








## Article

# HaCk: Hand Gesture Classification Using a Convolutional Neural Network and Generative Adversarial Network-Based Data Generation Model

Kalyan Chatterjee <sup>1</sup>, M. Raju <sup>1</sup>, N. Selvamuthukumar <sup>1</sup>, M. Pramod <sup>1</sup>, B. Krishna Kumar <sup>1</sup>,  
Anjan Bandyopadhyay <sup>2</sup> and Saurav Mallik <sup>3,4,\*</sup>

- <sup>1</sup> Department of Computer Science & Engineering, Nalla Malla Reddy Engineering College, Hyderabad 500088, Telangana, India; kalyanchatterjee@ieee.org or kalyanchatterjee.cse@nmrec.edu.in (K.C.); raju.cse@nmrec.edu.in (M.R.); selvamuthukumar.cse@nmrec.edu.in (N.S.); macherlapramod@gmail.com (M.P.); krishnakumar.cse@nmrec.edu.in (B.K.K.)
- <sup>2</sup> School of Computer Engineering, Kalinga Institute of Industrial Technology, Bhubaneswar 751024, Odisha, India; anjan.bandyopadhyayfcs@kiit.ac.in
- <sup>3</sup> Department of Environmental Health, Harvard T.H. Chan School of Public Health, Boston, MA 02115, USA
- <sup>4</sup> Department of Pharmacology & Toxicology, The University of Arizona, Tucson, MA 85721, USA
- \* Correspondence: sauravmtech2@gmail.com or smallik@hsph.harvard.edu

**Abstract:** According to global data on visual impairment from the World Health Organization in 2010, an estimated 285 million individuals, including 39 million who are blind, face visual impairments. These individuals use non-contact methods such as voice commands and hand gestures to interact with user interfaces. Recognizing the significance of hand gesture recognition for this vulnerable population and aiming to improve user usability, this study employs a Generative Adversarial Network (GAN) coupled with Convolutional Neural Network (CNN) techniques to generate a diverse set of hand gestures. Recognizing hand gestures using HaCk typically involves a two-step approach. First, the GAN is trained to generate synthetic hand gesture images, and then a separate CNN is employed to classify gestures in real-world data. The evaluation of HaCk is demonstrated through a comparative analysis using Leave-One-Out Cross-Validation (LOO CV) and Holdout Cross-Validation (Holdout CV) tests. These tests are crucial for assessing the model's generalization, robustness, and suitability for practical applications. The experimental results reveal that the performance of HaCk surpasses that of other compared ML/DL models, including CNN, FTCNN, CDCGAN, GestureGAN, GGAN, MHG-CAN, and ASL models. Specifically, the improvement percentages for the LOO CV Test are 17.03%, 20.27%, 15.76%, 13.76%, 10.16%, 5.90%, and 15.90%, respectively. Similarly, for the Holdout CV Test, HaCk outperforms HU, ZM, GB, GB-ZM, GB-HU, CDCGAN, GestureGAN, GGAN, MHG-CAN, and ASL models, with improvement percentages of 56.87%, 15.91%, 13.97%, 24.81%, 23.52%, 17.72%, 15.72%, 12.12%, 7.94%, and 17.94%, respectively.

**Keywords:** CNN; GAN; hand gesture; Holdout CV test; LOO CV test



**Citation:** Chatterjee, K.; Raju, M.; Selvamuthukumar, N.; Pramod, M.; Krishna Kumar, B.; Bandyopadhyay, A.; Mallik, S. HaCk: Hand Gesture Classification Using a Convolutional Neural Network and Generative Adversarial Network-Based Data Generation Model. *Information* **2024**, *15*, 85. <https://doi.org/10.3390/info15020085>

Academic Editors: Haifeng Wang, Norma B. Ojeda and Lu He

Received: 23 December 2023

Revised: 19 January 2024

Accepted: 26 January 2024

Published: 4 February 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Generative Adversarial Networks (GANs) play a crucial role in diverse areas of data science. One common hurdle in numerous data science projects revolves around insufficient or imbalanced datasets. Addressing this challenge necessitates the availability of diverse generators capable of producing data tailored to specific requirements.

In the context of hand gesture recognition (GR), a sizable dataset comprising diverse hand gesture images is typically necessary. Conventional methods involve manual capture through cameras or utilizing existing benchmark datasets online. In contrast, our proposed HaCk system introduces an alternative by generating these gestures using a pretrained generator. The resulting images exhibit variations in similarity, with each image in a specific

category representing the same gesture from different perspectives. Consequently, a CNN trained on this generated dataset can discern the inherent variations within each gesture.

The core concept of this study revolves around employing a dataset encompassing five categories, each comprising 6600 images of a specific hand gesture. This dataset is expanded via image augmentation techniques using an available online dataset. The GAN is trained with this augmented dataset, empowering the generator to produce diverse hand gestures in desired quantities. Subsequently, this generated dataset is utilized to train a CNN-based classifier, facilitating the straightforward classification of various hand gestures.

### 1.1. Research Gap

Existing hand-based GR systems typically utilize real hand gesture images as the training dataset. However, a limitation of this approach is the limited variation within the dataset. As a result, the model cannot handle unfavorable scenarios, such as unusual hand shapes, finger orientations, or cases where a person has only four fingers. Consequently, models trained solely on real hand gesture images can be apparent in such situations. Aside from that, we also identified the following limitations:

1. **Limited Diversity in Generated Data:** Many GAN-based data generation models tend to produce synthetic data that closely resemble the training data. However, these data need to enhance the diversity of generated hand gestures, ensuring that the CNN classifier is robust in recognizing a more comprehensive range of gestures and variations.
2. **Small Imbalanced Dataset:** Hand gesture datasets are often limited in size, leading to challenges in effectively training a CNN classifier. Addressing this gap may involve investigating techniques for data augmentation and class imbalances in generated datasets.
3. **Generalization to Real-World Conditions:** GAN-generated data may only partially capture the complexity of real-world scenarios, such as lighting, background, and noise variations.
4. **Optimal Architectures and Hyperparameters:** Identifying the most suitable CNN architectures and hyperparameters for hand gesture recognition tasks using GAN-generated data is an ongoing challenge.
5. **Efficiency and Real-Time Processing:** Real-time hand gesture recognition applications, such as sign language interpretation and gesture-based interfaces, require efficient CNN models. One research gap involves developing CNN architectures that balance accuracy and computational efficiency with real-time processing.

To overcome these challenges, we propose a solution in which we generate images of hand gestures that closely resemble real ones but exhibit all possible variations. These generated images are combined with real samples to form a comprehensive dataset. By training a CNN using this augmented dataset, the model gains the capability to handle the unusual scenarios mentioned earlier. Also, this can improve such systems' accuracy, robustness, and applicability.

### 1.2. Motivation

The development of the *Hand Gesture Classification Using a Convolutional Neural Network and Generative Adversarial Network-Based Data Generation Model (HaCk)* employed a combination of Convolutional Neural Network (CNN) and Generative Adversarial Network (GAN)-based data generation. The rationale behind this approach was to use the strengths of CNNs for effective hand gesture classification while leveraging GANs for generating synthetic data in order to enhance *HaCk's* robustness and generalization capabilities.

By incorporating a CNN, the authors aimed to exploit the network's ability to automatically learn hierarchical features from input images, making it suitable for hand gesture classification. Additionally, integrating a GAN-based data generation model allowed for the creation of diverse and realistic synthetic hand gesture data. This synthetic data aug-

mentation approach was expected to address potential limitations related to insufficient real-world data, thereby improving the model's overall performance and adaptability.

The discussion of possible outcomes involved exploring the effectiveness of the combined CNN and GAN approach in achieving accurate and reliable hand gesture classification results. The authors considered factors such as increased classification accuracy, enhanced *HaCk* robustness to variations in hand gestures, and the potential for the model to generalize well to previously unseen data. The incorporation of synthetic data through GANs was expected to contribute to a more comprehensive and versatile hand gesture classification system.

The significant contributions of this study are as follows:

1. This study proposes an unsupervised learning model focused on data efficiency to augment skeleton-based data, *HaCk*. *HaCk* can adapt to novel data containing previously unseen classes once trained.
2. As it does not necessitate prior knowledge and examination of input data during training, *HaCk* facilitates an automated cross-domain data augmentation procedure, requiring minimal hyperparameter adjustment.
3. *HaCk* utilizes real data for training customized gesture classifiers, allowing users to significantly reduce the effort needed to gather individualized training data.

The remaining sections of this study are organized as follows: Section 2 presents the literature survey conducted to develop our *HaCk* system. Section 3 outlines the system architecture and problem formulation of *HaCk*. Section 4 presents a description of the dataset used. Section 5 details the development process of the proposed *HaCk*. Section 6 evaluates the performance of the proposed model. Lastly, Section 7 summarizes the conclusions and outlines potential future work.

## 2. Literature Survey

In this section, we provide relevant studies related to the utilization of deep learning (DL) techniques for gesture recognition (GR), customized classifiers, domain translation, and style transfer networks (STNs).

### 2.1. The Application of DL Techniques in GR and Human Activity Recognition

Deep learning (DL) has emerged as a powerful tool for image recognition, feature detection, and pattern recognition. Similarly, notable advancements have been witnessed in Gesture Recognition (GR) and Human Activity Recognition (HAR) through DL techniques. It surpasses other ML techniques like SVM.

DL-based models have crucial advantages over ML models [1,2], as described below:

- DL models obviate the requirement for manual feature extraction, eliminating the dependence on domain-specific knowledge.
- DL models can learn more complex and profound features compared to ML-based heuristic approaches.
- DL models can utilize unlabeled data during training, whereas traditional ML methods rely heavily on a significant amount of labeled data.

Various strategies have been investigated and assessed for devising architectures of Deep Neural Networks (DNNs) in this domain. Since sensory data encompass prolonged time series signals, a vital consideration entails devising convolutional units that can adeptly capture temporal features. Initially, early studies employed 1D convolution for each sensor channel [3]. However, this structure proved inadequate in capturing the interaction between different sensor channels, prompting the introduction of recurrent neural networks (RNNs) and 2D convolution [4]. Furthermore, to model inter-channel relationships and long-term dependencies within sensory signals, researchers designed 2D convolution with RNNs and LSTM neural networks to capture the long-term sequential correlations [5–7].

Fang et al. introduced an algorithm for gesture recognition that integrates CNNs and Deep Convolutional Generative Adversarial Networks (DHaCk), namely CDCGAN. This algorithm enhanced traditional gesture recognition methods by showcasing increased resilience to illumination variations and background interference [8].

Tang et al. introduced the GestureGAN model, designed to translate wild hand gestures into gestures, emphasizing a deep understanding of the high-level mapping between the input source and the desired output target gesture. The enhancement in the performance of the GestureGAN model was achieved through improvements in the hand gesture classifier's accuracy and effectiveness [9].

Zhu et al. introduced a GGAN model for gesture recognition utilizing Generative Adversarial Networks. This GGAN model operates by employing a deep convolutional structure as the discriminator and a deep transpose convolutional structure as the generator [10].

Garg et al. introduced the Multiview Hand Gestures with Conditional Adversarial Network (MHG-CAN), which focuses on a gesture synthesis model that employs conditional translation. Additionally, the MHG-CAN model incorporates multiviews for recognizing hand gestures [11].

Barbhuiya et al. presented a resilient model for hand gesture recognition through the utilization of deep ensemble neural networks (i.e., the ASL model). The initial step involves creating a pretrained network using the VGG16 architecture, incorporating a self-attention layer within the VGG16 structure. The integration of this self-attention module facilitates the acquisition of distinctive image features, enhancing the differentiation among various gesture categories [12].

## 2.2. Customized Classifiers

All the aforementioned methods involve training for hand gesture recognition using pre-labeled data. Despite the applicability of these user-independent classifiers across various users, several studies have pointed out lower accuracy levels compared to customized classifiers trained with data explicitly tailored to the target user.

Weiss and Lockhart [13] presented evidence of the effectiveness of custom classifiers in a human activity recognition task using smartphone sensors.

Fallahzadeh and Ghasemzadeh [14] developed an activity recognition algorithm.

Siirtolas et al. [15] presented the efficacy of custom classifiers through incremental learning. An accuracy improvement of up to 3.99% was achieved in linear and quadratic discriminant analysis.

## 2.3. Domain Adaptation and STNs

We aim to develop customized classifiers without the need for specific examples of target gestures. To achieve this, machine learning (ML) and computer vision (CV) were used to introduce domain adaptation and various techniques of STNs.

Most studies leverage the Generative Adversarial Network, wherein a generator network (GN) creates a new image based on an input. In contrast, a discriminator network (DN) evaluates its authenticity and domain label. Traditionally, the supervised translation of images has relied on paired training data [16]. However, recent interest has shifted towards unsupervised translation, which eliminates the need for paired images based on the concept of cycle consistency [17]. This cycle consistency ensures that a translated image can be accurately converted to its original form.

However, this approach is limited to one-to-one translation between domains. To address this limitation, Almahairi et al. introduced Augmented CycleGAN [18]. Augmented CycleGAN allows for one-to-many or many-to-many translation mappings by incorporating auxiliary variables that capture variations independent of the content being translated.

Another noteworthy advancement is StyleGAN, proposed by Karras et al. [19]. The joint learning in StyleGAN enhances training efficiency and improves outcomes compared to pairwise transformations.

Considerable advancements have been made in computer vision and graphics image stylization techniques. One popular approach involves manipulating the intermediate features of a CNN. The gram matrix was utilized for CNN's middle feature extraction from a pretrained VGG network to represent style [20–28]. Similarly, Holden et al. proposed analogous ideas for motion stylization [29].

Wu et al. proposed patch swapping between feature maps of the input and style images to achieve stylization [20]. Sheng et al. proposed the features of whitening and de-whitening to preserve the input image's global structure while reflecting the reference image's style [27]. Dan [21] proposed an image style by adjusting the retrieval, description, and stylization. Johnson et al. proposed perceptual losses for real-time style transfer and super resolution [25].

Building upon these concepts, we use STNs for GR to train customized classifiers. We extend the capabilities of *HaCk* to facilitate the translation of sensory data among different users and gesture classes.

### 3. System Architecture and Problem Formulation

#### 3.1. System Architecture

The *HaCk* system comprises a Conditional Generative Adversarial Network and a Convolution Neural Network as a classifier trained using the dataset. This dataset was created by expanding the initially captured dataset through image augmentation techniques. After that, the Conditional Generative Adversarial Network underwent training, allowing its generator component to produce a dataset that closely resembles real data but includes variations, resulting in the *generated dataset*. As a result of this process, two distinct datasets were created: one containing *real images*, and the other containing images generated by *HaCk*.

Following the generation of this dataset, our proposed *HaCk* system proceeded to train two separate CNN models. One of these models was trained using the real data samples obtained from the original dataset. In contrast, the other was trained using the generated dataset produced by the Conditional Generative Adversarial Network generator. This means two CNN models were produced, each trained on a different dataset—one on real images and the other on synthetic images generated by *HaCk*.

In the final step of the system's operation, real-time images were captured through a webcam. These real-time images, which had not been previously seen by the system, were then used to make predictions. Both CNN models, the one trained on real data and the one trained on the generated dataset, were employed to predict the labels of these newly captured images. Once predictions had been made, the accuracy of both CNN networks was compared. This comparison assessed how well each CNN performed in recognizing and classifying the real-time images, providing insights into the system's effectiveness for hand gesture recognition.

Hence, our proposed *HaCk* system uses a conditional Generative Adversarial Network to generate synthetic data, trains two CNN models—one on real data and the other on the generated data—and then evaluates their performance in real-time image classification to gauge the utility of the generated data for improving the CNN-based hand gesture recognition system.

Figure 1 illustrates the system architecture of our proposed *HaCk* system.

#### 3.2. Problem Formulation

To design the objective function, let us assume that  $\mathbf{X}$  represents the input data representing hand gestures,  $\mathbf{Y}$  represents the corresponding class labels,  $\mathbf{C}$  is the CNN,  $\mathbf{G}$  is the GAN, and  $\mathbf{D}$  is the data generation model.

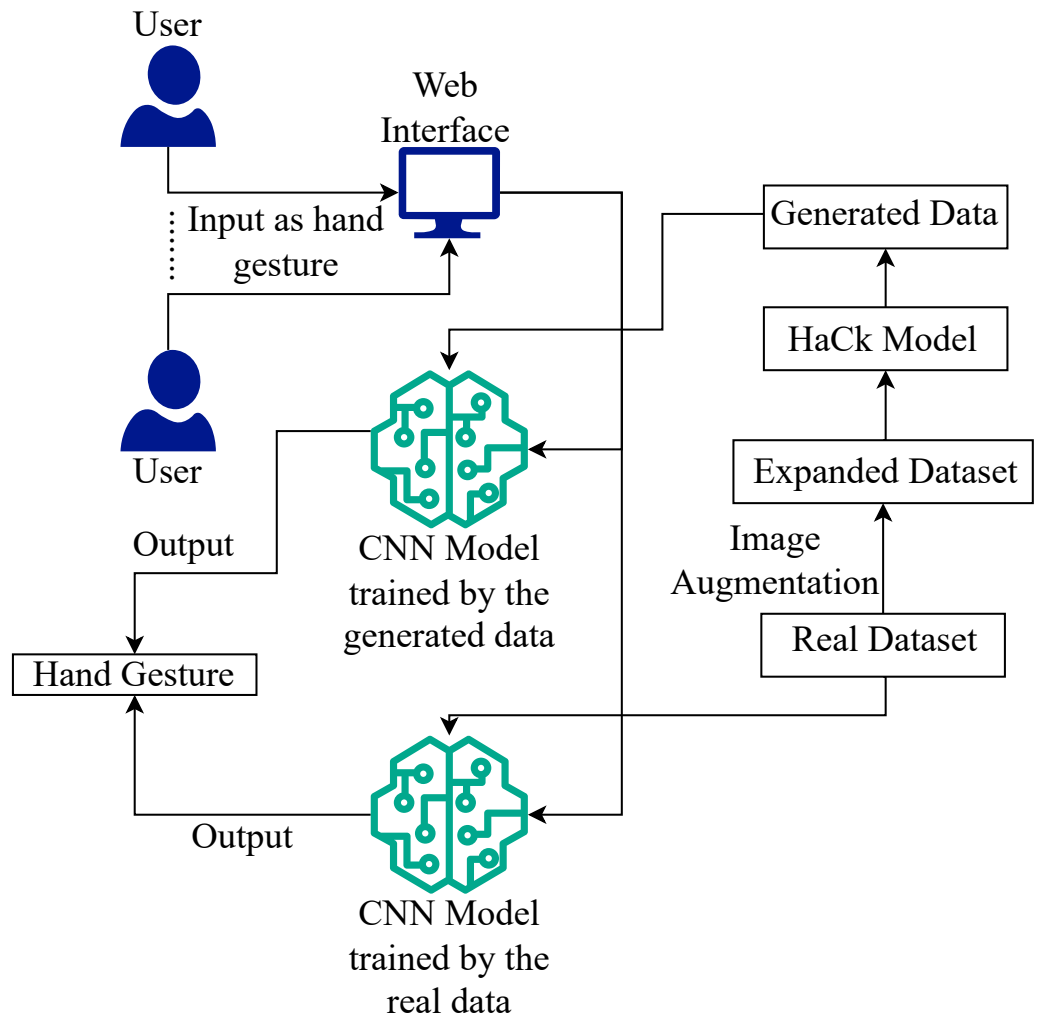


Figure 1. System architecture of the HaCk system.

### 3.2.1. CNN Formulation

The CNN  $C$  is designed to map input data  $X$  to class labels  $Y$ . Let  $W$  and  $b$  represent the weights and biases of the CNN layers.

$$Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]} \tag{1}$$

$$A^{[l]} = \sigma(Z^{[l]}) \tag{2}$$

$$\mathcal{L}_{CNN}(Y, \hat{Y}) = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^n (Y_{ij} \log(\hat{Y}_{ij}) + (1 - Y_{ij}) \log(1 - \hat{Y}_{ij})) \tag{3}$$

where:

- $Z^{[l]}$  is the linear output of layer  $l$ ;
- $A^{[l]}$  is the activation of layer  $l$ ;
- $\sigma$  is the activation function;
- $\mathcal{L}_{CNN}$  is the CNN loss function;
- $\hat{Y}$  is the predicted output;
- $m$  is the number of samples;
- $n$  is the number of classes.

### 3.2.2. GAN and Data Generation Formulation

The GAN  $\mathbf{G}$  is used to generate synthetic data  $\mathbf{X}_{\text{synthetic}}$ . The data generation model  $\mathbf{D}$  is employed to enhance the quality of generated data.

$$\mathbf{Z}_{\text{latent}} \sim \mathcal{N}(0, 1) \quad (4)$$

$$\mathbf{X}_{\text{synthetic}} = \mathbf{G}(\mathbf{Z}_{\text{latent}}) \quad (5)$$

$$\mathbf{X}_{\text{augmented}} = \mathbf{D}(\mathbf{X}, \mathbf{X}_{\text{synthetic}}) \quad (6)$$

$$\mathcal{L}_{\text{GAN}}(\mathbf{X}_{\text{synthetic}}) = \log(\mathbf{D}(\mathbf{X}_{\text{synthetic}})) \quad (7)$$

where:

- $\mathbf{Z}_{\text{latent}}$  is the latent vector;
- $\mathbf{X}_{\text{augmented}}$  is the augmented dataset;
- $\mathcal{L}_{\text{GAN}}$  is the GAN loss function.

## 4. Dataset

The hand gesture recognition database was obtained from [30], composed of a set of near-infrared images acquired by a Leap Motion sensor. The database comprises ten different hand gestures performed by ten subjects (five men and five women).

### Dataset Description

This image database utilizes the infrared data provided by the Leap Motion device for hand-related information. Consequently, this database is designed by collecting samples of various hand poses using a Leap Motion sensor positioned on a table. Subjects were seated near the sensor and moved their right hand within a range of 10 to 15 cm in front of it. A diverse set of 10 distinct hand gestures was performed by ten different individuals, consisting of five women and five men. Each subject and gesture combination resulted in the recording of 200 frames.

Figure 2 illustrates a selection of these hand gestures performed by different subjects. The top row, from left to right, showcases the following gestures: an open palm parallel to the sensor (Palm), a closed palm with the thumb and index fingers extended (L), a closed fist (Fist), a fist perpendicular to the sensor (Fist m), and a closed palm with the thumb extended (Thumb). The bottom row, also from left to right, displays the following gestures: a closed palm with the index extended (Index), an open palm with the index and thumb forming a circle (OK), an open palm perpendicular to the sensor (Palm m), a partially closed palm in a C shape (C), and an open palm with all fingers spread apart (Palm d).



Figure 2. Data samples [30].

## 5. Proposed HaCk System

This study presents a hand GR system capable of classifying different hand gestures into their respective classes and assigning appropriate labels. Our *HaCk* system can generate synthetic data that resemble real-world data while introducing variations. This capability is particularly beneficial in augmenting datasets for hand gesture recognition, thereby addressing potential limitations in the availability of diverse real-world data.



The reasons behind the selection of a CNN over all other classifiers, including Decision Trees (DTs), Logistic Regression (LR), K-Nearest Neighbor (KNN), Artificial Neural Networks (ANNs), and Support Vector Machines (SVMs), are as follows:

- **Feature Extraction:** CNNs are known for their ability to automatically learn and extract relevant features from image data. Also, CNNs are used for image classification tasks due to their hierarchical feature extraction capabilities.
- **Spatial Hierarchies:** CNNs are designed to capture spatial hierarchies in images, recognizing patterns at different scales. This architecture aligns with the hierarchical nature of hand gesture recognition, where specific hand shapes and movements may be composed of more minor, recognizable elements.
- **Computational Efficiency:** CNNs are known for their computational efficiency in processing image data, which can be important in real-time or resource-constrained applications.

*HaCk* comprises the following three phases: (i) data generation, (ii) training, and (iii) testing.

### 5.1. Phase I: Data Generation Phase

In this phase, we applied various image augmentation techniques, such as flipping, shifting, and rotation, to a dataset comprising five classes of hand gestures, with each class containing 550 images. This operation expanded the dataset, resulting in five classes with 6600 images in each class. The expanded dataset was then pre-processed and converted into NumPy arrays to train a conditional GAN. After training, the GAN's generator successfully generated images of different hand gestures, resulting in the generated dataset.

#### 5.1.1. Discriminator

The output layer of the model consists of only two neurons, forming a simple classifier. After each epoch, the model undergoes improvement by updating its weights using real and fake samples. The discriminator assigns a value of 0 to fake data and 1 to real data, allowing it to classify generated data from the generator as fake samples. The discriminator receives input data of the same dimension as the actual data, whether real or fake, and produces an output of either 0 or 1, corresponding to fake or real samples, respectively. The sigmoid activation function is utilized for model training. The loss function employed by the discriminator is binary cross-entropy, as described in Equation (8).

$$\Delta\theta_d = \frac{1}{m} \sum_{i=1}^m \left( \log D(x^{(i)}) + \log \left( 1 - D(G(z^{(i)})) \right) \right) \quad (8)$$

Here,  $\Delta\theta_d$  defines the loss function for the discriminator,  $m$  represents the total number of samples in the test dataset, and  $D$  refers to the discriminator, which is a function or model used in GANs to distinguish between real and generated data. The discriminator ( $D$ ) wants to maximize this stochastic function.  $x$  represents the real data samples,  $G$  refers to the generator, which is used in GANs to generate synthetic or fake data, and  $z$  represents the input noise vector used as an input to the generator to generate fake data.

Algorithm 1 outlines the employed discrimination procedure in detail.

#### 5.1.2. Generator

The generator in *HaCk* is responsible for generating data samples based on the input noise. It continually improves the performance of *HaCk* by learning from the feedback provided by the discriminator. Its main objective is to deceive the discriminator by generating data that appear to be fake but which do not contain fake information. The generator inputs random noise and produces generated data that resemble the original samples. The activation function used in the generator model is the hyperbolic tangent (tanh) function. Algorithm 2 outlines the employed generator procedure in detail.



**Algorithm 1** Discriminator Algorithm

---

```

1: Input: Real data samples  $\mathbf{X}$ , synthetic data samples  $\mathbf{X}_{\text{synthetic}}$ 
2: Output: Discriminator loss  $\mathcal{L}_D$ 
3: procedure TRAINDISCRIMINATOR( $\mathbf{X}$ ,  $\mathbf{X}_{\text{synthetic}}$ )
4:   Initialize discriminator parameters  $\theta_D$ 
5:   for epoch = 1 to  $N_{\text{epochs}}$  do
6:     for  $\mathbf{x} \in \mathbf{X}$  do
7:        $\mathcal{L}_{\text{real}} = -\log(D(\mathbf{x}))$ 
8:       Update  $\theta_D$  using gradient of  $\mathcal{L}_{\text{real}}$  w.r.t.  $\theta_D$ 
9:     for  $\mathbf{x}_{\text{synthetic}} \in \mathbf{X}_{\text{synthetic}}$  do
10:       $\mathcal{L}_{\text{synthetic}} = -\log(1 - D(\mathbf{x}_{\text{synthetic}}))$ 
11:      Update  $\theta_D$  using gradient of  $\mathcal{L}_{\text{synthetic}}$  w.r.t.  $\theta_D$ 
12:    $\mathcal{L}_D = \frac{1}{2}(\mathcal{L}_{\text{real}} + \mathcal{L}_{\text{synthetic}})$ 
13:   return  $\mathcal{L}_D$ 

```

---

**Algorithm 2** Generator Algorithm

---

```

1: Input: Latent vector  $\mathbf{Z}$ 
2: Output: Synthetic data sample  $\mathbf{X}_{\text{synthetic}}$ 
3: procedure GENERATEDATA( $\mathbf{Z}$ )
4:   Initialize generator parameters  $\theta_G$ 
5:   Generate synthetic data:  $\mathbf{X}_{\text{synthetic}} = G(\mathbf{Z}; \theta_G)$ 
6:   return  $\mathbf{X}_{\text{synthetic}}$ 

```

---

The loss function for the generator is similar to that of the discriminator, but in the case of the generator, the loss for real data samples is set to zero. As a result, the first term in the expression is eliminated, and the equation is simplified to the following mathematical expression, as shown in Equation (9).

$$\Delta\theta_g = \frac{1}{m} \sum_{i=1}^m \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \quad (9)$$

Here,  $\Delta\theta_g$  defines the loss function for the generator (G), and the generator (G) wants to minimize this stochastic function.

Figure 3 depicts the loss function for the generator (G).

### 5.2. Phase II: Training Phase

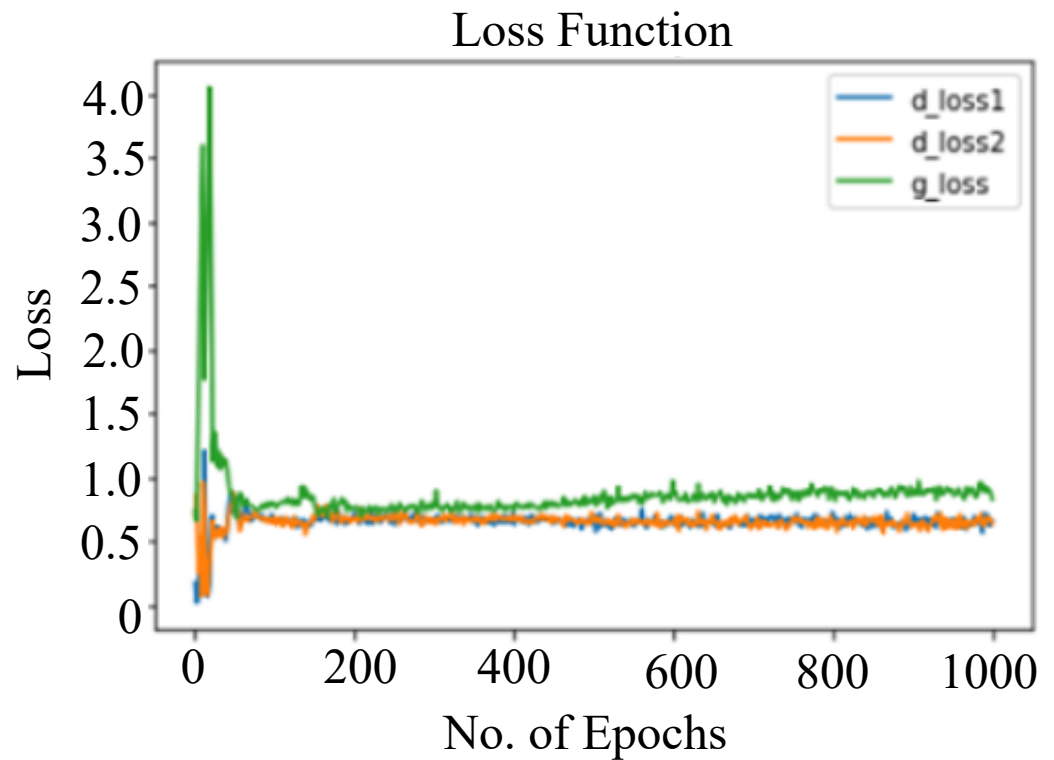
In the training phase, two CNNs are trained using different datasets. One model is trained using real data samples (hand gestures). In contrast, the other model is trained using generated data samples from the generator of the conditional GAN model, as described in Figure 4. Both CNN networks have the same configuration, and the hyperparameter setup is described in Table 1.

Choosing hyperparameters for CNN and GAN models is crucial to achieving an optimal performance. Therefore, we discuss the hyperparameters for each type of model, as follows:

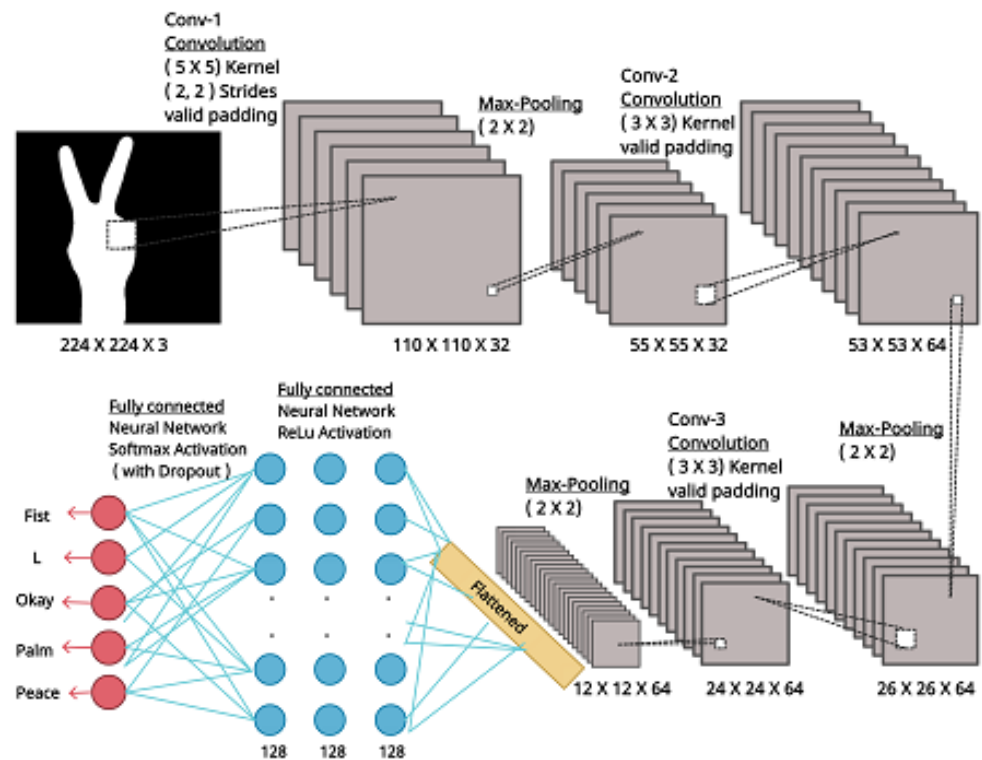
#### 1. Hyperparameters for CNN:

- (a) **Number of Layers:** This includes the number of convolutional layers, pooling layers, and fully connected layers. The architecture of the CNN dramatically impacts its ability to learn complex features from the data.
- (b) **Filter Size and Stride:** The size of convolutional filters (kernels) and the stride at which they move over the input data determine the spatial characteristics that the network can capture.

- (c) **Number of Filters:** The number of filters in each convolutional layer affects the depth of the network and its capacity to learn features at different levels of abstraction.
  - (d) **Activation Functions:** Choices like the use of Rectified Linear Units (ReLU) are standard, but other activation functions like Leaky ReLU or Sigmoid can also be used. The chosen activation functions affect the network's non-linearity.
  - (e) **Dropout Rate:** Dropout is a regularization technique that helps prevent overfitting by randomly dropping a fraction of neurons during training. The dropout rate is a hyperparameter determining the number of neurons to drop.
  - (f) **Batch Size:** The number of data samples used in each training iteration (mini batch) can impact training speed and generalization. It is important to find a balance between computational efficiency and convergence.
  - (g) **Learning Rate:** The learning rate determines the step size during gradient descent optimization. It must be carefully tuned to ensure convergence without overshooting or becoming stuck in local minima.
  - (h) **Weight Initialization:** How the network weights are initialized can affect training. Standard methods include random initialization and techniques like Xavier/Glorot initialization.
  - (i) **Optimizer:** The choice of optimization algorithms, such as Stochastic Gradient Descent (SGD), Adam, or RMSProp, can influence the convergence speed and final performance.
  - (j) **Padding:** Padding can be *valid* (no padding) or the *same* (zero-padding), and this affects the spatial dimensions of the output feature maps after convolution.
2. **Hyperparameters for GAN:**
- (a) **Generator Architecture:** Similar to CNN, the architecture of the generator network is crucial. This includes the number of layers, filter sizes, and activation functions.
  - (b) **Discriminator Architecture:** The discriminator must also be designed appropriately. It should be capable of distinguishing between real and generated data effectively.
  - (c) **Learning Rate:** The learning rates for both the generator and discriminator are essential. They can impact the stability of GAN training. Sometimes, different learning rates are used for each network.
  - (d) **Loss Functions:** GANs use two loss functions—generator loss (often a form of binary cross-entropy) and the discriminator loss. The choice of these loss functions can influence the quality of generated samples.
  - (e) **Noise Dimension:** GANs often take random noise as inputs to generate data. The dimensionality and distribution of this noise can impact the diversity and quality of generated samples.
  - (f) **Batch Size:** Similar to CNNs, the batch size used during GAN training can affect the stability and convergence of the model.
  - (g) **Training Duration:** Deciding when to stop training is essential. Training GANs can be tricky, and setting the number of epochs or other stopping criteria is a hyperparameter choice.
  - (h) **Regularization Techniques:** Techniques like weight clipping (for Wasserstein GANs), gradient penalties, and feature matching can stabilize training and improve sample quality.
  - (i) **Architecture Variants:** There are various GAN variants, such as Deep Convolutional GAN (DHaCK), Wasserstein GAN (WGAN), and more. The choice of GAN architecture should align with the specific task and data.
  - (j) **Data Preprocessing:** Data preprocessing, including normalization and scaling, can also be considered part of the hyperparameter selection process.



**Figure 3.** Plot of loss functions with numbers of epochs, where d\_loss1 and d\_loss2 are the discriminator’s loss on real and generated samples, respectively, and g\_loss is the loss of the generator (G).



**Figure 4.** The training phase using the CNN.

Each CNN model consists of an input layer with dimensions of  $224 \times 224 \times 3$ , matching the input image dimension. They also include three convolution layers: the first with a

kernel size of  $5 \times 5$ , 32 filters, 2,2 strides, and valid padding; the second with a kernel size of  $3 \times 3$ , 64 filters, and valid padding; and the third with a kernel size of  $3 \times 3$ , 64 filters, and valid padding. Three max pooling layers with a kernel size of  $2 \times 2$  are also inserted between the convolution layers. After flattening, there are three hidden layers, each with 128 neurons.

**Table 1.** Best Hyperparameter configuration.

Hyperparameter	Value
Loss	Validation MSE
Optimizer	Softmax
Metrics	MSE, and MAE
Batch size	68
Time step	1
Epochs	500
Number of CNN layers	300
Learning rate	0.211
Dropout	0.2
Activation function	ReLU

During training, the input dimensions ( $224 \times 224 \times 3$ ) are reduced to  $110 \times 110 \times 32$  after the first convolution operation, then further reduced to  $55 \times 55 \times 32$  after max pooling. Similarly, after the second convolution operation, the dimensions are reduced to  $53 \times 53 \times 64$ , and after max pooling, they become  $26 \times 26 \times 64$ . After the third convolution operation, the dimensions are reduced to  $24 \times 24 \times 64$ , and after max pooling, they become  $12 \times 12 \times 64$ .

Upon flattening, the total number of parameters is calculated as  $12 \times 12 \times 64 = 9216$ . This is connected to a hidden layer with 128 neurons, resulting in a total number of parameters of  $9216 \times 128 = 1,179,648$ . This is further connected to a second hidden layer, increasing the total number of parameters to  $1,179,648 \times 128 = 150,994,944$ . Finally, the second hidden layer is connected to a third hidden layer, resulting in a total number of parameters of  $150,994,944 \times 128 = 19,327,352,832$ . The output layer, consisting of five neurons, represents the five classes (Fist, L, OK, Palm, and Peace) for classification. After training, the model can predict the probability of an input image belonging to a particular class.

### 5.3. Phase III: Testing Phase

In the real-time testing environment, we utilized the webcam of a laptop to capture gestures continuously. We mapped different keyboard keys to various functionalities, such as using 'B' to capture the background, 'R' to reset the background, and the space bar to capture an image and predict its class using our trained CNN model. The predicted class was then displayed alongside the captured gesture in real time.

We employed the background subtraction technique to extract hand images from the overall captured images. This process involved converting the image from BGR to GRAY, keeping the hand white and the rest of the image black. By utilizing functions like `cvtColor` and `GaussianBlur`, we transformed the image into a binary image, enabling our model to recognize hand gestures of individuals with different complexions, such as fair or wheatish.

Subsequently, the pre-processed hand gesture images were passed through the CNN model for prediction. The model determined the class of the gesture—whether it was Fist, L, OK, Palm, or Peace. To initiate result prediction, we used the space bar. When pressed, the model employed its CNN network to predict the class or category of the image and calculate the probability of it belonging to each class. The class with the highest probability was designated as the class of the input image. We then combined the result with the output image, creating a black-and-white representation of the hand gesture, its label, and the corresponding action to be performed.

#### 5.4. Algorithm and Flow Chart of the HaCk System

The integration process of CNN and GAN for Hand Gesture Classification using HaCk can be described as follows:

##### 5.4.1. CNN Integration

The Convolutional Neural Network (CNN) serves as the primary classifier for hand gesture recognition. Its architecture is designed to effectively capture hierarchical features within the input images. The CNN comprises multiple convolutional layers, each followed by activation functions and pooling layers. These convolutional layers are instrumental in automatically learning spatial hierarchies and discriminating features, which are crucial for distinguishing various hand gestures.

In our integration, we adopted a modified VGG16 architecture as the backbone of our CNN. This architecture has proved to be effective in image classification tasks. Moreover, to enhance the network's attention to salient image features, a self-attention mechanism was embedded within the VGG16 structure. This self-attention module enables the model to focus on distinguishing features, improving the overall differentiability among different hand gesture categories.

##### 5.4.2. GAN Integration

To address data scarcity and potentially improve the model's robustness, we introduced a GAN for data generation. The GAN consists of a generator and a discriminator.

##### 5.4.3. Integration and Training Procedure

The integration of CNN and GAN involves a joint training procedure to improve both the classification performance and the quality of generated data, described as follows:

- *Pretraining the CNN:* The CNN is initially pretrained using available labeled data for hand gestures. This step provides the CNN with a foundation for recognizing real hand gestures accurately.
- *Adversarial Training:* GAN training involves a two-step process. First, the discriminator is trained on both real and synthetic hand gesture images, which are optimized for accurate discrimination. Subsequently, the generator is trained to produce synthetic images that can effectively deceive the discriminator. This adversarial training loop is iterated to refine both the generator and discriminator.
- *Fine-Tuning the CNN:* The CNN is then fine-tuned using the combined dataset of real and synthetic hand gesture images generated by the GAN. This step helps the CNN adapt to the augmented dataset, potentially improving its generalization to unseen gestures.

##### 5.4.4. Parameter Selection Rationale

The selection of hyperparameters, such as learning rates, batch sizes, and the architecture of both the CNN and GAN, is crucial for the success of the integrated model. These parameters were chosen through a systematic exploration and validation process, aiming for stable convergence during training and optimal performance on the task of hand gesture classification.

This integrated approach leverages the strengths of both CNN and GAN, enhancing the model's ability to recognize diverse hand gestures even in the presence of limited labeled data, as described in Table 1.

Algorithm 3 outlines the hand GR process implemented in HaCk, while the workflow of the proposed HaCk is illustrated in Figure 5.

---

**Algorithm 3** Hand Gesture Classification Using a Convolutional Neural Network and Generative Adversarial Network-Based Data Generation Model (*HaCk*)
 

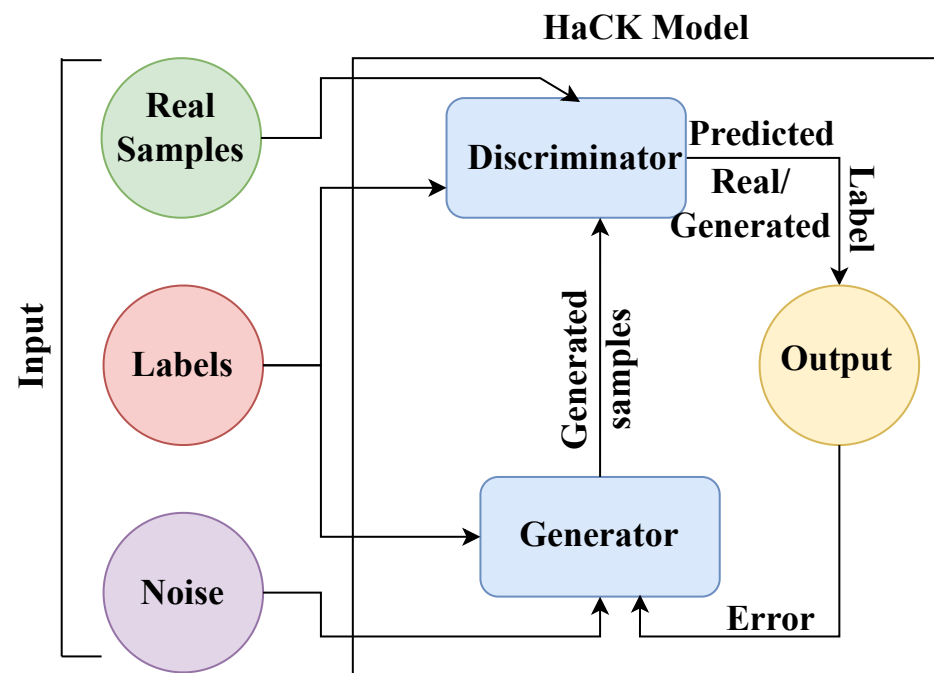
---

**Require:**

- 1: *G*: Real and generated hand gesture images from test data

**Ensure:**

- 2: *R*: Predicted class labels for input images
  - 3: **Begin**
  - 4: **Phase-I**
  - 5: Initialize the CNN-based classifier model
  - 6: Train the generator of the GAN model using the generated hand gesture images
  - 7: Generate a dataset containing both real and generated hand gesture images
  - 8: Preprocess the dataset by converting images to grayscale and resizing them to a fixed size
  - 9: Split the dataset into training and testing sets
  - 10: **Phase-II**
  - 11: Initialize the GAN model
  - 12: Train the CNN classifier model using the training set
  - 13: Test the trained model on the testing set
  - 14: Compute the accuracy of the classifier
  - 15: Predict the class labels for newly input hand gesture images using the trained model
  - 16: **End**
- 



**Figure 5.** Conditional GAN workflow.

## 6. Experimental Results and Discussion

In this section, we demonstrate the performance of our proposed *HaCk* model through a comparative analysis using LOO CV and Holdout CV tests.

The compared ML/DL models are: (i) CNN [31], (ii) FTCNN [32], (iii) CDCGAN [8], (iv) GestureGAN [9], (v) GGAN [10], (vi) MHG-CAN [11], and (vii) ASL [12] for LOO CV Test. For Holdout CV Test comparisons, the used ML/DL models are: (i) HU [33], (ii) ZM [33], (iii) GB [33], (iv) GB-ZM [33], (v) GB-HU [33], (vii) CDCGAN [8], (viii) GestureGAN [9], (ix) GGAN [10], (x) MHG-CAN [11], and (xi) ASL [12].

The evaluation of the GAN-generated data against real-world datasets is used for validating *HaCk*'s effectiveness, as described in Table 2.



**Table 2.** Evaluation of *HaCk*.

Dataset	Metric	Real-World Data	GAN-Generated Data	Combined Data (Real + GAN)
Test Set	Accuracy	0.85	0.78	0.87
	Precision	0.88	0.76	0.89
	Recall	0.82	0.80	0.85
	F1 score	0.85	0.78	0.87
<b>Robustness Testing</b>	Success rate	-	0.75	0.80

### 6.1. Model Evaluation

The network loss of *HaCk* is evaluated through the MSE and MAE metrics. Both MSE and MAE are standard loss computation metrics.

MSE is a commonly used metric for measuring the loss of a predictive model. It quantifies the average of the squared differences between the predicted values and the actual observed values. MSE can be calculated based on the following steps:

- For each data point in the dataset used, subtract the actual observed value (the ground truth) from the predicted value obtained from *HaCk*.
- Square the result of each subtraction to ensure that all differences are positive and emphasize larger errors.
- Calculate the average (mean) of all these squared differences.

On the other hand, MAE is another metric used to evaluate the loss of a predictive model. It calculates the average of the absolute differences between the predicted values and the actual observed values. MAE can be calculated based on the following steps:

- For each data point in the dataset used, subtract the actual observed value from the predicted value, taking the absolute value of the difference.
- Calculate the average (mean) of all these absolute differences.

MSE can be computed using (10).

$$MSE = 1/S * \left( \sum_{i=1}^S (y_i - \hat{y}_i)^2 \right) \quad (10)$$

MAE can be computed using (11).

$$MAE = 1/S * \left( \sum_{i=1}^S |y_i - \hat{y}_i| \right) \quad (11)$$

Here,  $y_i$  denotes the observed value,  $\hat{y}_i$  represents the predicted value, and  $S$  represents the total number of samples used during testing.

Therefore, MSE and MAE give more weight to more significant errors due to the squaring operation. As a result, these values penalize outliers more heavily and can be measured in squared units of the target variable, which may not always be interpretable. Lower MSE and MAE values indicate a better model performance, meaning that, close to 0, *HaCk*'s predictions are identical to the actual values.

Figures 6 and 7 depict the loss (MSE and MAE) of the proposed *HaCk* system for both the training and testing datasets. Observing these figures shows that the training and testing losses rapidly decrease during the initial epochs. Subsequently, after around 50 epochs, both plots stabilize, indicating that the training and testing losses converge to similar values. This implies a consistent trend between the training and testing datasets, which is desirable for sequential time series data. Furthermore, the loss remains approximately steady across 500 epochs.

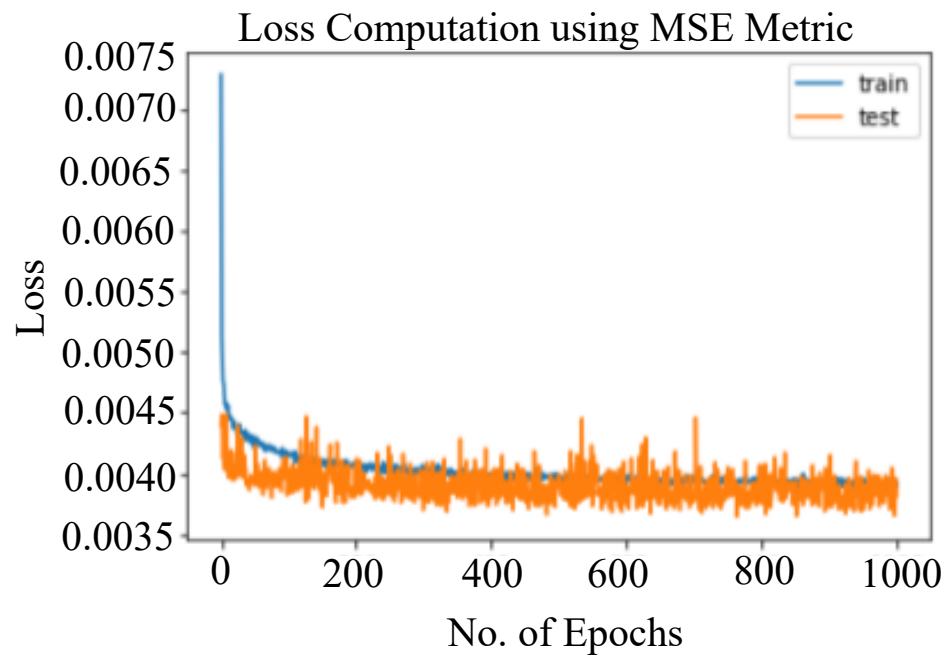


Figure 6. MSE of HaCk.

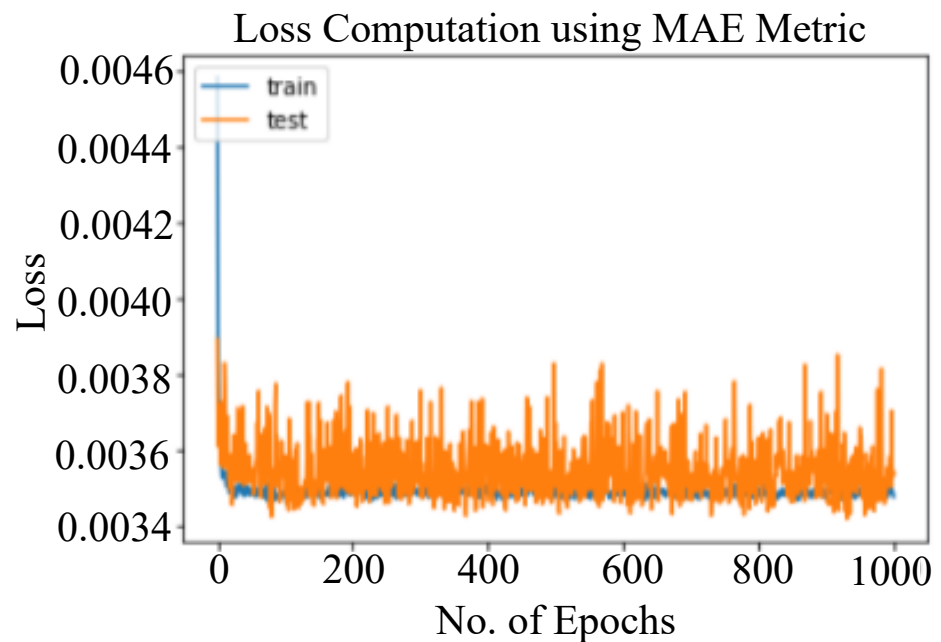


Figure 7. MAE of HaCk.

## 6.2. Hand GR

We used a separate testing dataset of 550 real images of various hand gestures, each labeled accordingly, to analyze the results, as illustrated in Figure 8. When both trained CNN models—one using real data samples and the other using generated data samples—were employed to predict the class of these testing images, the following statistics were obtained:

- The CNN model, trained using real data samples, achieved an accuracy of 93% in predicting the class of the testing hand gestures, as described in Figure 9.
- *HaCk*, trained using generated data samples, achieved an accuracy of 68% in predicting the class of the testing hand gestures, as depicted in Figure 10.

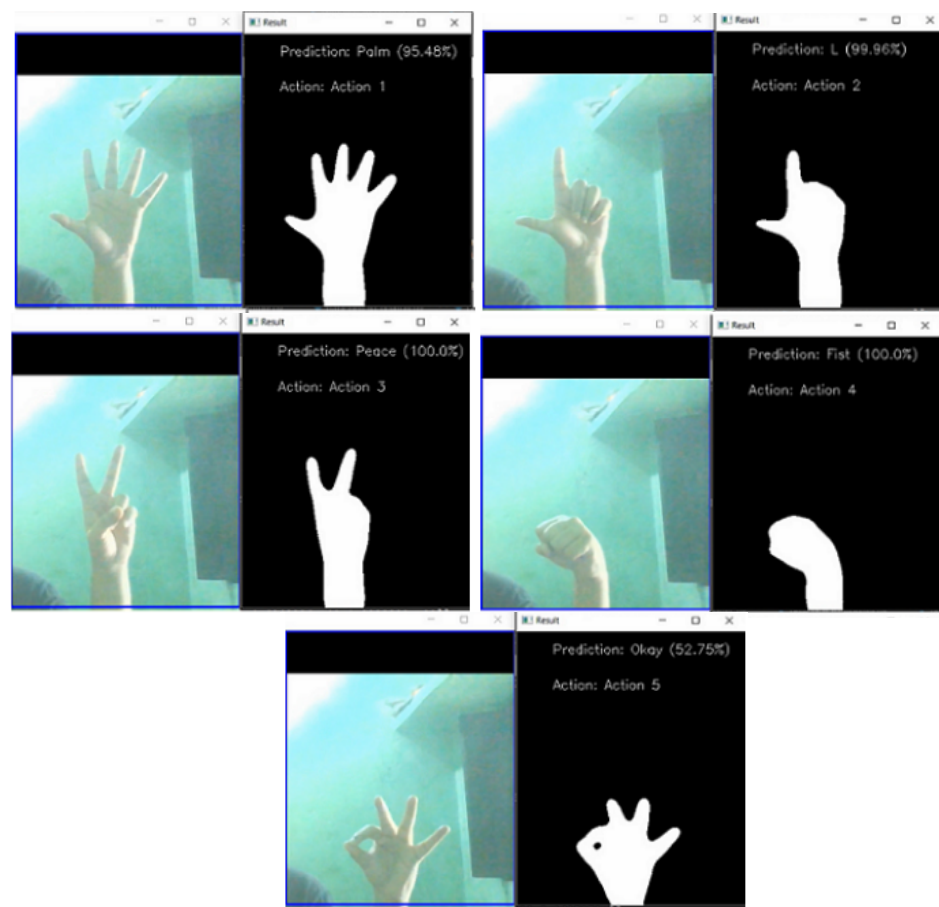


Figure 8. Hand GR is conducted by the continuous prediction of hand gestures during the testing phase in real time.



Figure 9. Result analysis using a heat map of CNN-based real samples.

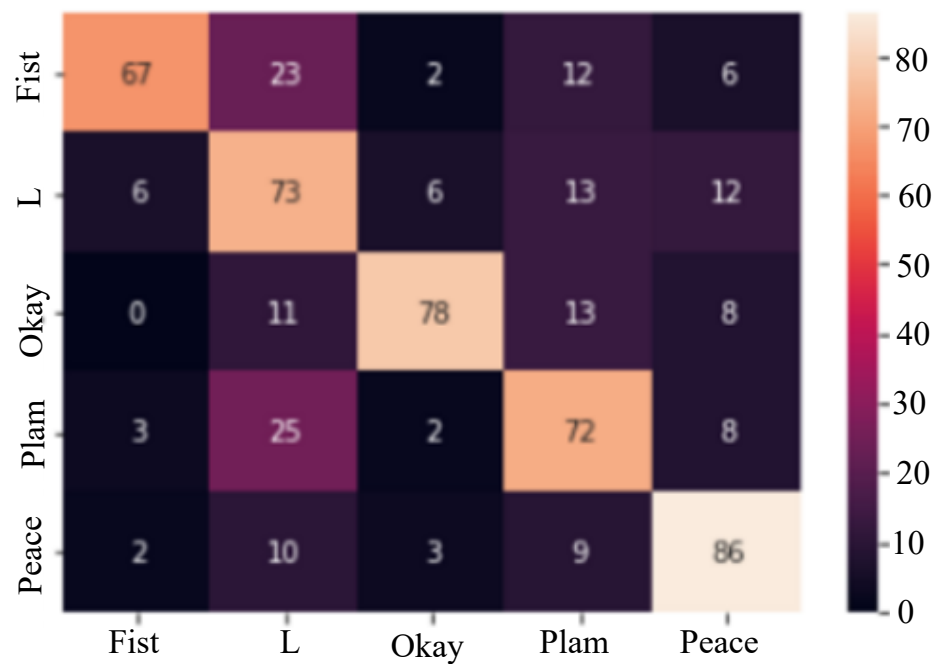


Figure 10. Result analysis using a heat map of CNN-based generated samples.

Upon comparing the accuracy using Figures 9 and 10, we observed a remarkable resemblance between the generated data samples and the properties of the real data samples.

### 6.3. Comparison of Hand GR with Other Existing Models

We evaluate the performance of the proposed *HaCk* by comparing it with a previously reported technique, employing both Leave-One-Out Cross-Validation (LOO CV) and Holdout Cross-Validation (Holdout CV) tests. The results are presented in Tables 3 and 4, respectively.

As shown in Table 3, the proposed *HaCk* exhibits a mean accuracy that is 17.03%, 20.27%, 15.76%, 13.76%, 10.16%, 5.90%, and 15.90% higher than the CNN, FTCNN, CDCGAN, GestureGAN, GGAN, MHG-CAN, and ASL models, respectively. These findings highlight the superior recognition accuracy of our proposed approach. Similarly, Table 4 also demonstrates *HaCk*'s superior mean accuracy performance compared to the earlier-reported technique during the Holdout CV test. From Table 4, we can observe that the proposed *HaCk* exhibits a mean accuracy that is 56.87%, 15.91%, 13.97%, 24.81%, 23.52%, 17.72%, 15.72%, 12.12%, 7.94%, and 17.94% higher than the HU, ZM, GB, GB-ZM, GB-HU, CDCGAN, GestureGAN, GGAN, MHG-CAN, and ASL models, respectively.

Table 3. Compared models using LOO CV test.

Models	Accuracy (%)
CNN [31]	78.93%
FTCNN [32]	75.69%
CDCGAN [8]	80.20%
GestureGAN [9]	82.20%
GGAN [10]	85.80%
MHG-CAN [11]	89.98%
ASL [12]	90.05%
<b><i>HaCk</i> (proposed)</b>	<b>95.96%</b>

**Table 4.** Compared Models using Holdout CV test.

Models	Accuracy (%)
HU [33]	41.05%
ZM [33]	82.01%
GB [33]	83.95%
GB-ZM [33]	73.11%
GB-HU [33]	74.40%
CDCGAN [8]	80.20%
GestureGAN [9]	82.20%
GGAN [10]	85.80%
MHG-CAN [11]	89.98%
ASL [12]	90.10%
<b>HaCk (proposed)</b>	<b>97.92%</b>

#### 6.4. Computational Complexity

The computational complexity of our proposed *HaCk* system concerns the amount of computational resources, such as time and memory, required to perform various operations within the system. Several factors influence this complexity, and these can vary depending on the specific architecture and size of the CNN and GAN models, the dataset size, and other computational considerations. Below are some aspects that contribute to the model's computational complexity:

1. **Model Architecture:** The number of layers, neurons, and connections in the CNN classifier, GAN generator, and discriminator networks significantly determine computational complexity. Deeper and larger networks generally require more computational resources.
2. **Training:** Training deep neural networks like CNNs and GANs involves forward and backward passes through the network, including matrix multiplications and gradient calculations. The number of training iterations and the mini batch size can affect the training time and memory usage.
3. **Data Augmentation:** If extensive data augmentation techniques are used to expand the dataset, this can increase computational complexity during training and inference, as additional transformations must be applied to the data.
4. **GAN Training:** Training a GAN involves multiple iterations of generator and discriminator updates, which can be computationally intensive, especially for more extensive networks.
5. **Real-Time Inference:** When using the system for real-time hand gesture recognition with a webcam feed, the computational complexity depends on the inference speed of the CNN model and the rate at which new frames are processed.
6. **Hyperparameter Search:** If hyperparameter tuning is performed, it adds an extra computational burden as the system evaluates multiple combinations of hyperparameters.
7. **Memory Usage:** The size of the neural network parameters and the dataset size can impact memory requirements. Large models and datasets may require more memory, potentially leading to memory limitations on some hardware.
8. **Optimization Techniques:** Optimization techniques, such as weight quantization, can influence the computational complexity and efficiency of the system.
9. **Hardware Resources:** The computational complexity depends on the available hardware resources. High-performance GPUs or TPUs can handle more complex models and larger datasets more efficiently than CPUs.

Hence, the computational complexity of *HaCk* is a multifaceted consideration influenced by factors such as model architecture, data size, training and inference procedures, and hardware resources.

### 6.5. Methodological Approaches

To enable a methodological comparison of the HaCk system, we use two models: the (i) Multisensor Guided Hand Gesture Recognition System (MGHGRS) [34], and the (ii) Wearable Respiratory and Activity Monitoring (WRAM) System [35].

Table 5 describes the methodological comparison among all state-of-the-art models.

From Table 5, we observed the need for the development of HaCk based on various factors, each of which plays a crucial role in meeting specific requirements, which can be described as follows:

- *Addressing Visual Impairment Challenges:* HaCk is explicitly designed to address visual impairment challenges, making it a suitable choice for applications where accessibility for individuals with visual impairments is a priority.
- *Non-Contact Interaction for Accessibility:* The HaCk system's emphasis on non-contact interaction aligns with scenarios where hands-free or touchless interaction is essential for user accessibility.
- *Diverse Set of Hand Gestures:* HaCk's ability to generate a diverse set of hand gestures suggests its versatility in accommodating a wide range of gestures. This could be advantageous in applications requiring a comprehensive set of recognizable hand movements.
- *Two-Step Recognition Approach:* The integration of CNN and GAN in a two-step recognition approach enhances the model's adaptability and performance.
- *Rigorous Evaluation through Cross-Validation:* HaCk undergoes rigorous evaluation through cross-validation tests, indicating a focus on generalization, robustness, and practical suitability. This makes it a favorable choice for applications where reliability and performance validation are critical.
- *Versatility in Cross-Validation Tests:* HaCk's versatility, as demonstrated by its success in both Leave-One-Out and Holdout Cross-Validation tests, suggests its potential applicability across different datasets and scenarios.
- *Ethical Considerations:* Ethical considerations related to the generation of synthetic data are acknowledged in relation to HaCk.

Therefore, the decision to develop HaCk was driven by specific application requirements such as the need for accessibility, non-contact interaction, diverse gesture recognition, modular recognition approaches, rigorous evaluation, versatility, and real-time responsiveness.

### 6.6. Real-World Application

Sign language interpretation using HaCk involves recognizing and translating sign language gestures into corresponding textual or spoken representations, which can be described as follows:

1. *Data Collection:*
  - (a) Gather a diverse dataset of sign language gestures representing various signs and expressions.
  - (b) Annotate the dataset with corresponding labels or translations.
2. *Model Training:*
  - (a) Train HaCk based on a CNN and GAN architecture.
  - (b) Train the CNN to recognize hand gestures from visual inputs. At the same time, employ the GAN for data augmentation or generating additional synthetic data to improve HaCk system's performance.
3. *Gesture Recognition:*
  - (a) Deploy the trained HaCk to recognize hand gestures in real-time or from recorded video inputs.
  - (b) Process the visual information using the CNN component so that the corresponding sign can be predicted.



4. *Translation:*
  - (a) Translate the recognized sign language gestures into textual or spoken form.
  - (b) Associate each recognized gesture with its corresponding meaning.
5. *Output Display:*
  - (a) Display the translated output, making it accessible to users who may not be proficient in sign language.
  - (b) Options include displaying the translation as text or using a speech synthesis system for spoken outputs.
6. *User Interaction:*
  - (a) Design the system to facilitate user interaction, allowing individuals to input sign language gestures through a camera or other input devices.
  - (b) Consider providing feedback to users, such as highlighting the recognized gestures on display.
7. *Adaptability and Robustness:*
  - (a) Ensure *HaCk*'s adaptability to different signing styles and variations in hand movements.
  - (b) Implement robustness features for lighting conditions, background noise, and occlusion variations.
8. *User Feedback and Iterative Refinement:*
  - (a) Encourage user feedback to improve *HaCk*'s accuracy and user experience.
  - (b) Consider implementing an iterative refinement process, updating *HaCk* based on observed limitations and user suggestions.
9. *Accessibility Features:*
  - (a) Implement accessibility features within *HaCk*, like options for adjusting the model's sensitivity, adapting to different signing speeds, and accommodating users with varying proficiency levels in sign language.
10. *Integration with Assistive Technologies:*
  - (a) Explore integration with assistive technologies to enhance accessibility for individuals with differing abilities.
  - (b) Consider compatibility with smart glasses and wearable technology for seamless interaction.
11. *Ethical Considerations:*
  - (a) Address all ethical considerations related to user privacy, informed consent, and the responsible use of AI technology in sign language interpretation.
12. *Continuous Improvement:*
  - (a) Implement feedback mechanisms for the continuous improvement of the *HaCk* system based on advancements in AI and user feedback.

Sign language interpretation using *HaCk* involves a comprehensive pipeline that combines computer vision, machine learning, and natural language processing to bridge communication gaps and enhance accessibility for individuals who use sign language.

**Table 5.** Methodological comparison among all state-of-the-art models.

Models	Network Architecture	Training Strategy	Data Preprocessing	Advantages	Limitations
MGHGRS [34]	A multi-sensor data fusion model comprising a multilayer RNN. The multilayer RNN consists of an LSTM module and a dropout layer.	Multiple label classification.	Signal processing and modeling approaches.	(i) Improved accuracy, (ii) real-time responsiveness, (iii) effective management of depth data challenges, (iv) occlusion handling, (v) adaptability to dynamic conditions, (vi) versatility in collaboration tasks, and (vii) a comparative edge over traditional ML algorithms.	(i) Hardware complexity, (ii) cost implications, (iii) adaptability to novel gestures, (iv) sensitivity to environmental changes, (v) dependency on real-time processing, and (vi) interference in dynamic environments.
WRAM system [35]	A fusion architecture incorporating multiple modalities is formed by amalgamating a hybrid hierarchical classification (HHC) algorithm that combines both deep learning and threshold-based methods.	Multiple users breathing in real-time.	Signal denoising and DL.	The WRAM system [35] offers significant advantages, ranging from accurate activity recognition and in-depth breathing pattern analysis to real-time monitoring capabilities, demonstrating its potential impact on healthcare and precision medicine.	(i) Limited generalization to diverse populations, (ii) dependency on participant compliance, (iii) sensitivity to device placement, (iv) challenges in handling unseen activities, (v) potential interference in real-world environments, (vi) data privacy and ethical considerations, and (vii) limited exploration of health conditions.
HaCk (proposed)	Integration of CNN and GAN.	Multiple label classification based on DL.	Data normalization and scaling.	(i) Addressing visual impairment challenges, (ii) non-contact interaction for accessibility, (iii) diverse set of hand gestures, (iv) two-step recognition approach, (v) evaluation through rigorous cross-validation, (vi) versatility in cross-validation tests, and (vii) contributing to accessibility technology and potential for real-time applications.	(i) Limited representativeness of visual impairment, (ii) dependency on non-contact interaction, (iii) synthetic gesture generalization challenges, and (iv) potential ethical considerations in synthetic data generation.

### 6.7. Discussion

Certainly, explaining why the specific technique of background subtraction was chosen for a CNN-based classifier for Hand Gesture Recognition (Hand GR) using a GAN-based data generation model and how it compares to other methods is crucial to provide insights into the design choices of the system. Below is an elaboration on this topic:

#### 6.7.1. Choice of Background Subtraction

Background subtraction is a common pre-processing technique in computer vision and gesture recognition applications, including Hand GR using CNNs and GANs. The rationale behind choosing background subtraction in this context can be explained as follows:

1. **Noise Reduction:** Hand gesture recognition relies heavily on accurately detecting the hand and its movements. A cluttered or dynamic background can introduce noise into the input data, making it challenging for the CNN-based classifier to focus on the hand itself. By subtracting the background, we isolate the region of interest (ROI), which is the hand, and reduce noise from the surrounding environment.
2. **Feature Extraction:** Background subtraction simplifies feature extraction. The CNN can then focus on extracting meaningful features from the hand region without being distracted by irrelevant information in the background. This can lead to more robust and accurate recognition of hand gestures.
3. **Consistency:** Lighting conditions and backgrounds can vary significantly in real-world scenarios. Using background subtraction helps achieve consistency in the input data, making it easier for the CNN model to generalize across different environments.
4. **Real-Time Processing:** For applications that require real-time processing, such as gesture recognition for interactive systems, background subtraction can be computationally efficient compared to more complex methods like background modeling and subtraction. This allows for faster inference times, which is critical for interactive applications.

Therefore, the choice of background subtraction in the context of the CNN-based Classifier for Hand GR using a GAN-based data generation model is motivated by its simplicity, real-time processing capabilities, noise reduction benefits, and suitability for various real-world scenarios. However, the selection of the technique should be based on the specific requirements and constraints of the application, considering factors like computational resources, environmental conditions, and the desired level of accuracy.

#### 6.7.2. Scalability, User Adaptability, and Computational Efficiency of the *HaCk* System

Let's consider a real-life example of *HaCk* for sign language interpretation. This example will help illustrate how *HaCk* addresses scalability, user adaptability, and computational efficiency.

- *Scalability:*
  - *Scenario:* The system is deployed in a public space, such as a transportation hub, where a diverse group of users may need to utilize a sign language interpretation service.
  - *Scalability Measures:*
    - \* The system is designed to handle a growing number of users, ensuring that the model's performance remains consistent even during peak usage times.
    - \* Training the model on a large and diverse sign language dataset allows it to recognize a broad range of gestures, accommodating various signing styles.
- *User Adaptability:*
  - *Scenario:* The system caters to users with different signing proficiency levels and physical characteristics.
  - *User Adaptability Measures:*

- \* *HaCk* includes adaptive features that adjust its recognition sensitivity based on individual user preferences, allowing both beginners and advanced signers to use the system effectively.
- \* Robustness in recognizing gestures from users with different hand shapes or sizes is ensured through extensive training on diverse datasets.
- *Computational Efficiency:*
  - *Scenario:* The system is integrated into a portable device that enables real-time sign language interpretation on the go.
  - *Computational Efficiency Measures:*
    - \* *HaCk* is optimized for efficient inference on the device, ensuring minimal latency in recognizing and interpreting gestures.
    - \* Quantization techniques are applied to reduce *HaCk*'s size, making it suitable for deployment on resource-constrained devices without compromising performance.
- *Operational Environments:*
  - *Scenario:* The system is used in various operational environments, including well-lit indoor spaces and outdoor environments with dynamic lighting conditions.
  - *Operational Environment Measures:*
    - \* *HaCk* is trained to be robust to different lighting conditions, and it undergoes thorough testing in diverse environments to ensure accurate gesture recognition.
    - \* Adaptive algorithms dynamically adjust recognition parameters based on environmental factors, maintaining their performance in challenging conditions.

Therefore, in this real-life example, *HaCk* is not only effective in controlled settings but also scalable, adaptable to diverse users, and computationally efficient in addressing the challenges of real-world sign language interpretation.

### 6.7.3. Ethical Considerations

Ethical considerations play a pivotal role in the development and deployment of AI-powered gesture recognition systems, especially when aiming to ensure inclusivity and accessibility for individuals with differing abilities. Below are the key ethical considerations in this context:

- *Inclusive Dataset Representation:*
  - *Ethical Concern:* Bias in training data that does not adequately represent individuals with differing abilities can lead to discriminatory outcomes.
  - *Mitigation:* Ensure diverse representation in training datasets, encompassing individuals with varying abilities, ages, genders, and cultural backgrounds.
- *Privacy and Informed Consent:*
  - *Ethical Concern:* Gesture recognition systems may capture sensitive information; obtaining informed consent is crucial to respect individuals' privacy.
  - *Mitigation:* Clearly communicate data collection purposes, offer opt-in/opt-out choices, and provide transparent explanations about how data will be used, stored, and shared.
- *Accessibility in Design:*
  - *Ethical Concern:* Excluding individuals with certain disabilities due to design limitations hinders inclusivity.
  - *Mitigation:* Prioritize universal design principles, ensuring that the gesture recognition system is accessible to individuals with diverse abilities and adaptable to assistive technologies.

- *Avoiding Reinforcement of Stereotypes:*
  - *Ethical Concern:* Biased algorithms can perpetuate stereotypes, affecting the fair treatment of individuals with varying abilities.
  - *Mitigation:* Regularly audit and test algorithms for bias, and implement strategies to minimize and rectify biases that may arise during training.
- *Interpretable and Explainable AI:*
  - *Ethical Concern:* Lack of interpretability in AI models may result in decisions that are difficult to explain or understand.
  - *Mitigation:* Prioritize the development of interpretable AI models to enhance transparency, accountability, and user trust.
- *Ensuring Fair Access:*
  - *Ethical Concern:* Limited accessibility to gesture recognition technology may widen existing disparities for individuals with differing abilities.
  - *Mitigation:* Strive to make the technology affordable, considerate of resource constraints, and compatible with a variety of devices to ensure widespread accessibility.
- *Continuous User Feedback and Adaptability:*
  - *Ethical Concern:* A lack of continuous feedback mechanisms may result in systems that do not adequately adapt to the needs of individuals with varying abilities.
  - *Mitigation:* Establish channels for ongoing user feedback, incorporate user perspectives in system refinement, and ensure adaptability to evolving accessibility requirements.
- *Guarding Against Unintended Consequences:*
  - *Ethical Concern:* Unintended consequences, such as misinterpretation of gestures or inadvertent exclusion, may occur.
  - *Mitigation:* Regularly assess the real-world impact of the gesture recognition system, conduct usability testing with individuals with differing abilities, and promptly address any issues identified.
- *Education and Awareness:*
  - *Ethical Concern:* Lack of awareness about the capabilities and limitations of gesture recognition systems can lead to unrealistic expectations.
  - *Mitigation:* Provide transparent information to users about the system's capabilities, potential errors, and ongoing efforts to enhance inclusivity.

By addressing these ethical considerations, we can contribute to the responsible and inclusive deployment of AI-powered gesture recognition technology, ensuring that it meets the needs of all users, including those with diverse abilities. Ongoing collaboration with relevant communities and stakeholders is crucial to the creation of systems that are both technologically advanced and ethically sound.

## 7. Conclusions and Future Work

In this study, we utilized a conditional Generative Adversarial Network (GAN) as a central component of our methodology. The primary objective of employing this GAN was to generate synthetic images that accurately represent various hand gestures. These synthetic images were subsequently subjected to a rigorous evaluation by comparing them with authentic hand gesture images. To achieve this comparison, we trained distinct Convolutional Neural Network (CNN) models using two separate datasets: the generated images and the genuine, real-world hand gesture images.

One noteworthy observation is that the generated images exhibit properties similar to the actual hand gesture images. This is a positive indication of the GAN's ability to produce realistic and contextually relevant data. However, it is essential to note that, in the spirit of continuous improvement and refinement, there remains room for enhancements and optimizations in future iterations of this research. While our current results are promising,

ongoing work and future investigations will further enhance the fidelity and accuracy of the generated images and the overall performance of the CNN models. This commitment to refinement underscores our dedication to advancing state-of-the-art methods in hand gesture recognition and underscores the dynamic and evolving nature of this field of research.

The study mentions that the generated images closely resemble real hand gesture images. However, the extent of the data variability and complexity in real-world scenarios has yet to be fully captured. The dataset may need more diversity in hand shapes, sizes, and environmental conditions, potentially limiting the model's robustness in practical applications where these factors vary widely.

**Author Contributions:** K.C.: manuscript writing, software development, experiment design, and conducting experiments. M.R., N.S., M.P., B.K.K. and A.B.: initial idea, manuscript writing, manuscript revision, experiment design, and data analysis. S.M.: resource management, data analysis, and manuscript revision. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data are contained within the article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

- Li, W.; Shi, P.; Yu, H. Gesture recognition using surface electromyography and deep learning for prostheses hand: state-of-the-art, challenges, and future. *Front. Neurosci.* **2021**, *15*, 621885. [[CrossRef](#)]
- Wang, J.; Chen, Y.; Hao, S.; Peng, X.; Hu, L. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognit. Lett.* **2019**, *119*, 3–11. [[CrossRef](#)]
- Dahou, A.; Al-qaness, M.A.; Abd Elaziz, M.; Helmi, A.M. MLCNNwav: Multi-level Convolutional Neural Network with Wavelet Transformations for Sensor-based Human Activity Recognition. *IEEE Internet Things J.* **2023**, *11*, 820–828. [[CrossRef](#)]
- Verma, K.K.; Singh, B.M. Deep multi-model fusion for human activity recognition using evolutionary algorithms. *Int. J. Interact. Multimed. Artif. Intell.* **2021**, *7*, 44–58. [[CrossRef](#)]
- Mekruksavanich, S.; Jitpattanakul, A. Deep convolutional neural network with rnns for complex activity recognition using wrist-worn wearable sensor data. *Electronics* **2021**, *10*, 1685. [[CrossRef](#)]
- Alessandrini, M.; Biagetti, G.; Crippa, P.; Falaschetti, L.; Turchetti, C. Recurrent neural network for human activity recognition in embedded systems using ppg and accelerometer data. *Electronics* **2021**, *10*, 1715. [[CrossRef](#)]
- Ordóñez, F.J.; Roggen, D. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors* **2016**, *16*, 115. [[CrossRef](#)]
- Fang, W.; Ding, Y.; Zhang, F.; Sheng, J. Gesture recognition based on CNN and DCGAN for calculation and text output. *IEEE Access* **2019**, *7*, 28230–28237. [[CrossRef](#)]
- Tang, H.; Wang, W.; Xu, D.; Yan, Y.; Sebe, N. Gesturegan for hand gesture-to-gesture translation in the wild. In Proceedings of the 26th ACM International Conference on Multimedia, Seoul, Republic of Korea, 22–26 October 2018; pp. 774–782.
- Zhu, W.; Yang, Y.; Chen, L.; Xu, J.; Zhang, C.; Guo, H. Application of Generative Adversarial Networks in Gesture Recognition. In Proceedings of the 2022 WRC Symposium on Advanced Robotics and Automation (WRC SARA), Beijing, China, 20 August 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6.
- Garg, M.; Ghosh, D.; Pradhan, P.M. Generating multiview hand gestures with conditional adversarial network. In Proceedings of the 2021 IEEE 18th India Council International Conference (INDICON), Guwahati, India, 19–21 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
- Barbhuiya, A.A.; Karsh, R.K.; Jain, R. ASL hand gesture classification and localization using deep ensemble neural network. *Arab. J. Sci. Eng.* **2023**, *48*, 6689–6702. [[CrossRef](#)]
- Javed, A.R.; Faheem, R.; Asim, M.; Baker, T.; Beg, M.O. A smartphone sensors-based personalized human activity recognition system for sustainable smart cities. *Sustain. Cities Soc.* **2021**, *71*, 102970. [[CrossRef](#)]
- Fallahzadeh, R.; Ghasemzadeh, H. Personalization without user interruption: Boosting activity recognition in new subjects using unlabeled data. In Proceedings of the 8th International Conference on Cyber-Physical Systems, Pittsburgh, PA, USA, 18–20 April 2017; pp. 293–302.
- Siirtola, P.; Röning, J. Context-aware incremental learning-based method for personalized human activity recognition. *J. Ambient. Intell. Humaniz. Comput.* **2021**, 10499–10513. [[CrossRef](#)]



16. Boroujeni, S.P.H.; Razi, A. IC-GAN: An Improved Conditional Generative Adversarial Network for RGB-to-IR image translation with applications to forest fire monitoring. *Expert Syst. Appl.* **2024**, *238*, 121962. [[CrossRef](#)]
17. Wang, G.; Shi, H.; Chen, Y.; Wu, B. Unsupervised image-to-image translation via long-short cycle-consistent adversarial networks. *Appl. Intell.* **2023**, *53*, 17243–17259. [[CrossRef](#)]
18. Almahairi, A.; Rajeshwar, S.; Sordoni, A.; Bachman, P.; Courville, A. Augmented cycleGAN: Learning many-to-many mappings from unpaired data. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 195–204.
19. Karras, T.; Laine, S.; Aila, T. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4401–4410.
20. Wu, B.; Ding, Y.; Dong, Q. Fast continuous structural similarity patch based arbitrary style transfer. *Appl. Sci.* **2019**, *9*, 3304. [[CrossRef](#)]
21. Ruta, D.S. Learned Representations of Artistic Style for Image Retrieval, Description, and Stylization. Ph.D. Thesis, University of Surrey, Guildford, UK, 2023.
22. Gupta, V.; Sadana, R.; Moudgil, S. Image style transfer using convolutional neural networks based on transfer learning. *Int. J. Comput. Syst. Eng.* **2019**, *5*, 53–60. [[CrossRef](#)]
23. Gu, S.; Chen, C.; Liao, J.; Yuan, L. Arbitrary style transfer with deep feature reshuffle. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8222–8231.
24. Jung, D.; Yang, S.; Choi, J.; Kim, C. Arbitrary style transfer using graph instance normalization. In Proceedings of the 2020 IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, 25–28 October 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1596–1600.
25. Johnson, J.; Alahi, A.; Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In Proceedings of the Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part II 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 694–711.
26. Wang, Z.; Zhao, L.; Chen, H.; Qiu, L.; Mo, Q.; Lin, S.; Xing, W.; Lu, D. Diversified arbitrary style transfer via deep feature perturbation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 7789–7798.
27. Sheng, L.; Lin, Z.; Shao, J.; Wang, X. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8242–8250.
28. Suzuki, N.; Watanabe, Y.; Nakazawa, A. Gan-based style transformation to improve gesture-recognition accuracy. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*; Association for Computing Machinery: New York, NY, USA, 2020; Volume 4, pp. 1–20.
29. Holden, D.; Saito, J.; Komura, T. A deep learning framework for character motion synthesis and editing. *Acm Trans. Graph. (Tog)* **2016**, *35*, 1–11. [[CrossRef](#)]
30. Banerjee, T.; Srikanth, K.P.; Reddy, S.A.; Biradar, K.S.; Koripally, R.R.; Varshith, G. Hand Sign Recognition using Infrared Imagery Provided by Leap Motion Controller and Computer Vision. In Proceedings of the 2021 International Conference on Innovative Practices in Technology and Management (ICIPTM), Noida, India, 17–19 February 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 20–25.
31. Chevtchenko, S.F.; Vale, R.F.; Macario, V.; Cordeiro, F.R. A convolutional neural network with feature fusion for real-time hand posture recognition. *Appl. Soft Comput.* **2018**, *73*, 748–766. [[CrossRef](#)]
32. Sahoo, J.P.; Prakash, A.J.; Pławiak, P.; Samantray, S. Real-time hand gesture recognition using fine-tuned convolutional neural network. *Sensors* **2022**, *22*, 706. [[CrossRef](#)] [[PubMed](#)]
33. Sahoo, J.P.; Sahoo, S.P.; Ari, S.; Patra, S.K. RBI-2RCNN: Residual block intensity feature using a two-stage residual convolutional neural network for static hand gesture recognition. *Signal Image Video Process.* **2022**, *16*, 2019–2027. [[CrossRef](#)]
34. Qi, W.; Ovrur, S.E.; Li, Z.; Marzullo, A.; Song, R. Multi-Sensor Guided Hand Gesture Recognition for a Teleoperated Robot Using a Recurrent Neural Network. *IEEE Robot. Autom. Lett.* **2021**, *6*, 6039–6045. [[CrossRef](#)]
35. Qi, W.; Aliverti, A. A Multimodal Wearable System for Continuous and Real-Time Breathing Pattern Monitoring during Daily Activity. *IEEE J. Biomed. Health Inform.* **2020**, *24*, 2199–2207. [[CrossRef](#)] [[PubMed](#)]

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