

Article

SDViT: Stacking of Distilled Vision Transformers for Hand Gesture Recognition

Chun Keat Tan ¹, Kian Ming Lim ^{1,*} , Chin Poo Lee ¹ , Roy Kwang Yang Chang ¹  and Ali Alqahtani ^{2,3} 

¹ Faculty of Information Science and Technology, Multimedia University, Jalan Ayer Keroh Lama, Malacca 75450, Malaysia; 1181101099@student.mmu.edu.my (C.K.T.); cplee@mmu.edu.my (C.P.L.); kychang@mmu.edu.my (R.K.Y.C.)

² Department of Computer Science, King Khalid University, Abha 61421, Saudi Arabia; amosfer@kku.edu.sa

³ Center for Artificial Intelligence (CAI), King Khalid University, Abha 61421, Saudi Arabia

* Correspondence: kmlim@mmu.edu.my

Abstract: Hand gesture recognition (HGR) is a rapidly evolving field with the potential to revolutionize human–computer interactions by enabling machines to interpret and understand human gestures for intuitive communication and control. However, HGR faces challenges such as the high similarity of hand gestures, real-time performance, and model generalization. To address these challenges, this paper proposes the stacking of distilled vision transformers, referred to as SDViT, for hand gesture recognition. An initially pretrained vision transformer (ViT) featuring a self-attention mechanism is introduced to effectively capture intricate connections among image patches, thereby enhancing its capability to handle the challenge of high similarity between hand gestures. Subsequently, knowledge distillation is proposed to compress the ViT model and improve model generalization. Multiple distilled ViTs are then stacked to achieve higher predictive performance and reduce overfitting. The proposed SDViT model achieves a promising performance on three benchmark datasets for hand gesture recognition: the American Sign Language (ASL) dataset, the ASL with digits dataset, and the National University of Singapore (NUS) hand gesture dataset. The accuracies achieved on these datasets are 100.00%, 99.60%, and 100.00%, respectively.

Keywords: hand gesture recognition; sign language recognition; vision transformer; knowledge distillation; stacking



Citation: Tan, C.K.; Lim, K.M.; Lee, C.P.; Chang, R.K.Y.; Alqahtani, A. SDViT: Stacking of Distilled Vision Transformers for Hand Gesture Recognition. *Appl. Sci.* **2023**, *13*, 12204. <https://doi.org/10.3390/app132212204>

Academic Editor: Andrea Prati

Received: 7 October 2023

Revised: 24 October 2023

Accepted: 25 October 2023

Published: 10 November 2023



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

1. Introduction

Hand gestures are powerful tools for communication, and they are often used in conjunction with speech to enhance the expression of intentions and convey information. They also serve as a standalone method of communication, particularly in sign language, especially for individuals who are mute or deaf. This research paper primarily focuses on American Sign Language (ASL), which is recognized as the most commonly utilized sign language. Despite the general awareness of sign language, few people understand it and can effectively communicate using it. To address this issue, a hand gesture recognition (HGR) system is needed.

HGR models hold significant promise for a wide range of applications. In interactive computing interfaces, HGR can enable intuitive and hands-free control, finding applications in gaming, virtual reality (VR), and augmented reality (AR) systems. In the healthcare domain, HGR can be utilized for developing assistive technologies, allowing individuals with mobility impairments to control devices through hand gestures. HGR also plays a crucial role in human–robot interactions, where robots can interpret and respond to human gestures, enhancing collaboration and communication in various settings. Moreover, HGR has potential applications in sign language interpretation, contributing to improved communication accessibility for individuals with hearing impairments. The technology can also be applied in security and surveillance, allowing for gesture-based commands for

monitoring and controlling security systems. As HGR technology advances, its applications are expected to expand, offering innovative solutions in diverse fields.

HGR can be broadly categorized into two types: static HGR [1–8] and dynamic HGR [9–13]. The former involves classifying hand gestures from images, while the latter involves classifying hand gestures from videos. Dynamic hand gesture recognition is more complex than static HGR due to the numerous possibilities and combinations of hand gestures that can occur in a video. Static HGR can be further divided into two main types. The first type relies on vision, where images are captured using a camera. The second type involves wearable devices such as data gloves, which can capture hand gesture information more accurately but are inconvenient and expensive. Identifying hand gestures in both scenarios poses numerous obstacles, such as fluctuations in lighting circumstances, intricate backgrounds, resemblances between different hand gestures, and differences in hand dimensions and skin complexion. This research will specifically focus on vision-based static HGR.

Vision-based static HGR can be categorized into two main approaches: hand-crafted and deep learning. The hand-crafted approach involves applying various feature extraction techniques, such as discrete wavelet transform (DWT) [14–17], histogram of oriented gradient (HOG) [18–20], and scale invariant feature transform (SIFT) [21], to extract features from the hand gestures. These extracted characteristics are subsequently input into a classification algorithm for the purpose of gesture categorization. Alternatively, the deep learning approach utilizes deep learning networks, such as convolutional neural networks (CNNs) [1–6,22–29], artificial neural networks (ANNs) [30], and autoencoders [31,32], to automatically extract features from the hand gestures. Deep learning networks showcase adaptability to various challenges in static hand gesture recognition, learning from extensive datasets and accommodating diverse environmental factors such as lighting conditions, complex backgrounds, and variations in hand size and skin tone. However, existing methods encounter several limitations that present research gaps to be addressed. One common challenge is the limited capacity of current models to effectively capture global dependencies and intricate relationships among different parts of static hand gestures. Additionally, variations in hand pose and lighting conditions pose difficulties for existing models, impacting their robustness in real-world scenarios. The sensitivity of models to background complexity, especially when distinguishing hand gestures from intricate or cluttered backgrounds, is another notable limitation. Furthermore, the generalization capabilities of existing models across different hand sizes and skin tones remain as an ongoing concern. Discriminating between visually similar gestures, where subtle differences may be challenging to discern, poses a significant obstacle. Moreover, some existing models, particularly those based on complex architectures, may suffer from computational inefficiency and large model sizes, limiting their practical deployment.

In order to address the research gaps, the proposed SDViT model incorporates the vision transformer (ViT) model along with a self-attention mechanism that empowers the model to grasp intricate connections among image sections, ultimately enhancing its capability to recognize these gestures effectively. To reduce the size of the ViT model, a knowledge distillation process is applied to obtain a smaller yet equally accurate distilled student ViT model. The final prediction is obtained through ensemble learning stacking, which combines the predictions of four distilled student ViT models. This approach reduces the risk of overfitting and improves prediction accuracy, resulting in better overall performance.

The primary contributions of this research paper can be outlined as follows:

- The ViT model, equipped with a self-attention mechanism, effectively captures complex relationships among image patches, enhancing its capability to handle challenges such as variations in pose, lighting, background, occlusions, and the issue of similarity between hand gestures.
- Knowledge distillation is employed to distill the knowledge from a larger, more complex model into a smaller, computationally efficient version, enabling accurate and efficient hand gesture recognition with a reduced model size.

- Ensemble learning, through the combination of multiple models, effectively reduces the risk of overfitting and enhances the accuracy of HGR predictions.

2. Related Work

Researchers have addressed the issue of static hand gesture recognition by adopting various approaches over the years. These approaches can be broadly categorized into the traditional hand-crafted approach and the more recent deep learning approach. In the hand-crafted approach, features of hand gesture images were extracted using predefined techniques such as the local histogram feature descriptor (LHFD), discrete wavelet transform (DWT), K convex hull, localized contour sequence (LCS), block-based technique, HOG, SIFT, and non-negative matrix factorization (NMF). These features were then utilized for hand gesture classification.

The LHFD feature extraction method proposed by Reddy et al. [33] involved dividing the pre-processed image of hand gestures into 16 blocks, calculating the histogram for each block, and concatenating the histograms to form a feature vector. However, LHFD was sensitive to image scaling and rotation, limiting its effectiveness for images with different sizes and orientations. DWT was then introduced as an alternative, capturing the frequency domain information of an image more efficiently than LHFD. Several authors [14,16,17] utilized DWT for feature extraction in hand gesture image classification with slight variations in implementation. For example, Sahoo et al. [14] used the F-ratio to select DWT coefficients, reducing dimension and computation time while achieving higher accuracy than LCS and block-based methods. Candrasari et al. [16] employed a four-step process to extract features using DWT, selecting sub-bands and decomposition levels for accuracy improvement. Parvathy et al. [17] combined 2D-DWT for feature extraction with a modified speeded up robust features (SURF) algorithm and bag of feature (BoF) method for classification using a support vector machine (SVM).

Later, the K convex hull method emerged as a replacement for DWT, capturing the contour and shape information of hand gestures. Islam et al. [34] proposed the K convex hull method, combining K curvature and convex hull algorithms. The convex hull algorithm was applied to the segmented hand region, drawing a polygon around the hand gesture image, while the Sobel edge detection algorithm was used to detect the contour. The K curvature algorithm found the angle between two points for fingertip detection. The method also incorporated automatic pixel segmentation, eccentricity, elongatedness, and rotation algorithms to diversify the number of features and increase image samples for improved accuracy. However, the K convex hull method only captured contour information and may not have considered regional characteristics, limiting its effectiveness in distinguishing between similar hand gestures. LCS and block-based methods captured both contour and regional information, providing a more comprehensive feature representation for hand gestures. Ghosh et al. [35] proposed a combination of LCS and block-based feature extraction methods along with a genetic algorithm (GA)-based feature subset selection method. The GA selected the optimal feature subset using a chromosome bit pattern, and classification was performed using a K-means and least mean square (LMS)-based improved radial basis function (RBF) neural network.

HOG was proposed as a replacement for LCS and block-based methods as they may not have captured the fine-grained details of hand gestures. HOG captures gradient information in localized regions, enabling the detection of finer details and a more robust feature extraction process. Gupta et al. [18] utilized a combination of HOG and SIFT for feature extraction and employed K-nearest neighbors (KNNs) for classification. They created a dataset consisting of hand gestures performed with one or two hands, presenting a unique approach. Lahiani et al. [19] combined local binary patterns (LBPs) and HOG for feature extraction and used adaptive boosting (AdaBoost) for classification, achieving improved accuracy compared with individual feature extractors. Bamwenda et al. [20] described HOG as a histogram-based technique unaffected by changes in object geometry and illumination, making it suitable for real-time hand gesture detection. However, the

HOG approach was sensitive to changes in illumination and contrast, affecting recognition accuracy, especially in real-world scenarios. SIFT was more robust to illumination and contrast changes, detecting scale, rotation, and illumination invariant key points and extracting local features based on gradient directions. Ma et al. [21] proposed a method that combined SIFT for feature extraction with a sparse autoencoder neural network. NMF was later introduced as an alternative to SIFT, extracting features robust to changes in lighting and noise. Zhuang et al. [36] proposed using NMF for the subspace analysis of hand gesture images, achieving robust feature extraction. However, these handcrafted feature extraction methods required extensive domain expertise, trial and error for optimal feature selection, and may not be generalized well to new datasets. Additionally, handcrafted features may not capture complex relationships and patterns in the data necessary for accurate classification.

In contrast, deep learning approaches are capable of automatically learning high-level representations of the data through multiple layers of abstraction, reducing the need for manual feature engineering and improving performance in complex tasks. Consequently, researchers shifted towards employing deep learning approaches such as CNNs [1–6,22–27], artificial neural networks (ANNs) [30], and autoencoders [31,32] instead of traditional hand-crafted methods. Tan et al. [3] proposed a CNN model with spatial pyramid pooling (CNN-SPP), utilizing SPP instead of max pooling or average pooling to capture more spatial information and facilitate better learning. The CNN-SPP model with two layers of SPP demonstrated an increased average test accuracy for both augmented and non-augmented datasets compared with the typical CNN without SPP. In another work, Tan et al. [6] introduced an improved iteration of DenseNet known as EDenseNet by modifying the transition layer of the original DenseNet. The adjustment entailed altering the initial layer within the transition layer and introducing a fresh layer preceding the pooling layer, which led to enhanced feature transmission and a decrease in undesirable features. EDenseNet showed performance improvements of 0.6% and 0.45% for non-augmented and augmented data, respectively, compared with the original DenseNet.

Gao et al. [22] proposed a parallel CNN architecture consisting of RGB-CNN and Depth-CNN networks for HGR. The predictions from both channels were combined using a probability equation to achieve the final prediction. This approach improved the recognition accuracy of hand gestures by 3% to 11.9% compared with a single-channel CNN approach. Similarly, Khari et al. [1] utilized transfer learning and the refinement of a pretrained VGG19 CNN model for RGB and depth images in ASL classification. Both the RGB and depth VGG19 models underwent training on an expanded ASL dataset, and for the purpose of classification, the color and depth information were combined. In addition, they also applied early stopping to prevent overfitting during training. In a later work by Sahoo et al. [23], the fully connected layer of a pretrained CNN model (AlexNet) was used for feature extraction, and principal component analysis (PCA) was applied to reduce the feature dimension before being classified by an SVM. The aim of reducing the feature dimension was to eliminate redundant and unnecessary features from data with higher dimensions while preserving crucial information within the feature vector.

A similar work by Ozcan et al. [2] proposed an HGR approach, which applied transfer learning with a pretrained AlexNet CNN model and refined the model using the artificial bee colony (ABC) algorithm. The ABC algorithm was chosen for its ability to explore local solutions, handle objective costs, and provide flexibility. Cheng et al. [24] proposed a joint network comprising a superposed network of multiple restricted Boltzmann machines (RBMs) for unsupervised feature extraction and a CNN for supervised feature extraction. The features extracted from both networks were combined and classified to achieve the best performance. Mujahid et al. [4] proposed an HGR model based on the YOLOv3 and DarkNet-53 CNN neural network. The dataset was labeled using YOLO annotation, and the YOLOv3 algorithm extracted values from the bounding box formed in the image. These values underwent a sigmoid function before being passed to classification. Dadashzadeh et al. [25] presented a two-stage HGR method called HGR-Net, which con-

sisted of three CNN models. In the initial phase, a CNN model was trained to delineate hand gesture areas by separating them from the image background. Subsequently, in the second phase, two CNN models underwent training: one on the resulting image from the first phase and the other on the original hand gesture image. The outcomes from both models were combined using a fusion function. In [26], Alani et al. proposed an adapted deep CNN model for HGR. They used He uniform initialization for the ReLU layers and Xavier uniform initialization for the SoftMax layer along with L2 regularization. Xie et al. [27] introduced a fine-tuning method based on InceptionV3 for static HGR. Their method involved transfer learning and fine-tuning of InceptionV3 in a two-stage process along with feature concatenation of RGB and depth images. Early stopping callback and data augmentation were implemented to avoid overfitting and improve the model's recognition ability.

A later work by Ewe et al. [5] proposed a lightweight VGG16-RF ensemble classifier for vision-based HGR. They used VGG16 as the feature extractor due to its ability to extract low- and high-level features and prevent underfitting. To optimize the balance between accuracy and training time, they removed the fifth convolutional block. The random forest classifier was chosen for its ability to grow up to 100 trees, providing optimal performance and processing time. Badi et al. [30] presented two models for hand gesture classification: hand contour-based neural networks and complex moments-based neural networks. The hand contour-based networks consisted of five networks, while the region-based complex moments networks consisted of four networks. Both models were tested sequentially, and the testing process stopped when one network recognized the gesture. The hand contour-based networks were faster, while the complex moments-based networks were more accurate. In [31], Oyedotun et al. proposed a method called stacked denoising autoencoders (SDAEs) for learning distributed and hierarchical features of the training data. They employed the denoising autoencoder (DAE) to acquire meaningful features by reconstructing flawed input data at the output layer. The fine-tuning of the DAE for classification tasks was conducted using the backpropagation algorithm. Another work by Bobic et al. [32] introduced a sparse autoencoder with five hidden layers to discover interesting structures within data. They gradually reduced the parameters from 2500 to 800 to 400 to 200 to 100 and all the way down to 50 to extract features from the training set. After each dimension reduction, the SoftMax layer was trained for hand gesture classification. Among the five layers, the second layer with 400 parameters performed the best.

In the realm of static hand gesture recognition, prevailing deep learning methods, particularly those reliant on CNN-based networks, grapple with distinct challenges, including limited receptive fields, constrained adaptability, and susceptibility to overfitting. The inherent limitation of a limited receptive field in CNNs hampers their ability to capture extensive spatial relationships in static hand gestures. Moreover, the adaptability of CNNs is often constrained, posing challenges when confronted with diverse datasets and varying environmental conditions. The issue of overfitting, a common concern in CNN-based models, necessitates innovative approaches to enhance their generalization capabilities. To address these research gaps, this paper introduces the SDViT model as a promising solution. The ViT model leverages self-attention mechanisms, facilitating global interactions among all input features and mitigating the limited receptive field problem inherent in CNNs. Furthermore, the proposed knowledge distillation technique serves as a remedy to the generalization constraints of CNNs. By distilling knowledge from a larger and more complex model to a smaller and simpler network, the ViT model gains improved generalization abilities across diverse tasks and input data. The utilization of stacking further contributes to overcoming overfitting concerns, enhancing model performance by amalgamating predictions from multiple distilled ViT models. This multifaceted approach addresses critical limitations in existing CNN-based methods, positioning the SDViT model as a robust alternative for advancing static hand gesture recognition.

3. Stacking of Distilled Vision Transformers (SDViT)

This paper proposes stacking of distilled vision transformers, named as SDViT, for hand gesture recognition. The proposed SDViT method leverages transfer learning and fine-tuning of pretrained vision transformers to capture the global dependencies in a hand gesture image. Knowledge distillation is then employed, with the pretrained ViTs serving as both teacher and student models. The larger fine-tuned ViT model acts as the teacher, while the smaller fine-tuned ViT model acts as the student. Through this method, the distilled knowledge helps the smaller student model to generalize better. In addition, stacking is proposed to ensemble multiple distilled student models. The design of the proposed SDViT approach is illustrated in Figure 1.

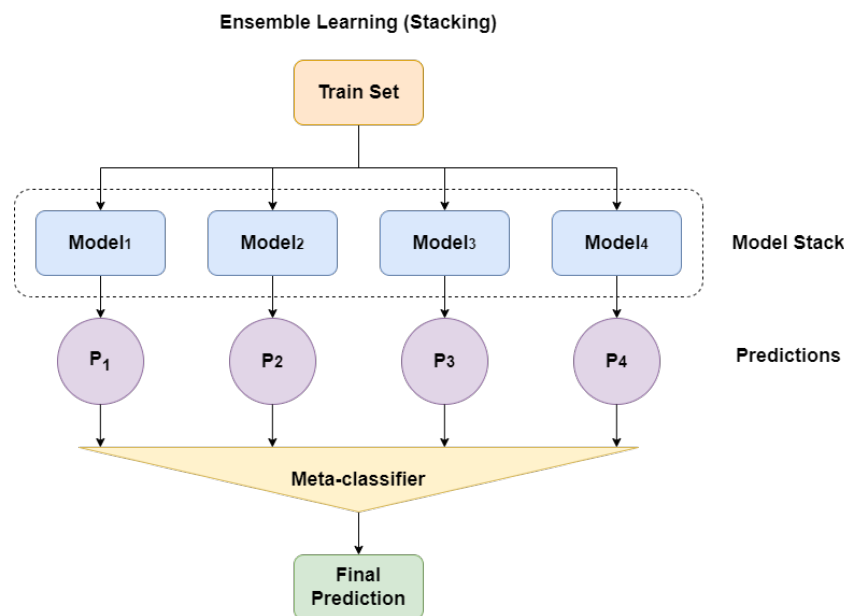


Figure 1. Network architecture of the proposed SDViT approach.

3.1. Vision Transformer

The vision transformer (ViT) is a modified version of the original transformer architecture, initially designed for natural language processing (NLP) tasks, adapted to solve vision-based tasks. It operates by dividing input images into patches, which are then linearly embedded and passed through the transformer. This treatment of image patches as tokens resembles the processing of text sequences in NLP applications, where tokens are fed into the transformer to predict class labels for images. The modifications to the original transformer architecture to create ViT were introduced by the Google Research Brain team [37]. The architecture of the ViT model in the proposed SDViT, as depicted in Figure 2, comprises several key components: the linear projection, transformer encoder, multi-head self attention (MSA) layer, and the multi-layer perceptron (MLP) layer. According to the researchers in [37], ViT offers significant improvements in both computational efficiency and accuracy compared with state-of-the-art (SOTA) CNN models. Nonetheless, ViT demonstrates a lesser inclination for translation equivariance and locality in its inductive bias compared with CNN models. This implies that ViT may not generalize well when trained on small datasets, and the authors recommend training ViT on at least 14 million images to mitigate this weakness. To address this limitation, the ViT model implemented in this paper utilizes the ViT model pretrained on the ImageNet21k + ImageNet2012 dataset. The transfer learning of the pretrained ViT model helps to mitigate the weak inductive bias issue associated with the ViT model.

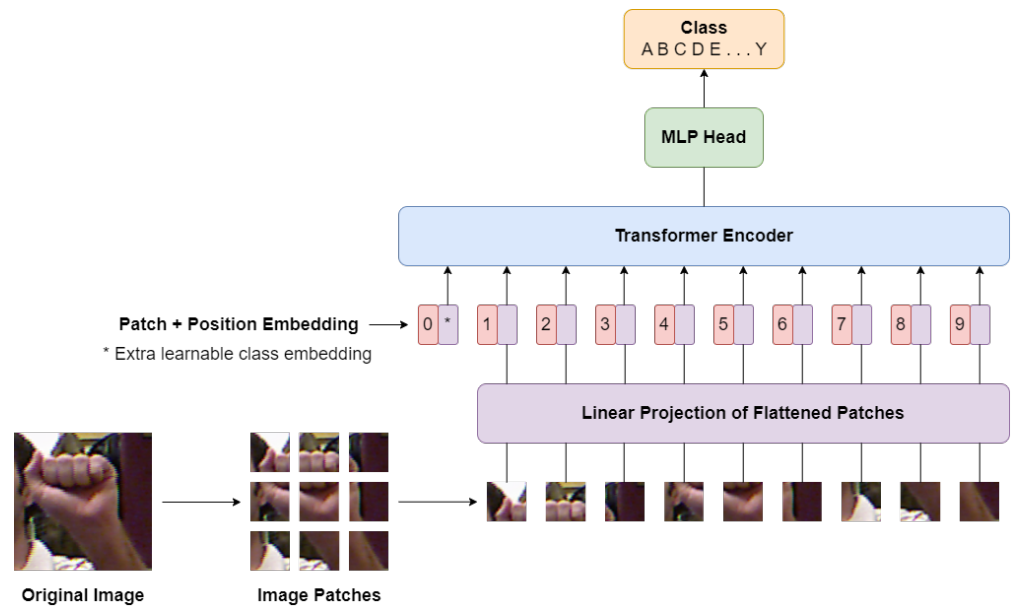


Figure 2. Network architecture of the vision transformer in the proposed SDViT approach.

Given the hand gesture image, the initial step involves resizing it to a dimension of 256×256 to ensure that it can be subdivided into patches measuring 32×32 . The resized image is subsequently divided into patches of consistent size, resulting in a series of flattened 2D patches denoted as $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$. The number of patches N is determined by the image dimensions and patch size, with $N = \frac{HW}{P^2}$. Here, H and W represent the height and width of the image, while C signifies the channel count and P denotes the patch size. N also serves as the effective sequence length applicable to the transformer.

3.1.1. Linear Projection of Flattened Patches

Before feeding the patches into the transformer encoder blocks, the image patches undergo a linear projection process. Each individual patch is transformed into a vector and associated with a lower-dimensional embedding through the utilization of a trainable matrix denoted as E . In order to incorporate positional details and support the learning process, one-dimensional positional embeddings denoted as E_{pos} , which are acquired through training, are incorporated. The role of positional embedding is to inject information about the relative or absolute position of each patch in the image by using sine and cosine functions with different frequencies to encode positional information. As ViTs do not have a built-in understanding of the spatial arrangement of pixels, positional embedding helps the model differentiate between different regions of the image and capture spatial relationships. Without positional embedding, the model might treat all patches equally, neglecting the essential spatial structure of the input. Additionally, a trainable class embedding x_{class} is included in the sequence of embedded patches, similar to the concept of the class token used in bidirectional encoder representations from transformers (BERT). The outcome of this linear projection process, labeled as z_0 , can be expressed as follows:

$$z_0 = [x_{class}; x_p^1 E; x_p^2 E; \dots; x_p^N E] + E_{pos}, \tag{1}$$

where $E \in \mathbb{R}^{(P^2 \cdot C) \times D}$, $E_{pos} \in \mathbb{R}^{(N+1) \times D}$, and x_p^n represents the flattened patch vector for the n th patch. The patch embeddings are employed as the input for the transformer encoder, enabling the ViT model to grasp overall patterns and relationships within the image. Simultaneously, it retains a degree of spatial information by utilizing patches.

3.1.2. Transformer Encoder

The transformer encoder plays a crucial role in the transformer model. It comprises several layers, denoted as L , each of which consists of an MSA layer and a position-wise feedforward layer (MLP), as depicted in Figure 3. Layer normalization (LN) is employed before the commencement of each block, and residual connections are implemented following the conclusion of each block. LN is applied to stabilize the model representations. Following the MSA layer, another layer normalization step is performed, and the resulting sequence is passed through the MLP layer. Equations (2) and (3) represent the flow of information within the transformer encoder block. Overall, the transformer encoder leverages self-attention mechanisms and MLPs to effectively grasp global interdependencies and efficiently handle representation processing.

$$z'_\ell = \text{MSA}(\text{LN}(z_{\ell-1})) + z_{\ell-1}, \quad \ell = 1 \dots L \tag{2}$$

$$z_\ell = \text{MLP}(\text{LN}(z'_\ell)) + z'_\ell, \quad \ell = 1 \dots L \tag{3}$$

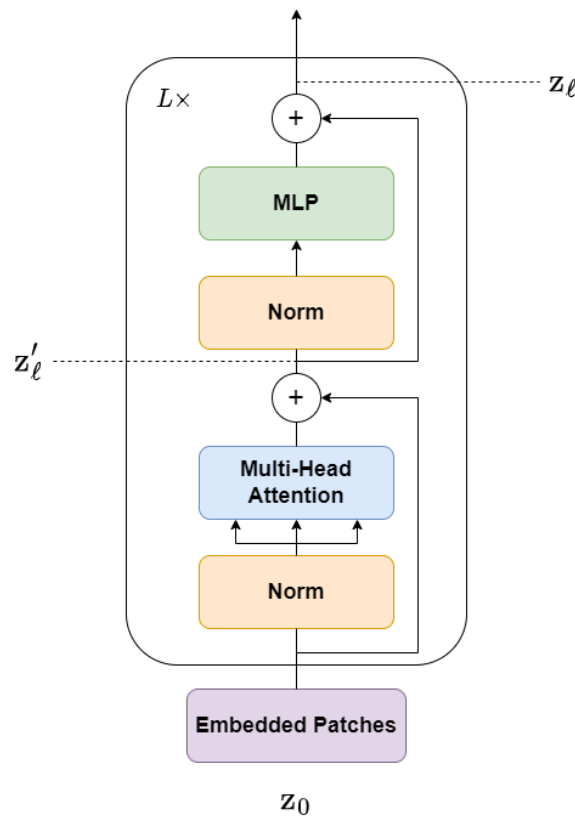


Figure 3. The design of the transformer encoder.

- Multi-head Self-Attention

The MSA layer in the ViT model captures contextual relationships among patch embeddings through self-attention. It utilizes numerous self-attention heads to assess the significance of patch embeddings by considering their interactions. Within the MSA layer, there are query, key, and value projections; dot product attention; and linear projection layers. It takes the patch embeddings as its input, carries out linear projections to produce query, key, and value matrices, and computes attention scores. These attention scores are employed to weigh the values, resulting in a weighted summation. The outcomes of this self-attention process are aggregated along the dimension of heads and subsequently subjected to a linear projection to return them to their original dimension. This process enables the MSA layer to capture patch relationships and produce the final output.

- Multi-Layer Perceptron

Within the ViT encoder, there is an MLP block responsible for modifying the representations acquired from the MSA layer. This block is composed of two linear transformations separated by a Gaussian error linear unit (GELU) activation function. The inclusion of the MLP introduces non-linear characteristics, enabling the model to capture intricate connections. The result of the MLP block is obtained through a series of operations, which involve multiplying the input by a weight matrix, adding a bias, applying the GELU activation function, and then repeating the process with another weight matrix and bias. This sequence of operations is iterated multiple times within the ViT encoder to learn progressively more abstract features from the input image.

The transformer encoder extracts features using self-attention in the MSA layer to capture global interdependencies among patch embeddings. Attention scores computed through query, key, and value projections facilitate the weighing of values, capturing contextual relationships. The resulting representations pass through the MLP block, introducing non-linear characteristics to progressively learn more abstract features. This feature extraction process enables the ViT model to capture intricate patterns and relationships within images, making it well suited for a variety of computer vision tasks.

In the encoder block, the first token z_L^0 is then selected and normalized to generate the image representation \mathbf{y} .

$$\mathbf{y} = \text{LN}\left(\mathbf{z}_L^0\right). \quad (4)$$

This representation is subjected to pooling in order to derive a solitary feature vector. Afterwards, it is flattened, normalized through batch normalization, and passed through a fully-connected layer with SoftMax activation for the ultimate classification. While training, the rectified Adam optimizer effectively adjusts the model's parameters. The categorical cross-entropy loss is employed to gauge the dissimilarity between predicted and actual probability distributions, with SoftMax probabilities and labels being utilized to compute this loss. To mitigate overfitting and enhance performance, the training process incorporates early stopping and adaptive learning rate callbacks.

3.2. Knowledge Distillation

Knowledge distillation is a powerful method for compressing models, allowing the transfer of expertise from a large, intricate model to a smaller one throughout training. This method is commonly known as the teacher–student model, with the larger model assuming the role of the teacher and the smaller model functioning as the student. The motivation behind knowledge distillation stems from the fact that large models possess a significant amount of knowledge capacity due to their numerous layers and parameters. However, this knowledge is not fully utilized when deploying these large models to devices that have limited memory and computational resources. To overcome this limitation, a model compression technique was initially proposed by [38] and later generalized by [39] as knowledge distillation. The fundamental concept of knowledge distillation is depicted in Figure 4, which illustrates the network architecture used in the proposed SDViT approach. Both the teacher and student models utilized in the knowledge distillation process are based on transfer learning from pretrained vision transformers, with varying fine-tuning layers. During training, the teacher model transfers its knowledge to the student model by guiding its learning process. This knowledge transfer is achieved by instructing the student model to replicate not only the final predictions made by the teacher model but also the internal representations and patterns learned by the teacher. Through this distillation of knowledge from the teacher model, the student model can capture and leverage the valuable insights and decision-making capabilities of the larger model while maintaining a compact size. This enables the student model to achieve comparable performance to the teacher model, making it a highly efficient solution for real-world applications.

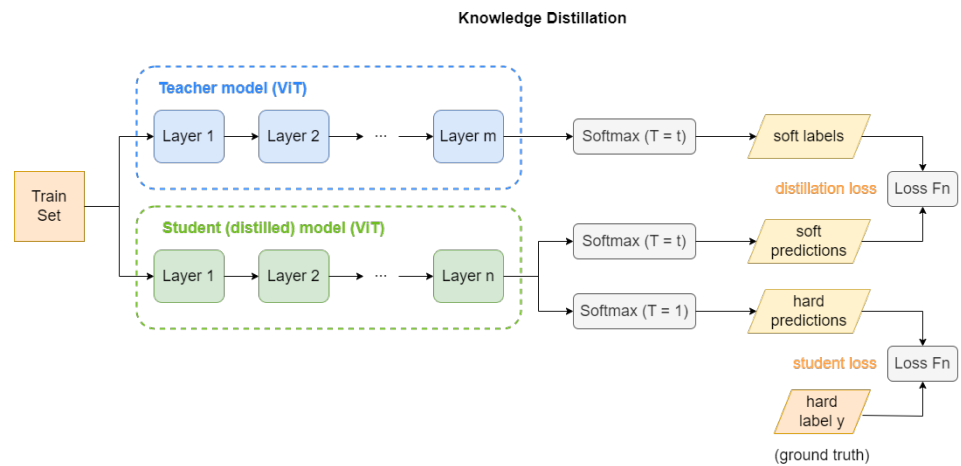


Figure 4. Network architecture of knowledge distillation of pretrained vision transformer model.

In this paper, response-based knowledge distillation is employed. The teacher model’s output probabilities or logits are utilized as the knowledge, which provide valuable information about the confidence or certainty of the model’s predictions. By incorporating this response-based knowledge into the training process of the student model, it can learn to mimic the teacher’s decision-making process and improve its own predictive capabilities. In addition, we adopt the widely used offline distillation approach. Offline distillation involves employing a pretrained teacher model to steer and inform the training of the student model through knowledge transfer. This method offers ease of implementation and is commonly employed in deep learning applications.

During the knowledge distillation process, knowledge transfer from the teacher model to the student model takes place through the minimization of a loss function. The objective in this process is to match the class probability distribution generated by the teacher model, which is derived by applying a SoftMax function to the teacher model’s logits. However, the teacher model’s probability distribution often assigns a high probability to the correct class while assigning close to zero probabilities to other classes. This limited distribution fails at offering extra information beyond the actual labels present in the dataset. To overcome this limitation, SoftMax temperature [39] is employed to calculate the probability p_i of class i directly from the logits z using the equation:

$$p_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)} \tag{5}$$

This equation represents the calculation of class probabilities, where the temperature parameter T plays a crucial role. A temperature of one yields the standard SoftMax function, while increasing the value of T results in a softer probability distribution. This softer distribution provides more information about which classes the teacher model considers similar to the predicted class. To incorporate this knowledge transfer, the same temperature parameter T is employed when applying the SoftMax function to the student’s logits during the computation of the loss function with respect to the teacher’s soft targets. This loss function is commonly known as the distillation loss.

In addition to the distillation loss, it is advantageous to ensure that the distilled model can accurately predict labels based on the true data. Consequently, the student loss is computed, which reflects the typical loss between the student’s forecasted class probabilities and the actual ground truth labels. The comprehensive loss function, which combines both the distillation loss and the student loss, is formulated as follows:

$$\mathcal{L}(x; W) = \alpha * \mathcal{H}(y, \sigma(z_s; T = 1)) + \beta * \mathcal{H}(\sigma(z_t; T = \tau), \sigma(z_s; T = \tau)) \tag{6}$$

In the given equation, x represents the input, W denotes the student model's parameters, y represents the actual ground truth label, and \mathcal{H} represents the cross-entropy loss function. The SoftMax function, which depends on the temperature parameter T , is denoted by σ . The coefficients α and β are utilized to balance the contributions of the distillation loss and the student loss, where $\beta = 1 - \alpha$. The logits of the student and teacher models are denoted as z_s and z_t , respectively, with the subscripts s and t indicating the student and teacher models.

From the loss function, it is observed that the values of τ , α , and β are hyperparameters. The temperature parameter, τ , controls the richness of information in the soft labels distribution. Increasing the temperature leads to a more diverse distribution. However, it is important to note that using higher temperatures may not always be beneficial for the student model, as excessively high temperatures can make it difficult for the student model to capture all of the rich information. It is generally recommended to use lower temperatures, although the most suitable temperature value could differ depending on the unique attributes of the student model.

On the other hand, α and β dictate the weighted mean of both the distillation loss and the student loss, with β being defined as $1 - \alpha$. The authors have observed that setting α to be much smaller than β leads to the best results. This implies that the distillation loss has a higher weight in the overall loss function, emphasizing the transfer of knowledge from the teacher to the student. It should be noted that the values of τ , α , and β are determined through experimentation and fine-tuning.

3.3. Stacking

Ensemble learning is a powerful machine learning technique designed to improve predictive performance by amalgamating the forecasts from numerous models [40]. There are different approaches to implementing ensemble learning, including bagging, stacking, and boosting. In this paper, we focus specifically on the stacking method, which involves training multiple models on the same dataset and combining their predictions using a meta-learner. By employing a stacking approach, we aim to harness the collective intelligence of multiple models and take advantage of their complementary strengths to enhance the accuracy and effectiveness of our predictive model. The configuration of ensemble learning with stacking is depicted in Figure 5.

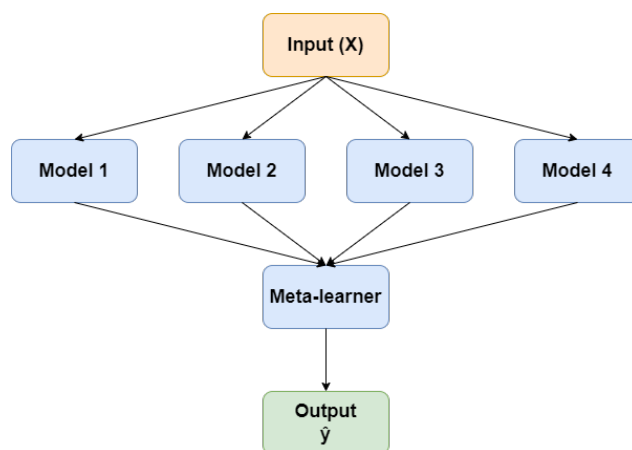


Figure 5. Structure of the stacking ensemble method.

One common approach to stacking is the two-level hierarchy of models. In this approach, the individual models, also known as level-0 models or first-level learners, make predictions on the data. The predictions from these models are then used as inputs for the meta-learner or level-1 model, which combines them to produce the final prediction. While various machine learning models can be used as the meta-learner to aggregate the predictions, logistic regression is commonly employed. This choice ensures that the

complexity of the model is primarily focused on the lower-level ensemble member models. By doing so, the simple combiner model can effectively learn to utilize the predictions of the lower-level models and determine which classifiers are more likely to succeed in different regions of the feature space, combining their predictions accordingly. By employing stacking in ensemble learning, we can exploit the diversity and collective intelligence of multiple models, thereby improving the comprehensive predictive capabilities and constructing a more resilient and precise model. The training procedure for the proposed SDViT approach is outlined in Algorithm 1.

Algorithm 1 The procedural steps for training the proposed SDViT method

Require: Dataset $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$, First level learning algorithm A_1, \dots, A_T , Second level learning algorithm A

- 1: **for** $t = 1, 2, \dots, T$ **do**
- 2: $m_t = A_t(D)$
- 3: **end for**
- 4: $D' = \Phi$
- 5: **for** $i = 1, 2, \dots, T$ **do**
- 6: **for** $t = 1, 2, \dots, T$ **do**
- 7: $f_{it} = m_t(x_i)$
- 8: **end for**
- 9: $D' = D' \cup \{(f_{i1}, f_{i2}, \dots, f_{it}), y_i\}$
- 10: **end for**
- 11: $m' = A(D')$
- 12: **return** $M(x) = m'(m_1(x), m_2(x), \dots, m_T(x))$

The ensemble learning training process begins with taking an input of hand gesture dataset D consisting of pairs (x_i, y_i) , where x_i represents the input data and y_i is the corresponding label of the hand gesture samples. Additionally, it requires two sets of learning algorithms: A_1, A_2, \dots, A_T for the first level and A for the second level. The algorithm starts by performing first-level learning. It iterates over T steps, where in each iteration, it employs a distinct first-level learning algorithm denoted as A_t on the dataset D . This process generates a set of first-level models m_1, m_2, \dots, m_T , each capturing different aspects and representations of the input data, which in our case is the set of knowledge-distilled ViT models. Next, the algorithm proceeds with feature extraction. For each data point x_i in the original dataset D , it applies the set of first-level models to obtain a set of intermediate features $f_{i1}, f_{i2}, \dots, f_{it}$. These intermediate features serve as representations that capture different levels of abstraction and contextual information from the input hand gesture data.

To further enhance the training process, an augmented dataset D' is created. Initially, D' is an empty set denoted by Φ . The algorithm then iterates over each data point x_i and collects the intermediate features $f_{i1}, f_{i2}, \dots, f_{it}$ obtained in the previous step. It combines these features with the original label y_i and adds the augmented pair $((f_{i1}, f_{i2}, \dots, f_{it}), y_i)$ to the augmented dataset D' . This augmentation expands the dataset by incorporating the intermediate features alongside the original data. With the augmented dataset in place, the algorithm proceeds to the second-level learning. It applies the second-level learning algorithm A to the augmented dataset D' , resulting in the training of the final model m' . This model integrates the information and representations learned from both the original input data and the intermediate features obtained from the first-level models.

Finally, the trained model $M(x)$ is defined as the composition of the final model m' and the set of first-level models m_1, m_2, \dots, m_T . Given an input data point x , the model $M(x)$ leverages the combined knowledge and representations from the first- and second-level models to make the final prediction. By stacking multiple levels of learning and incorporating intermediate representations, the algorithm aims to enhance the performance

and representation power of the final model. This enables the model to effectively grasp intricate patterns and connections within the hand gesture data.

4. Experiment and Analysis

In this section, we employ three benchmark datasets that are commonly used by researchers in the field of static hand gesture recognition [3,6,26,41–44] to assess the effectiveness of the proposed SDViT method. These datasets encompass the ASL, ASL with digits, and NUS hand gesture datasets.

4.1. Datasets

The ASL dataset, created by the authors in [45], serves as the first benchmark dataset for evaluation. It consists of 65,774 image samples representing 24 distinct hand gestures, encompassing the letters A to Y. It is worth noting that this dataset does not include the dynamic hand gestures J and Z. To introduce variation in the samples, the images were signed by five different individuals. Each class within the dataset contains a varying number of image samples, with the letter W having the highest count of 6221 samples, while the letter F has the lowest count of 5235 samples. Figure 6 displays the image samples corresponding to each category in the ASL dataset.



Figure 6. Illustrations of images representing each category within the ASL dataset. The label of each category is shown at the top of each subfigure.

The second dataset is known as the ASL with digits dataset, which was created by the authors in [46]. This dataset comprises 36 unique classes of hand gestures, encompassing the letters A to Z in addition to the numbers 0 to 9. It consists of hand gestures performed by five different signers to introduce variation, resulting in a total of 2515 image samples. Each class in this dataset is signed 25 times by the first and second signer except for the letter T, which is signed 20 times. The third and fourth signers perform each class five times, while the final signer performs each class ten times. Image samples representing each category in the ASL with digits dataset are depicted in Figure 7.

For the NUS hand gesture dataset [47], it comprises 10 classes and includes 2000 image samples. The 10 classes of hand gestures correspond to the letters A to J. To enhance the diversity in hand gestures, this dataset incorporates the largest number of signers among the three datasets, totaling 40 signers. Each signer performs each class five times. Image samples representing each category in the NUS hand gesture dataset are exhibited in Figure 8.



Figure 7. Illustrations of images representing each category within the ASL with digits dataset. The label of each category is shown at the bottom of each subfigure.

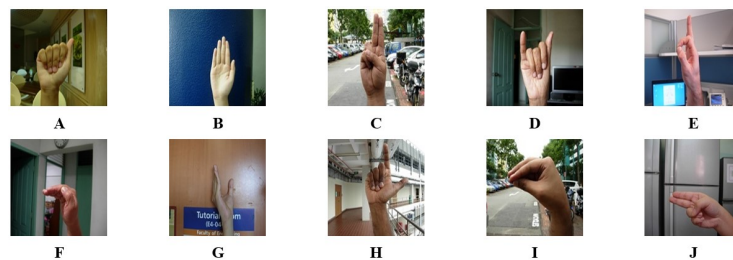


Figure 8. Illustrations of images representing each category within the NUS hand gesture dataset. The label of each category is shown at the bottom of each subfigure.

Figure 9 provides a visual representation of image samples from the three datasets, depicting the diverse challenges encountered in static hand gesture recognition. These challenges encompass five key categories: pose, lighting, background, occlusions, and similarity. In the pose challenges, images from the same class exhibit subtle variations in pose due to different signers. This introduces potential confusion for both human experts and machine learning models, as the nuanced differences in signing might lead to misinterpretations. Addressing lighting challenges, the displayed image samples capture hand gestures under suboptimal lighting conditions. Such conditions pose a difficulty in accurately discerning the hand gestures, thereby presenting a challenge for robust recognition. The background challenges showcase image samples against complex backgrounds. This complexity has the potential to mislead machine learning models, as elements of the background may be misconstrued as integral parts of the hand gesture. The occlusion challenge is exemplified by image samples where certain parts of the hand gestures, such as the thumb finger or a significant portion of the gesture, are obscured. This introduces ambiguity and difficulty in the interpretation of the hand gestures. Lastly, the similarity challenge is demonstrated through image samples where subtle differences exist between hand gestures of different classes. For instance, distinctions in the positioning of the thumb differentiate similar gestures, such as numbers and alphabets. These minute differences can be challenging for

both human experts and machine learning models to accurately discern. The presence of these challenges in static hand gesture recognition underscores the complexity of the task. The potential for confusion or misinterpretation not only poses difficulties for human experts but also demands sophisticated approaches in machine learning models to effectively navigate and overcome these challenges.

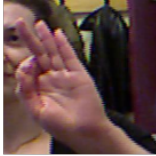

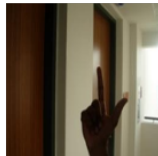

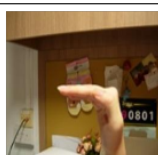
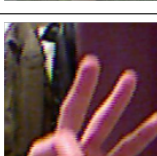
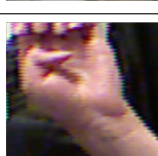

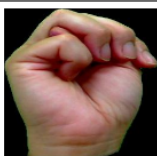
Challenges	Samples		
Pose			
			
Lighting			
Background			
Occlusions			
Similarity			
	Number 6	Alphabet M	Number 2
			
	Alphabet W	Alphabet N	Alphabet V

Figure 9. Illustrations of images representing each category of challenges found within the three datasets.

4.2. Experimental Setup

To ensure robust experimentation, this paper employs five-fold cross-validation (CV), where the dataset is split into five separate subsets or folds. During each fold, four subsets will be used for training purposes, while the remaining subset will serve as the test set. This process will be repeated, with each subset being used as the test set in turn. By employing five-fold CV, every subset of the dataset will serve as the test set at least once, thereby

reducing the risks of biased estimation and overfitting. Therefore, this method is commonly used by researchers in the field [3,6,17,23,41] as it is reliable for assessing model performance and mitigating the impact of data distribution variations among different subsets.

Table 1 illustrates the dataset division into folds and their assignments for training and testing. The dataset is segregated into five folds, numbered from 1 to 5. In each fold, the training sets are composed of four folds, while the remaining fold is used as the test set. This systematic arrangement ensures that the entire dataset is employed for both training and testing in a balanced manner throughout the CV process. Overall, employing five-fold CV facilitates a thorough assessment of the model's performance as it assesses its generalization capabilities across multiple partitions of the dataset. This methodology serves as a reliable way for verifying the model's effectiveness and reduces the impact of data distribution variations among different subsets.

Table 1. Distribution of the dataset among training and testing sets.

Dataset	Folds	Training Set (80%)	Testing Set (20%)	Total (100%)
ASL	5	52,619	13,155	65,774
ASL with Digits	5	2012	503	2515
NUS Hand Gesture	5	1600	400	2000

4.3. Experimental Analysis

The proposed SDViT approach consists of three components: the vision transformer, knowledge distillation, and ensemble learning stacking. To optimize the performance of the pretrained vision transformer through transfer learning and fine-tuning, a set of experiments is being carried out to identify the best combination of hyperparameters on the three benchmarking datasets. In the initial stage, all images from the three datasets are resized to dimensions of 256×256 which is a common approach utilized by existing researchers [48–52] for transformer-based models. This higher resolution is advantageous for fine-tuning the ViT compared with the pretrained ImageNet21k dataset, which has an image dimension of 224×224 . Research by [37] shows that feeding higher resolution images into the pretrained ViT with the same patch size leads to better performance due to the larger effective sequence length generated. Next, the input images undergo preprocessing in accordance with the ViT model's requirements. Subsequently, a flattening layer, batch normalization layer, and dense layer with SoftMax activation are applied for making predictions. The model is configured by having the rectified Adam optimizer employing a learning rate of 0.0001, by using categorical cross-entropy as the loss function, and by using accuracy as the evaluation metric.

In addition to the training setup, two essential callback methods, namely early stopping and reduce learning rate on plateau, are incorporated into the model. Early stopping is a fundamental technique that prevents overfitting and improves model generalization. It monitors the accuracy on a validation set during training and halts the process if the performance degrades over a set number of epochs. The parameters for early stopping include monitoring accuracy, a minimum delta of 0.0001, patience of 5, mode set to max, restore best weights set to true, and verbose of 1. The critical aspect of "restore best weights" becomes apparent in its role as a safeguard against overfitting. When set to true, this parameter ensures that, upon early stopping, the model reverts to the set of weights that achieved the highest accuracy on the validation set. This strategic addition is pivotal for preserving the model's ability to generalize well to new data. By restoring the weights associated with optimal validation performance, the model is less prone to memorizing noise or idiosyncrasies in the training data, promoting better generalization to real-world scenarios and unseen data. Simultaneously, the adaptive learning rate technique, known as reduce learning rate on plateau, is employed to dynamically adjust the learning rate during training. This technique contributes to the convergence of the model, preventing it from converging too quickly or too slowly. The parameters for reduce learning rate on

plateau involve monitoring accuracy, a factor of 0.2, patience of 2, verbose of 1, minimum delta of 0.0001, minimum learning rate of 0.000001, and mode set to max. The adaptive learning rate proves beneficial by adjusting the learning rate when a specified metric, such as accuracy, shows no improvement for a set number of epochs. These callback methods, when integrated into the training process, enhance the model's efficiency, stability, and generalization performance.

Two experiments were conducted on the pretrained ViT. The first experiment focused on comparing the ViT configured with a patch size of 32×32 , referred to as ViT B32, with fine-tuning of all available layers using weights from ImageNet21k and ImageNet2012 datasets. The results can be observed from Table 2, while the best results were achieved when using weights from ImageNet21k + ImageNet2012, with testing accuracies of 99.97% for the ASL dataset, 99.01% for the ASL with digits dataset, and 99.85% for the NUS hand gesture dataset. On average, across all three datasets, the accuracy reached 99.61%. The second experiment involved fine-tuning the ViT B32 with ImageNet21k + ImageNet2012 weights by unfreezing different numbers of layers, ranging from 3 to 19 layers. The results can be observed from Table 3, while the best results were obtained when fine-tuning 12 layers, achieving testing accuracies of 99.98% on the ASL dataset, 99.40% on the ASL with digits dataset, and 99.85% on the NUS hand gesture dataset. On average, across all three datasets, the accuracy reached 99.74%.

Table 2. Results of weights experiment of ViT on the three benchmark datasets.

Weights	Datasets			Average
	ASL	ASL with Digits	NUS Hand Gesture	
ImageNet21k	99.94%	98.53%	99.15%	99.21%
ImageNet21k + ImageNet2012	99.97%	99.01%	99.85%	99.61%

Table 3. Results of fine tuning experiment of ViT on the three benchmark datasets.

Fine Tune Layers	Datasets			Average
	ASL	ASL with Digits	NUS Hand Gesture	
3	99.93%	97.46%	98.20%	98.53%
4	99.95%	98.57%	99.15%	99.22%
5	99.97%	98.69%	99.05%	99.24%
6	99.98%	98.77%	99.35%	99.36%
7	99.98%	98.85%	99.40%	99.41%
8	99.98%	99.08%	99.60%	99.55%
9	99.98%	99.09%	99.80%	99.62%
10	99.99%	99.36%	99.80%	99.72%
11	99.98%	99.24%	99.80%	99.68%
12	99.98%	99.40%	99.85%	99.73%
13	99.96%	98.85%	99.70%	99.50%
14	99.95%	98.81%	99.75%	99.50%
15	99.96%	98.89%	99.85%	99.57%
16	99.96%	98.49%	99.70%	99.38%
17	99.97%	98.85%	99.75%	99.52%
18	99.96%	98.69%	99.90%	99.52%
19	99.97%	99.01%	99.85%	99.61%

To perform knowledge distillation on the pretrained vision transformer (ViT), a series of experiments were carried out to identify the best hyperparameter setup. The teacher and student models used in the knowledge distillation process have the same configurations as the pretrained ViT experiments but with different fine-tuning layers. The student model is a scaled-down version of the ViT, while the teacher model is a larger ViT. The model was then configured using the rectified Adam optimizer, utilizing a learning rate of 0.0001. Accuracy was used as the evaluation metric. The student loss function was categorical cross-entropy with label smoothing of 0.2, and the distillation loss function was Kullback–Leibler divergence, with an alpha value of 0.1 and a temperature of 10. Additionally, early stopping and reduce learning rate on a plateau were implemented during the training phase.

The first experiment for knowledge distillation focused on comparing teacher and student models with different fine-tuning layers, as more fine-tuning layers result in a more complex model. Various combinations of teacher–student models were tested, including models with more, fewer, or the same number of fine-tuning layers. The best results were obtained when the teacher model had 12 fine-tuning layers and the student model had 11 fine-tuning layers. With this setup, the model achieved accuracy scores of 99.99% on the ASL dataset, 99.36% on the ASL with digits dataset, and 99.90% on the NUS hand gesture dataset. On average, across all three datasets, the accuracy reached 99.75%. The second experiment focused on the alpha and temperature hyperparameters in the knowledge distillation process. The teacher and student models from the previous experiment, with 12 and 11 fine-tuning layers, respectively, were used. The most favorable outcomes were obtained when using the default temperature of 10 and an alpha value of 0.5, resulting in accuracies of 99.99% on the ASL dataset, 99.40% on the ASL with digits dataset, and 99.95% on the NUS hand gesture dataset. On average, across all three datasets, the accuracy reached 99.78%. The model was then saved in the h5 format for the stacking process, preserving the acquired knowledge and configurations to contribute effectively to the ensemble learning approach.

Finally, to leverage ensemble learning stacking with the knowledge-distilled models, an experiment was conducted using the distilled student model from the previous phase. The objective was to find the optimal combination of distilled student models that enhanced the overall performance. The predictions of each distilled student model on the dataset, which were loaded from the h5 model saved in the previous experiment, were stacked together before being passed to the logistic regression model for training. The best results were achieved by stacking four different models. The first combination consists of the distilled student model with 12 fine-tuned layers, a teacher model with 11 fine-tuned layers, and an alpha value of 0.5. The second combination involves the distilled student model with 11 fine-tuned layers, a teacher model with 12 fine-tuned layers, and an alpha value of 0.5. The third combination includes the distilled student model with 12 fine-tuned layers, a teacher model with 11 fine-tuned layers, and an alpha value of 0.1. Lastly, the fourth combination comprises the distilled student model with 11 fine-tuned layers, a teacher model with 12 fine-tuned layers, and an alpha value of 0.1. As a result, the proposed method achieved outstanding performance. The testing accuracy reached 100.00% on the ASL dataset, 99.60% on the ASL with digits dataset, and 100.00% on the NUS hand gesture dataset, leading to an average accuracy of 99.87% across all three datasets. This demonstrates the effectiveness of the ensemble learning stacking approach when applied to the knowledge-distilled models.

An additional experiment was also conducted to assess and validate the performance of the stacked distilled ViT (SDViT) model in comparison with the stacked ViT model. In this experiment, we stacked ViT models with a configuration similar to that of SDViT. The best result was achieved by stacking four different models, specifically, the top-performing ViT models from our pretrained ViT experiment, namely the ViT models fine-tuned with 10, 11, 12, and 19 layers. However, as shown in Table 4, while the stacked ViT demonstrated strong performance, it slightly lagged behind SDViT on the ASL with digits dataset, with a performance difference of just 0.04%. It is also worth noting that stacked ViT also utilized a

higher number of trainable parameters, exceeding SDViT by about 9.5 million trainable parameters. Therefore, our findings suggest that knowledge distillation not only aids in model compression but also improves the model performance.

Table 4. Results of stacked ViT compared against the proposed SDViT method on the three benchmark datasets.

Method	No. of Trainable Parameters	Dataset			Average
		ASL	ASL with Digits	NUS Hand Gesture	
Stacked ViT	278,843,904	100.00%	99.56%	100.00%	99.85%
SDViT	269,345,280	100.00%	99.60%	100.00%	99.87%

4.4. Experimental Results and Discussions

The proposed SDViT approach achieved remarkable testing accuracies on different datasets. Specifically, it achieved a 100.00% accuracy on the ASL dataset, 99.60% accuracy on the ASL with digits dataset, and 100.00% accuracy on the NUS hand gesture dataset. On average, across all three datasets, the testing accuracy stands at an impressive 99.87%. Table 5 shows a more comprehensive view of the performance of the proposed SDViT approach in each cross-validation set for the three datasets.

Table 5. Results of the proposed SDViT method for each cross-validation set on the three benchmark datasets.

Dataset	Cross-Validation Set	Test Accuracy (%)	F-Score
ASL	1	100.00	1.00
	2	100.00	1.00
	3	99.99	1.00
	4	100.00	1.00
	5	99.99	1.00
	Average		100.00
ASL with digits	1	99.20	1.00
	2	100.00	1.00
	3	100.00	1.00
	4	99.20	1.00
	5	99.60	1.00
	Average		99.60
NUS hand gesture	1	100.00	1.00
	2	100.00	1.00
	3	100.00	1.00
	4	100.00	1.00
	5	100.00	1.00
	Average		100.00

Figures 10–12 present the cost function against epochs graphs for the proposed SDViT approach across three distinct datasets. The incorporation of early stopping in the training phase is evident in these graphs, showcasing the model’s adaptive behavior in halting the training process when the loss ceases to decrease. This strategic intervention signifies that the model ceases to enhance its generalization capability. This implementation of early stopping serves a dual purpose. Firstly, it optimizes computational resources by concluding the training process upon reaching an optimal point, eliminating the need for training to a fixed number of epochs. This results in a more efficient use of computational power and time, preventing unnecessary prolongation of the training phase. More critically, the early

stopping mechanism acts as a safeguard against overfitting. By recognizing when the loss on the validation set plateaus or starts to rise, the model prevents itself from becoming overly specialized to the training data. This proactive measure ensures that the model not only performs well on the training set but also exhibits robust generalization to new, unseen data.

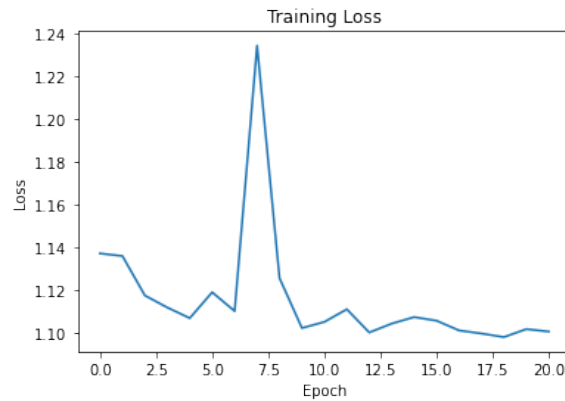


Figure 10. Cost function against epochs graph of the proposed SDViT approach on the ASL dataset.

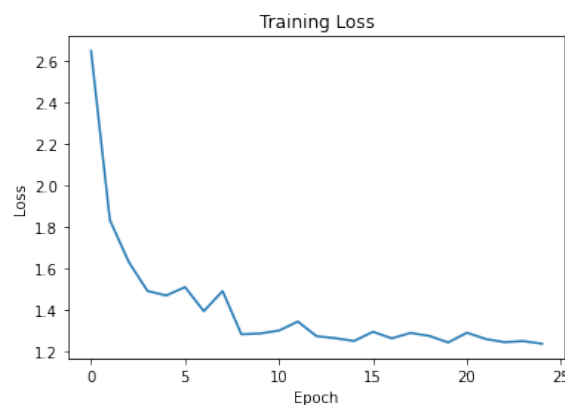


Figure 11. Cost function against epochs graph of the proposed SDViT approach on the ASL with digits dataset.

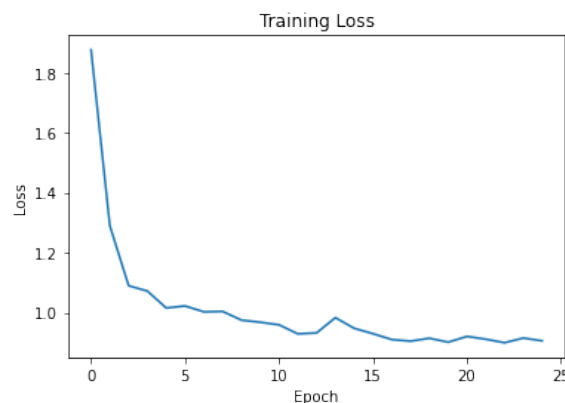


Figure 12. Cost function against epochs graph of the proposed SDViT approach on the NUS hand gesture dataset.

Figure 13 illustrates examples of misclassified samples from the benchmark datasets, mainly consisting of the four misclassified samples from the ASL with digits dataset. Two samples of the hand gesture for the letter O were misclassified as the number 0, one sample of the hand gesture for the letter V was misclassified as the number 2, and one sample of the

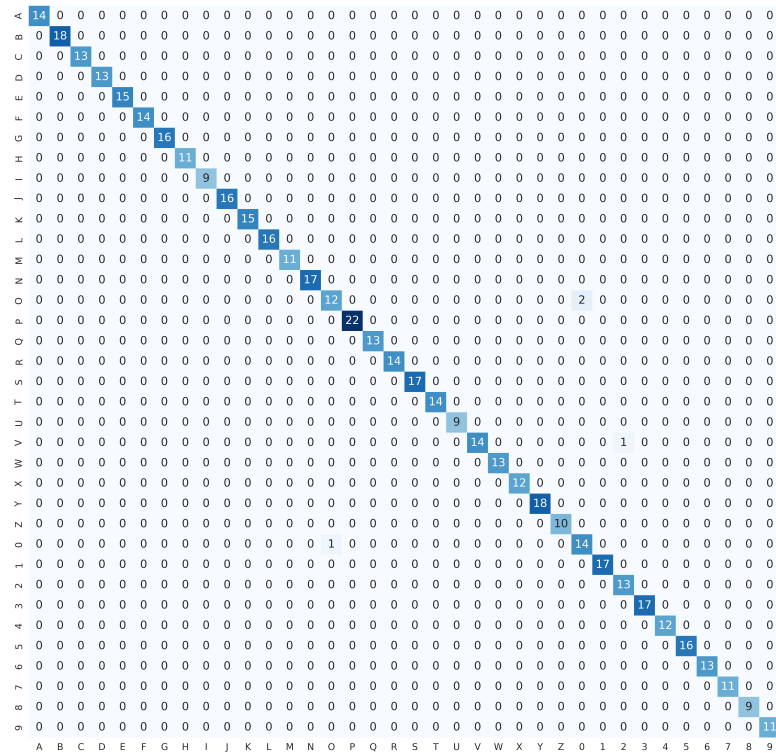


Figure 15. Confusion matrix of the proposed SDViT method on the ASL with digits dataset.

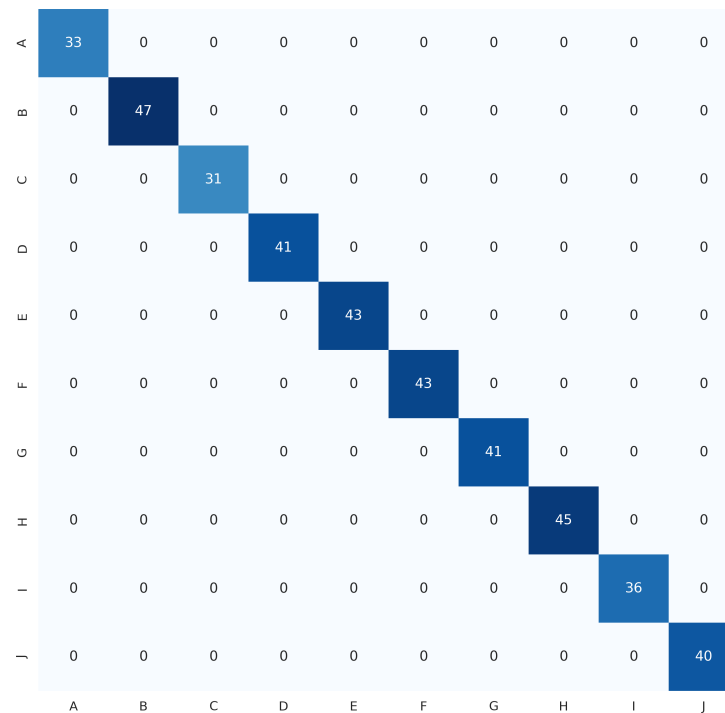


Figure 16. Confusion matrix of the proposed SDViT method on the NUS hand gesture dataset.

Table 6 presents a performance evaluation comparing the proposed SDViT with existing SOTA approaches on the ASL dataset, ASL with digits dataset, and NUS hand gesture dataset. Noticeably, the proposed SDViT method outshines the compared methods in all three datasets, achieving the highest classification accuracy. Specifically, it achieved a remarkable classification accuracy of 100.00%, 99.60%, and 100.00% on the ASL dataset, ASL with digits dataset, and NUS hand gesture dataset, respectively. Overall, when consid-

ering the average accuracy across all three datasets, the proposed method demonstrated an impressive accuracy of 99.87%, outperforming the SOTA approaches under comparison.

Table 6. Accuracy percentages for classification (%) of the SOTA methods and the proposed SDViT method across the three hand gesture datasets.

Method	ASL Dataset	ASL with Digits Dataset	NUS Hand Gesture Dataset	Average Accuracy
Deep learning with CNN [41]	99.96	-	94.70	-
CNN Baseline A [42]	99.85	98.69	89.15	95.90
CNN Baseline B [43]	99.78	98.65	89.30	95.91
ADCNN [26]	98.50	98.49	83.10	93.36
DAG-CNN [44]	99.89	98.13	91.05	96.36
EdenseNet [6]	99.89	98.85	96.75	98.50
CNN-SPP [3]	99.94	99.17	95.95	98.35
SDViT (Ours)	100.00	99.60	100.00	99.87

5. Conclusions

In conclusion, this paper proposes stacking of distilled vision transformers, referred to as SDViT, for hand gesture recognition. The proposed SDViT encompasses several key components that contribute to its effectiveness. Firstly, transfer learning and fine-tuning of pretrained ViTs are employed, leveraging their superior performance and computational efficiency compared with SOTA CNNs. By incorporating early stopping and reduce learning rate on plateau techniques, the pretrained ViTs are fine-tuned to achieve optimal results. Additionally, knowledge distillation techniques are applied using the pretrained ViTs as both teacher and student models. This process not only enhances the performance of the models but also reduces their overall size, resulting in reduced computational resource requirements. Furthermore, the knowledge distilled student models are ensembled using a stacking approach. This approach combines the predictions of multiple stacked models, effectively reducing bias and enhancing model stability and accuracy.

While the proposed SDViT model demonstrates remarkable performance in hand gesture recognition, it is essential to acknowledge certain limitations and identify potential areas for improvement. One notable aspect is the reliance on distilled student models which, despite their computational efficiency, might not capture all of the nuanced features present in the original ViT. Additionally, the generalization of the model across diverse datasets and real-world scenarios warrants further investigation. The sensitivity of the model to variations in lighting conditions, occlusions, or dynamic backgrounds could be a potential challenge that needs attention. Furthermore, the ensemble learning approach, while effective, introduces additional computational overhead, and optimizing the ensemble size and configuration could be explored for efficiency. These considerations pave the way for future research to refine and extend the capabilities of the SDViT model in addressing real-world challenges and diverse application scenarios.

The techniques and insights from SDViT, which include transfer learning, knowledge distillation, and ensemble learning, present a versatile toolkit with applications extending beyond hand gesture recognition. One notable application is in the realm of object recognition, where SDViT's transfer learning facilitates swift adaptation to new datasets, leveraging pretrained ViTs for robust feature extraction and hierarchical representations. The knowledge distillation process ensures the creation of compact yet accurate models that are suitable for resource-constrained environments. The ensemble learning aspect contributes to the model's robustness, enhancing accuracy and generalization by aggregating diverse perspectives from multiple distilled models. Additionally, in medical image analysis, SDViT's ability to capture intricate patterns and relationships proves valuable. Transfer learning aids in scenarios with limited labeled medical datasets, while knowledge distillation creates more efficient models without sacrificing diagnostic capabilities. The ensemble learning approach ensures reliability across diverse patient profiles and imaging conditions. These examples illustrate how SDViT's methodologies adapt to specific com-

puter vision tasks, offering improvements in performance, efficiency, and interpretability, thus showcasing its potential for broader applications.

In our extensive evaluations, the proposed SDViT consistently outperformed existing methods on three benchmark datasets. Achieving a remarkable testing accuracy of 100.00% on the ASL dataset, 99.60% on the ASL with digits dataset, and 100.00% on the NUS hand gesture dataset and an average accuracy of 99.87% across all three datasets, the results underscore the model's robustness and effectiveness. The superior performance positions SDViT as a state-of-the-art solution for vision-based static hand gesture recognition, demonstrating its potential for real-world applications across diverse domains within computer vision.

Author Contributions: Conceptualization, C.K.T. and K.M.L.; methodology, C.K.T., K.M.L., C.P.L., R.K.Y.C. and A.A.; software, C.K.T. and K.M.L.; validation, C.K.T., K.M.L., C.P.L., R.K.Y.C. and A.A.; formal analysis, C.K.T.; investigation, C.K.T. and K.M.L.; resources, C.K.T. and K.M.L.; data curation, C.K.T. and C.P.L.; writing—original draft preparation, C.K.T. and K.M.L.; writing—review and editing, C.K.T., K.M.L., C.P.L., R.K.Y.C. and A.A.; visualization, C.K.T. and C.P.L.; supervision, K.M.L. and R.K.Y.C.; project administration, K.M.L. and A.A.; funding acquisition, K.M.L. and A.A. All authors have read and agreed to the published version of the manuscript.

Funding: The research in this work was supported by Telekom Malaysia Research & Development under grant number RDTC/231084 and by the Deanship of Scientific Research, King Khalid University, Saudi Arabia, under grant number RGP2/332/44.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

References

1. Khari, M.; Garg, A.K.; Crespo, R.G.; Verdú, E. Gesture Recognition of RGB and RGB-D Static Images Using Convolutional Neural Networks. *Int. J. Interact. Multimed. Artif. Intell.* **2019**, *5*, 22–27. [[CrossRef](#)]
2. Ozcan, T.; Basturk, A. Transfer learning-based convolutional neural networks with heuristic optimization for hand gesture recognition. *Neural Comput. Appl.* **2019**, *31*, 8955–8970. [[CrossRef](#)]
3. Tan, Y.S.; Lim, K.M.; Tee, C.; Lee, C.P.; Low, C.Y. Convolutional neural network with spatial pyramid pooling for hand gesture recognition. *Neural Comput. Appl.* **2021**, *33*, 5339–5351. [[CrossRef](#)]
4. Mujahid, A.; Awan, M.J.; Yasin, A.; Mohammed, M.A.; Damaševičius, R.; Maskeliūnas, R.; Abdulkareem, K.H. Real-time hand gesture recognition based on deep learning YOLOv3 model. *Appl. Sci.* **2021**, *11*, 4164. [[CrossRef](#)]
5. Ewe, E.L.R.; Lee, C.P.; Kwek, L.C.; Lim, K.M. Hand Gesture Recognition via Lightweight VGG16 and Ensemble Classifier. *Appl. Sci.* **2022**, *12*, 7643. [[CrossRef](#)]
6. Tan, Y.S.; Lim, K.M.; Lee, C.P. Hand gesture recognition via enhanced densely connected convolutional neural network. *Expert Syst. Appl.* **2021**, *175*, 114797. [[CrossRef](#)]
7. Tan, Y.S.; Lim, K.M.; Lee, C.P. Wide Residual Network for Vision-based Static Hand Gesture Recognition. *IAENG Int. J. Comput. Sci.* **2021**, *48*, 906–914.
8. Lim, K.M.; Tan, A.W.; Tan, S.C. A four dukkha state-space model for hand tracking. *Neurocomputing* **2017**, *267*, 311–319. [[CrossRef](#)]
9. Chen, X.; Wang, G.; Guo, H.; Zhang, C.; Wang, H.; Zhang, L. Mfa-net: Motion feature augmented network for dynamic hand gesture recognition from skeletal data. *Sensors* **2019**, *19*, 239. [[CrossRef](#)]
10. Rahim, M.A.; Islam, M.R.; Shin, J. Non-touch sign word recognition based on dynamic hand gesture using hybrid segmentation and CNN feature fusion. *Appl. Sci.* **2019**, *9*, 3790. [[CrossRef](#)]
11. Vaitkevičius, A.; Taroza, M.; Blažauskas, T.; Damaševičius, R.; Maskeliūnas, R.; Woźniak, M. Recognition of American sign language gestures in a virtual reality using leap motion. *Appl. Sci.* **2019**, *9*, 445. [[CrossRef](#)]
12. Dong, Y.; Liu, J.; Yan, W. Dynamic hand gesture recognition based on signals from specialized data glove and deep learning algorithms. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 1–14. [[CrossRef](#)]
13. Athira, P.; Sruthi, C.; Lijiya, A. A signer independent sign language recognition with co-articulation elimination from live videos: An Indian scenario. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 771–781. [[CrossRef](#)]
14. Sahoo, J.P.; Ari, S.; Ghosh, D.K. Hand gesture recognition using DWT and Fratio based feature descriptor. *IET Image Process.* **2018**, *12*, 1780–1787. [[CrossRef](#)]

15. Aamir, M.; Pu, Y.F.; Rahman, Z.; Tahir, M.; Naeem, H.; Dai, Q. A framework for automatic building detection from low-contrast satellite images. *Symmetry* **2018**, *11*, 3. [[CrossRef](#)]
16. Candrasari, E.B.; Novamizanti, L.; Aulia, S. Discrete Wavelet Transform on static hand gesture recognition. *J. Phys. Conf. Ser.* **2019**, *1367*, 012022. [[CrossRef](#)]
17. Parvathy, P.; Subramaniam, K.; Venkatesan, G.K.P.; Karthikaikumar, P.; Varghese, J.; Jayasankar, T. Development of hand gesture recognition system using machine learning. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 6793–6800. [[CrossRef](#)]
18. Gupta, B.; Shukla, P.; Mittal, A. K-nearest correlated neighbor classification for Indian sign language gesture recognition using feature fusion. In Proceedings of the 2016 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 7–9 January 2016; pp. 1–5.
19. Lahiani, H.; Neji, M. Hand gesture recognition method based on HOG-LBP features for mobile device. *Procedia Comput. Sci.* **2018**, *126*, 254–263. [[CrossRef](#)]
20. Bamwenda, J.; Özerdem, M.S. Recognition of Static Hand Gesture with Using ANN and SVM. *Dicle Univ. J. Eng.* **2019**, *10*, 561–568.
21. Ma, L.; Huang, W. A static hand gesture recognition method based on the depth information. In Proceedings of the 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Hangzhou, China, 27–28 August 2016; Volume 2, pp. 136–139.
22. Gao, Q.; Liu, J.; Ju, Z.; Li, Y.; Zhang, T.; Zhang, L. Static hand gesture recognition with parallel CNNs for space human-robot interaction. In Proceedings of the Intelligent Robotics and Applications: 10th International Conference, ICIRA 2017, Wuhan, China, 16–18 August 2017; pp. 462–473.
23. Sahoo, J.P.; Ari, S.; Patra, S.K. Hand gesture recognition using PCA based deep CNN reduced features and SVM classifier. In Proceedings of the 2019 IEEE International Symposium on Smart Electronic Systems (iSES)(Formerly iNiS), Rourkela, India, 16–18 December 2019. [[CrossRef](#)]
24. Cheng, W.; Sun, Y.; Li, G.; Jiang, G.; Liu, H. Jointly network: A network based on CNN and RBM for gesture recognition. *Neural Comput. Appl.* **2019**, *31*, 309–323. [[CrossRef](#)]
25. Dadashzadeh, A.; Targhi, A.T.; Tahmasbi, M.; Mirmehdi, M. HGR-Net: A fusion network for hand gesture segmentation and recognition. *IET Comput. Vis.* **2019**, *13*, 700–707. [[CrossRef](#)]
26. Alani, A.A.; Cosma, G.; Taherkhani, A.; McGinnity, T.M. Hand gesture recognition using an adapted convolutional neural network with data augmentation. In Proceedings of the 2018 4th International Conference on Information Management (ICIM), Oxford, UK, 25–27 May 2018. [[CrossRef](#)]
27. Xie, B.; He, X.; Li, Y. RGB-D static gesture recognition based on convolutional neural network. *J. Eng.* **2018**, *2018*, 1515–1520. [[CrossRef](#)]
28. Aamir, M.; Rahman, Z.; Dayo, Z.A.; Abro, W.A.; Uddin, M.I.; Khan, I.; Imran, A.S.; Ali, Z.; Ishfaq, M.; Guan, Y.; et al. A deep learning approach for brain tumor classification using MRI images. *Comput. Electr. Eng.* **2022**, *101*, 108105. [[CrossRef](#)]
29. Guan, Y.; Aamir, M.; Hu, Z.; Dayo, Z.A.; Rahman, Z.; Abro, W.A.; Soothar, P. An Object Detection Framework Based on Deep Features and High-Quality Object Locations. *Trait. Signal* **2021**, *38*, 719–730. [[CrossRef](#)]
30. Badi, H. Recent methods in vision-based hand gesture recognition. *Int. J. Data Sci. Anal.* **2016**, *1*, 77–87. [[CrossRef](#)]
31. Oyedotun, O.K.; Khashman, A. Deep learning in vision-based static hand gesture recognition. *Neural Comput. Appl.* **2017**, *28*, 3941–3951. [[CrossRef](#)]
32. Bobic, V.; Tadic, P.; Kvascev, G. Hand gesture recognition using neural network based techniques. In Proceedings of the 2016 13th Symposium on Neural Networks and Applications (NEUREL), Belgrade, Serbia, 22–24 November 2016. [[CrossRef](#)]
33. Reddy, D.A.; Sahoo, J.P.; Ari, S. Hand gesture recognition using local histogram feature descriptor. In Proceedings of the 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 11–12 May 2018; pp. 199–203.
34. Islam, M.M.; Siddiqua, S.; Afnan, J. Real time Hand Gesture Recognition using different algorithms based on American Sign Language. In Proceedings of the 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Dhaka, Bangladesh, 13–14 February 2017. [[CrossRef](#)]
35. Ghosh, D.K.; Ari, S. On an algorithm for Vision-based hand gesture recognition. *Signal Image Video Process.* **2016**, *10*, 655–662. [[CrossRef](#)]
36. Zhuang, H.; Yang, M.; Cui, Z.; Zheng, Q. A method for static hand gesture recognition based on non-negative matrix factorization and compressive sensing. *IAENG Int. J. Comput. Sci.* **2017**, *44*, 52–59.
37. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
38. Bucilă, C.; Caruana, R.; Niculescu-Mizil, A. Model compression. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, 20–23 August 2006; Volume 2006. [[CrossRef](#)]
39. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
40. Brownlee, J. A Gentle Introduction to Ensemble Learning Algorithms. *Mach. Learn. Mastery* **2021**. Available online: <https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/> (accessed on 15 August 2023).
41. Adithya, V.; Rajesh, R. A deep convolutional neural network approach for static hand gesture recognition. *Procedia Comput. Sci.* **2020**, *171*, 2353–2361.

42. Flores, C.J.L.; Cutipa, A.G.; Enciso, R.L. Application of convolutional neural networks for static hand gestures recognition under different invariant features. In Proceedings of the 2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON), Cusco, Peru, 15–18 August 2017; pp. 1–4.
43. Ahuja, R.; Jain, D.; Sachdeva, D.; Garg, A.; Rajput, C. Convolutional neural network based american sign language static hand gesture recognition. *Int. J. Ambient. Comput. Intell. IJACI* **2019**, *10*, 60–73. [[CrossRef](#)]
44. Arenas, J.O.P.; Moreno, R.J.; Beleño, R.D.H. Convolutional neural network with a dag architecture for control of a robotic arm by means of hand gestures. *Contemp. Eng. Sci.* **2018**, *11*, 547–557. [[CrossRef](#)]
45. Pugeault, N.; Bowden, R. Spelling it out: Real-time ASL fingerspelling recognition. In Proceedings of the 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, Spain, 6–13 November 2011; pp. 1114–1119. [[CrossRef](#)]
46. Barczak, A.L.C.; Reyes, N.H.; Abastillas, M.; Piccio, A.; Susnjak, T. A New 2D Static Hand Gesture Colour Image Dataset for ASL Gestures. *Res. Lett. Inf. Math. Sci.* **2011**, *15*, 12–20.
47. Pisharady, P.K.; Vadakkepat, P.; Loh, A.P. Attention based detection and recognition of hand postures against complex backgrounds. *Int. J. Comput. Vis.* **2013**, *101*, 403–419. [[CrossRef](#)]
48. Gruber, I.; Krnoul, Z.; Hruz, M.; Kanis, J.; Bohacek, M. Mutual support of data modalities in the task of sign language recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 3424–3433.
49. Geng, J.; Wen, L.; Geng, J. TCCNN: Transformer ConCated Convolutional Neural Networks for Hand Gesture Recognition. In Proceedings of the CAIBDA 2022, 2nd International Conference on Artificial Intelligence, Big Data and Algorithms, Nanjing, China, 17–19 June 2022; pp. 1–3.
50. Cao, Z.; Li, Y.; Shin, B.S. Content-Adaptive and Attention-Based Network for Hand Gesture Recognition. *Appl. Sci.* **2022**, *12*, 2041. [[CrossRef](#)]
51. Liu, Y.; Li, X.; Yang, L.; Bian, G.; Yu, H. A CNN-Transformer Hybrid Recognition Approach for sEMG-based Dynamic Gesture Prediction. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 2514816. [[CrossRef](#)]
52. Yoo, C.H.; Yoo, J.H.; Kim, H.W.; Han, B. Pointing Gesture Recognition via Self-Supervised Regularization for ASD Screening. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 2–3 October 2023; pp. 3036–3043.

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.