Dynamic Gesture Recognition using a Transformer and Mediapipe

Asma H.Althubiti, Haneen Algethami
Department of Computer Science
College of Computers and Information Technology
Taif University, Taif, 21944, Saudi Arabia

***Abstract*—There is a rising interest in dynamic gesture recognition as a research area. This is the result of emerging global pandemics as well as the need to avoid touching different surfaces. Most of the previous research has focused on implementing deep learning algorithms for the RGB modality. However, despite its potential to enhance the algorithm's performance, gesture recognition has not widely utilised the concept of attention. Most research also used three-dimensional convolutional networks with long short-term memory networks for gesture recognition. However, these networks can be computationally expensive. As a result, this paper employs pre-trained models in conjunction with the skeleton modality to address the challenges posed by background noise. The goal is to present a comparative analysis of various gesture recognition models, divided based on video frames or skeletons. The performance of different models was evaluated using a dataset taken from Kaggle with a size of 2 GB. Each video contains 30 frames (or images) to recognise five gestures. The transformer model for skeleton-based gesture recognition achieves 0.99 accuracy and can be used to capture temporal dependencies in sequential data.**

Keywords*—*Gesture recognition; self-attention; transformer encoder; skeleton; transfer learning

I. Introduction

In an increasingly interconnected world where human-computer interaction plays a pivotal role, the ability to accurately recognise and interpret gestures has emerged as a critical component of next-generation technology. Consequently, the growing importance of gesture recognition has prompted extensive research and development efforts to advance this technology's capabilities and applications. Gesture recognition is an interesting field of computer vision and is linked to solving many day-to-day problems and simplifying human life [1]. In addition to everyday applications such as clinical operation [2], sign language [3], robots [4], virtual environment [5], home automation [6], personal computers and tablets [6], and gaming, driver behaviour [7], [8], [9]. In recent years and with the appearance of epidemics, there has been an urgent need to develop this field, and gesture recognition is beginning to be used in remote areas of the world [10], [11].

Dynamic gesture recognition allows users to perform natural and intuitive gestures to control devices, which can be more engaging and easier to learn than traditional input methods like keyboards and mouse. It provides an alternative input method for individuals with physical disabilities who may find it difficult to use conventional interfaces. Touchless systems can be tailored to recognize a wide range of gestures, accommodating different abilities and preferences. In environments where hygiene is crucial, such as hospitals, clean rooms, or public kiosks, touchless interaction reduces the risk of contamination and the spread of pathogens. Touchless systems are particularly useful in scenarios where users need to interact with devices while keeping their hands free, such as in kitchens, workshops, or while driving.

Dynamic gesture recognition can be integrated into various applications, from gaming and virtual reality to smart home systems and industrial automation. This versatility makes it a valuable component in a wide range of touchless interaction systems. These systems can be designed to recognize context-specific gestures, making interactions more efficient and reducing the cognitive load on users. For example, different gestures can be used for different modes of an application or device. Dynamic gestures can be used to perform complex commands that would be cumbersome with traditional input methods. For instance, gestures can control the playback of media, navigate through interfaces, or manipulate virtual objects in 3D space.

Gesture recognition techniques can be categorised into three main approaches: the glove-based hand approach [12], the radar-based hand gesture approach [13], and computer vision-based hand gesture recognition [6]. In the first approach, the precise coordinates of the palm and fingers can be determined to facilitate accurate gesture recognition. According to the angle of bending, several sensors used the same technique, including the curvature sensor, the angular displacement sensor, the optical fibre transducer, the flex sensor, and the accelerometer. Due to the high cost of these sensors, it is difficult to detect hand gestures on gloves due to their different physical principles [6]. The second approach is that a radar transmitter transmits a radio wave towards a target, and the radar receiver intercepts the reflected energy. In this technology, radar waves bounce off your hand and back to the receiver, allowing it to interpret changes in shape or movement. This technology is still being investigated [14]. Computer vision-based hand gesture recognition involves using algorithms and image analysis to interpret and understand hand movements and gestures captured by cameras or sensors, facilitating natural human-computer interaction.

The objective of motion recognition research is to delineate human gestures and subsequently employ these gestures for device control or information transmission [15]. Gestures can be broadly categorised into two types: dynamic (temporal) and static (spatial). In static gesture recognition, an image of a hand captured at a specific moment is utilised, with recognition outcomes relying on its position, contour, and texture within the image. An image sequence captured over a continuous period can be used to recognise dynamic gestures. The recognition result is not only affected by the appearance of the hand but also by the temporal characteristics describing its trajectory [16]. Dynamic gestures differ from static gestures in that they are more varied, more expressive, and more practical [17]. There are three goals for dealing with hand gestures: detection, tracking, and recognition. Similarly to image classification, videos recorded in a real-life scenario may contain a variety of interferences. Some examples of such interferences include blurry gestures that occur when a camera shakes or a performer moves suddenly during video recording [17]. Variable illumination intensities, intricate backgrounds, and unique hand gestures made by various people are limitations of dynamic hand gesture recognition [18].

In the context of video-based gesture recognition, the fundamental challenge revolves around the identification of specific actions being performed. Gesture recognition encompasses the broader scope of action recognition within a video, while gesture recognition detection and segmentation entail specialised methods for pinpointing and analysing individual instances of gestures embedded within the video stream. Deep learning models require access to large video datasets to support the acquisition of precise action representations, which is a challenging task given the challenging dimensionality and vast amount of video data. In addition, these models must be able to extract both spatial and temporal details from video clips, which will make it easier to recognise complex gestures [19].

In the world of computer vision research, there is a fast-growing trend towards using transformer architectures. These new approaches are making action recognition much more accurate and efficient, marking a significant change in how things are done in this field [19]. When it comes to transformer learning, building deep convolutional neural networks from the ground up, such as AlexNet, GoogleNet, and ResNet, requires access to large, carefully annotated datasets. ImageNet is one of the most important datasets used as a reference [20]. Pre-trained deep CNN models can serve as fixed feature extractors or fine-tuners with limited data [15]. Attention is the sole concept driving this transformation, which is perhaps the most powerful concept in deep learning today [18]. In gesture recognition, attention focuses on specific data components, improving the modelling of spatial and temporal interactions by suppressing redundancy. In general, transformer models typically demand large-scale datasets for effective training. Nevertheless, the availability of such expansive datasets is often limited.

The existing literature on gesture recognition has not fully leveraged attention mechanisms, particularly in conjunction with the Skeleton modality. There is a pressing need for research that integrates attention mechanisms to dynamically focus on relevant spatial and temporal aspects of gesture sequences, especially utilizing the rich information provided by skeleton data. Investigating advanced fusion strategies that combine skeleton data with other modalities through attention mechanisms can potentially bridge this gap and lead to significant improvements in gesture recognition accuracy and robustness.

To the best of our knowledge, few research studies have utilised the skeleton modality for gesture recognition. The graph convolution network [21] might not be as useful when working with transfer learning that uses new models that have already been trained and transformers for gesture recognition [22], [23]. So, the goal of this study is to look into self-attention techniques with transfer learning using transformers that have already been trained to recognise dynamic gestures. The dataset used in this study is taken from Kaggle with a size of 2 GB. Each video contains 30 frames (or images) to recognise five gestures. Therefore, our contribution in pursuit of this objective is to:

- Investigate the performance of different gesture recognition models, such as MobileNet, VGG19 with attention, Densenet121 with attention, and Resenet50 with attention, based on pixel intensity or key points.
- Assess the effectiveness of selected feature extraction models in extracting relevant features from video frames and classifying gestures accurately using the video frame-based approach.
- Test how accurate and dependable skeleton-based gesture recognition is by using GRU, LSTM, and transformer models with skeleton data obtained from MediaPipe.
- Explore the performance of selected gesture recognition models as they undergo training and fine-tuning on relevant datasets while evaluating their ability to adapt to real-world situations.

In the following section, a review of dynamic gesture recognition literature is presented in Section II. After that, in the methodology section, along with the transfer learning models, machine learning (ML) and deep learning (DL) models that are commonly used in gesture recognition are also described in Section III. Then, the proposed methodology is outlined while providing insights into our experimental setup, dataset exploration, and the tools and resources utilised to conduct our research in Section IV. In Section V, the findings are presented, and the results are analysed in Section VI. Finally, the work is concluded in Section VII by summarizing key findings and potential future directions.

II. Literature Review

The focus of this paper is dynamic gesture recognition. Hence, static gesture recognition methods are not mentioned in this section.

Data acquisition is fundamental in gesture recognition research, with access to comprehensive datasets being critical. The Ego Gesture dataset, introduced in 2018, is a significant resource comprising over 24,000 RGB-D video samples and three million frames across 50 subjects. It includes 83 different types of static and moving gestures, making it one of the biggest egocentric gesture datasets made for interacting with wearable tech [24]. The NV Gesture database, established in 2019, offers a unique resource with multiple sensors and viewpoints in an indoor car simulator setting. It includes 25 gesture classes primarily designed for human-computer interfaces, totalling 1532 videos. These videos are weakly segmented, meaning gestures are not explicitly labelled within them, and are split into 1050 training videos and 482 test videos, each featuring one gesture [25]. The REHAP dataset stands out with over a million hand posture samples, divided into REHAP-1 and REHAP-2. REHAP-1 contains 600,000 images from 20 individuals, captured with a resolution of 160 × 120 in-depth using Time-of-Flight (ToF) sensors. In contrast, REHAP-2 has a higher resolution of 320 × 240 and includes RGB images. This dataset offers multi-viewpoint images, enhancing its versatility compared to others [26]. For dynamic hand gestures, two publicly accessible databases provided skeleton information: the DHG-14/28 Dataset and the SHREC'17 Track Dataset. The DHG-14/28 Dataset encompassed 14 hand gestures, captured five times by 20 volunteers, resulting in 2800 sequences [27]. In specific real-time scenarios, custom databases tailored to the environmental context have been utilised, as summarised in Table I.

A. Machine Learning Algorithms

Artificial neural network (ANN) is a computer model based on the biological neural networks in the brain. It is more of a framework than a fixed algorithm, and it can handle complex data by learning from a set of training examples how to do certain tasks.

Dynamic gesture recognition involves analyzing an image sequence captured continuously, and its recognition outcome relies not just on the hand's appearance but also on temporal features that characterize the hand's trajectory within the sequence, making dynamic gestures more diverse, expressive, and applicable compared to static gestures [17]. Different research studies have utilized various methods for dynamic gesture recognition, including the Hidden Markov Model and Dynamic Time Warping.

1) Hidden Markov model (HMM): In the dynamic gesture recognition task, each gesture category corresponds to an HMM. HMMs are traditional models used to solve problems based on time series or state sequences. Training involves dividing each gesture sample into categories and then using forward and backward algorithms to train a matching HMM model for each category. To produce the test sample, all HMM models are traversed to calculate their probability values [17].

2) Dynamic Time Warping (DTW): DTW is a measure of the similarity between two-time series of different lengths and was originally used for speech recognition. Dynamic time-based regularization measures the similarity between two videos. Even if two videos are similar, they may not be aligned in time, so alignment must be completed before comparing them. A dynamic time regularization algorithm was developed by Corradino to recognize dynamic movements [17].

B. Deep Learning Algorithm

Deep learning employs multilayer architectures to learn from data, providing accurate predictions. Research utilizing deep learning often extracts features directly, with common methods in gesture recognition including Convolutional neural networks (CNNs), Long short-term Memory networks (LSTMs), and Graph Convolutional neural networks (GCNs). Models and features such as modularity type used and the number of gestures of deep learning algorithms used in this domain are stated in Table II.

1) Convolutional Neural Networks: CNN, a neural network with specialised layers [41], plays a vital role in image (2D) and video (3D) analysis. Several studies [27] [14], [42], [30] leverage 3DCNN for recognition and classification. In [27], a novel approach combines geometry algorithms and deep learning for accurate hand gesture recognition (97.12% accuracy) compared to 2DCNN (64.28%). The study in [30] presents real-time fingertip detection and gesture recognition using RGB-D cameras and 3DCNN, achieving 92.6%. The study in [30], propose a framework that enhances unimodal networks with multi-modality knowledge for improved accuracy. The research in [31] adopt Faster-RCNN for tiny object detection, with a designed bi-stream attention module. The study in [34] introduce SEMN, focusing on skeleton edge movement for human action recognition through deep spatial-temporal blocks.

2) Long-short Term Memory Networks: LSTM, known for its cell state concept (alongside the hidden RNN state), excels in handling sequential data [41]. This architecture, featuring a forget gate, is widely employed in sequence analysis [22], [25]. [35], explore two CNN networks to model spatial and temporal information using RGB and optical flow images, leveraging LSTM's ability to tackle gradient disappearance. The research in [17] highlight LSTM's advantage in processing longer sequences compared to standard RNNs. The study in [18] introduced STSNN, comprising four modules: short-term sampling, feature extraction with ConvNet, long-range temporal feature learning with LSTM, and hand gesture classification (achieving 95.73% on the jester dataset).

3) Graph Convolution Neural Network: Graph CNNs, or Convolutional Graph Neural Networks, extend classical CNNs to analyze graph data like molecules, point sets, and social networks. They were applied to extract skeleton modality from RGB data in research papers [21], [37]. The study in [21], FGCN employs a multi-stage progressive approach via Feedback Graph Convolutional Networks for spatial-temporal feature extraction.The research in [37] introduces RS-GCN models, featuring a multi-stream graph convolutional network (GCN) to reduce noise and enhance discriminative features across skeleton joints for improved action model robustness.

C. Transfer Learning for Gesture Recognition

In recent years, transfer learning has gained significant traction in various research papers, addressing classification challenges by leveraging pre-trained models. Transfer learning involves training an agent on a source task and then using its learned features to improve performance on a target task [30]. The process often entails transferring parameters and models from a convolutional neural network trained on a large dataset to a smaller gesture dataset. Numerous research papers [38], [39], [9], [22], [23], [40] have explored different transformers for recognition. A transfer learning-based method for recognising gestures in study [27] that uses the AlexNet network model and convolution layer weight parameters from large datasets got very good results. In [38], a lightweight VGG16 feature extractor and Random Forest ensemble classifier were used to focus on VGG16 layers. Transfer learning is used to avoid underfitting, and a 99.89% accuracy rate is reached. The study in [38] presents a fine-tuned VGG19 model for static gesture recognition, combining multiple training stages. In study [9],

TABLE I. Gesture Recognition Datasets

Database	Samples	Labels	Subject	Scenes	Modalities	Task	Ref.
Ego Gesture	24,161	83	50	6	RGB-D	Classification detection	[28]
NV Gesture	1,352	25	—-	Multi	RGB-D	Classification+detection	[29], [26], [30]
DHG-H128	175	14	—-	——	SKELETON	Classification	[31]
REHAP	600,000	——	——	——-	RGB-D	Classification	[7]

TABLE II. Deep Learning Algorithm for Gesture Recognition

Model	Modality	Data	No. gestures	Stream	Fusion	Stage	Technique	ACU	Ref	Year
3DCNN	RGB-D	Manually	7	1	Decision	1	Geometric	92.60%	[27]	2020
3DCNN	RGB-D	Manually	7	1	Decision	1	Geometric	97.12	[14]	2020
3DCNN	RGB-D	Manually	7	1	Decision	1	Geometric	92.60	[32]	2020
3D-CNNs	RGB-D+OPTICAL FLOW	Ego-Gesture	50	1	Data-level+decision level	1	MTUT	92.48	[30]	2022
3D-CNNs	RGB-D	Fine Gym	99	1	Decision level	2	Heatmap	94.3%	[33]	2021
CNN	Skelton	Penn action	15	3	Feature level	1	Heatmap	98.19%	[34]	2021
Faster CNN-bi	RGB+estimate poes	DHG-H128	10	2	Feature-decision	2	——	92.4	[31]	2022
Lstm (GL-Lstm)	Human Skelton	NTU	60	1	Feature-level	2	gematric	98.6%	[35]	2020
Lstm (STSNN)	RGB+Optical flow	20N-gesture	27	3	Data level Feature level	2	———	95.73%	[18]	2021
LSTM Media Pipe	RGB	ASL	30	1	———	1	————-	99%	[27]	20
3DCNN-LSTM	RGB	22 participates	27	3	Feature level	2	———	93.95	[36]	2020
GCNN (FGCN)	Skelton	NTU-RGB+D	60	2	Decision	1	Zooming	96.25%	[21]	2020
GCNN (RC-GCN)	Skelton	NTU-RGB+D	60	3	Data level-Feature level	1	———	80%	[37]	2020
Alex _Net	RGB Static	Manually	5	1	SoftMax	1	——-	99%	[27]	2019
VGG16+RF	RGB Static	NUS hand	10	1	Decision	2	———	99.89%	[38]	2022
VGG19	RGB +RGB-D	ASL		2	Feature-Level	1	————	94.8%	[39]	2019
ALEX-NET	RGB	Manually	2	1	SoftMax	1	GMM	91%	[9]	2019
VGG19	RGB Static	LIS	26	1	SoftMax	1	Cross-Entropy	99%	[22]	2020
Dense Net	RGB	Manually	12	1	SoftMax	1	HOS/SVM	95.70%	[23]	2021
VGG19+ logistic regression	YouTube	YouTube	11	1	Feature _level	2	10-fold	98.49%	[40]	2021

used a deep CNN model and a transfer learning approach for driving-related activity recognition. The model segments raw RGB images using a GMM algorithm to improve identification accuracy. The study in [23] observes higher test accuracy on single-user datasets but recommends caution when interpreting these results. In study [27], MediaPipe hand landmarks were applied in addition to LSTMs for effective gesture recognition, achieving a high accuracy rate of 0.99.

Additionally, [43], utilised the Pilled dataset to train object detection methods like RetinaNet, SSD, and YOLO v3, with YOLO v3 demonstrating faster convergence and training time. MediaPipe, an open-source framework developed by Google, illustrated in Fig. 1, offers versatile machine-learning solutions for real-time pose, hand movement, and facial landmark detection [44]. MediaPipe's holistic pipeline is used to identify landmarks from the face, hands, and body pose, particularly for hand- and finger-tracking solutions [44]. MediaPipe Holistic Hands detects approximately 21 3D hand landmarks in real-time, combining a palm detection model with hand keypoint localization [44]. The framework is designed for processing perceptual data, including images, videos, and audio, using machine learning to achieve real-time hand tracking and gesture recognition.

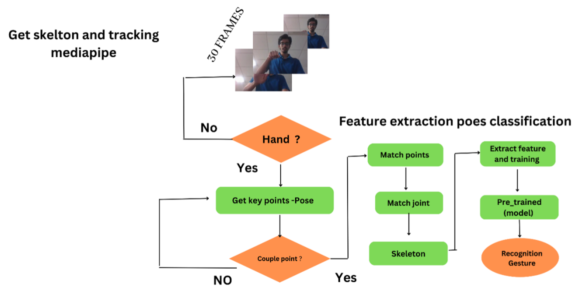


Fig. 1. Framework for MediaPipe.

D. Discussion

Deep learning has taken a prominent role in gesture recognition, particularly outperforming traditional machine learning methods. Skeleton-based models, favored for their robustness in dynamic and complex environments, have seen substantial adoption in computer vision, especially when coupled with deep learning techniques [21], [37]. Graph Convolutional Networks (GCNs) play a pivotal role in this context. Transformer-based models have shown their effectiveness, especially in large-scale databases [27]. While many studies employ transformers on RGB data with limited attention mechanisms, some explore the potential of attention transformers in skeleton data, as seen in by [23] where they extract skeleton data from RGB using MediaPipe. A noteworthy observation is that decision fusion at the last layer consistently outperforms data-level and feature-level fusion methods, achieving an accuracy of 96%, as illustrated in Fig. 2. Additionally, descent transfer learning has proven superior to models, as illustrated in Fig. 4. Hence, this result highlight its potential in enhancing gesture recognition performance

III. Methodology

A. Problem Formulation and General Framework

The overall framework consists of two key experiments. In the first experiment, we perform video frame-based analysis

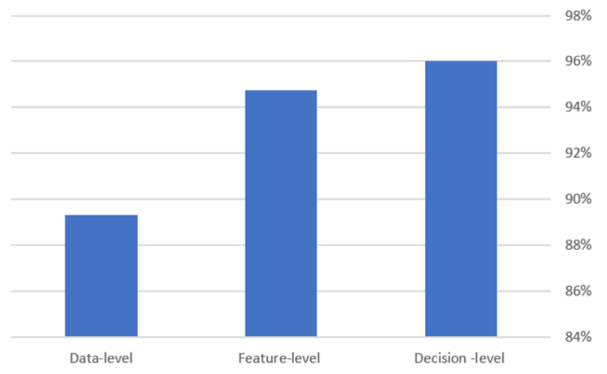


Fig. 2. Gesture recognition performance depends on the type of fusion.

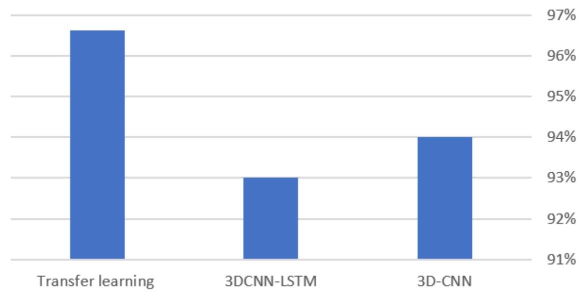


Fig. 4. Gesture recognition performance depends on the transfer learning used.

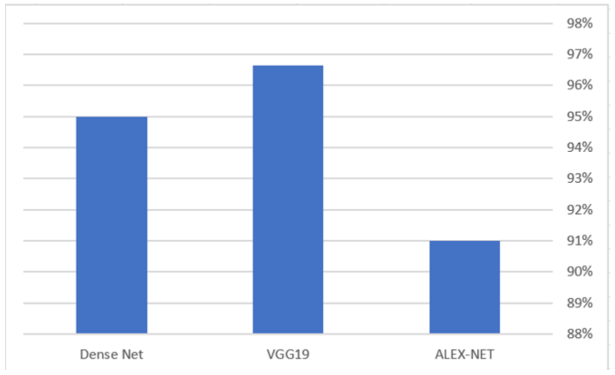


Fig. 3. Overarching framework for our proposed gesture recognition performance.

utilizing various feature extraction models, such as MobileNet, VGG19 (with attention), Densenet121 (with attention), and Resenet50 (with attention). The second experiment centers around skeleton-based gesture recognition, employing the MediaPipe library to process video frames. Within this experiment, three data modules—GRU, LSTM, and transformer—are utilized. These models undergo training and fine-tuning with suitable datasets, and their performance is evaluated based on accuracy metrics.

These models are pre-trained and utilize the MediaPipe library for the detection of hand landmarks (see Fig. 6) and poses, facilitating the process of gesture recognition. The architectural layout of these proposed models is illustrated in Fig. 2. The model workflow typically involves a sequence of raw images, usually consisting of around 30 frames.

Fig. 3 illustrates the overarching framework for our proposed gesture recognition approach. The process begins with initial preprocessing steps, where frames from the database are resized. Subsequently, these resized frames are passed through the MediaPipe library, which extracts landmarks from RGB models. The database is then split into training and testing subsets. The general framework includes three different model experiments, each of which includes an evaluation stage.

In the first experiment, the initial model is trained using transfer learning with VGG19, which does not incorporate attention mechanisms. The architecture of the transfer learning VGG19 is outlined above, and Fig. 11 provides a visual representation of the model pipeline. Accuracy is evaluated following this experiment.

The second experiment involves training the database with the DenseNet121 architecture, which incorporates attention mechanisms. The DenseNet121 architecture is described above, and it consists of dense blocks, transition layers, ReLU activation functions, 3D-Averagepooling layers, and fully connected layers. Fig. 12 illustrates the layer pipeline for this model. Testing is carried out, and accuracy is measured.

The final experiment trains Model 3, which is passed through DenseNet121 with an added self-attention block. The self-attention architecture is detailed in this paper, and a step-by-step implementation is presented.

1) Insert feature map
2) Initialize weight
3) Derive key, query, and value
4) Calculate attention scores
5) Calculate SoftMax
6) Multiply scores with values
7) Sum weighted values to get output.
8) Fully connected and SoftMax test and compare these Models.

B. Proposed Transformer based on Pixel Intensity and Skeleton

1) Feature Extractor Layers: The feature extractor in our study is initialised with pre-trained weights obtained from the ImageNet dataset [45]. We have removed the top layer of the network for our specific task. Our input images are standardized to a fixed size, with a height of 224 pixels and a width of 224 pixels, and they are represented in the RGB color space (3 channels).

For feature extraction, we employed three distinct techniques utilizing well-established convolutional neural network architectures: VGG19, DenseNet121, and ResNet50. These architectures have demonstrated strong performance in various computer vision tasks and were chosen to capture diverse image features relevant to our research.

2) Feature Extraction by VGG19:

- Convolutional layer with 64 filters of size 3*3, stride of 1, and padding of 1.

- Convolutional layer with 64 filters of size 3*3 stride of 1, and padding of 1.
- A max pooling layer with assize 2*2 and stride of 2 is applied to reduce the spatial dimensions by half.
- A convolutional layer with 128 filters of the size of 3*3, a stride of 1, and padding of 1.
- Convolutional layer with 128 filters of size 3*3, a stride of 1, and padding of 1.
- Max pooling layer with a pool size of 2*2 and stride of 2 is applied to reduce the spatial dimensions by half.
- Convolutional layer with 256 filters of size 3*3, stride of 1, and padding of 1.
- Convolutional layer with 256 with filters of size 3*3, stride of 1, and padding of 1.
- Convolutional layer with 256 filters of size, stride of 1, and padding of 1. A
- Convolutional layer with 256 filter size 3*3, stride of 1, and padding of 1.
- Max pooling layer: Another max pooling layer with a pool size of 2*2 and stride of 2 is applied to reduce the spatial dimensions by half.
- Convolutional layer with 512 filters of size 3*3, stride of 1, and padding of 1.
- Convolutional layer with 512 filters of size 3*3, stride of 1, and padding of 1.
- Convolutional layer with 512 filters of size 3*3, stride of 1, and padding of 1.
- Convolutional layer with 512 filters of size 3*3, stride of 1, padding of 1.
- Max pooling layer with a pool size of 3*3 and stride of 2 is applied to reduce the spatial dimensions by half.

3) Feature Extractor by DenseNet121: The second-stage feature extractor utilized in our study employs DenseNet121, a convolutional neural network architecture introduced by [46]. This architecture, comprising a total of 6.9 million parameters, has consistently demonstrated state-of-the-art performance on several image classification benchmarks, including the renowned ImageNet dataset.

DenseNet121 is characterized by a unique structure consisting of multiple convolutional layers organized into three dense blocks. Within each dense block, the convolutional layers are intricately connected through dense connections, promoting rich feature reuse and gradient flow throughout the network. Following each dense block, the feature maps undergo processing through a transition layer. These transition layers serve a dual purpose: they reduce the spatial dimensions of the feature maps while also compressing their depth

- The first layer is a convolutional layer with 64 filters of the size of 7*7 and a stride of 2. This layer applies filters to the input image to extract low-level features.
- Bach normalization layer: A batch normalization layer is added after the convolutional layer to normalize the output and improve training stability.
- Activation layer: An activation function (ReLU) is added after the batch normalization layer to introduce nonlinearity into the model.
- Max pooling layer: A max pooling layer with a pool size of 3*3 and stride of 2 is added after the activation function to reduce the spatial dimensions of the feature maps.
- Dense block 1: The first dense block consists of multiple layers that are densely connected to each other. Each dense block contains serval bottleneck layers, which are composed of batch normalization, ReLU, and convolutional layers with smaller filters (1*1 and 3*3). The output from each bottleneck layer is concatenated with all previous outputs in the dense block.
- Transition block 1: A transition block is added after each dense block to reduce the number of feature maps and spatial dimensions before passing them on to the next dense block. The transition block consists of batch normalization, ReLU, and convolutional layers with filter size 1*1 for compression, followed by an average pooling operation with pool size 2*2.
- Dense block 2-4: Three more dense blocks are added after transition block 1, each consisting of multiple bottleneck layers that are densely connected.
- transition block 2: Another transition block is added after the last dense block to further reduce the number of feature maps and spatial dimensions.
- Global average pooling layer: A global average pooling layer is added after the final transition block to reduce the spatial dimensions of the feature maps to a single value per feature map.

4) Features an Extractor by Resenet50: In our third step, we harness the power of ResNet50 as our chosen feature extractor. ResNet50 is a distinguished member of the ResNet family, celebrated for its profound depth and consistent excellence in the realm of computer vision.

With a network architecture comprising 50 layers, ResNet50 stands as a testament to the innovation brought about by the ResNet family. Its design incorporates skip connections, or residual connections, which fundamentally address the issue of vanishing gradients in exceptionally deep neural networks. These residual connections empower the training of remarkably deep networks, while still preserving their accuracy and effectiveness The ResNet50 architecture consists of:

- A Convolutional layer with 64 filters and a kernel size of 7*7.
- A max pooling layer with a pool size of 3*3 and stride of 2.
- A series of residual blocks (16 in total), each containing multiple convolutional layers with different filter

sizes and numbers, as well as skip connections that bypass some of the convolutional layers.

- A global average pooling layer that averages the feature maps across spatial dimensions.

5) Features Extractor by MidiPipe: In the fourth step of our feature extraction process, we employ the use of Medipipe to extract key points. Developed by Google, MediPipe presents a versatile pipeline for real-time computer vision and machine learning applications.

MediPipe operates with a modular approach, where different graphs are utilized for specific tasks, each equipped with its own set of parameters, methods, and output configurations tailored to the task's requirements. The functionality and utility of Medipipe depend on the particular graph chosen for the task at hand. Detailed information about available graphs, their usage, and the associated input/output streams for each task can be found in the comprehensive MediPipe documentation.

6) The Mechanism of Transformer Encoder: By employing a Transformer Encoder, our model architecture adeptly captures extensive dependencies within sequential tensors. This Transformer Encoder is structured with a combination of self-attention mechanisms and feed-forward layers.

1) Positional Embedding layers: The inclusion of positional information within the layers enhances the model's performance, particularly in sequence-related tasks such as time series analysis. Position embeddings serve the crucial role of enabling the model to differentiate the order and placement of elements within the sequence. This additional context empowers subsequent layers to more effectively capture patterns and dependencies. To calculate the position encoding for a sequence of frames, we employ the following formula:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$
$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i+1}{d}}}\right) \quad (1)$$

   Where:
   - pos is the position of the frame in the sequence (0-indexed).
   - i is the index of the dimensionality of the embedding vector.
   - d represents the total dimensionality of the embedding vector.

2) Multi-head attention layers: The input embedding, enriched with positional encoding, undergoes a series of layers featuring self-attention mechanisms. During the generation of the output representation, the model assesses the significance of various elements through self-attention. The multi-head attention mechanism empowers the model to simultaneously focus on distinct segments of the tensor sequence. Given a sequence of tensor frames with a specific length and dimensional embedding, the multi-head attention mechanism computes a weighted sum of values based on the similarity between keys and queries. This output is subsequently processed through a feed-forward neural network. Mathematically, the multi-head attention mechanism can be expressed as follows:

$$\begin{aligned} Q &= X \cdot W_q \\ K &= X \cdot W_k \\ V &= X \cdot W_v \end{aligned} \quad (2)$$

   Following this, the positional encodings are propagated through a series of stacked layers, each housing self-attention mechanism. During the generation of the output representation, the model meticulously assesses the significance of individual elements through its self-attention mechanism. What sets it apart is the multi-head attention mechanism, a pivotal component that enables the model to simultaneously focus on various segments within the tensor sequence.
   For a given sequence of tensor frames, characterized by a specific length and dimensional embedding, the multi-head attention mechanism performs a critical operation—it computes a weighted sum of the values, leveraging the similarity between keys and queries to do so. This calculated output is then channeled through a feed-forward neural network, further enhancing the model's capacity to capture intricate patterns and dependencies within the data. Mathematically, the multi-head attention mechanism can be expressed as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \ldots, head_h)W^O,$$
$$\text{where } head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \quad (3)$$

   The multi-head attention is a hyperparameter. Each attention head will compute self-attention scores for each position in the input sequence using different learned weights as follows.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{dk}}\right)V \quad (4)$$

   The results from each attention head are merged and then processed through a linear layer, yielding the ultimate output of the multi-head attention layer. Throughout the training process, the model refines its understanding by iteratively adjusting the weights for each attention head and the linear layer using backpropagation. These dynamically learned weights empower the model to simultaneously focus on various aspects of the input sequence, fostering the computation of intricate relationships across different positions within the sequence.

3) Feed-Forward Neural Network: Following the self-attention layers, each position within a sequence undergoes individual processing via a feed-forward network, consisting of fully connected layers with non-linear activation functions. The class encapsulates the essential operations required to encode an

input sequence using self-attention and feed-forward mechanisms within the transformer model. Subsequently, the encoder's output is subjected to a global max-pooling layer, followed by the application of a dropout rate of 0.5. Finally, a dense layer with softmax activation is employed, generating probabilities for class recognition. The model is compiled using the Adam optimizer with a learning rate set to 0.0001, and it employs a categorical cross-entropy loss function for training.

IV. Experimental Setup

A. Dataset

The dynamic gesture database is an open-source dataset of a size 2GB. The file contains a 'train' and a 'test' folder with two CSV files for the two folders. These folders are in turn divided into subfolders where each subfolder represents a video of a particular gesture. Each subfolder, a video, contains 30 frames (or images). Note that all images in a particular video subfolder have the same dimensions, but different videos may have different dimensions. Specifically, videos have two types of dimensions - either 360x360 or 120x160 (depending on the webcam used to record the videos). Hence, you will need to do some pre-processing to standardize the videos.

- **Thumbs Up:** Increase the volume.
- **Thumbs down:** Decrease the volume.
- **Left swipe:** 'Jump' backwards 10 seconds
- **Right swipe:** 'Jump' forward 10 seconds
- **Stop:** Pause the movie

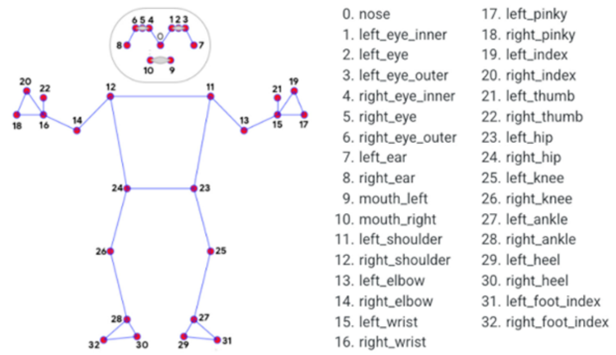


Fig. 5. Body landmarks extracted from media pipe library.

B. Baseline

The two models will be used to be compared next term. Conv3D and 3DCNN+LSTM have the accuracy of Conv3D and 3DCNN with LISTM. For the research paper model, the first model is Conv3D applied to RGB-D and achieved 81% accuracy. The second model, 3DCNN+LSTM also applied to RGB-D and achieved 70% accuracy; it was applied to the same Database as the research paper.

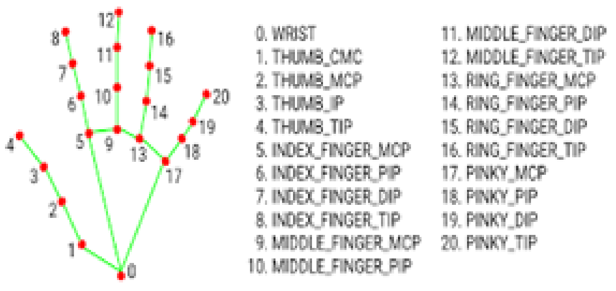


Fig. 6. Hand landmarks extract from mediapipe library.

C. Resources, Materials and Tools

In this research, we employ a range of valuable resources. Our experimental platform of choice is Google Colab, where we utilize Python 3.10. The hardware configuration features an Intel Core i10 processor complemented by 24GB of RAM. Our deep learning endeavors are powered by the TensorFlow library, Version 3.9, enabling us to construct and train models effectively. TensorFlow offers a versatile suite of components, including layers, optimizers, evaluation metrics, and essential elements like transformer encoders and MLP layers. Among its many components, we harness MultiHeadAttention, Dropout, LayerNormalization, Conv1D, Dense, and others to tailor our models precisely to our needs. Furthermore, we incorporate the Medipipe library, an essential tool for extracting human poses from frames, enhancing the depth and richness of our research.

D. Evaluation

In our assessment of the proposed dynamic gesture recognition system, we employ a comprehensive range of evaluation measures based on the four primary outcomes used to evaluate classifiers: true positives, false positives, true negatives, and false negatives. These measures allow us to gauge the system's effectiveness thoroughly.

One of the key metrics used in evaluating the system's performance is accuracy, also known as the recognition rate. To determine accuracy, we divide the number of correctly classified instances of a particular gesture by the total number of instances of that specific gesture. This fundamental measure provides insights into the system's ability to correctly identify and classify gestures within the given dataset as show in Eq. (5).

$$Accuracy = \frac{\text{Correctly Recognized Samples}}{\text{Total Samples}} \times 100 \quad (5)$$

Another key evaluation metric is the F1-score, derived from the harmonic mean of precision and recall, which assigns equal importance to both metrics in its calculation. It falls within the range of 0 to 1, with 1 representing an ideal score, signifying flawless precision and recall. A higher F-score signifies superior performance, reflecting excellence in both precision and recall. The formula for F1-Score is shown in Eq. (6). Note that the Recognition Rate is the total number of correctly identified probe images divided by the total number of probe images.

TABLE III. Right Arm & Hand pose landmarks used in GRU, LSTM, Transformer

MediaPipe Skeltone	Landmarks Area	Landmarks	Dimensions	Attributes
Body Pose	Right arm, wrist, 3 fingers	12, 14, 16, 18, 20, 22	x, y, z	18
Body Pose	Right shoulder & elbow	(12, 14, 24), (14, 12, 16)	radians*	2
Hand Pose	Full Hand	1 to 21	x, y, z	63
				83

TABLE IV. Right Arm & Hand pose landmarks used in Transformer (Mid body)

Media Pipe Skeltone	Landmarks Area	Landmarks	Dimensions	Attributes
Body Pose	Left & right arm and mid-body	11 to 24	X, Y, Z	56
Body Pose	Right/left shoulders & elbows	(12,14,24)(14,12,16) (12,11,15),(11,13,23)	radians*	4
				60

$$F1 - score = 2 \times \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

V. Experiments and Results

The feature extractor is initialized with pre-trained weights obtained from the ImageNet dataset, with the top layer removed. The input shape is defined as (image height = 224, width = 224, channels = 3). Hyperparameters are predefined parameters configured prior to the learning process, influencing how the model transfers knowledge from one task to another. In this study, we partitioned our dataset into distinct training and testing subsets, encompassing 85% (663 instances) and 15% (100 instances) of the video data, respectively. Table V shows the hyperparameters used for the experiment.

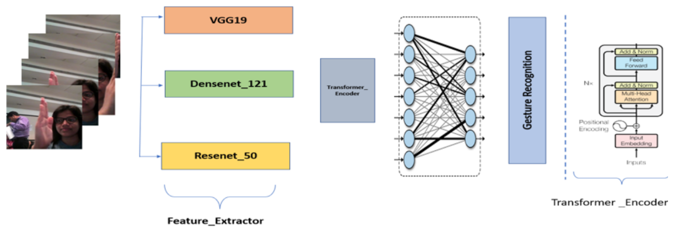
TABLE V. Hyperparameters and Values

Hyperparameter	Value
Optimizer	Adam
Activation functions	RELU
Last Activation functions	Softmax
Training data	85%
Test data	15%
Data Augmentation	Rotation
Learning rate	0.001
Number of Epochs	100

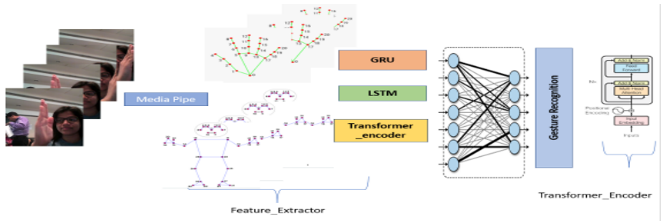
A. Gesture Recognition by Frame-based (pixel intensity)

1) Transfer Learning VGG19 With Transformer Encoder: Our approach commenced with the integration of a pre-trained VGG19 model from the Keras deep learning library is illustrated in Fig. 7. To tailor this model for video sequence analysis, we meticulously adjusted the input layer to accommodate sequences of 30 frames, each characterized by dimensions of 224 $\times$ 2224 pixels. To streamline computation, we strategically froze all layers of the VGG19 architecture except for the final layers, which processed the 512 feature maps extracted from each of the 30 frames.

Subsequently, a transformer encoder is incorporated as block sourced from TensorFlow. This encoder block consists of multiple layers, integrating self-attention mechanisms and feedforward neural networks. The preprocessing entailed the incorporation of positional embeddings, followed by multi-head attention mechanisms with a dropout rate of 0.3. Our experimentation involved varying the number of heads across settings, including 1, 2, 4, 6, and 8. The sequence length was consistently set to 512, and we adjusted the dense dimension accordingly to enhance model performance.



(a) Transfer learning with transformer encoder



(b) Skeleton-based gesture recognition with transformer encoder.

Fig. 7. General framework for dynamic gesture recognition.

Normalization techniques are applied after the multi-head attention layers to ensure stability during training. Following this, we channeled the encoder's output through a global max-pooling layer, introducing a dropout rate of 0.5 for regularization purposes. This was concluded with a dense layer employing softmax activation to yield probabilities for class recognition. To compile the model, the Adam optimizer is used with a learning rate of 0.0001. The categorical cross-entropy loss function is harnessed. This meticulously engineered architecture and training process aim to optimize our model's performance for the task at hand.

2) Transfer Learning DensNet121 With Transformer Encoder: We applied a similar technique to DenseNet121. The feature extractor was constructed using Keras' DenseNet implementation, which had been pre-trained on the ImageNet dataset. We removed the final classification layer to fine-tune the model and utilized global max pooling as the feature extractor's output by setting the pooling parameter to 'max.' Input size was set again to 224×2224 pixels.

We then processed this data, which had dimensions of (663, 30, 1024), by passing it through a custom position embedding layer in TensorFlow. This layer accepts sequences of frames and applies positional embeddings. The positional embeddings were learned through embedding layers with an input dimension of 1024. Next, the data was processed through a custom 'Transformer Encoder' layer in TensorFlow. This layer incorporated a multi-head attention mechanism with varying numbers of heads (1, 2, 4, 6, and 8) and key dimensions set to 4. Afterward, residual connections were introduced using 'LayerNormalization.'

Following the 'Transformer Encoder,' the output underwent further processing. It passed through a feedforward neural network with two 'Dense' layers, with another round of residual connections using 'LayerNormalization.' The output of the 'Transformer Encoder' was further refined through a 1D global max-pooling layer, a dropout layer with a rate of 0.6, and a dense output layer with softmax activation.

To complete the model, we compiled it using the Adam optimizer and utilized a sparse categorical cross-entropy loss function. The model's performance was evaluated using metrics such as F1 score, accuracy, and others.

3) Transfer Learning Resenet50 With Transformer Encoder: To process sequential frames through ResNet-50, a deep convolutional neural network implemented using the TensorFlow deep learning framework, we followed a similar systematic approach. The ResNet-50 model, pre-trained on the ImageNet dataset and with its top classification layer removed, was employed. We specified 'average' pooling and set the input shape as (224, 224, 3). Each frame resulted in 2048 features after ResNet feature extractor.

Subsequently, the data underwent position embedding through custom layers, a critical step in distinguishing positions within the sequence and capturing temporal dependencies. Following this, the data was directed to a transformer encoder, implemented as a custom Keras layer. This encoder featured multi-head attention with variable numbers of heads (1, 2, 4, 6, 8) and key dimensions set at 4, along with the incorporation of residual connections through 'LayerNormalization.'

The primary objective of the transformer encoder was to encode input sequences by concurrently attending to all positions, thereby learning representations that encompass both local and global dependencies. The multi-head attention mechanism facilitated the capture of diverse relationships across different positions within the sequence. The dense projection introduced non-linear transformations before undergoing further normalization.

The output from the 'TransformerEncoder' layer was subsequently subjected to a 1D global max-pooling layer, followed by a dropout layer (rate = 0.6), and ultimately a dense output layer with softmax activation. For model compilation, we employed the Adam optimizer and utilized a sparse categorical cross-entropy loss function. The model was rigorously evaluated using F1 score, and accuracy.

B. Gesture Recognition by GRU, LSTM, and Transformer based on the Skeleton

1) Media Pipe with GRU: Leveraging Mediapipe library, we utilize pre-trained models designed for the estimation of human body pose based on video frames. Each frame of the video undergoes processing, resulting in the creation of a set of key points that collectively represent the skeleton. These key points effectively capture the spatial configuration of the body joints.

Our model architecture consists of multiple GRU (Gated Recurrent Unit) layers, which are followed by dense layers. The first GRU layer encompasses 64 units and returns sequences. We employ the Rectified Linear Unit (ReLU) activation function in this layer. The second GRU layer, consisting of 128 units, similarly returns sequences. The third GRU layer comprises 64 units.

After the GRU layers, we introduce three dense layers, each with varying numbers of units: 64, 32, and the final layer with five units dedicated to gesture recognition. The ultimate dense layer utilizes the softmax activation function, producing the output layer responsible for gesture classification.

2) Mediapipe with LSTM: The model receives as input a sequence of video frames and employs a structured architecture comprising five LSTM units. Each LSTM unit is designed with two LSTM layers: one dedicated to processing upper features and the other for lower features. These LSTM units effectively capture the spatial configuration of the body joints.

Within this model, multiple LSTM layers are employed, with each LSTM unit contributing to the overall sequence processing. The outputs from these LSTM units are concatenated and subsequently passed through a fully connected layer, culminating in the recognition of the final five gestures. The ultimate dense layer employs the softmax activation function to generate the output layer, responsible for gesture classification.

3) MediaPipe with a Transformer: This experiment is designed to assess the performance of a transformer-encoder model, leveraging skeleton key point coordinates extracted from video frames. The model architecture has a transformer-encoder that works with frames in a sequence and then a multi-layer perceptron (MLP) that is connected to it through a feed-forward layer. Key point features, derived from joints, are extracted using the Medipipe library, as illustrated in Fig. 5 and 6. Details regarding the landmark features are illustrated in Table III and IV.

Then, the data goes through a position embedding layer and is put through multi-head attention mechanisms that have certain hyperparameters, such as a head size of 2 and a range of heads (1, 2, 4, 6, 8) for each attention block. Layer normalization is applied to enhance model stability and convergence.

Following the attention blocks, feed-forward layers with two dense layers employing ReLU activation functions are applied. Global Average Pooling is employed to reduce data dimensionality while retaining critical information by aggregating output representations across time steps. Subsequently, a series of fully connected layers follow the pooling layer.

Finally, the model's output tensor, featuring a shape corresponding to five distinct classes, is generated through a dense layer utilizing softmax activation. Training is accomplished using the Adam optimizer with a learning rate set to 0.001, and the model employs sparse cross-entropy loss as its objective function.

VI. Results and Discussion

The experiment results reveal diverse levels of performance across the tested models. Accuracy, F1 score (indicating recognition rates for each class), and model parameters are key aspects to consider when evaluating these models. Parameter is a transformer model that refers to that model learns from the training data. these parameters are adjusted during the training. process to minimize the error loss function and improve the model's performance. Increasing the number of parameters increases the risk of overfitting where the model becomes too specialized to the training data performance poorly on unseen data.

A. Comparative between Attention based on (pixel-intensity) based Video and Key Point based Skeleton

According to the Table VI in our comparative analysis, it's evident that model performance varies significantly based on data type and architecture. Models that rely on pixel intensity, such as DenseNet121 with attention, VGG19 with attention, and Resenet50 with attention, excel at capturing intricate spatial details from video frames. However, they tend to require a higher number of parameters and achieve relatively lower accuracy when compared to skeleton-based models, which include LSTM, GRU, and the transformer. Interestingly, the skeleton-based models achieve comparable accuracy with significantly fewer parameters. Among the skeleton-based models, the transformer with attention stands out due to its remarkable ability to capture long-range dependencies and concentrate on relevant skeleton data. Consequently, it exhibits high accuracy, making it a promising choice for tasks requiring precise recognition and classification

B. Comparative between Densnet-121 and Resenet50 and VGG19 for Extract Features

In Table VII, the experiment results vividly illustrate the varying performance of the evaluated models. MobileNet stands out with an accuracy of 0.8312 and an F1 score of 0.831. On the other hand, VGG19 with attention exhibits an accuracy of 0.73 and an F1 score of 0.7254. Notably, VGG19 lacks an explicit attention mechanism, which hinders its capability to focus on pertinent information while suppressing noise or irrelevant features within the input data.

DenseNet121 with attention emerges as a top performer, achieving an impressive accuracy of 0.91 and an F1 score of 0.9094. This excellence can be attributed to the combination of DenseNet's robust feature extraction capabilities with attention mechanisms that emphasize relevant information through dense connections. This approach allows the model to leverage information from earlier layers to compute subsequent layer features, facilitating the capture of both low-level and high-level features and enriching information throughout the model.

Similarly, ReseNet50 with transformer encoder achieves remarkable accuracy (0.88) by leveraging the advantages of residual connections and attention mechanisms to capture intricate frame details effectively. This combination enhances the model's ability to understand and interpret complex visual data.

In summary, the experiment results underscore the significant impact of attention mechanisms on model performance, with DenseNet121 and ReseNet50 demonstrating the potential of combining robust feature extraction and attention to achieve superior accuracy.

C. Comparative between Attention by Sequence (LSTM and GRU) and Parallel Attention by (Transformer Encoder)

The GRU model attained an accuracy of 0.7778 and an F1 score of 0.7718, with a parameter count of 0.140 million. In contrast, the LSTM model demonstrated superior performance, achieving an accuracy of 0.968 and an F1 score of 0.966, while having a parameter count of 0.00945 million. Notably, the GRU model exhibited the lowest accuracy among the three models, which can be attributed to its relatively simpler architecture and fewer parameters. This simplicity may hinder its ability to capture complex patterns and long-range dependencies present in video skeleton data.

Conversely, the transformer model outperformed both the GRU and LSTM models with an accuracy of 0.993 and an impressive F1 score of 0.992. The transformer's strength lies in its ability to weigh the importance of different tensor sequences, enabling it to focus on relevant features and effectively capture long-range dependencies. Understanding the temporal relationships between various joint points is particularly helpful in the context of skeleton data from videos (see Table VIII) using the transformers' attention mechanism. Unlike recurrent models such as GRU and LSTM, transformers can process tensor sequences in parallel, facilitating faster computation and leveraging this advantage for enhanced performance.

D. Comparative between Skeleton-based Transformer-Encoder

Table IX illustrates the impact of various body poses and hand poses on attention mechanisms. When using a long tensor sequence that includes the middle body and arms, the model encountered challenges in accurately predicting hand poses, leading to lower accuracy. In contrast, focusing the attention on the right arm and full hand resulted in a more precise and effective recognition rate. Additionally, the first two models employed a single long tensor (frames*frames) for each video, while the third model processed 30 frames with feature-length, leading to enhanced accuracy in the attention model.

Fig. 8 shows the accuracy of different models over 100 epochs, categorized by architecture and data type. The plot showcases how model accuracy evolves with increasing

TABLE VI. Comparative between Attention based on (pixel-intensity) based Video and Key Point based Skeleton

Model	Data Type	Pretrained Checkpoint	Parameters (M)	Input Size	Learning Rate (LR)	Accuracy
MobileNet	Video Frames	mobilenet	4.103	224x224	0.001	0.825
VGG19 + Transformer	Video Frames	Vgg19	1.075	224x224	0.001	0.730
DenseNet + Transformer	Video Frames	Densenet121	4.247	224x224	0.001	0.910
ResNet + Transformer	Video Frames	Resenet50	16.883	224x224	0.001	0.880
GRU	Skeleton from Video	None (GRU layers)	0.140	83 sk. plt	0.001	0.772
LSTM	Skeleton from Video	None (LSTM layers)	0.009	83 sk. plt	0.001	0.966
Transformer Enconder	Skeleton from Video	None (Attention Layers)	0.024	83 sk. plt	0.001	0.982

TABLE VII. Comparative Analysis of Video Frames-based Models for Gesture Recognition

Model	Data Type	Pretrained-Checkpoint	Parameters (M)	Input Size	LR	Accuracy	F1
MobileNet	Video Frame	Mobilenet	4.103	224x224	0.001	0.8312	0.831
VGG with attention	Video Frame	Vgg19	1.075	224x224	0.001	0.73	0.7254
DenseNet with attention	Video Frame	Densenet121	4.247	224x224	0.001	0.91	0.9094
ResNet with attention	Video Frame	Resenet50	16.883	224x224	0.001	0.88	0.8782

TABLE VIII. Comparative Performance of Models Trained on Skeleton Data from Videos

Model	Data Type	Pretrained-Checkpoint	Parameters (M)	Input Size	LR	Accuracy	F1
GRU	Skeleton from Video	None (GRU layers)	0.140	83 sk. plt	0.001	0.7778	0.7718
LSTM	Skeleton from Video	None (LSTM layers)	0.00945	83 sk. plt	0.001	0.968	0.966
Transformer	Skeleton Video	None (Attention Layers)	0.02440	83 sk. plt	0.001	0.9815	0.981

TABLE IX. Comparative Results for Different Models on Body-Pose and Hand-Pose Attributes

Model	Attributes	Hand-Pose Attribute	Accuracy	F1
Transformer with Atta (middle_body n arms)-Long tensor input to attention layer	60*	0	0.779	0.7767
Transformer with Attan (right-arm+full_hand)-long tensor input to attn layer	20*	63	0.981	0.981
Transformer with attn (right_arm-full_hand)-frame-wise input to tensor layer	20*	63	0.993	0.992

epochs. In the video-based gesture recognition category, MobileNet achieves 0.825 accuracy, while VGG19 with a transformer encoder achieves 0.73. DenseNet121 with a transform encoder stands out with an impressive 0.91 accuracy, and ResNet with a transformer encoder follows closely at 0.88. In the skeleton data category, including GRU, LSTM, and transformer encoder models, which do not use pre-trained weights, we observe varying performances. GRU achieves 0.732 accuracy with 83 skeleton plots (sk. plt), LSTM achieves 0.8738 accuracy, and the transformer encoder shines with an impressive 0.99 accuracy.

Fig. 9 centered on the analysis of skeleton-based video data. The relative effectiveness of the GRU and LSTM models becomes apparent when compared to the transformer encoder, as evident from the distinctive green line on the graph. These models do not rely on pre-trained weights and exhibit variability in terms of the number of layers and parameters. Specifically, with 83 skeleton plots (sk. plt) as input data, the GRU model achieves an accuracy of 0.732, while the LSTM model attains a higher accuracy of 0.8738. The transformer encoder surpasses them all with a remarkable accuracy of 0.99

Fig. 10 depicts the accuracy attained by three distinct models in the recognition of body-pose attributes across 100 training epochs. Notably, among the three models, the transformer encoder configured with frame-wise input for ‘right arm and full hand’ consistently achieves the highest level of accuracy. The graph visually highlights an initial surge in accuracy followed by a sustained upward trend

E. Comparative Results for Different Models of Action Recognition

Table X provides accuracy scores for four distinct models (VGG19, DenseNet121 with transformer encoder, ResNet with transformer encoder, and MobileNet with transformer encoder) across various gesture categories.

1) Stop Gesture: Both the DenseNet121 and ResNet models with attention achieved a perfect accuracy rate of 1.000 in recognizing the “Stop Gesture.” This indicates that these models accurately identified the stop gesture in all instances, showcasing the effectiveness of the attention mechanism in capturing relevant features and patterns.

2) Thumbs Down:: The VGG19 model exhibited the lowest accuracy at 0.4375 when recognizing the “Thumbs Down” gesture. This suggests that the model faced challenges in capturing the distinctive features or patterns associated with thumbs-down gestures. It implies that VGG19’s architecture

TABLE X. Comparative Results for Transfer Learning Models with Transformer Encoder on Gesture Recognition

Action	VGG19	Densenet121 with Attention	ResNet with Attention	MobileNet
Left-swipe	0.7222	0.7778	0.7222	0.7879
Right-swipe	0.7826	0.913	0.8261	0.8235
Stop-Gesture	0.8122	1	1	0.7742
Thumbs-Up	0.8095	0.9524	0.9048	0.9697
Thumbs-Down	0.4375	0.875	0.9375	0.7931

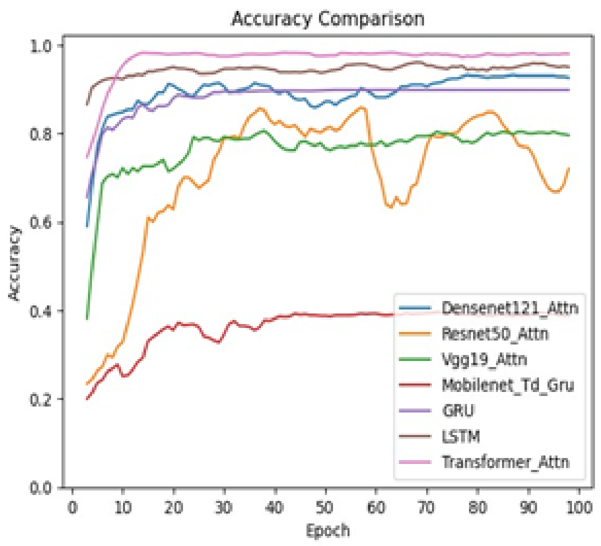


Fig. 8. Performance over 100 epochs using transfer learning with attention mechanisms.

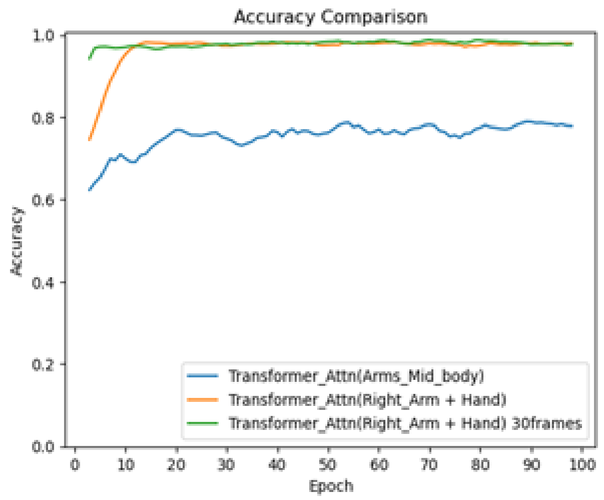


Fig. 10. Performance over 100 epochs, focusing on changes in joint training for the skeleton.

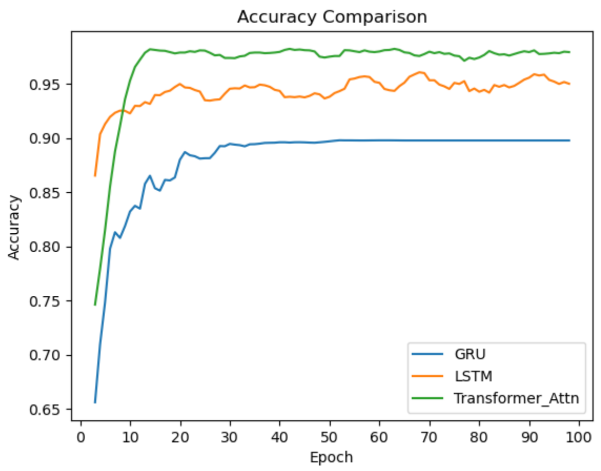


Fig. 9. Performance over 100 epochs using attention-based joint extraction from frames in the skeleton.

might not be sufficiently intricate to capture the nuances of this specific gesture.

For the skeleton-based models (GRU, LSTM, and Transformer), the highest accuracy was achieved by both LSTM and Transformer models in recognizing the “Stop Gesture,” with a perfect accuracy score of 1.000 as shown by Table XI. Similarly, both LSTM and the Transformer demonstrated perfect accuracies of 1.000 in identifying the “Thumbs Up” gesture. However, the GRU model exhibited the lowest accuracy at 0.5517 when recognizing the “Thumbs Down” gesture. This variance in performance suggests that while LSTM and Transformer models excel at capturing long-term dependencies, the GRU model may struggle to capture complex temporal patterns, despite its recurrent neural network nature similar to LSTM.

Table XII demonstrates variations in model performance driven by specific attributes related to body and hands. Overall, the model with attention focusing on the right arm and full hand with frame-wise input to the tensor layer outperformed both the model with attention on the middle body and arms and the model with attention on the right arm with full-hand long tensor input to attention.

Fig. 11 shows the model performance based on pixel intensity for four different models: DenseNet121 with transformer encoder, ResNet50 with transformer encoder, VGG19 with transformer encoder, and MobileNet with GRU. The graph highlights the consistently strong performance of DenseNet121 with attention across various gestures, consistently achieving high accuracy levels. ResNet with attention also demonstrated

TABLE XI. COMPARATIVE RESULTS FOR TRANSFORMER ENCODER BASED ON SKELETON ON GESTURE RECOGNITION

Action	GRU	LSTM	Transformer
Left swipe	0.8889	0.9222	0.9944
Right swipe	0.9677	0.9513	0.9957
Stop Gesture	0.7576	1	1
Thumbs Up	0.7273	0.9467	1
Thumbs Down	0.5517	0.9818	0.9688

TABLE XII. ACCURACY PERFORMANCE FOR EACH CLASS DEPENDING ON KEY-POINT

Action	Transformer with Attn (middle_body n arms)	Transformer with Atta (right-arm+full_hand)	Transformer with Attention (right-arm+full_hand) frame-wise input to tensor layer
Left-Swipe	0.7833	0.9667	0.9667
Right-Swipe	0.9913	0.9739	1
Stop-Gesture	0.7182	1	1
Thumbs-Down	0.7667	0.9857	0.9952
Thumbs-Up	0.5688	0.9751	1

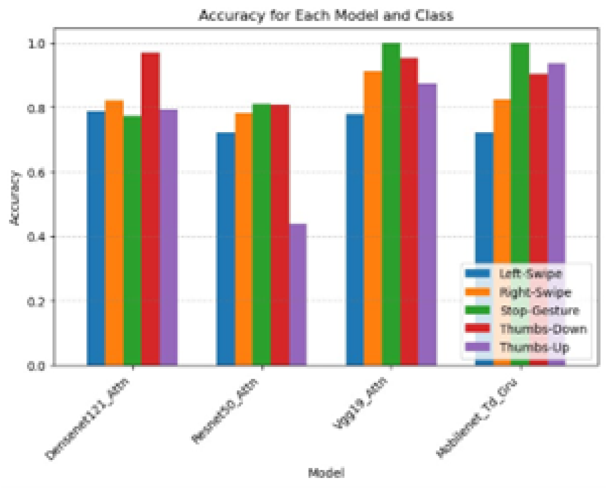


Fig. 11. Performance based on pixel intensity.

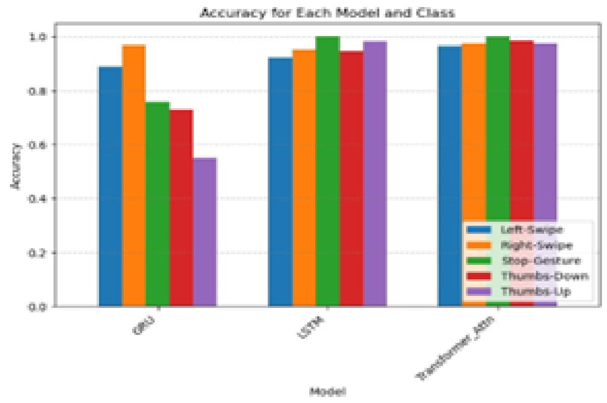


Fig. 12. Performance of models on recognizing the five different gestures based on the skeleton (key points).

competitive results across most gestures. MobileNet's performance varied depending on the gesture, while VGG19 showed moderate performance.

Fig. 12, presents the performance of GRU, LSTM, and Transformer Encoder models for recognizing five different gestures based on skeleton (key points). The graph reveals that the Transformer Encoder achieved the highest accuracy of 0.99 for the 'Left swipe' gesture, closely followed by LSTM at 0.9222, and GRU at 0.88. For the 'Right swipe' gesture, the Transformer model demonstrated a remarkable accuracy of 0.9957, surpassing LSTM with 0.9513 and GRU with 0.9677. Both LSTM and Transformer Encoder achieved perfect accuracy scores of 1, while GRU achieved a slightly lower accuracy of 0.7576. Additionally, in recognizing the 'Thumbs-Up' gesture, the Transformer model achieved an accuracy of 1, outperforming GRU with 0.7273 and LSTM with 0.9467.

Fig. 13 presents a comprehensive view of gesture recognition by different models, showcasing varying levels of accuracy across different gestures. In the first set of models (VGG19, DenseNet121, ResNet, and MobileNet with GRU), we notice fluctuations in accuracy among various gestures. For instance, DenseNet121 with attention excels in recognizing "Stop Gesture" and "Thumbs Up," while MobileNet performs exceptionally well in identifying "Left Swipe."

The second set of models (GRU, LSTM, and Transformer) also demonstrates variation in accuracy across gestures. Notably, the Transformer model consistently exhibits high accuracy, highlighting its effectiveness in gesture recognition. However, it's worth noting that GRU and LSTM models achieve relatively high accuracy for specific gestures like "Right Swipe" and "Stop Gesture." Therefore, the choice of the model should align with the specific gesture recognition task at hand.

F. Comparative Num-head-att on Pixel Intensity and Key Point of Coordinate Joint

The Tables (XIII, XIV, XV, XVI) demonstrate that the selection of multi-head attention configurations can substantially influence model performance. The choice between using pixel intensity or joint key points depends on the specific architecture and data type. For instance, when achieving high performance, using eight heads is essential for pixel intensity, whereas only two heads are needed for joint key points.

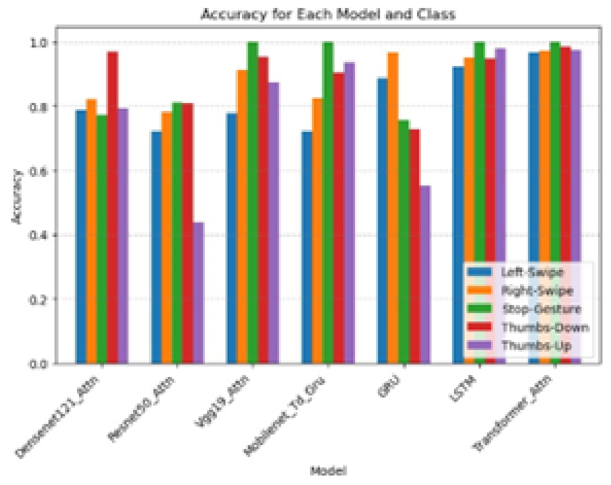


Fig. 13. Comparative analysis of model performance for five different gestures based on both pixel intensity and skeleton (key point) data across various models.

TABLE XIII. Comparative when Different Multi-head Attention is used on Transformer-Encoder by DenseNet121 Transfer Learning

Multi-head Attention Par	Parameter	Acc(%)
1	4,247,561	82
2	8,444,937	88
4	16,859,689	84
6	33,629,193	90
8	33,629,193	96

VII. Conclusion

Dynamic gesture recognition is a key enabler for touchless interaction systems, offering numerous benefits such as natural and intuitive user experiences, enhanced hygiene and safety, and improved accessibility. By leveraging this technology, developers can create more engaging, efficient, and user-friendly systems across various domains, from consumer electronics and smart homes to healthcare and industrial applications.

This paper conducts a comprehensive review of prior research in gesture recognition within the domains of machine learning and deep learning. It scrutinizes these studies based on modalities, the number of streams, stages employed, and algorithms utilized. The primary focus of this paper is to compare the impact of attention mechanisms on performance, particularly concerning skeleton modality. Evaluation metrics include accuracy, recall, and precision. Our study aspires to achieve superior performance compared to existing research in this area.

In our investigation, we demonstrate the potential of employing pre-trained models and transformer-based architectures for both video frame and skeleton-based gesture recognition. Notably, attention mechanisms applied to keypoint coordinates yield enhanced performance. While our current study focuses on sequence data in the spatial domain, future research may delve into the frequency domain. The objective of this paper is to provide a comparative analysis of gesture recognition using video frames and skeletons within a transformer framework. Our findings enable researchers to make informed decisions tailored to their specific needs, considering the strengths and weaknesses of each model.

TABLE XIV. Comparative when Different Multi-head Attention is used on Transformer-Encoder by DenseNet121 Transfer Learning ResNet50

Multi-head Attention Par	Parameter	Acc(%)
1	16,883,721	60
2	33,667,081	89
4	67,233,801	94
6	100,800,621	90
8	134,367,241	92

TABLE XV. Comparative when Different Multi-head Attention is used on Transformer-Encoder by Transfer Learning VGG19

Multi-head Attention Num	Parameters	Acc(%)
1	1,075,209	69
2	2,125,321	59
4	4,225,545	78
6	6,325,759	76
8	8,425,993	82

This study contributes to the field of action recognition by highlighting the effectiveness of different models and elucidating their architectural distinctions. Future research avenues could explore hybrid models that combine video frames (pixel intensity) and skeleton information (key point coordinates). Additionally, investigating approaches within the frequency domain, as an alternative to the spatial domain, holds promise for advancing gesture recognition technology. Another future direction is to recognize gestures under various conditions, by using data augmentation techniques to simulate different lighting conditions, backgrounds, and noise levels. This helps the model. Also, integrate data from multiple modalities (e.g., RGB, depth, infrared, and skeleton data) to provide a more comprehensive understanding of the gesture. This helps the model to be more robust to variations in any single modality.

References

[1] B. Van Amsterdam, I. Funke, E. Edwards, S. Speidel, J. Collins, A. Sridhar, J. Kelly, M. J. Clarkson, and D. Stoyanov, "Gesture recognition in robotic surgery with multimodal attention," *IEEE Transactions on Medical Imaging*, vol. 41, no. 7, pp. 1677–1687, 2022.

[2] R. A. Salvador and P. Naval, "Towards a feasible hand gesture recognition system as sterile non-contact interface in the operating room with 3d convolutional neural network," *Informatica*, vol. 46, no. 1, 2022.

[3] M. Al-Hammadi, G. Muhammad, W. Abdul, M. Alsulaiman, M. A. Bencherif, T. S. Alrayes, H. Mathkour, and M. A. Mekhtiche, "Deep learning-based approach for sign language gesture recognition with efficient hand gesture representation," *IEEE Access*, vol. 8, pp. 192 527–192 542, 2020.

[4] B. Hu and J. Wang, "Deep learning based hand gesture recognition and uav flight controls," *International Journal of Automation and Computing*, vol. 17, no. 1, pp. 17–29, 2020.

[5] S. Shriram, B. Nagaraj, J. Jaya, S. Shankar, and P. Ajay, "Deep Learning-Based Real-Time AI Virtual Mouse System Using Computer Vision to Avoid COVID-19 Spread," 2021. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC8560261/

[6] M. Oudah, A. Al-Naji, and J. Chahl, "Hand gesture recognition based on computer vision: a review of techniques," *journal of Imaging*, vol. 6, no. 8, p. 73, 2020.

[7] K. K. Verma, B. M. Singh, and A. Dixit, "A review of supervised and unsupervised machine learning techniques for suspicious behavior recognition in intelligent surveillance system," *International Journal of Information Technology*, vol. 14, pp. 1–14, 2019.

TABLE XVI. Comparative when Different Multi-head Attention is used on Transformer-Encoder on the Skeleton (Right Arm with Full Hands) Frame-wise Input to Tensor Layer

Multi-head Attention Num	Parameters	Acc(%)
1	31,881	0.993
2	59,686	0.993
4	115,296	0.991
6	170,906	0.993
8	226,516	0.980

[8] O. Sangjun, R. Mallipeddi, and M. Lee, "Real time hand gesture recognition using random forest and linear discriminant analysis." in *HAI*, 2015, pp. 279–282.

[9] Y. Xing, C. Lv, H. Wang, D. Cao, E. Velenis, and F.-Y. Wang, "Driver activity recognition for intelligent vehicles: A deep learning approach," *IEEE transactions on Vehicular Technology*, vol. 68, no. 6, pp. 5379–5390, 2019.

[10] A. R. Elshenaway and S. K. Guirguis, "On-air hand-drawn doodles for iot devices authentication during covid-19," *IEEE Access*, vol. 9, pp. 161 723–161 744, 2021.

[11] N. Mohammadzadeh, M. Gholamzadeh, S. Saeedi, and S. Rezayi, "The application of wearable smart sensors for monitoring the vital signs of patients in epidemics: a systematic literature review," *Journal of ambient intelligence and humanized computing*, vol. 14, pp. 1–15, 2020.

[12] S. N. R. Kantareddy, Y. Sun, R. Bhattacharyya, and S. E. Sarma, "Learning gestures using a passive data-glove with rfid tags," in *2019 IEEE international conference on RFID technology and applications (RFID-TA)*. IEEE, 2019, pp. 327–332.

[13] S. Ahmed, K. D. Kallu, S. Ahmed, and S. H. Cho, "Hand gestures recognition using radar sensors for human-computer-interaction: A review," *Remote Sensing*, vol. 13, no. 3, p. 527, 2021.

[14] B. Hu and J. Wang, "Deep learning based hand gesture recognition and uav flight controls," in *2018 24th International Conference on Automation and Computing (ICAC)*, 2018, pp. 1–6.

[15] O. Güler and İ. Yücedağ, "Hand gesture recognition from 2d images by using convolutional capsule neural networks," *Arabian Journal for Science and Engineering*, vol. 47, no. 2, pp. 1211–1225, 2022.

[16] W. Zhang and J. Wang, "Dynamic hand gesture recognition based on 3d convolutional neural network models," in *2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*. IEEE, 2019, pp. 224–229.

[17] S. Yuanyuan, L. Yunan, F. Xiaolong, M. Kaibin, and M. Qiguang, "Review of dynamic gesture recognition," *Virtual Reality & Intelligent Hardware*, vol. 3, no. 3, pp. 183–206, 2021.

[18] W. Zhang, J. Wang, and F. Lan, "Dynamic hand gesture recognition based on short-term sampling neural networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 1, pp. 110–120, 2020.

[19] A. Ulhaq, N. Akhtar, G. Pogrebna, and A. Mian, "Vision transformers for action recognition: A survey," 2022.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[21] H. Yang, D. Yan, L. Zhang, Y. Sun, D. Li, and S. J. Maybank, "Feedback graph convolutional network for skeleton-based action recognition," *IEEE Transactions on Image Processing*, vol. 31, pp. 164–175, 2021.

[22] V. J. Schmalz, "Real-time italian sign language recognition with deep learning," in *CEUR Workshop Proceedings*, vol. 3078. CEUR Workshop Proceedings, 2022, pp. 45–57.

[23] J. de Lope and M. Graña, "Deep transfer learning-based gaze tracking for behavioral activity recognition," *Neurocomputing*, vol. 500, pp. 518–527, 2022.

[24] Y. Zhang, C. Cao, J. Cheng, and H. Lu, "Egogesture: A new dataset and benchmark for egocentric hand gesture recognition," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1038–1050, 2018.

[25] O. Köpüklü, A. Gunduz, N. Kose, and G. Rigoll, "Real-time hand gesture detection and classification using convolutional neural networks," in *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, 2019, pp. 1–8.

[26] Z. Cao, Y. Li, and B.-S. Shin, "Content-adaptive and attention-based network for hand gesture recognition," *Applied Sciences*, vol. 12, no. 4, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/4/2041

[27] X. Wu, X.-r. Song, S. Gao, and C.-b. Chen, "Gesture recognition based on transfer learning," in *2019 Chinese Automation Congress (CAC)*. IEEE, 2019, pp. 199–202.

[28] Y. Zhang, C. Cao, J. Cheng, and H. Lu, "Egogesture: A new dataset and benchmark for egocentric hand gesture recognition," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1038–1050, 2018.

[29] O. Köpüklü, A. Gunduz, N. Kose, and G. Rigoll, "Real-time hand gesture detection and classification using convolutional neural networks," 2019.

[30] M. Abavisani, H. R. V. Joze, and V. M. Patel, "Improving the performance of unimodal dynamic hand-gesture recognition with multimodal training," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1165–1174.

[31] Q. Gao, Y. Chen, Z. Ju, and Y. Liang, "Dynamic hand gesture recognition based on 3d hand pose estimation for human–robot interaction," *IEEE Sensors Journal*, vol. 22, no. 18, pp. 17 421–17 430, 2021.

[32] D.-S. Tran, N.-H. Ho, H.-J. Yang, E.-T. Baek, S.-H. Kim, and G. Lee, "Real-time hand gesture spotting and recognition using rgb-d camera and 3d convolutional neural network," *Applied Sciences*, vol. 10, no. 2, p. 722, 2020.

[33] H. Duan, Y. Zhao, K. Chen, D. Lin, and B. Dai, "Revisiting skeleton-based action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2969–2978.

[34] H. Wang, B. Yu, K. Xia, J. Li, and X. Zuo, "Skeleton edge motion networks for human action recognition," *Neurocomputing*, vol. 423, pp. 1–12, 2021.

[35] Y. Han, S.-L. Chung, Q. Xiao, W. Y. Lin, and S.-F. Su, "Global spatio-temporal attention for action recognition based on 3d human skeleton data," *IEEE Access*, vol. 8, pp. 88 604–88 616, 2020.

[36] L. Gionfrida, W. M. Rusli, A. E. Kedgley, and A. A. Bharath, "A 3dcnn-lstm multi-class temporal segmentation for hand gesture recognition," *Electronics*, vol. 11, no. 15, p. 2427, 2022.

[37] Y.-F. Song, Z. Zhang, C. Shan, and L. Wang, "Richly activated graph convolutional network for robust skeleton-based action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 5, pp. 1915–1925, 2020.

[38] E. L. R. Ewe, C. P. Lee, L. C. Kwek, and K. M. Lim, "Hand gesture recognition via lightweight vgg16 and ensemble classifier," *Applied Sciences*, vol. 12, no. 15, p. 7643, 2022.

[39] R. G. Crespo, E. Verdú, M. Khari, and A. K. Garg, "Gesture recognition of rgb and rgb-d static images using convolutional neural networks," *IJIMAI*, vol. 5, no. 7, pp. 22–27, 2019.

[40] T. Ahmad, J. Wu, I. Khan, A. Rahim, and A. Khan, "Human action recognition in video sequence using logistic regression by features fusion approach based on cnn features," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 11, 2021.

[41] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, and V. Zocca, *Python Deep Learning: Exploring deep learning techniques and neural network architectures with Pytorch, Keras, and TensorFlow*. Packt Publishing Ltd, 2019.

[42] A. Mujahid, M. J. Awan, A. Yasin, M. A. Mohammed, R. Damaševičius, R. Maskeliūnas, and K. H. Abdulkareem, "Real-time hand gesture recognition based on deep learning yolov3 model," *Applied Sciences*, vol. 11, no. 9, p. 4164, 2021.

[43] L. Tan, T. Huangfu, L. Wu, and W. Chen, "Comparison of retinanet, ssd, and yolo v3 for real-time pill identification," *BMC medical informatics and decision making*, vol. 21, pp. 1–11, 2021.

[44] B. Subramanian, B. Olimov, S. M. Naik, S. Kim, K.-H. Park, and J. Kim, "An integrated mediapipe-optimized gru model for indian sign language recognition," *Scientific Reports*, vol. 12, no. 1, p. 11964, 2022.

[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[46] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.