar Shukla. "Re...	<1%
35	Publication	Samuel, Benjamin. "Real-Time Action Classification for Olympic Saber Fencing Usi...	<1%
36	Internet	journal.uui.ac.id	<1%
37	Submitted works	De Montfort University on 2025-06-26	<1%
38	Publication	S. Renjith, Poorna S S. "A novel spatio-temporal memory routing network for isol...	<1%

39	Publication	Abdenmour Boulesnane, Lyna Bellil, Maissoun Ghouzlen Ghiri. "A hybrid CNN-ran...	<1%
40	Submitted works	International University Schloss Reichardshausen on 2022-04-04	<1%
41	Publication	Basel A. Dabwan, Mukti E. Jadhav. "Arabic Sign Language Detection Using Deep L...	<1%
42	Publication	Kok-Kwang Phoon, Chong Tang, Zi-Jun Cao. "Machine Learning for Data-Centric G...	<1%
43	Submitted works	Majan College on 2026-06-03	<1%
44	Submitted works	Universiti Teknologi MARA on 2026-07-08	<1%
45	Publication	Cenk Temizel, Salih Tutun, Celal Hakan Canbaz, Ekrem Alagoz et al. "Artificial Inte...	<1%
46	Submitted works	Coventry University on 2024-11-27	<1%
47	Publication	Farahmandnia, Feraidoon. "Enhanced DDoS Attack Detection Through Hybrid Ma...	<1%
48	Publication	Hamzah Luqman, El-Sayed M. El-Alfy, Galal M. BinMakhashen. "Joint space repres...	<1%
49	Submitted works	ICTS on 2026-07-31	<1%
50	Submitted works	Jordan University of Science & Technology on 2021-01-15	<1%
51	Submitted works	Majmaah University on 2019-10-23	<1%
52	Submitted works	RV College of Engineering, Bengaluru on 2026-07-27	<1%

53	Publication	Stein, Kyle Richard. "Integrating Transformers for Cyber Defense Under Unseen a...	<1%
54	Submitted works	Taibah University on 2026-05-09	<1%
55	Submitted works	Tilburg University on 2026-06-22	<1%
56	Submitted works	University of South Florida on 2026-07-14	<1%
57	Submitted works	University of Sydney on 2025-05-30	<1%
58	Submitted works	University of the Western Cape on 2010-06-29	<1%
59	Internet	ebin.pub	<1%
60	Internet	vpshs.eq.edu.au	<1%
61	Internet	www.aanda.org	<1%
62	Publication	"Universal Access in Human-Computer Interaction. Applications and Practice", Sp...	<1%
63	Publication	Alsharif, Bader. "Advances in Real-Time American Sign Language Recognition Sys...	<1%
64	Publication	Amit Kumar Tyagi, Ajith Abraham. "Recurrent Neural Networks", CRC Press, 2022	<1%
65	Submitted works	Arab American University - Jenin on 2026-07-01	<1%
66	Submitted works	Canvas on 2026-06-22	<1%

67	Publication	Iacopo Carnacina, Mawada Abdellatif, Manolia Andredaki, James Cooper, Darren ...	<1%
68	Submitted works	Liverpool John Moores University on 2023-10-30	<1%
69	Submitted works	Nanyang Polytechnic on 2026-02-08	<1%
70	Submitted works	Siksha 'O' Anusandhan University on 2026-05-29	<1%
71	Publication	Sunil Kumar, Rama Krishna Gupta Potnuru. "Empowering Emerging Economies th...	<1%
72	Submitted works	Thesis 2026 on 2026-03-22	<1%
73	Submitted works	Thesis 2026 on 2026-05-12	<1%
74	Internet	baptiste-wicht.com	<1%
75	Internet	icaiit.org	<1%
76	Submitted works	Higher Education Commission Pakistan on 2018-12-23	<1%
77	Publication	Mogeeb A. A. Mosleh, Abdu H. Gumaiei. "An Efficient Bidirectional Android Transla...	<1%
78	Publication	Sarra Rouabhi, Redouane Tlemsani, Nabil Neggaz. "Real-time mobile application f...	<1%
79	Submitted works	The Hong Kong Polytechnic University on 2013-08-12	<1%
80	Submitted works	University of Nottingham on 2024-05-03	<1%

81	Submitted works	
University of Sheffield on 2007-06-21		<1%
82	Publication	
Wahiba Ismaiel, Lilia kechiche, Yassine Aribi, Omer Salih Dawood Omer, Walied M...		<1%
83	Publication	
Asmaa Alayed. "Machine Learning and Deep Learning Approaches for Arabic Sign...		<1%
84	Publication	
Bhabani Prasanna Pattanaik, Dharendra Kumar Shukla, Sambit Satpathy, Kamala...		<1%

Midocean Journal for Research and Studies<https://mojr.midocean.edu.km/issues>

e-ISSN: 3078-5669

**Deep Learning Techniques for Arabic Sign Language Recognition**

تقنيات التعلم العميق للتعرف على لغة الإشارة العربية

Sultan Ali Senan

ORCID ID: 0009-0001-2302-7041

Artificial Intelligence – College of Informatics – Midocean University – United Arab Emirates

Email: 253002563@st.midocean.edu.km

Dr. Amjad Qtaish

ORCID ID: 0000-0001-5210-7086

Artificial Intelligence – College of Informatics – Midocean University – United Arab Emirates

Email: amjadqtaish@midocean.edu.km

سلطان علي سنان

الذكاء الاصطناعي – كلية المعلوماتية – جامعة ميد أوشن – الإمارات العربية المتحدة

د. أمجد قطيش

الذكاء الاصطناعي – كلية المعلوماتية – جامعة ميد أوشن – الإمارات العربية المتحدة

Abstract:

Objective: To design, implement and evaluate a deep learning based recognition system for the static Arabic Sign Language manual alphabet, and to conduct a controlled comparative benchmark between conventional machine learning algorithms and deep learning architectures on the same standardized dataset in order to determine the optimal architecture with respect to performance against training cost. **Methodology:** The publicly available Arabic Alphabets Sign Language dataset, consisting of 54,049 grayscale images divided into 32 alphabet classes, has been employed. Seven models have been implemented: four conventional classifiers, such as Random Forest, Linear Support Vector Machine, HistGradientBoosting and Multilayer Perceptron, and three deep neural architectures, including custom convolutional network, Light ResNet and MobileNetV2 transfer learning model. Stratified train/val split of 80%/20% with a fixed random seed has been used. **Results:** Proposed convolutional network demonstrated highest validation accuracy of 94.88%, MobileNetV2 – 93.64% and Light ResNet – 80.97%. As for the classifiers, Random Forest was the most efficient with 80.25% accuracy and Linear SVM – the worst with 47.85%. Training time of the proposed model was 103.54 seconds compared to 280.73 seconds for transfer learning model. Analysis per class of the proposed model provides the highest macro-averaged F1 score of 0.9490 with residual errors being concentrated in a small group of visually similar letters. **Conclusion:** Custom-designed compact architecture can achieve high validation accuracy on grayscale domain-specific imagery with significantly lower computation costs compared to larger pre-trained one. The study is limited to static alphabet signs in stratified (not signer-independent) split, and the prototype developed produces text output on screen only; speech synthesis has not been implemented and is seen as future work.

Keywords: Arabic Sign Language, Deep Learning, Convolutional Neural Network, Model Benchmarking, Static Alphabet Recognition, Computational Cost.

المستخلص:

الهدف: تصميم وتنفيذ وتقييم نظام للتعرف على أبجدية لغة الإشارة العربية الساكنة باستخدام التعلم العميق، وإجراء دراسة مقارنة مضبوطة بين خوارزميات التعلم الآلي التقليدية ومعماريات التعلم العميق على مجموعة البيانات المعيارية نفسها، بهدف تحديد المعمارية الأنسب من حيث الموازنة بين الأداء وتكلفة التدريب. المنهجية: اعتمدت مجموعة البيانات المتاحة للعموم Arabic Alphabets Sign Language والمكونة من 54,049 صورة رمادية موزعة على 32 فئة أبجدية. نُفذت سبعة نماذج: أربعة مصنفات تقليدية هي الغابة العشوائية وآلة المتجهات الداعمة الخطية والتعزيز التدرجي المُدرج والإدراك متعدد الطبقات، وثلاث معماريات عصبية عميقة هي شبكة التلافيف مخصصة ونسخة مخففة من ResNet ونموذج النقل التعلّمي MobileNetV2. استُخدم تقسيم طبقي بنسبة 80% للتدريب و20% للتحقق مع بذرة عشوائية ثابتة. النتائج: حققت الشبكة الالتفافية المقترحة أعلى دقة تحقق بلغت 94.88%، تلتها MobileNetV2 بنسبة 93.64% ثم ResNet المخففة بنسبة 80.97%. أما بين المصنفات التقليدية فكانت الغابة العشوائية الأفضل بدقة 80.25%، بينما سجلت آلة المتجهات الداعمة الخطية أدنى نتيجة عند 47.85%. بلغ زمن تدريب النموذج المقترح 103.54 ثانية مقابل 280.73 ثانية لنموذج النقل التعلّمي. ويعطي تحليل الأداء لكل فئة متوسط F1 مرجحاً كلياً قدره 0.9490، مع تركّز الخطأ المتبقي في مجموعة صغيرة من الحروف المتشابهة بصرياً. الخلاصة: يمكن لمعمارية مدمجة مصممة لغرض محدد أن تبلغ دقة تحقق عالية على صور رمادية خاصة بالمجال مع استهلاك حسابي أقل بكثير من نموذج مُدرّب مسبقاً وأكبر حجماً. تقتصر الدراسة على الإشارات الأبجدية الساكنة ضمن تقسيم طبقي غير مستقل عن المُشير، ويُخرج النموذج الأولي المنشور نصّاً على الشاشة فقط؛ ولم يُنفذ التركيب الصوتي للكلام وقد حُدّد كعمل مستقبلي.

الكلمات المفتاحية: لغة الإشارة العربية، التعلم العميق، الشبكات العصبية الالتفافية، المقارنة المرجعية للنماذج، التعرف على الأبجدية الساكنة، التكلفة الحسابية.

1. Introduction:

Communication is one of the basic needs of humankind, yet millions of deaf and hard-of-hearing individuals continue to experience difficulties in their everyday life due to the issues related to accessibility. Sign languages are considered to be the most widespread means of communication among deaf communities; however, automatic translation from sign language into written and spoken language is an extremely challenging computational task. Arabic Sign Language is utilized in over twenty Arab countries, yet it remains relatively poorly researched in artificial intelligence field as compared to widely researched sign languages, such as American and British Sign Languages.

In contrast to other sign languages, Arabic Sign Language is a complete language and does not represent the manual form of spoken Arabic language. This language has its own grammar and conveys the meaning through combining manual features, namely hand shape, orientation, location, and movement, as well as non-manual features like facial expression and head movement. The manual alphabet, comprising 32 signs for all letters of written Arabic language, has a special status in Arabic Sign Language and is mostly used for spelling proper names, technical terms, and words without signs. Therefore, automatic detection of the manual alphabet is the crucial aspect of Arabic Sign Language recognition and the primary goal of this study.

The most distinctive characteristic of the technology of Arabic Sign Language is the regional diversity. In the absence of the centrally organized process of development of Arabic Sign Language unlike other sign languages, differences in vocabulary and articulation emerge in different Arab countries. This feature makes building datasets difficult and can explain why there

are less publicly available Arabic Sign Language resources than American and British Sign Language resources.

Modern methods of deep learning technology caused revolution in computer vision and pattern recognition. Convolutional Neural Networks turned out to be effective at recognizing visual patterns, so they could become a perfect tool for sign language recognition that involves identifying hand shape, finger configuration, and spatial orientation. However, the issue of the optimal architecture to be used with any particular sign language dataset is not usually determined by conducting controlled experiments; instead, people make their decisions depending on various results that are obtained under heterogeneous conditions.

1.1 Statement of the Problem

Despite many achievements made in the field of sign language recognition, technological support for Arabic Sign Language remains rather insufficient. Deaf and hard of hearing individuals in the Arab region suffer from the problems of communication that arise because of lack of automatic sign language recognition systems and the shortage of certified interpreters. The latter problem is stated in the latest literature on the subject: according to Noor (2024), the lack of certified interpreters in Saudi Arabia is the main driving factor behind the development of automatic recognition; Podder et al. (2023) make the same argument in favor of the need for the signer-independent systems. Latif et al. (2019) note that lack of publicly available Arabic Sign Language resources hindered the progress of technology development. As a result, the mentioned factors restrict deaf and hard of hearing individuals' access to public services, education, and work.

There is also a lack of research that compares traditional machine learning techniques and different deep learning approaches systematically on the same standardized data set under the same experimental conditions. Such comparison is needed in order to assess which method performs better and by how much better it performs compared to other approaches. Moreover, most of the past studies ignore the computation complexity during evaluation and usually do not implement their models in real life.

1.2 Research Questions

That is why the research questions that need answering in this work are as follows:

- Which algorithms show better performance compared to other approaches in terms of the recognition accuracy on the same Arabic Sign Language data set and under the same experimental conditions?
- Which approach can be recommended for the practical implementation due to the high accuracy, adequate training time, and simplicity of implementation?
- Can the performance of a purpose-built custom CNN algorithm exceed that of transfer learning involving large pre-trained networks on grayscale images of signs utilized in sign language?

The two additional questions of what the performance is like at the letter level and how well the models generalize to signers unseen during training are acknowledged as crucial but are left out of the evaluation methodology employed in this study. These limitations are specifically addressed in Section 4.8 and form the first two items on the future work agenda in Section 5.

1.3 Goals of the Study

- Data acquisition, validation and processing of the image dataset of the Arabic Sign Language alphabet.
- Custom design and training of a sign language recognition convolutional model on a standard PC.
- In order to apply conventional machine learning algorithms as well as different deep learning techniques for comparative analysis.
- In order to conduct comparative analysis of all models using a common framework in terms of validation accuracy and training time.
- In order to utilize the best model in a browser-based interactive translation prototype for Arabic Sign Language, which generates text output.

1.4 Importance of the Research

In technological terms, the importance of the research is that it will allow conducting a comparison of seven models on a publicly available large dataset for Arabic Sign Language, with the empirical results in terms of accuracy and computational efficiency being provided for all seven techniques. In social terms, the importance of the research is that it offers a practical tool aimed at helping deaf people access the services, education, and labor market.

1.5 Scope and Delimitations

The scope of the study includes recognition of 32 static signs representing the letters of the Arabic manual alphabet. The recognition of dynamic signs and translation of words and complete sentences into spoken language are out of scope of the current research. The evaluation method used here is stratified random partitioning and not a signer-independent protocol. All deep learning models were trained for a fixed number of five epochs on CPU architecture. The deployment of the developed system outputs recognized letters and assembled words in the text form on the screen; the audible synthesis of speech was not done. Hence, this study is always referred to as sign language recognition with text output, and no claim of speech recognition or speech synthesis is made.

2. Previous Studies:

The creation of the Arabic Alphabets Sign Language dataset can be regarded as a milestone in the progress of the recognition technology of Arabic Sign Language (Latif, Mohammad, Alghazo, AlKhalaf and AlKhalaf, 2019). It comprises 54,049 grayscale images of 32 Arabic alphabet signs collected from over forty subjects and resized to 64×64 pixels. Prior to this dataset being available to researchers, the majority of studies had to work with private datasets, which prevented the comparison between different works. The publication of this dataset allows using it as a common benchmark for research in the area, and it is the dataset chosen for use in this study.

Noor (2024) developed a hybrid system for recognition of Arabic Sign Language combining convolutional and recurrent neural networks. When tested on the custom dataset of 4,000 images of ten dynamic and ten static gesture words, the convolutional network achieved 94.40% accuracy, and the recurrent network scored 82.70%. The main motivation for developing such a system was the deficiency of certified interpreters in Saudi Arabia, which is also an issue of the current research. The limitations of this work include small size of the dataset used and restriction to the vocabulary consisting of twenty words.

A relevant piece of work by Podder et al. (2023) aims at solving the problem of signer-independent Arabic Sign Language recognition, where the task of the system is to recognize gestures performed by individuals not included in the training sample. The system uses either MobileNetV2 or ResNet18 backbone combined with long short-term memory and a custom self-attention mechanism. The system is reported to achieve 87.69% accuracy, 88.57% precision, and F1-score of 87.72% on RGB video with face and hand segmentation, which is almost eleven percentage points better than the state-of-the-art scores reported on signer-independent tasks. The limitation of this work is a multi-stage pipeline consuming considerable amount of resources.

Bani Baker (2023) investigated the impact of transfer learning on the Arabic alphabet sign recognition in the situation of the small training set. It was discovered that in this case transfer learning gives better results than training a model from scratch. The current study examines whether the advantage of transfer learning holds true when there are more than fifty thousand training samples available, and the images are single-channel grayscale rather than RGB photography.

Apart from the Arabic context, recent developments in this area of research include the work done in the domain of American Sign Language recognition by Kumar (2023) in real time using hand landmarks and convolutional classifiers, reaching the accuracy level of 99.95%, and the work in the field of word-level recognition of Indian Sign Language done by Das (2023) using VGG-19 and bidirectional long short-term memory with histogram-based keyframe selection and achieving the average accuracy of 87.67%. As pointed out by Hussain (2023), the research and evaluation practices in this area of research are not evenly distributed among regional languages, thus complicating the knowledge accumulation process.

Table 1 provides the summary of the studies which are most relevant to the current work, including datasets used in those studies, methods and metrics applied there and their limitations. Heterogeneity visible in that table – different corpora, class number, input modality and evaluation protocol – is the reason for the controlled single execution benchmark presented in this paper.

Table (1): Comparative Summary of Related Work in Sign Language Recognition

Study	Dataset	Classes	Method	Reported Metrics	Stated Limitation
Latif et al. (2019)	ArASL, 54,049 grayscale images, 40+ signers	32	Public dataset release; 64x64 grayscale standardisation	Benchmark corpus (no classifier proposed)	Static alphabet only; single capture context
Noor (2024)	Custom set, 4,000 images	20 words	Hybrid CNN + LSTM	CNN 94.40%; LSTM 82.70% accuracy	Small custom corpus; 20-word vocabulary
Podder et al. (2023)	Custom RGB video, face-hand segmented	Words	MobileNetV2/ResNet18 + LSTM + SelfMLP	87.69% acc.; 88.57% prec.; 87.72% F1 (signer-independent)	Multi-stage pipeline; high computational cost
Bani Baker (2023)	Public ArSL alphabet images	32	Fine-tuned pretrained CNNs (transfer learning)	Transfer learning exceeded from-scratch training	Limited to small static alphabet sets
Das (2023)	ISL video word dataset	Words	VGG-19 + BiLSTM with histogram keyframe selection	87.67% average accuracy	Validated on ISL only

Study	Dataset	Classes	Method	Reported Metrics	Stated Limitation
Kumar (2023)	ASL gesture images	Letters	MediaPipe landmarks + CNN	Up to 99.95% accuracy	Validated on ASL only; landmark-dependent
Hussain (2023)	Survey (multiple corpora)	-	Literature review	Identifies uneven regional coverage	No primary experimental contribution
This study	ArASL, 54,049 grayscale images	32	Seven models benchmarked in one execution	Custom CNN 94.88% validation accuracy; training time reported	Static signs; stratified, not signer-independent split

2.1 Classical Approaches and the Value of Being a Baseline

Before the era of deep learning, the recognition of sign language used to be performed by combining classical computer vision algorithms and traditional machine learning classifiers. In the typical algorithmic pipeline, one would start by segmenting the hand region depending on the skin color, extract handcrafted features, such as histograms of oriented gradients or scale-invariant feature transforms and classify them using support vector machines or decision trees. Although such approaches demonstrate good performance for restricted vocabularies, they exhibit poor generalizability to the lighting conditions, backgrounds and signers' appearance changes, as the handcrafted features were designed instead of being learned.

However, classical approaches still prove themselves to be highly valuable benchmarks for comparisons. Assessing what can be achieved with a well-tuned classical classifier is a reasonable benchmark in evaluating the gain in performance that may be provided by the use of deep learning. The Random Forests classifier trains an ensemble of decision trees on bootstrapped datasets, which makes it a highly robust method that requires no hyperparameters tuning. The gradient boosting trains an ensemble of models iteratively, and therefore every next model tries to minimize the residuals produced by the previously trained models from the ensemble. The support vector machines try to find a maximum margin separating hyperplane in the feature space, and their performance is greatly affected by the separability of the feature space.

One of the constraints that arises in the context of these methods is the problem of dimensionality. The 64x64 image provides a feature vector of 4,096 dimensions, and using such classifiers on thousands of samples with such dimensionality poses great requirements to the memory capacity and processing power. Standard ways of solving this problem include stratified sampling of the training set and downscaling of the images. Both methods are employed in the described experimental setup in Section 3. This constraint is worth mentioning separately because it demonstrates the real scalability problem of classical approaches when applied to image data.

2.2 Theoretical Foundation of Convolutional Architectures

The Convolutional Neural Network is well-suited for image classification because of two structural assumptions that are particularly applicable to visual data. Firstly, it is the assumption of locality: pixels adjacent to each other in space are more likely to be connected semantically than distant pixels, and therefore the filters operate on the local neighbourhoods. Secondly, it is the assumption of translation equivariance: a visual pattern does not lose its identity when its location in the frame changes, and the same filter may be used to find this pattern at any spatial location.

These two assumptions result in the decrease of the number of parameters comparing to fully-connected neural networks.

These assumptions are especially important in the context of sign language recognition, since the identity of the sign is defined by the spatial relationships between fingers and palm, i.e. by the local spatial relationship. The sign is the same whether it is located slightly to the left or to the right from the center of the frame. However, the classical classifier that works with the feature vector of pixel values does not take into account these assumptions and treats every location independently. This explains the great empirical difference mentioned in Section 4.

In the same way, the regularization method has its importance. This is done via Dropout method suggested by Srivastava et al. (2014). According to this method, certain units of the network are dropped at random while training the network so that the network does not depend on any particular pathway and constructs many different pathways, which makes the representation distributed, and hence helps to avoid overfitting. The current network adopts medium dropout in the convolutional blocks and large dropout prior to the last layer.

2.3 Transfer Learning and Domain Relatedness

This approach relies on an existing model which has been trained to solve some problem and takes it as a basis, training only the classification head of the network to adapt the model to a different problem. One of the most common architectures employed for such purposes is MobileNetV2 by Sandler et al. (2018). It is characterized by depthwise separable convolutions and inverted residual blocks.

Rationale for the Use of Transfer Learning Transfer learning is grounded on the hypothesis that lower level visual features (edges, basic textures, etc.) can be efficiently reused in various visual tasks. However, the extent of this advantage heavily depends on the similarity of domains (source and target). In case the target domain contains imagery that is substantially different from the source domain, some part of the learned feature will be ineffective or even detrimental. For instance, grayscale sign language imagery (64 by 64 pixels) differs considerably from the color photos on ImageNet; hence, transfer learning must be validated on this data set.

2.4 Research Gap

Three research gaps have been identified. First, it is quite uncommon to compare different approaches to sign language recognition using the same standardised Arabic Sign Language data set, and it is hard to tell which group of methods should be used over the other one. Second, the computation cost is not usually provided along with the accuracy rate even though it is one of the important factors that need to be considered when developing an assistive system for routine use. Third, many researchers just stop at the evaluation step while their task is to develop a system based on a trained classifier. The methodology described in Section 3 solves the problem: the standardised ArASL data set and the single-execution experiment used in this research solve the first gap, the training time reported together with accuracy in Section 3.7 solves the second gap, and the prototype developed in Section 3.8 solves the third gap. However, the research does not address another related gap: the validation carried out is not signer independent as the split adopted in this research is not signer independent.

3. Methodology:

3.1 Datasets and Data Integrity Verification

For this research, the Arabic Alphabets Sign Language dataset containing 54,049 grayscale images of 32 different classes was used. To check the integrity of the dataset, a procedure was applied that entailed conducting a recursive search in the dataset directory in order to match each filename with its absolute path and verify its existence, size not being equal to zero, and possibility of decoding an image. No missing and corrupted files were detected in the process and all 54,049 images were successfully verified. Then the classes were converted into integers from 0 to 31. The features of the corpus are presented in Table 2. Class sizes vary from 1,293 to 2,114 images, the mean number being 1,689 and imbalance ratio 1.63 between the largest and smallest class; the corpus can be considered moderate rather than perfectly balanced and the stratified split takes into account these parameters. The corpus is publicly available in Mendeley Data (<https://data.mendeley.com/datasets/y7pckrw6z2/1>, DOI: 10.17632/y7pckrw6z2.1) under the Creative Commons Attribution 4.0 licence and is described in the data article of Latif et al. (2019). One feature of the corpus influences the way the results reported below should be interpreted: the published metadata lacks the signer identifier for each image, hence there is no way of splitting the data by the signer. Therefore, the stratified random split in Section 3.6 allows images generated by the same signer to appear both in train and validation sets. The reported accuracy should thus be understood as performance of the signer-dependent protocol and not the generalization performance on the unseen signer; this restriction is mentioned again in Sections 4.3 and 4.8.

Table (2): Properties of the Arabic Alphabets Sign Language Dataset

Property	Value
Dataset name	Arabic Alphabets Sign Language (ArASL)
Source	Latif et al. (2019), publicly available
Total images	54,049
Number of classes	32 Arabic alphabet signs
Image resolution	64 x 64 pixels
Colour format	Grayscale (single channel)
Number of signers	More than 40 participants
Integrity check result	54,049 valid; 0 missing; 0 corrupt

3.2 Datasets Features and Capture Variability

Before developing any models, an exploratory analysis of the intensity statistics for various sample classes was conducted in order to understand the characteristics of the corpus visuals. It appeared that the mean pixel intensity was class-dependent: the majority of sample classes had the mean intensity of about 170 to 192 on the regular eight-bit scale – that is consistent with the imagery captured against the light background. There was, however, one class that was characterized by the deviation in mean intensity, with a minority of images captured under the different lighting and the mean intensity of about 66 compared to 191 in the rest of the class.

It means that there is some heterogeneity of lighting both in and between the sample classes. This leads to two important remarks. The first one is that variability may be beneficial for the robustness

of a trained model with respect to various lighting conditions, similar to explicit corpus augmentation. However, if certain lighting conditions correlate with certain classes, there is a risk that a model might partially exploit this fact instead of focusing exclusively on the hand configuration as the discriminating feature. While in the current experimental setting this danger does not seem to be realized, the fact is recorded in the interest of transparency and will be discussed again as a limitation in Section 4.8.

3.3 Preprocessing

There are two preprocessing pipelines depending on the model family. In case of deep learning architectures, the whole image set is preloaded into memory as an unsigned eight-bit array of approximately 211 MB, eliminating the need to reload and decode the images during training. Then the data pipeline converts the images to the floating point format and normalizes their intensity to the unit interval, shuffling using the buffer of 10,000 samples, batching at 64 and prefetching in order to overlap data loading with computation.

The traditional classifiers require the images to be flattened to the one-dimensional feature vectors, thus introducing a scaling constraint. At the native resolution, the whole dataset yields the matrix of approximately 221 millions of elements, surpassing practical memory limitations for such methods as SVM. Thus, two common mitigations were employed: firstly, a random subset of 5,000 training and 2,000 validation samples were selected without repetition; since the subset is quite large relatively to the 32 classes, the proportions of the classes are approximately retained, although the sampling is not stratified. Secondly, spatial downsampling was conducted from 64 by 64 to 32 by 32 pixels, thus reducing the dimensionality to 1,024 features per sample. The pixel values were then normalized to the unit interval. The deep learning pipeline was implemented with the help of TensorFlow/Keras, and the classical classifiers with scikit-learn (Pedregosa et al., 2011); the versions of the libraries are listed in the requirements file accompanying the source code. It means that the two families of the models were not trained on the same input data: the deep architectures were provided with all 43,239 training images at full resolution of 64 by 64, while the classical classifiers had a 5,000-image stratified subset of the images downsampled to 32 by 32. It was imposed by the limitations of memory, not by the choice, but it means that the accuracy and training time numbers reported in Section 4 are not the like-for-like comparison of the two families. The difference between the families should thus be considered as indicative of the practical scalability difference of the approaches, not as a controlled comparison of the architectural capacities. Comparisons inside each family — among the three deep architectures, and among the four classical classifiers — were performed on the same inputs and are thus directly comparable.

3.4 Proposed Architecture

The suggested convolutional neural network is provided with a grayscale image of 64 by 64 in size and consists of three convolutional layers in sequence and a fully-connected head. Each convolution layer performs convolution with a filter of 3 by 3 size with the rectified linear unit activation function, then performs max-pooling with a window of 2 by 2 and finally applies dropout with the rate of 0.25. The number of the filters increases gradually from 16 to 32 and to 64 in the last layer allowing to learn the abstract representation while decreasing the spatial resolution.

After that, the output of the third block is flattened and passed to the dense layer with 128 neurons and rectified linear unit activation function. This is followed by the dropout with the rate of 0.5 and finally by the dense layer with 32 neurons and the softmax activation yielding the probability distribution of 32 classes. The gradual dropout is used in order to introduce more regularization to the densely connected part of the model, which comprises the largest share of its parameters.

Table (3): Layer Configuration of the Proposed Convolutional Network

There are 322,464 parameters to train in total and 91.5% of them are located in the first dense layer, and that is the reason why the higher dropout of 0.5 is applied to the fully-connected head but not to the convolutional blocks.

Layer	Output Shape	Kernel / Pool	Activation	Trainable Parameters
Input	64 x 64 x 1	-	-	0
Conv2D (16 filters)	62 x 62 x 16	3 x 3	ReLU	160
MaxPooling2D	31 x 31 x 16	2 x 2	-	0
Dropout (rate 0.25)	31 x 31 x 16	-	-	0
Conv2D (32 filters)	29 x 29 x 32	3 x 3	ReLU	4,640
MaxPooling2D	14 x 14 x 32	2 x 2	-	0
Dropout (rate 0.25)	14 x 14 x 32	-	-	0
Conv2D (64 filters)	12 x 12 x 64	3 x 3	ReLU	18,496
MaxPooling2D	6 x 6 x 64	2 x 2	-	0
Dropout (rate 0.25)	6 x 6 x 64	-	-	0
Flatten	2,304	-	-	0
Dense (128 units)	128	-	ReLU	295,040
Dropout (rate 0.5)	128	-	-	0
Dense (32 units)	32	-	Softmax	4,128
Total	-	-	-	322,464

3.5 Baseline Models

Six baseline models have been considered: four classical classifiers, implemented using scikit-learn (Pedregosa et al., 2011) and two different deep architectures, one of which is reduced residual network according to He et al. (2016) design and the second one is the MobileNetV2 transfer-learning model proposed by Sandler et al. (2018). The configuration of the models and their choice are described in Table 4.

Table (4): Configuration of Baseline Models

Model	Configuration	Rationale for Inclusion
Random Forest	100 estimators, parallel execution	Robust ensemble baseline requiring minimal tuning
Linear SVM	C = 1.0, squared hinge loss, dual = False	Tests linear separability of the raw pixel space

Model	Configuration	Rationale for Inclusion
HistGradientBoosting	Maximum 100 boosting iterations	Represents modern gradient boosting methods
Multilayer Perceptron	Hidden layers of 128 and 64 units, Adam solver	Neural baseline lacking spatial inductive bias
Light ResNet	Five layers with two residual skip blocks	Tests whether residual depth benefits this dataset
MobileNetV2	Frozen ImageNet backbone, grayscale-to-RGB layer	Tests transferability of natural-image features

3.6 Training Protocol

Stratified split of the dataset has been performed in order to divide samples into training (80%) and validation (20%) sets while keeping class distribution the same. There are 43,239 images in the training set and there are 10,810 images in the validation set. The random seed value equal to 42 has been used in all experiments to make them reproducible. Deep learning models have been trained with Adam (Kingma and Ba, 2015) optimizer, sparse categorical cross-entropy loss function (labels are integer-encoded), learning rate of 0.001, batch size of 64, five epochs and on CPU hardware (Intel Core i7, 16 GB RAM). Five epochs have been selected as a budget that is identical for all three deep learning models, so that the accuracy difference reflects the efficiency of conversion of a fixed budget into performance, but not the full convergence behaviour. It is known that there is a drawback of such choice since deeper networks, like residual network and transfer-learning model, might not reach their limit within five epochs. However, this is discussed as a limitation in Section 4.8 and as a future work in Section 5. Training and validation accuracy and loss per epoch has been recorded by benchmark script and is stored along with the source code. They are not included here since available plot curves correspond to a longer training procedure than the five-epoch budget discussed in this paper.

More importantly, though, all seven models were trained and validated in a single automated run with the same partition and evaluation code. This is an important methodology point: if results are combined from independent runs, then accuracy differences may be due to variations in data splitting or preprocessing, not because of the differences in capability of the architecture. Thus, training of all models in a single run helps to get rid of such a confounding variable and ensure that accuracy differences are indeed due to the different architectures of the models. The set of hyperparameters used to train all models is presented in Table 5.

Table (5): Training Hyperparameters

Parameter	Value
Framework	TensorFlow / Keras
Optimizer	Adam
Initial learning rate	0.001
Loss function	Sparse categorical cross-entropy
Batch size	64

Parameter	Value
Epochs (deep learning models)	5
Train / validation split	80% / 20%, stratified
Training samples	43,239
Validation samples	10,810
Random seed	42
Training hardware	CPU-based environment

3.7 Evaluation Metrics

Validation accuracy was selected as the main performance metric: it is defined as the fraction of the validation samples assigned to the correct class. Since there are 32 balanced classes (imbalanced classes ratio is 1.63), and stratification preserves class balances, then accuracy is a meaningful summary statistic. Training time was chosen as the secondary metric, since computational cost affects the ability of the architecture to be used and retrained on a regular computer. There are two caveats. Firstly, average accuracy over 32 classes does not tell us which exact letters a model confuses: some letters of the manual alphabet look very similar, so precision, recall and F1-score per class, along with the confusion matrix are required for the purpose; both are calculated in Section 4.6. Secondly, the split we use is a train-validation split, not train-validation-test split, so the reported accuracies are validation accuracies and are not an unbiased estimate of the model performance on completely unseen data. Claims in Section 4 are made appropriately.

In order to ensure completeness of the paper, the definitions of classification metrics are provided below. Those metrics are calculated for every class as a positive class in the one-versus-rest approach: TP, TN, FP and FN stand for true positives, true negatives, false positives and false negatives respectively. Accuracy is the ratio of true positives and true negatives to the total number of predictions. Precision is the ratio of true positives to the total number of true and false positives. Recall is the ratio of true positives to the total number of true and false negatives. Finally, F1-score is the harmonic mean of precision and recall.

3.8 Interactive Translation Prototype

The best model out of the benchmark was used in a translation prototype in the form of a client-server application. The prototype generates text output only; no sound generation part was included. Server side is implemented with the help of FastAPI web framework and includes the prediction endpoint: it takes the frame of an image uploaded to it, decodes it, makes it grayscale, resizes to the 64x64 pixels, required by the model as input, normalizes pixel values and returns the predicted class and its confidence. The model is loaded once at server start-up and held in memory, so that the cost of each request is only the cost of inference, and not the cost of loading the model.

Client side is implemented as a web interface that requires no installation by the user. It has access to the device camera, displays live video feed from it, captures frames every 350 milliseconds and sends it to the prediction endpoint. Predicted letters are added to the text area consecutively; controls for adding a space, removing the last character and clearing the current text are provided. Stabilization algorithm accepts predicted letter only if it is predicted consistently for several

frames, so that transient misclassification due to hand transition does not affect the output. Since the interval between frames is fixed and equal to 350 milliseconds, the prototype works at about 2.9 frames per second; it is a design parameter of the client, not a measured throughput. End-to-end latency, inference time per frame, camera specification and deployment hardware are reported in Section 4.5. Term real-time is avoided in favor of interactive-rate throughout the paper.

3.9 Reproducibility Measures

Various ways have been employed to ensure the independence of the reproduction of the results. For instance, a deterministic seed of 42 is used for all stochastic components of the pipeline including stratified partitioning, shuffling of the training batches, bootstrap sampling of the samples in the Random Forest ensemble, and weight initialization of the neural networks. Therefore, rerunning the benchmark on the same data always yields the same partitions and similar results.

In this benchmark, all seven models are trained and evaluated sequentially in one execution as a part of a single script rather than several scripts whose output would be later combined. Upon completion, this script produces a report containing both the accuracy and the training time of each model and visualization of the results. Since all figures in Section 4 are produced by the program automatically and not copied manually from the results of the experiments, transcription errors can be excluded as one of the sources of discrepancies.

The dataset in question is publicly available and the preprocessing procedure (normalization, stratification, and spatial downsampling) is deterministic since the fixed seed is used. The model is saved to disk with explicit mapping between indices and class labels being preserved, so that the ordering of the classes used in the training is kept during the inference. Although this may seem trivial, this is an essential detail, because any disagreement about the class labels would not affect the training process but would lead to incorrect predictions of the models.

4. Results and Discussion:

This section presents the results of the comparative benchmark and their analysis related to the research questions presented in Section 1.2. Table 6 and Table 7 summarize the data partition and the accuracy and training time of all seven considered models respectively.

Table (6): Dataset Split Summary

Subset	Number of Images	Percentage	Number of Classes
Training	43,239	80%	32
Validation	10,810	20%	32
Total	54,049	100%	32

Table (7): Accuracy of Validation and Training Time of all Tested Models

Model	Model Type	Validation Accuracy	Training Time (s)
Custom CNN (proposed)	Deep Learning	94.88%	103.54

Model	Model Type	Validation Accuracy	Training Time (s)
MobileNetV2 (transfer learning)	Deep Learning	93.64%	280.73
Light ResNet	Deep Learning	80.97%	152.44
Random Forest	Traditional ML	80.25%	1.90
HistGradientBoosting	Traditional ML	77.75%	237.27
Multilayer Perceptron	Traditional ML	61.60%	18.72
Linear SVM	Traditional ML	47.85%	210.60

The results reveal huge variations across all tested methods. The proposed convolutional network was the most accurate with the validation accuracy of 94.88%, while 93.64% of the MobileNetV2 transfer learning model followed. The accuracy of the Light ResNet architecture is 80.97%, which is low not only for the deep learning models but also only slightly better than the best classifier. Among the traditional classifiers, Random Forest is the most accurate with 80.25%, then HistGradientBoosting with 77.75%, Multilayer Perceptron with 61.60% and Linear Support Vector Machine with 47.85%. The visual representation of the comparison is presented in Figure 1. There are two key factors that influence the reading of Table 7. The training times were measured on the same machine using the same CPU and in one execution run, hence, they can be compared; however, the classical classifiers were trained on the downsampled inputs mentioned in Section 3.3, which means that the training times for them correspond to the smaller workload and, consequently, cannot be compared to the deep learning architectures. In particular, 1.90 seconds of training time for Random Forest correspond to the 5,000 samples of 1,024 features rather than 43,239 samples of 4,096 features. Comparisons within the groups are based on identical inputs and, hence, they are directly comparable.

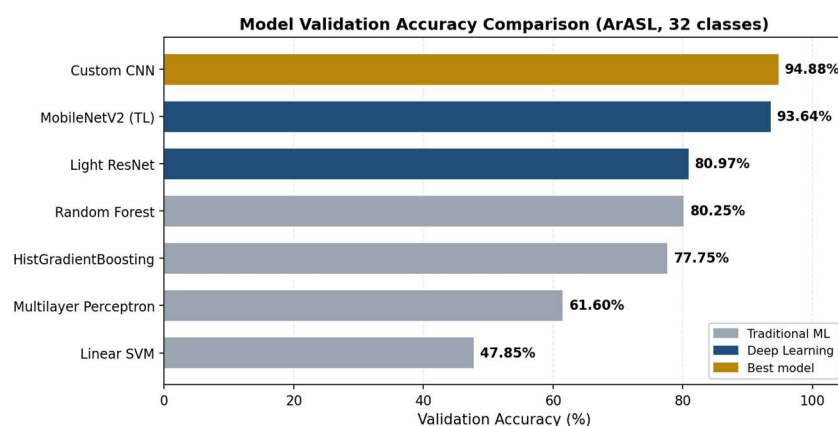


Figure (1): Comparison of Model Validation Accuracy

As for the classification methods, two most accurate convolutional models are in one group above 93%, while five others form a cluster ranging from 47% to 81%. It is important to pay special attention to the Light ResNet, which is a deep learning model, but whose accuracy is placed between the tree-based ensemble models rather than with the other convolutional models. Thus, having a deep learning model does not ensure high accuracy; the key is matching the architecture

67

to the input data. It is important to note that all the figures presented here are the validation accuracies obtained with a single stratified split using a single random seed. Neither test set was kept aside, nor any cross-validation was conducted, so all these figures do not have any estimate of the variance and, hence, may slightly overestimate the performance on the unseen data. The ranking of the architectures is consistent and sufficiently different at the top of the table to be useful, while the figures can be viewed as the single-run measurements rather than benchmark values.

At the same time, another significant factor is the computational cost. Not only did the proposed model achieve the highest accuracy, but it also did so in 103.54 seconds, compared with 280.73 seconds of the MobileNetV2 training time, which is shown in Table 8 below. Hence, the proposed model was slightly more accurate and required approximately 37% of the training time of the transfer learning model.

Table (8): Performance versus Training Cost of Deep Learning Architectures

Model	Validation Accuracy	Training Time (s)	Relative Cost
Custom CNN (proposed)	94.88%	103.54	1.00x (reference)
Light ResNet	80.97%	152.44	1.47x
MobileNetV2 (transfer learning)	93.64%	280.73	2.71x

4.1 Deep Learning vs Machine Learning

The disparity between the two classes is huge; the proposed architecture outperformed the best machine learning model by 14.63 percentage points. This discrepancy comes about as a result of the innate architectural distinction in the models. While the traditional classification algorithms work on flattened pixel vectors and do not consider the spatial relationship of the pixels that characterize hand shapes, the proposed models incorporate convolutional layers that retain the spatial relationship among adjacent pixels. This is how the performance difference is theoretically predicted in section 2.2.

Just as the disparity in performance between the two classes of models above is educative, the one within the family of machine learning models is equally enlightening. Even though random forest outperformed Linear SVM by 32.40 percentage points while employing the same input data, the message here is that selection of algorithm within a family is as crucial as the class itself. Indeed, the performance pattern where ensemble methods outperform neural networks and neural networks outperform linear classifiers is an indicator of their ability to learn non-linear relationship in the pixel intensity space. The linear classifier's performance was expected because the pixel intensity space is not linearly separable.

4.2 Custom Architecture vs Transfer Learning

However, the fact that the proposed network matched and even outperformed MobileNetV2 requires analysis because it is assumed that transfer learning should work better in this case. The problem is that domain mismatch takes place when training a network. MobileNetV2 has been trained on ImageNet which is a collection of natural coloured images where some features like

colour gradients and complicated textures cannot be applied to the task of discrimination of low-resolution monochrome images of hand shapes.

Also, the result is consistent with the common principle according to which the benefits of transfer learning become lower as the difference between source and target domains increases. Moreover, the dataset has more than fifty thousand training images which is sufficient to train a network without the use of transfer learning. Thus, the obtained result contradicts the findings of Bani Baker (2023) who states that there is an advantage of transfer learning for Arabic sign recognition due to insufficient data.

The value of 80.97% for the Light ResNet is consistent with the observation that additional complexity does not automatically yield improvements. Residual connections were added to allow the training of deep networks but, in the current case, the dataset is simple visually because it consists of low-resolution single-channel images of hand shapes. Thus, under the five-epochs regime which is common for all tested networks, additional depth and number of parameters provide no benefits to the architecture. In order to achieve its maximum performance, it needs some more epochs. Under the limited computational budget, the custom compact architecture works better.

4.3 Comparison with Prior Studies

Table 9 places the obtained accuracy rate into the context with previous researches. It should be noted that other datasets, number of classes and evaluation techniques were used in prior studies.

Table (9): Comparison with Previous Sign Language Recognition Works

Study	Dataset	Approach	Accuracy
Proposed Custom CNN (this study)	ArASL, 54,049 images, 32 classes	Custom three-block CNN	94.88%
Noor (2024) - CNN	Custom set, 4,000 images	Hybrid CNN component	94.40%
Noor (2024) - LSTM	Custom set, 4,000 images	Hybrid LSTM component	82.70%
Podder et al. (2023)	Custom RGB video set	MobileNetV2 with LSTM	87.69%
Das (2023)	ISL video word dataset	VGG-19 with BiLSTM	87.67%

The 94.88% attained here lies in the same category as the 94.40% found by Noor (2024) but is found on a corpus more than thirteen times bigger including all 32 letters of the manual alphabet. However, the two cannot be compared because Noor (2024) evaluates a 20-word vocabulary including dynamic gestures on a custom corpus whereas the current work evaluates 32 static alphabet classes on ArASL. Even greater caution is needed when making any comparison between the two studies because Podder et al. (2023) found 87.69% using a signer-independent evaluation protocol on RGB video, which is a much more difficult test than stratified signer-dependent evaluation performed here on grayscale still images. Therefore, the difference in evaluation protocols, inputs, and tasks contributes much to the difference between the two numbers as the models themselves. In other words, the current result cannot be considered to exceed theirs until

their evaluation protocol is replicated independently. The information provided in Table 9 below should be taken into account only as context but not as a ranking.

4.4 Practical Implications

The results have several important practical implications for developing assistive sign language recognition systems. The first one is related to architecture selection. Researchers in the field of sign language recognition cannot expect the transfer learning from a large generic model to give the best results in the case of enough in-domain data. The small architecture trained from scratch will be much better and cheaper in that case.

The second implication is related to the deployment of the proposed system. Training the proposed model took less than two minutes on CPU only. This makes the method accessible for people who do not have access to GPU hardware.

Finally, the last implication is related to the benchmarking practices in sign language recognition. Difference in accuracy between the proposed model and MobileNetV2 of 1.24 percentage points would not seem significant when reported alone. However, the difference in computation costs of 63% makes the report about the cost reporting as a benchmarking practice much more significant.

The fourth implication relates to the questions of accessibility and sustainability. If the assistive technology designed for the Deaf communities in the discussed regions without the special computing facilities is supposed to be maintained locally, then it should be trainable and based on ordinary computing hardware. The model requiring less than two minutes of central processing unit training time can easily be retrained locally anytime new information becomes available. The second model with extended graphics processing unit training time cannot be retrained at all; therefore, the efficiency advantage also implies sustainability in addition to the experiment results.

Finally, a general methodological implication for sign language recognition system can be identified from the experiments. The common idea that bigger pretrained models will be more effective in transfer learning is obviously not true in this particular case; moreover, it would not have been noticed without combining transfer learning with the from-scratch training of the model. Thus, for the purpose of comparative evaluation, the baseline model should be purposefully developed for the task at hand.

4.5 Evaluation of the Deployed System

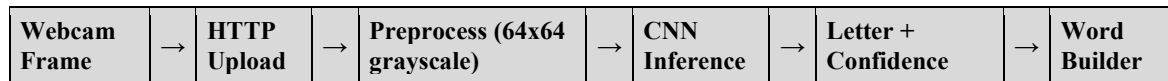
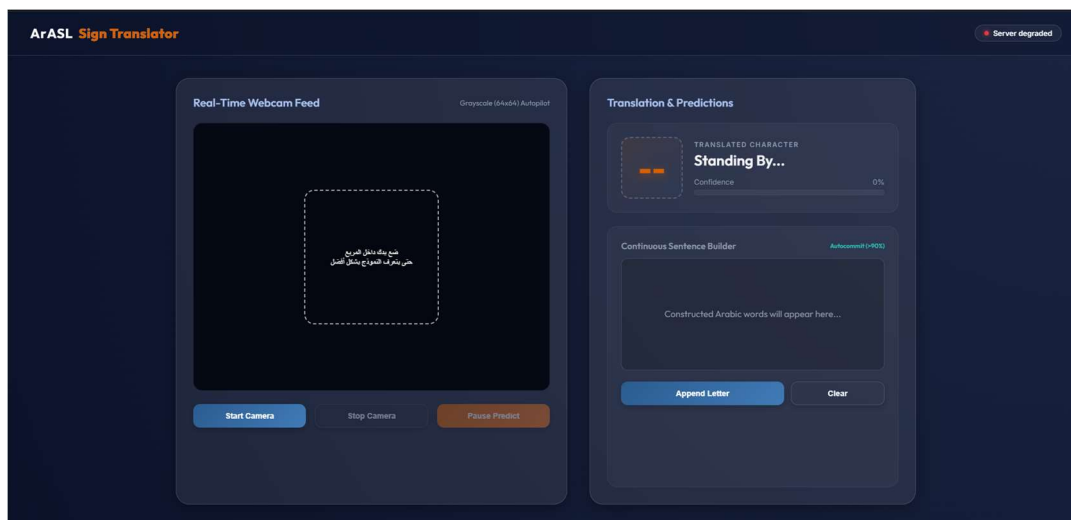
The trained model was used in the interactive translation prototype introduced in Section 3.8 and tested in the operational conditions. The prototype maintained interactive rate sufficient for the natural signing speed, the full process of image capturing, transmission, preprocessing, prediction, and display. This conclusion is qualitative in nature; further details are provided below. The word building module effectively generated words from the recognized letters, while the stability mechanism efficiently prevented temporary misclassification of hands while switching from one sign to another.

Practically, two observations were made while testing the system in operational conditions. First, the recognition accuracy depends on lighting and background conditions because the model was trained on images captured in a controlled environment. Second, the confidence score displayed along with predictions provides valuable feedback to the user because it indicates that hand position and frame need to be adjusted if the score is persistently low.

Table (10): Deployment Characteristics of the Prototype

Measure	Value
Client frame capture interval (ms)	350 (fixed design parameter; 2.86 frames per second by construction)
Autocommit confidence threshold	Prediction committed when confidence exceeds 90%
Camera resolution used for testing	1280 x 720 (720p)
Deployment hardware	Intel Core i7 CPU, 16 GB RAM
Number of signs exercised in operational testing	15 of the 32 alphabet classes
Number of participants in operational testing	1 (the author)
Audio or speech output	Not implemented

Pipeline of the processing performed by the deployed prototype is illustrated in Fig. 2, and its browser interface while performing in Fig. 3. Operational testing described below was exploratory in nature rather than a controlled user study, and the reaction time was qualitatively adequate for a natural signing rhythm, but was not measured: neither frame-by-frame inference time, nor end-to-end latency or throughput measurements have been made. Therefore, table 10 describes the deployment configuration rather than performance profiling. Measuring of quantitative reaction time along with usability assessment with several participants, is identified as future work in section 5. Users should be aware of the fact that the operational testing was performed on one participant and 15 of 32 alphabet classes, and consequently is reported as feasibility study rather than performance demonstration.

**Figure (2): Processing Pipeline of the Deployed Prototype****Figure (3): Browser Interface of the Deployed Prototype During Operation**

4.6 Performance Analysis per Alphabet Class and Confusion Analysis

Aggregate accuracy does not specify what letters are recognized accurately and where the errors lie. Therefore, Table 11 gives precision, recall, and F1-score for all the classes on the validation split, and Figure 4 shows the normalized confusion matrix for the same.

Table (11): Per-class Precision, Recall and F1-Score on the Validation Set

Letter	Class label	Precision	Recall	F1-Score	Support
ع	ain	0.9586	0.9858	0.9720	423
ال	al	0.9628	0.9628	0.9628	269
ا	aleff	0.9466	0.9551	0.9508	334
ب	bb	0.9439	0.9860	0.9645	358
د	dal	0.9515	0.9602	0.9559	327
ظ	dha	0.9494	0.9246	0.9369	345
ض	dhad	0.9755	0.9551	0.9652	334
ف	fa	0.8995	0.8696	0.8843	391
ق	gaaf	0.8528	0.9003	0.8759	341
غ	ghain	0.9604	0.9798	0.9700	396
ه	ha	0.9392	0.8742	0.9055	318
ح	haa	0.9404	0.9311	0.9357	305
ج	jeem	0.8981	0.9387	0.9180	310
ك	kaaf	0.9507	0.9239	0.9371	355
خ	khaa	0.9323	0.9439	0.9381	321
لا	la	0.9855	0.9742	0.9798	349
ل	laam	0.9727	0.9700	0.9714	367
م	meem	0.9741	0.9603	0.9672	353
ن	nun	0.9723	0.9643	0.9683	364
ر	ra	0.9723	0.9518	0.9619	332
ص	saad	0.9361	0.9657	0.9506	379
س	seen	0.9572	0.9543	0.9557	328
ش	sheen	0.9643	0.9867	0.9754	301
ت	ta	0.9229	0.9559	0.9391	363
ط	taa	0.9443	0.9212	0.9326	368
ث	thaa	0.9419	0.9178	0.9297	353
ذ	thal	0.9281	0.9399	0.9340	316
ة	toot	0.9507	0.9693	0.9599	358
و	waw	0.9710	0.9781	0.9745	274
ي	ya	0.9795	0.9738	0.9767	344
ئ	yaa	0.9922	0.9768	0.9844	259
ز	zay	0.9615	0.9091	0.9346	275

Letter	Class label	Precision	Recall	F1-Score	Support
—	Macro average	0.9496	0.9488	0.9490	10,810
—	Weighted average	0.9491	0.9488	0.9488	10,810

Overall performance is balanced in terms of the letters: macro-averaged precision, recall, and F1-score equal 0.9496, 0.9488, and 0.9490, correspondingly; the difference between the macro-average and the weighted average is negligible, which matches the nature of an only slightly unbalanced dataset. None of the classes collapses. F1-score varies from 0.8759 for gaaf to 0.9844 for yaa, giving a range of about eleven percent points, and thus the overall result of 94.88% is rather representative than averaging a few successful and a number of failing classes.

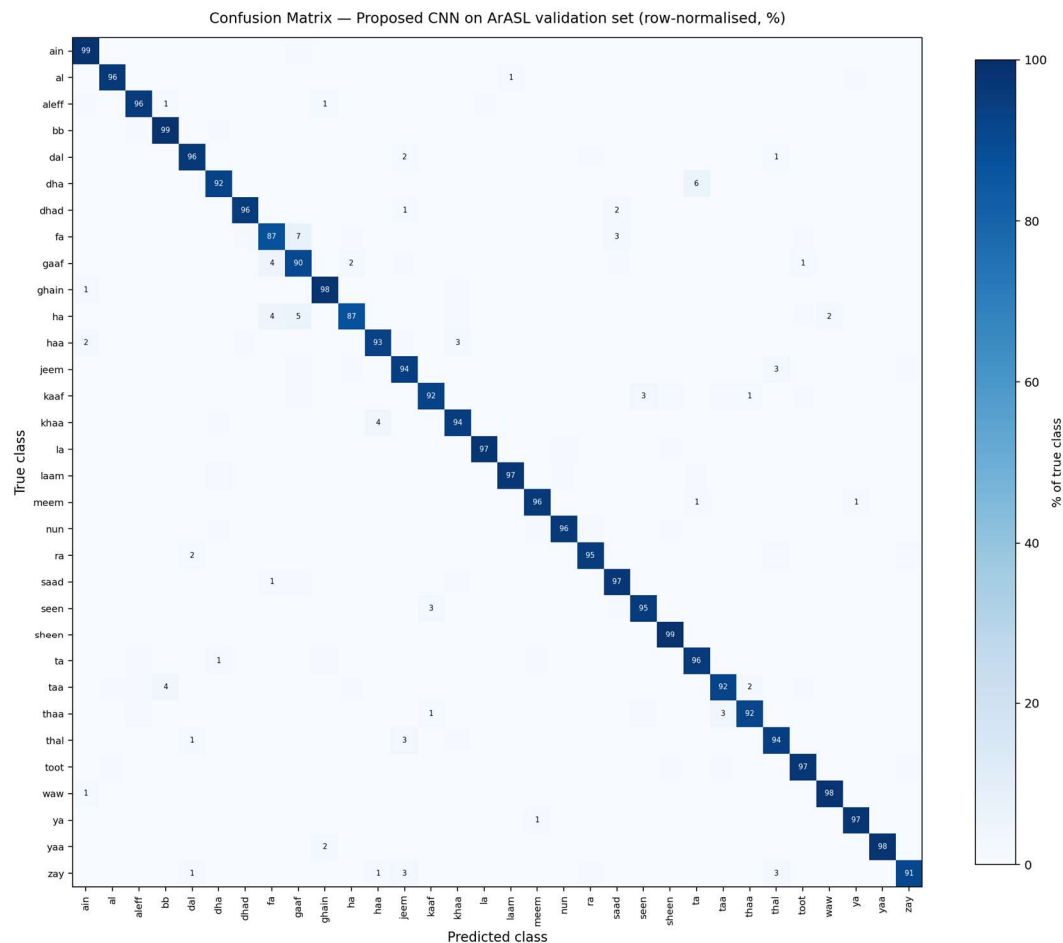


Figure (4): Confusion Matrix of the Proposed CNN on the Validation Set (row-normalized, percentage of each true class)

The confusion pattern is linguistic in nature. The most prevalent confusion is fa classified as gaaf (27 times, 6.9% of fa), and the converse confusion is present as well (14 times, 4.1%); these letters are differentiated in writing by the dot quantity only, while in the sign language their only difference is a slight modification in finger position. Similarly, the confusion pair is observed for khaa classified as haa (3.7%) and thaa classified as taa (3.4%), also differing only in one base handshape. The third pattern comprises ha, which is assigned to gaaf (5.3%) and fa (4.1%).

Two implications arise. First, the error is clustered in visually adjacent character pairs as opposed to being evenly spread out; thus, a targeted intervention becomes possible: more examples or data augmentation targeting those groups, fa, gaaf and ha specifically, will yield an especially high proportion of the total error recovered. Second, in regards to the practical deployment of the prototype, these are the characters that will have the highest frequency of instances, in which the confidence threshold cannot be met and the user is forced to move their hand.

4.7 Answers to the Research Questions

Based on the obtained results, it is possible to provide direct answers to the three research questions stated in Section 1.2. In regards to the first research question about the comparison of deep learning and traditional machine learning algorithms in regards to the accuracy attained based on the same dataset and identical conditions, it is easy to see that both of the two best performing CNNs have an accuracy exceeding 93%, whereas none of the traditional classifiers reached the mark of 81%. The difference of 14.63 percentage points between the proposed network and the best-performing traditional classifier is consistent with the structural argument provided in Section 2.2. The margin should be understood in conjunction with the caution expressed in Section 3.3: the two families of models were not trained on identical inputs, so the margin represents the gap between the approaches in feasible configurations rather than an isolated measure of architectural capacity.

As far as the second research question about the identification of the architecture that provides the best trade-off between accuracy, training speed and feasibility of deployment is concerned, it is evident that the proposed custom CNN was the winner. It showed the highest accuracy in comparison with all seven architectures examined and needed less time for training than any of other deep learning models. Moreover, a relatively low number of parameters makes inference at interactive speeds on standard hardware possible. Special attention should be paid to the Random Forest architecture, which required the least amount of time for training – 1.90 seconds. However, this achievement was made with the use of downsized sample and feature space and its accuracy of 80.25% was insufficient for practical purposes.

In regards to the third research question about the comparison of the performance of the proposed network with transfer learning in terms of the task of classification of grayscale sign imagery, the answer is positive under the current conditions. The proposed custom network was better than the competing MobileNetV2 in regards to accuracy (outperforming it by 1.24 percentage points) and took only about 37% of its training time. The reason for this result is in the inconsistency between the domains of the ImageNet dataset (natural photography) and the used sign imagery (grayscale images of hands). The limited data volume, which is supposed to make transfer learning effective, is no longer relevant.

4.8 Limitations

Several limitations bound the results. One of the most important is the evaluation protocol. Since ArASL does not provide per-image signer identification information, it was not possible to do the split signer-independently, and images from the same signer could end up in the train and validation datasets. This means that the reported 94.88% reflects the performance of the models trained on signer-dependent data that would not generalize to a new signer because the models could learn specific to the signer hand geometry, skin coloration, or framing. This limitation is noted in Sections 3.1, 3.6 and 4.3 and concerns all accuracy figures in this paper. Another limitation is the scope: the current work deals with the static signs of the Arabic manual alphabet

28
20

only and does not consider the dynamic word-level signs or even the continuous signing that requires temporal modeling of the sequence of frames. Second, all the imagery was taken from one and only one dataset, collected in a certain geographic area under controlled conditions.

Third, no independent test set was used and no cross-validation was performed, so all the figures are single-run validation accuracies without any measure of variance. Fourth, all deep models were trained under the five-epoch budget, which may be insufficient for the deeper residual architecture and the transfer learning model to reach their respective ceilings; their figures represent thus performance under constraint rather than best attainable accuracy. Fifth, the classical classifiers were trained on a reduced and downsampled subset, so the cross-family comparisons should be regarded as indicative and not controlled. Sixth, the class-level analysis of Section 4.6 was done on the validation set as well, inheriting its signer-dependence and revealing a confusion structure that may be different under a signer-independent protocol. Finally, the deployment evaluation was done in an exploratory manner and not in a proper user study; the quantitative latency results of Table 10 remain to be collected.

5. Conclusion

This paper developed and evaluated a deep learning-based recognition system for the static Arabic Sign Language manual alphabet, benchmarking seven models on the publicly available ArASL corpus of 54,049 images in a single execution. The custom convolutional network showed the highest validation accuracy, 94.88%, and the highest macro-averaged F1-score, 0.9490, with no collapsed class, being the fastest to train among the deep architectures, and was deployed in a browser-based translation prototype generating on-screen text.

The experimental results confirm three hypotheses. First, under the configurations feasible for each architecture type, the convolutional architectures proved much better than the classical classifiers, outperforming the latter by 14.63 percentage points – as was predicted by the structural argument about the preservation of the spatial relationships in the convolutional layers. Second, the compact purpose-designed architecture turned out to match and slightly surpass a much larger pretrained model in this particular domain, requiring approximately 37% of its training time – thus confirming the hypothesis about the diminishing returns of transfer learning when the domain distance is large and in-domain data plentiful. Third, increased architectural depth did not result in higher accuracy under the five-epoch budget, as exemplified by the 80.97% accuracy of the reduced residual network.

These results are valid within a certain envelope. All accuracies represent single-run validation figures under a stratified split that is not signer-independent, no test set was used, the two model families were trained on different inputs, and the deep architectures were trained under a constrained epoch budget. These results should be viewed as a controlled comparison of the architectures under a fixed and limited computational budget and not as converged state-of-the-art figures. Subject to the above envelope, the study provides the following contributions: a seven-model benchmark on a large public Arabic Sign Language corpus that eliminates the confounding factor of aggregation of results across separate experiments; the reporting of the computational cost along with the accuracy, which is still uncommon practice in this literature; and a prototype of the practical deployment rather than a classifier evaluation.

Four directions emerge from this work. The first and most immediate is methodological: replicating the benchmark experiment under a signer-independent protocol, with an independent

test set and cross-validation, will establish how much of the claimed accuracy comes from actual generalisation, and will allow the performance to be compared with signer-independent experiments such as Podder et al. (2023). The per-class analysis in Section 4.6 has already shown where the remainder of the errors occurs, namely among fa, gaaf and ha, whose manual forms are distinguished only in subtle details, and the obvious follow-up is to act on this information by collecting or synthesising more data for these particular classes rather than enriching the whole alphabet equally.

The second direction is extending capability. Incorporating an Arabic text-to-speech component closes the communication loop by producing recognised words as audible speech; this is the obvious extension of the current prototype, and the reason why the current work has been presented as recognising signs and translating into text rather than speech recognition. Recurrent or transformer architectures will extend the system's capabilities from isolated letter classification to continuous signing.

The third direction is about robustness and efficiency. Extending the corpus to include different lighting conditions, backgrounds, camera angles and regional varieties of Arabic Sign Language will improve the system's performance under uncontrolled conditions, addressing the capture heterogeneity discussed in Section 3.2. Running the benchmark experiment again using hardware acceleration and extended training time will establish if the ranking of architectural choices made here is stable, especially if it holds for the residual network once allowed to converge. Finally, the fourth direction is evaluating the system deployment: a structured usability study including several participants along with the latency and throughput measurements presented in Table 10 will be needed before any claims about usability can be justified.

All these directions comprise the programme of building a fully functional translation system for Arabic Sign Language. This paper demonstrates that the underlying task of recognition is tractable within modest computing capabilities, which is a precondition for this program and an encouraging one for practitioners working under constraints.

References list

- 6 Bani Baker, R. (2023). Image-based Arabic sign language recognition system using transfer deep learning models. *Applied Computational Intelligence and Soft Computing*, 2023, 5195007. <https://doi.org/10.1155/2023/5195007>
- 38 Das, S. (2023). A deep learning based recognition model for Indian sign language. *Neural Computing and Applications*, 35, 1469-1481
- 4 He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770-778). <https://doi.org>
- 15 Hussain, A. (2023). Recent advancements in Indian sign language recognition systems: towards enhanced communication and accessibility for the deaf and hearing-impaired. *Asian Journal of Engineering Sciences*, 12(2).
- 23 Adam: A method for stochastic optimization. Kingma, D. P., & Ba, J. (2015). In *International Conference on Learning Representations Proceedings*.

- Kumar, R. (2023). Sign language detection using CNNs in real-time with MediaPipe. arXiv preprint ar
- 9 Latif, G., Mohammad, N., Alghazo, J., AlKhalaf, R., & AlKhalaf, R. (2019). ArASL: Arabic alphabets sign language dataset.
- 26 Noor, T. H. (2024). Real-time Arabic sign language recognition using a hybrid deep learning architecture. Sensors, 24(11), 3683
- 5 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., & Grisel, O. (2011). Scikit-learn: Machine learning in python. Journal of
- 1 Podder, K. K., Ezeddin, M., Chowdhury, M. E. H., Sumon, M. S. I., Tahir, A. M., & Ayari, M. A. (2023). Signer-independent recognition system of Arabic sign language based on deep learning model. Sensors,
- 1
- 2 Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4
- 11 Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine