The Indonesian Journal of Computer Science
www.ijcs.net
Volume 14, Issue 1, February 2025
https://doi.org/10.33022/ijcs.v14i1.4493

Evaluating the Impact of Deep Learning Model Architecture on Sign Language Recognition Accuracy in Low-Resource Context

Tebatso Gorgina[1], Absolom Muzambi[2], Bester Chimbo[3]
moapetg@unisa.ac.za[1], 50417347@mylife.unisa.ac.za[2], chimbb@unisa.ac.za[3]
[1,2,3]School of Computing, University of South Africa, Johannesburg, South Africa

Article Information

Received : 7 Nov 2024
Revised : 28 Jan 2025
Accepted : 7 Feb 2025

Keywords

Sign Language Recognition (SLR), Deep Learning Models, Transformer Models, Low-Resource Datasets Environments

Abstract

Deep learning models are well-known for their reliance on large training datasets to achieve optimal performance for specific tasks. These models have revolutionized the field of machine learning, including achieving high accuracy rates in image classification tasks. As a result, these models have been used for sign language recognition. However, the models often underperform in low-resource contexts. Given the country-specific nature of sign languages, this study examines the effectiveness and performance of Convolutional Neural Networks (CNN), Artificial Neural Networks (ANN), hybrid model (CNN + Recurrent Neural Networks (RNN)), and VGG16 deep learning architectures in recognizing South African Sign Language (SASL) under a data-constrained context. The models were trained and evaluated using a dataset of 12420 training images representing 26 static SASL alphabets, and 4050 validation images. The paper's primary objective is to determine the optimal methods and settings for improving sign recognition models in low-resource contexts. The performance of the models was evaluated across multiple image dimensions trained for 60 epochs to analyze each model's adaptability and efficiency under varying computational parameters. The experiments showed that the ANN and CNN models consistently achieved high accuracy with lower computational requirements, making them well-suited for low-resource contexts.

A. Introduction

Deep learning has emerged as one of the most transformative technologies in the field of machine learning, specifically in natural language processing tasks such as speech processing and text generation and in the computer vision field for image recognition and classification tasks [1]. Deep learning models and architectures have the ability to automatically extract relevant features from raw data through multiple layers of processing without the need for manual feature extraction. However, the success of these models often relies heavily on the availability of large training datasets that allow the models to learn patterns, features, and relationships within the training data [2-4]. This reliance on big data is both an advantage and a disadvantage, as it enables high accuracy rates in high-resource data environments while presenting significant challenges in low-resource contexts.

Deep learning architectures and models have demonstrated a lot of success in several areas including image classification. Models such as Convolutional Neural Networks (CNNs), Artificial Neural Networks (ANNs), recurrent neural networks (RNNs), hybrid models, and, recently, transformer-based models such as VGG16 have achieved state-of-the-art accuracy rates in computer vision-related or image processing tasks [5, 6]. The success in these tasks has led to the application of deep learning models in more specialized tasks, such as sign language recognition, where interpreting visual gestures is critical for communication [7].

Sign language recognition is an important application of deep learning models. This task directly addresses communication barriers for individuals with hearing and speaking impairments. Deep learning models have enabled the development of end-to-end systems capable of automating the interpretation and translation of sign language gestures by using images or sometimes videos as input data [7]. These systems have the potential to improve accessibility and inclusion in various social and professional contexts. Similar to how translation tools help bridge language barriers, they can facilitate communication between individuals who understand sign language and those who do not [8]. Each country has its own unique set of gestures and alphabets. This means that models trained on one dataset may not generalize well to another sign language. The country-specific nature of sign languages requires the development of models tailored to specific languages, such as South African Sign Language (SASL).

Another challenge in developing these systems is the large datasets required to train the models. Towards this end, the objective of this study is to evaluate the effectiveness of several deep learning models in recognizing SASL in low-resource conditions. This study experiments with CNNs, ANNs, a hybrid architecture combining CNNs and RNNs, and VGG16, a widely used CNN variant. Each model has distinct architectural strengths, which may influence their performance in recognizing static sign language alphabets with limited training data. CNNs are widely used due to their ability to extract spatial features [9] while RNNs perform well in capturing temporal dependencies [10]. On the other hand, ANNs provide the capability to model complex, non-linear relationships between inputs and outputs which provides versatility in recognizing patterns in sign language data [11]. VGG16 is mostly used for its functionality to extract fine-grained features from images [12].

A dataset consisting of 12,420 training images and 4,050 images representing 26 static SASL was used to conduct the experiments. The dataset presents a realistic low-resource scenario, allowing for the assessment of the models' adaptability and generalization in a data-constrained environment. The study aims to identify the optimal model and training configuration by experimenting with different image dimensions. These experiments are designed to reveal the impact of data scarcity on each model's ability to learn and recognize signs accurately.

This paper is organized as follows: Section B presents the related works. Section C outlines the research methodology, including the experimental framework, evaluation metrics, and experiments, along with the results. Lastly, Section D concludes the paper.

B. Related Works

Several comparative studies have been conducted to analyze the performance of the different machine and deep-learning models for automated sign-language recognition. Authors in [13] conducted research to compare the performance of a CNN model and support vector machine (SVM) model on the recognition and translation of Tanzanian Swahili sign language words. The CNN model attained a 96% accuracy rate compared to the SVM model, which performed at a 95% accuracy rate. The research that was done by [14] examined the performance of two methods, namely SVM and K-Nearest Neighbors (KNN), using scale-invariant feature transform (SIFT) as a feature extraction methodology. The SVM classifier outperformed the K-NN classifier in the context of Arabic sign language. Based on these two studies, CNN demonstrated superior performance compared to SVM, while SVM outperformed KNN.

In their study, [15] compared single deep learning architecture with a hybrid architecture. The authors employed a hybrid architecture of a fusion layer (FL), a bidirectional gated recurrent unit (BGRU), and a temporal convolution (TCOV). The fusion of these features allows the model to combine information from both short-term and long-term temporal contexts, enhancing its ability to capture intricate patterns in sign language gestures. The TCOV records change in the immediate past, the BGRU keeps track of changes in the context that happen over longer periods of time and across other dimensions of time, while the FL learns the correlations between the TCOV and BGRU outputs by linking (fusing) the embedded features. The results of the study indicated that the hybrid architecture outperformed the single deep learning model by 6.1 in terms of word error rate (WER).

In other studies, a model trained on one sign language is fine-tuned for another sign language. Authors in [16] did this, where a model trained for Arabic sign language was fined tuned for Indian sign language. The authors employed transfer learning to extend the model's capacity to recognize additional classes, effectively addressing the shortage of training data needed for retraining. For recognizing sign languages from video streams, [17] used CNN and bidirectional long short-term memory (BiLSTM) to learn complex dynamic gestures by calculating the forward and backward hidden sequences to iterate long short-term memory (LSTM) combinations. Bi-LSTMs were used since unidirectional recurrent neural networks (RNNs) can only compute hidden steps from previous time steps.

Authors in [18] employed CNN and RNN hybrid architecture for American sign language. In this study, CNN was used to extract spatial characteristics from a video feed that was intended for sign language identification. For the experimentation, LSTM and RNN models were used to extract time-related information from video sequences using the softmax function, and the CNN's pooling layer. The researchers reported a highest accuracy of 93% with softmax layer and 58% with the pool layer. The researchers noted the potential for improvement, suggesting that alternative RNN architectures, such as GRU and independent RNNs, could enhance the performance of the pooling layer. Additionally, they proposed that replacing the Inception model with Capsule Networks might further improve the CNN's accuracy.

A unique methodology that was centered on isolating the hand by using depth information derived from RGB images was proposed in [19] for the ASL fingerspelling dataset. The authors proposed a principal component analysis network (PCANet), a modified version of CNN that retrieved characteristics from depth images instead of immediately categorizing them. In another study by [20], CNN was used as the fundamental framework for ASL recognition by utilizing a pre-trained VGG-16 transfer model for static gesture recognition and a sophisticated deep learning-based architecture for dynamic gesture identification. Their method incorporated spatiotemporal characteristics by including components such as ConvLSTM and 3D CNN.

C. Materials and Methods

Experimental Framework

To conduct the experiment, the SASL dataset was sourced from *Realsals.com*. The dataset was pre-processed and augmented following the procedure outlined in [21]. It was then split into training and validation sets to evaluate the performance of the models. Figure 1 depicts an illustration of the experimental framework.

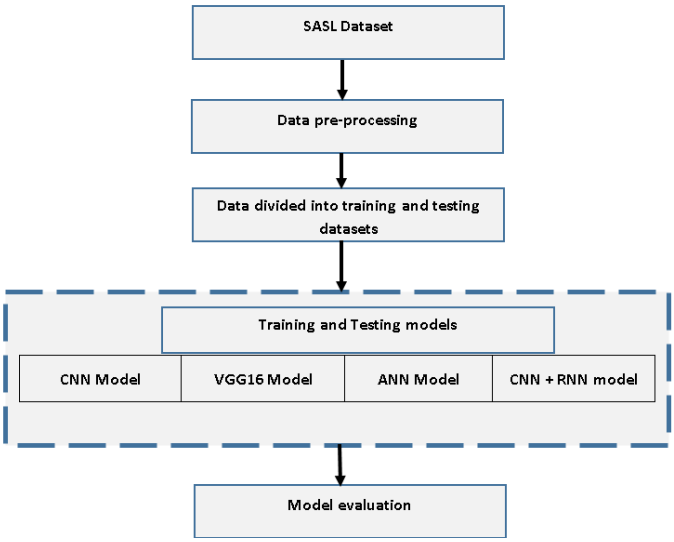


Figure 1. Experimental Framework

The dataset consists of static images representing each letter in the SASL alphabet, illustrated in Figure 2.

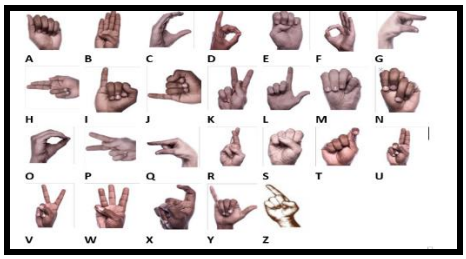


Figure 2. SASL static images

In the dataset, each alphabet letter has 460 images generated for training and 150 for testing using data augmentation. The total dataset consisted of 12 420 images for model training and 4 050 for validation.

CNNs have been proven to process visual data effectively and are commonly used for sign language recognition tasks [22, 23]. They are artificial neural networks designed to process grid-like data, such as photographs, by using convolutional layers that learn hierarchical features from the data they are trained on. The term "convolutional neural network" refers to a specialized type of artificial neural network designed to process and analyze visual data by mimicking the human visual system. Compared to traditional neural networks, CNN layers detect features and patterns automatically, enabling them to identify key features from hand gestures, such as finger positions and movements, enabling accurate classification of signs [23]. This eliminates the need for manual feature extraction, which can be time-consuming and less accurate, especially with complex image backgrounds. They have been proven to achieve high accuracy in classifying static sign language images. However, they are not effective in capturing temporal dependencies in sign language sequences, which are crucial for understanding the dynamic nature of signs.

VGG16 architecture is an extension of CNN architecture specifically designed to capture complex features from images REF. The trained VGG16 model is used for various tasks, including sign language recognition, particularly for classifying static hand gestures by extracting spatial features from input images. While CNNs are primarily developed to process visual data such as images and videos using convolutional layers, pooling layers, and fully connected layers in the network, the VGG16 has a more defined architecture with 16 weight layers [5].

RNNs are mostly used to process sequential data, such as words, phrases, or time series data that are defined by the complex rules of grammar and semantics REF [10]. They mimic sequential data conversion similar to the way humans process data. RNNs contain a memory that stores information about previous states' computations, which is one of its key features, the feedback connections. The hidden units feed into themselves via feedback loops where there is only one set of input, hidden, and output units for each layer. In sign language recognition, RNNs process temporal sequences where Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU) capture long-range dependencies and temporal progression of gestures across frames [1, 10].

ANNs use principles borrowed from biological neural networks to simulate the way the human brain is structured. Neurons in an artificial neural network are linked in a hierarchical structure, much like the human brain. These network nodes are often referred to as nodes. The knowledge of a network is represented

by the weights of these links or connections [21]. Artificial neural networks (ANNs) differ in type based on their topology and learning techniques. In sign language recognition, ANNs learn and extract relevant features from sign language gestures, such as the shape, position, and motion of hands or fingers. This study used a feedforward neural network with three node levels: input, hidden, and output layers. With the exception of the input nodes, every node is a neuron that has a non-linear activation function. Information is introduced into the network at the input layer and progresses sequentially through each subsequent layer until it reaches the ultimate output layer [21].

Hybrid models leverage the strengths of the combined architectures. In this case, the combination of CNNs and RNNs allows for both spatial and temporal feature extraction, which is particularly useful in tasks like sign language recognition [24]. This combination enables the model to process both individual frame details and the overall gesture dynamics, improving accuracy and robustness in recognizing complex sign language gestures.

Evaluation Metrics

To evaluate the performance of the proposed model, metrics such as accuracy (A), precision (P), recall (R), and F1-score (F) were employed. These metrics rely on values like true negatives (tn), true positives (tp), false negatives (fn), and false positives (fp). A true positive occurs when the model correctly classifies a positive instance, whereas a false positive happens when a negative instance is wrongly classified as positive. True negatives indicate the correct classification of negative instances, and false negatives represent positive instances misclassified as negative. The following equations were used to calculate the metrics: accuracy (1), precision (2), recall (3), and F1-score (4).

$$A = (tn + tp)/(tp + fp + tn + fn) \quad (1)$$

Accuracy is the proportion of true predictions, which are the correctly predicted instances, to the total number of predictions made by the model. It quantifies the overall correctness of the model's predictions.

$$P = (tp)/(tp + fp) \quad (2)$$

Precision is the ratio of true positive predictions, the correctly predicted positive instances to the sum of true positive and false positive predictions which are all instances predicted as positive, irrespective of the result. It measures the accuracy of positive predictions made by the model.

$$R = (tp)/(tp + fn) \quad (3)$$

Recall is the ratio of true positive predictions to the sum of true positive and false negative predictions, instances that are positive but incorrectly predicted as negative. It measures the model's ability to identify all positive instances in the dataset.

$$F1 = 2(P*R)/(P+R) \quad (4)$$

The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall.

Experiments

This study used Keras to conduct the experiments. Keras is an open-source library for developing deep-learning applications coded in Python [18]. The code was implemented in a Colab environment. The Colab environment provides free access to computing resources, including GPU, however, additional GPUs were purchased.

Table 1. Experiment 1: 224x224 images with 60 epochs

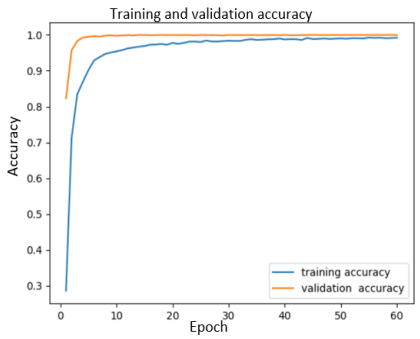
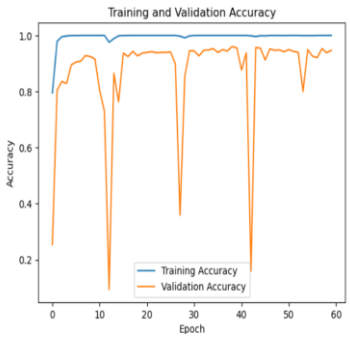
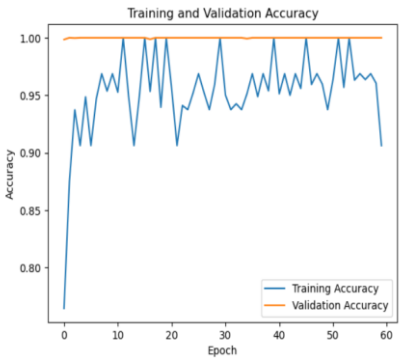
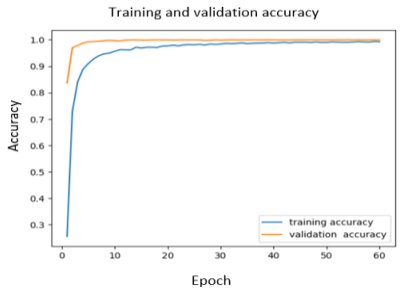
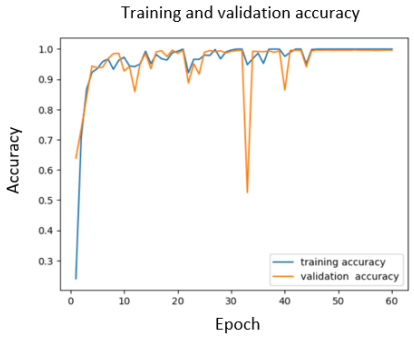
Experiment 1: 224x224 images with 60 epochs	
CNN	ANN
	
Hybrid	VGG16
	

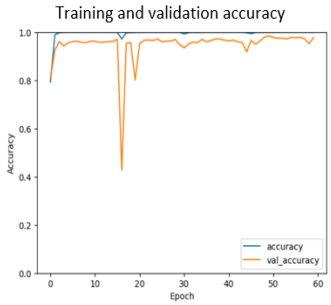
Table 2. Experiment 2: 128x128 images with 60 epochs

Experiment 2: 128x128 images with 60 epochs	
CNN	ANN





Hybrid



VGG16

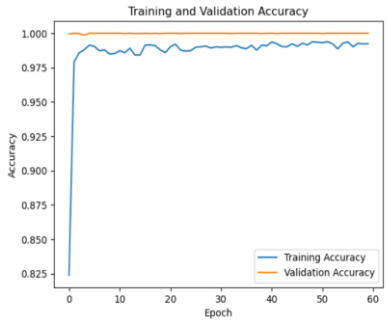
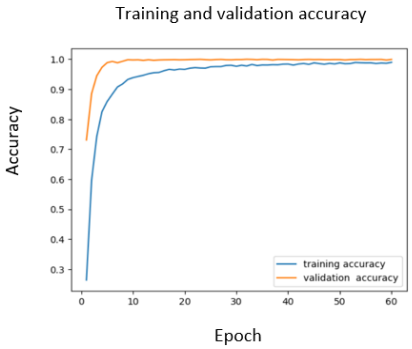


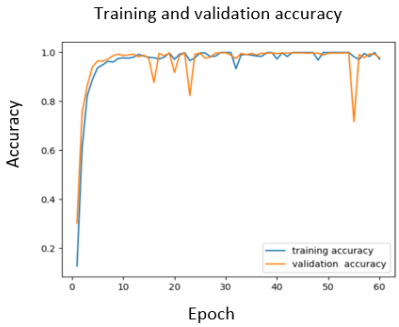
Table 3. Experiment 1: 50x50 images with 60 epochs

Experiment 2: 50x50 images with 60 epochs

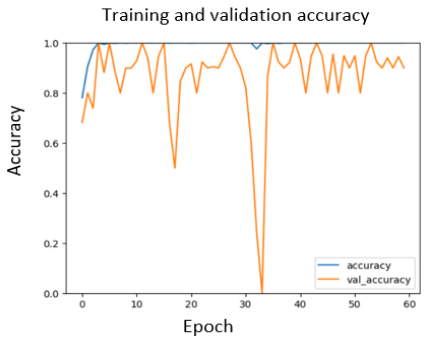
CNN



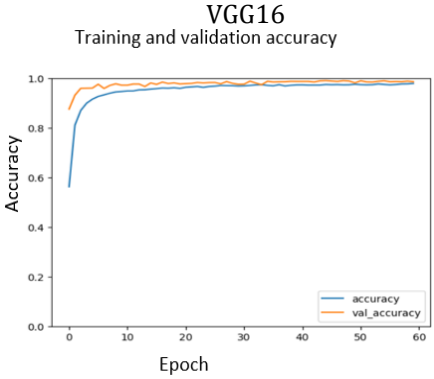
ANN



Hybrid



VGG16



Experiment Results and Discussion

The results for Experiment 1 are presented in Table 4, for Experiment 2 in Table 5, and for Experiment 3 in Table 6.

Table 4. Experiment 1 results: 224x224 images with 60 epochs

Model	Training time (s)	Accuracy	Validation accuracy	Training loss	Validation loss	Precision	Recall	F1_score
CNN	15288	0.9915	0.9995	0.0331	0.0017	0.9607	0.9700	0.9653
ANN	11556	0.9992	0.98883	0.0046	0.0594	0.97	0.97	0.97
CNN+RNN(hybrid)	13824	1.000	0.9469	0.00037	0.1730	0.9852	0.9825	0.9838
VGG16	64910	0.9021	1.00000	0.4204	0.00000015	0.8925	0.8900	0.8912

The 224x224 pixels images experiment demonstrated high accuracy levels but required significant computational resources and took the longest training time. The image size of 224x224 was chosen to accommodate the VGG16 pre-trained model as this was the required image size for the model. The CNN achieved a high training accuracy of 99.15% and an almost perfect validation accuracy of 99.95%. This suggests that the model can generalize well when using larger images. However, the training time was substantial at 15.288 seconds, indicating that high accuracy with this model comes at a computational cost. On the other hand, the ANN performed comparably in terms of accuracy, with a training accuracy of 99.92%. Despite this, its validation accuracy fluctuated. This fluctuation can be attributed to the challenges in maintaining consistency across training and validation data. This instability suggests that while ANN models are computationally efficient, their performance may be affected in low-resource settings where stable generalization is essential. The hybrid model (CNN + RNN) achieved 100% training accuracy but had a reduced validation accuracy of 94.69%, which can be the result of overfitting. The high accuracy paired with long training times (13.824 seconds) indicates that this model might require further adjustments, such as regularization, to better generalize while remaining computationally efficient. VGG16, while showing excellent validation accuracy of 100%, had the lowest training accuracy at 90.21% and the longest training time of 64.910 seconds, making it less suited for low-resource environments without further optimization.

Table 5. Experiment 2 results: 128x128 images with 60 epochs

Model	Training time (s)	Training accuracy	Validation accuracy	Training loss	Validation loss	Precision	Recall	F1_score
CNN	4560	0.9895	0.9995	0.0311	0.0026	0.9789	0.9767	0.9775
ANN	1340	1.000	0.9964	0.000083	0.0164	0.9845	0.9833	0.9838
CNN+RNN(hybrid)	5112	1.000	0.9778	0.000106	0.0778	0.9923	0.9825	0.9874
VGG16	46860	0.9906	1.0000	0.0290	0.00000035	0.9711	0.9828	0.9769

In Experiment 2, the models were trained on smaller, 128x128 images. This led to reductions in training time but maintained good performance. CNN's training accuracy was slightly lower at 98.95%, but its validation accuracy remained stable at 99.95%, and the training time decreased to 4.560 seconds. This result highlights CNN's adaptability and suggests that it performs efficiently with reduced computational demands when image size is optimized. ANN achieved perfect training accuracy at 100% and showed stable validation accuracy at 99.64%, with a drastically reduced training time of only 1.340 seconds. This performance reflects ANN's potential for high accuracy and fast processing in low-

resource environments. This is important to note, given that the reduced image size did not significantly impact its ability to generalize. The hybrid model also showed strong results with 100% training accuracy and an improved validation accuracy of 97.78%, alongside a training time of 5.112 seconds. However, the need for a slightly longer training time compared to ANN suggests that, while effective, the hybrid model requires more computational resources. VGG16 showed an improvement, achieving a training accuracy of 99.06% and perfect validation accuracy, with a reduced training time of 46.860 seconds. Despite the improvement, its extended training time still indicates potential limitations for its application in low-resource settings.

Table 6. Experiment 3 results: 50x50 images with 60 epochs

Model	Training time (s)	Accuracy	Validation accuracy	Training loss	Validation loss	Precision	Recall	F1_score
CNN	4280	0.9880	0.9995	0.0391	0.0010	0.9651	0.9542	0.9775
ANN	1080	1.000	0.9964	0.000082	0.0162	0.9876	0.9833	0.9854
CNN+RNN(hybrid)	5390	1.000	0.90000	0.000052	1.2938	0.9923	0.9825	0.9873
VGG16 model	8328	0.9797	0.9864	0.0592	0.0370	0.9711	0.9828	0.9769

Using 50x50 images in experiment 3 yielded the fastest training times across all models, enhancing computational efficiency. CNN maintained a high training accuracy of 98.80% and a stable validation accuracy of 99.95%, with a training time of only 4.280 seconds. This suggests that CNN's performance can be preserved even with minimal image resolutions, making it well-suited for low-resource environments where efficiency is important. ANN continued to excel with 100% training accuracy and a stable validation accuracy of 99.64%, requiring only 1.080 seconds to complete training. This reinforces ANN's suitability for low-resource settings, where computational efficiency and generalization stability are important. The hybrid model achieved 100% training accuracy, but validation accuracy fell to 90%, indicating that the model struggled to generalize effectively with smaller images despite a reduced training time of 5.390 seconds. This performance suggests that while hybrid models are effective in richer data environments, they may be less ideal in low-resource contexts with smaller image inputs. VGG16 showed a training accuracy of 97.97% and a validation accuracy of 98.64%, with a significantly reduced training time of 8.328 seconds. Although VGG16's accuracy metrics were high, its training time suggests it is still relatively computationally resource-intensive. Figure 3 visually illustrates training time for all the models.

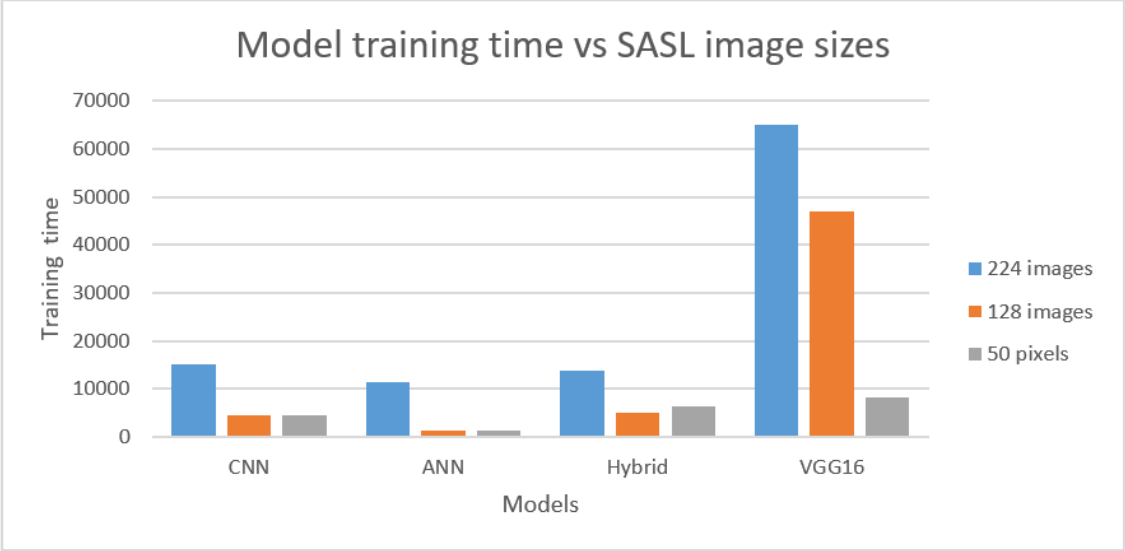


Figure 3. Model training time

As illustrated, VGG16 takes more training time across all image sizes, indicating that it is the most computationally intensive model among those tested. This extended training time proves that VGG16 requires higher processing power and memory capacity, which can be a limitation in low-resource environments, both in datasets and computation. Despite its high validation accuracy, the model's efficiency is compromised when compared to lighter models like ANN and CNN, which achieve similar accuracy with significantly less processing time. Figure 4 illustrates model accuracy across all models and image sizes.

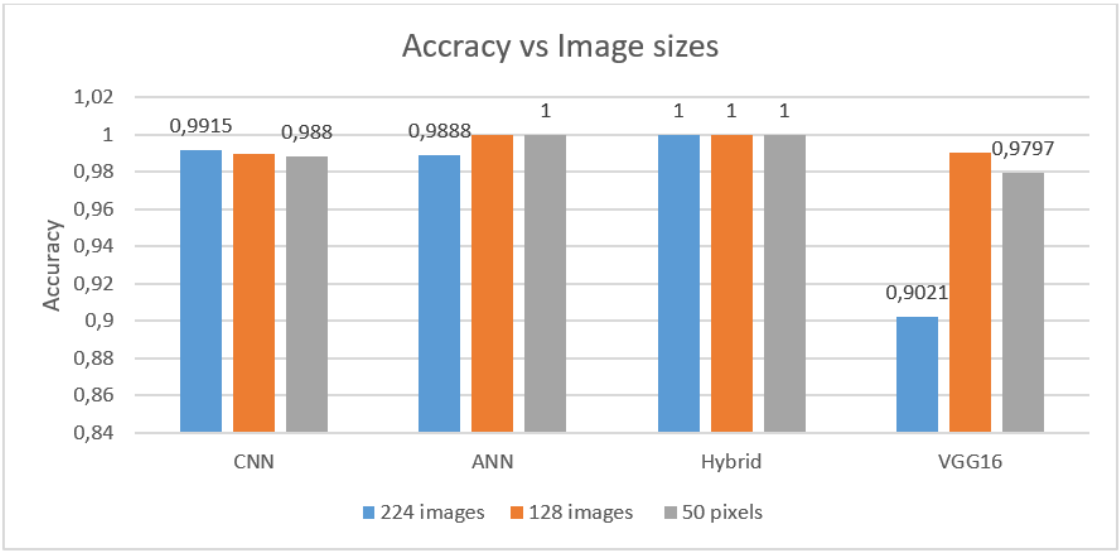


Figure 4. Model accuraries

Hybrid model has the highest rounded accuracy among the models as depicted in Figure 4. This is due to its combined architecture of CNN and RNN layers, these enables it to capture both spatial and temporal features effectively. This combination of the two architectures allows the model to generalize well and

recognize complex patterns within the dataset, making it effective for sign language recognition. However, the hybrid model requires more training time than both the CNN and ANN models because of the additional computational complexity involved in processing sequential dependencies through RNN layers alongside spatial features with CNN layers. This increased training time results in higher accuracy, but it can be a drawback in low-resource environments where efficiency is essential.

D. Conclusion

The conducted experiments indicated that optimizing model performance with smaller image sizes balances accuracy and computational demands. Across all experiments, the results indicate that the ANN and CNN models offer the most consistent performance while requiring lower training times, especially with 128x128 and 50x50 images. As per the experiment, these models appear better suited for environments with limited computational resources and datasets as they achieve high accuracy with minimal compromise on validation stability. The hybrid model and VGG16 were computationally intensive, this indicates that the model requires further optimization, such as transfer learning or fine-tuning, to be more practical in low-resource settings. One factor that played a role in this context was reducing image size. This effectively decreased training time across all four models, with minimal impact on accuracy. This further proves the feasibility of using smaller image resolutions for sign language recognition tasks in data-scarce contexts.

E. References

[1] I. Goodfellow, "Deep learning," ed: MIT press, 2016.

[2] O. Koller, N. C. Camgoz, H. Ney, and R. Bowden, "Weakly supervised learning with multi-stream CNN-LSTM-HMMs to discover sequential parallelism in sign language videos," *IEEE transactions on pattern analysis and machine intelligence,* vol. 42, no. 9, pp. 2306-2320, 2019.

[3] L. Liu, L. Gao, W. Lei, F. Ma, X. Lin, and J. Wang, "A Survey on Deep Multi-modal Learning for Body Language Recognition and Generation," *arXiv preprint arXiv:2308.08849,* 2023.

[4] M. D. Nareshkumar and B. Jaison, "A Light-Weight Deep Learning-Based Architecture for Sign Language Classification," *Intelligent Automation & Soft Computing,* vol. 35, no. 3, 2023.

[5] D. Alexey, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv: 2010.11929,* 2020.

[6] L. Liu, W. Zhou, W. Zhao, H. Hu, and H. Li, "Multi-modal sign language spotting by multi/one-shot learning," in *European Conference on Computer Vision*, 2022: Springer, pp. 256-270.

[7] T. Ananthanarayana *et al.*, "Deep learning methods for sign language translation," *ACM Transactions on Accessible Computing (TACCESS),* vol. 14, no. 4, pp. 1-30, 2021.

[8] S. Albanie *et al.*, "BSL-1K: Scaling up co-articulated sign language recognition using mouthing cues," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, 2020: Springer, pp. 35-53.

[9] M. M. Rahman, M. S. Islam, M. H. Rahman, R. Sassi, M. W. Rivolta, and M. Aktaruzzaman, "A new benchmark on american sign language recognition using convolutional neural network," in *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*, 2019: IEEE, pp. 1-6.

[10] C. K. Lee, K. K. Ng, C.-H. Chen, H. C. Lau, S. Y. Chung, and T. Tsoi, "American sign language recognition and training method with recurrent neural network," *Expert Systems with Applications,* vol. 167, p. 114403, 2021.

[11] J. Ekbote and M. Joshi, "Indian sign language recognition using ANN and SVM classifiers," in *2017 International conference on innovations in information, embedded and communication systems (ICIIECS)*, 2017: IEEE, pp. 1-5.

[12] T. N. Abu-Jamie and S. S. Abu-Naser, "Classification of sign-language using vgg16," 2022.

[13] K. Myagila and H. Kilavo, "A comparative study on performance of SVM and CNN in Tanzania sign language translation using image recognition," *Applied Artificial Intelligence,* vol. 36, no. 1, p. 2005297, 2022.

[14] A. Tharwat, T. Gaber, A. E. Hassanien, M. K. Shahin, and B. Refaat, "Sift-based arabic sign language recognition system," in *Afro-European Conference for Industrial Advancement: Proceedings of the First International Afro-European Conference for Industrial Advancement AECIA 2014*, 2015: Springer, pp. 359-370.

[15] S. Wang, D. Guo, W.-g. Zhou, Z.-J. Zha, and M. Wang, "Connectionist temporal fusion for sign language translation," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 1483-1491.

[16] A. Shahin and S. Almotairi, "Automated Arabic sign language recognition system based on deep transfer learning," *IJCSNS Int. J. Comput. Sci. Netw. Secur,* vol. 19, no. 10, pp. 144-152, 2019.

[17] R. Cui, H. Liu, and C. Zhang, "A deep neural framework for continuous sign language recognition by iterative training," *IEEE Transactions on Multimedia,* vol. 21, no. 7, pp. 1880-1891, 2019.

[18] K. Bantupalli and Y. Xie, "American sign language recognition using deep learning and computer vision," in *2018 IEEE international conference on big data (big data)*, 2018: IEEE, pp. 4896-4899.

[19] O. K. Oyedotun and A. Khashman, "Deep learning in vision-based static hand gesture recognition," *Neural Computing and Applications,* vol. 28, no. 12, pp. 3941-3951, 2017.

[20] D. Bendarkar, P. Somase, P. Rebari, R. Paturkar, and A. Khan, "Web based recognition and translation of American sign language with CNN and RNN," 2021.

[21] T. G. Moape, A. Muzambi, and B. Chimbo, "Convolutional Neural Network Approach for South African Sign Language Recognition and Translation," in *2024 Conference on Information Communications Technology and Society (ICTAS)*, 2024: IEEE, pp. 101-106.

[22] A. Jana and S. S. Krishnakumar, "Sign language gesture recognition with convolutional-type features on ensemble classifiers and hybrid artificial neural network," *Applied Sciences,* vol. 12, no. 14, p. 7303, 2022.
[23] B. Bhandari, "Comparative study of popular deep learning models for machining roughness classification using sound and force signals," *Micromachines,* vol. 12, no. 12, p. 1484, 2021.
[24] A. M. Buttar, U. Ahmad, A. H. Gumaei, A. Assiri, M. A. Akbar, and B. F. Alkhamees, "Deep learning in sign language recognition: a hybrid approach for the recognition of static and dynamic signs," *Mathematics,* vol. 11, no. 17, p. 3729, 2023.