Tugay MANDAL, et. al. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 14, Issue 3, March, 2024, pp: 65-75

RESEARCH ARTICLE OPEN ACCESS

3D Game Objects Action Recognition With 3D Lenet 5

Tugay MANDAL*, Sina APAK**
**(Department of Computer Science, İstanbul Aydin University, Türkiye)*
*** (Department of Computer Science, İstanbul Aydin University, Türkiye)*

ABSTRACT
This article presents a novel approach for 3D game object action recognition using a customized 3D LeNet-5 architecture. In response to the increasing complexity of modern gaming environments, our proposed model is designed to effectively capture spatial and temporal features that are crucial for accurate action recognition. Utilizing a diverse dataset, our proposed methodology exhibits state-of-the-art performance in recognizing a wide range of in-game actions. Through comparative analysis, we emphasize the computational efficiency and accuracy advantages of our proposed 3D PSO-LeNet-5 model over existing methods. This research not only advances the field of 3D game object action recognition but also demonstrates the potential applicability of our proposed model in video analysis and surveillance, leading to enhanced user experiences in interactive and immersive gaming scenarios.
Keywords-3D LeNet-5, Coordinate, Game Object, PSO,

Date of Submission: 09-03-2024 Date of acceptance: 23-03-2024

I. INTRODUCTION

In recent years, the concepts of gesture recognition have been an area of research. The advances in technology such as memory and processor capabilities, and the concept of metaverse at hand, increase the urge to advance in the area. More often 3D environments such as video games, virtual reality, or in the fields of video analysis and surveillance, there has been a necessity to delve beyond mere motion detection towards identifying specific movements based on user gestures. This article aims to focus on detecting which movements are being performed based on user hand gestures. The 3D Convolutional Neural Network (3D CNN) is a widely used algorithm in image processing. This article will utilize the 3D LeNet 5 algorithm based on this approach. Additionally, the Particle Swarm Optimization (PSO) algorithm will be applied to enhance the hyperparameter optimization. To ensure compatibility with real-time data acquisition during the study, the dataset used will not be sourced from a public database but rather collected during the research process. A known game engine was employed in the process of obtaining this dataset.

1.1 RELATED WORK

While investigating the problem of how to recognize gesture recognition, several proposals of methodologies and optimization techniques which include the use of sensors and use of other technologies such as cameras and LEDs should be considered.

Vaitkevičius et al.[1] introduced a method for recognizing gestures in virtual reality using the Leap Motion device. This device captures images of a user's hands using specialized cameras and LED technology, creating a spatial representation of hand movements. From these images, details like hand coordinates, angles of rotation, or palm center locations are extracted and stored in a database. By employing a Hidden Markov Classification algorithm, the system can recognize different gestures with approximately 86% accuracy based on this collected information. This approach allows for accurate gesture identification within virtual reality environments using data derived from the Leap Motion device.

Li et al.[2] introduced an innovative gesture recognition algorithm centered around fusing image information. They began by delving into the fundamentals of VR environments, highlighting the intricate connections between users and these virtual spaces, as well as how VR influences individuals. Their approach involved devising a model for combining information from multiple sensors. This method, which fused data from various sensors, attained an impressive success rate of 96% in recognizing gestures. Notably, it outperformed several other machine learning methods in terms of

recognition speed, showcasing its efficiency in real-time gesture recognition within VR settings.

Yu et al.[3] introduced a gesture recognition technique tailored for VR environments, leveraging multi-scale fusion and two forms of image fusion. Their method incorporates the powerful deep learning model YOLOv3, allowing for robust analysis of gestures within this setting. Notably, their approach demonstrated superior performance when compared to other methods such as R-CNN and SSD. While the proposed method offered swift identification of gestures, there were instances where the recognition accuracy fell short. Despite its ability to quickly identify gestures, it encountered challenges in achieving precise recognition across all scenarios.

Lv et al.[4] introduced an innovative hand gesture recognition algorithm that relies on data collected from a data-glove. Their method involves extracting distinctive features from this glove data and employing a Support Vector Machine (SVM) for classifying these features to predict the corresponding gesture label. Despite its efficiency, the algorithm encountered limitations in recognizing gestures with varying periodic patterns. This means that gestures with different rhythms or timing weren't accurately identified, presenting a challenge for the algorithm in distinguishing such variations.

A new deep learning model is proposed by Ur et al. [5] which learns spatial-temporal features of dynamic hand gesture sequences in a video-stream. The setup involves using a 3D-CNN, which stands for a three-dimensional convolutional neural network. Following this, there's an LSTM network, short for long short-term memory. This part of the architecture is capable of learning patterns over time, making it great for capturing both spatial (related to space or location) and temporal (related to time) features within the video frames. All of this happens even when dealing with challenging aspects like complex backgrounds and varying lighting conditions.

Park et al. [6] introduced a novel approach to enhance the accuracy of hand gesture recognition techniques by combining 2D-FFT and CNN. They initially pre-processed the collected data using 2D-FFT, 8bit-Normalization, and Resizing methods to optimize the input for CNN classification. This processed data was fed into both the Double Parallel CNN model and the Two-stage Serial CNN model for simulation. In their evaluation, they compared the performance of their proposed model with several established models by inputting the pre-processed data. While it's not guaranteed that all preprocessing methods would improve accuracy, their experiments revealed that integrating 2D-FFT preprocessing with the CNN deep learning model consistently increased the classification accuracy of hand gestures.

Unlike the proposals that have been considered, the approach that is proposed in this paper, although it is possible to use technologies such as VR and its controllers, only the coordinate data that is given by the technology is enough. Other factors are handled by the given game engine and other software which will be further discussed in the paper. The approach is useful for identifying gestures in 3D video game environments but is not limited to it, as it could be used in other areas such as VR and video recognition.

II. METHODOLOGY

In this study, input data from game engine objects are collected with position reactions (X, Y, and Z). This collected data is then trained and tested based on a three-dimensional convolutional neural network to detect the movement of game objects' hands. When preparing the data, the five most commonly used gestures in games under different conditions and movement speeds will be labeled. After labeling this data, the 3D Lenet-5 model with six hidden layers as the output layer for optimization of the number of hidden layers, weights, and nodes of the 3D LeNet-5 model using the Particle Swarm Optimization (PSO) algorithm in case of overfitting and underfitting problems during the model training will be fed. Fig 1represents this research methodology.

2.1 3D LeNet 5

Yann LeCun created LeNet-5 back in 1998 as a deep learning framework primarily designed to identify handwritten and machine-printed characters. This architecture, depicted in Fig 2, comprises an input layer, hidden layers, and an output layer. Within this structure, the hidden layers play a crucial role in extracting and categorizing the distinctive features of objects. Meanwhile, the output layer produces an integer that signifies the identified category or class.

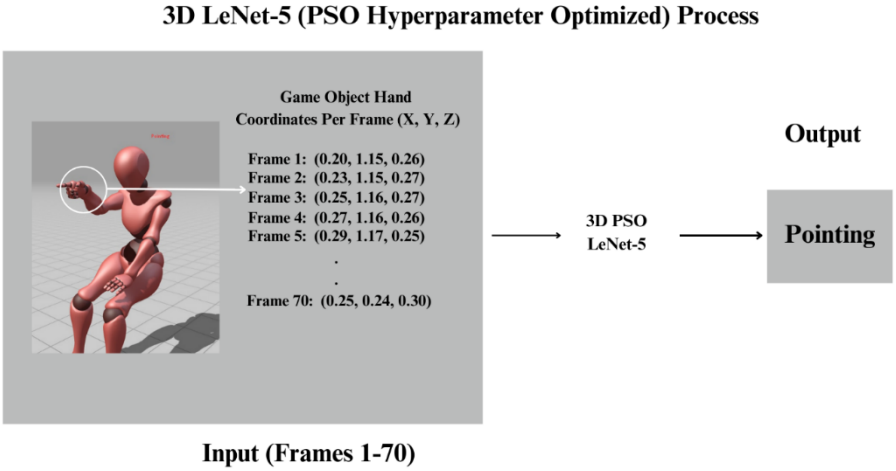


Fig 1.Basic process of recognizing gestures.

In a convolutional neural network, the hidden layer typically comprises Convolutions, Sub-sampling, and fully connected layers. In the model illustrated in Fig 2, there are two convolution layers labeled C1 and C3. C1 uses 32 convolutional kernels of size 5 × 5, while C3 employs 64 convolutional kernels of the same size. During image recognition, these kernels convolve over 5 × 5 regions of the input image, generating feature maps that capture robust features, especially those resilient to changes in scale. When the convolution step size is set to 1, the convolutional operation, expressed in equation (1), involves the convolutional kernel moving across the input image in 5 × 5 regions, extracting and emphasizing specific features within each region.

$$H_{i,j}^{s} = \sum_{m=0}^{k} \sum_{n=0}^{k} w_{m,n}^{s} x_{m+i,n+j} + b^{s}$$

Equation(1)

Equation (1) outlines the convolutional operation where X represents an image with a depth of 1 and dimensions $u \times v$. W signifies the convolutional kernel sized $k \times k$. S corresponds to different feature graphs produced by multiple convolutional kernels [7]. Within these feature graphs, $H_{i,j}^{s}$ denotes the specific element at row i, column j in the feature graph H^{s} matrix. Additionally, b^{s} represents the bias value associated with each convolutional kernel. CNNs employ pooling layers like S2 and S4, enabling dimension reduction and eliminating redundant features through successive layers. This process fosters a degree of translational invariance within the extracted features. The fully connected layers, F5 and F6, resemble traditional multi-layer perceptron neural networks. Here, the image features derived from convolution and pooling layers are combined with weights in these fully connected layers. The subsequent classification of features is

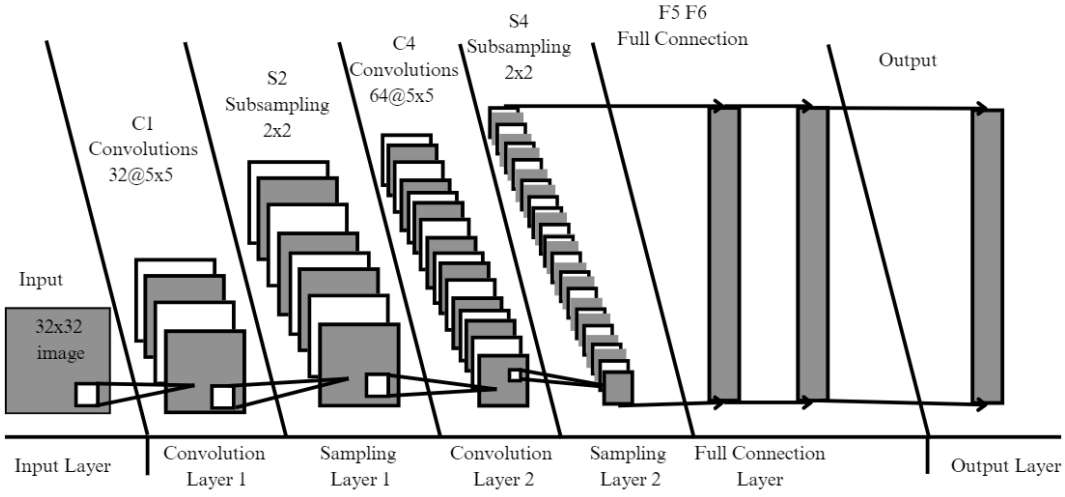


Fig 2. The LeNet-5 convolutional neural network model

achieved by applying the Sigmoid function[8][9].

The proposed 3D LeNet architecture is structured with two 3D convolution layers, followed by pooling, flattening, fully connected layers, and an output layer. It operates on a 3D volume of (x, y, z) coordinate data with dimensions 70 x 70 x 70 x 3. In the initial convolution layer, Rectified Linear Unit (ReLU) activation functions are employed, and 32 kernels, each sized (3, 3, 3), are utilized. This generates (68, 68, 68, 32) feature maps by convolving the input with these filters. After activation, an average pooling process with a kernel size of 2 x 2 x 2 is applied, halving the spatial dimensions. The subsequent convolution phase involves 64 kernels, also sized (3, 3, 3), with ReLU activation. Following pooling, the output shape becomes (16, 16, 16, 64) in terms of feature maps. After the second average pooling step, the extracted features are flattened into a vector, creating an input of size 262,144 for the fully connected layers. For the fully connected layers, ReLU activation functions are applied with 120 neurons. Finally, the output layer comprises a single neuron employing the sigmoid function for classification [10]. To optimize hyperparameters, Particle Swarm Optimization (PSO) is employed, a process that will be elaborated upon later in the methodology. The entire process is represented in Fig 3.

2.2 PSO

Basically, PSO utilizes a swarm of candidate solutions, referred to as particles, moving in a search space based on a formula. Particle movements are guided by both the swarm's best-known position and their own. Improved positions influence the swarm's movements, with the process repeated in the hope of discovering a satisfactory solution, though not guaranteed. [11].

The operational design of PSO incorporates five principal tenets inspired by the following animal behaviors:

- Proximity: This principle highlights the algorithm's proficiency in carrying out fundamental computations related to time and space.
- Stability: The swarm maintains its behavioral traits consistently even when the environment undergoes alterations.
- Quality: It demonstrates a strong sensitivity to shifts in environmental conditions, promptly adjusting its behavior in response to changes in quality.
- Diverse response: It showcases the algorithm's ability to adapt without constraints to varying environmental dynamics.
- Adaptability: The algorithm possesses the capability to assess whether environmental changes necessitate a change in behavior, exhibiting discernment in its responses.

Fig 4 presents a schematic of the basic PSO algorithm, outlining the seven primary steps executed in each iterative process.

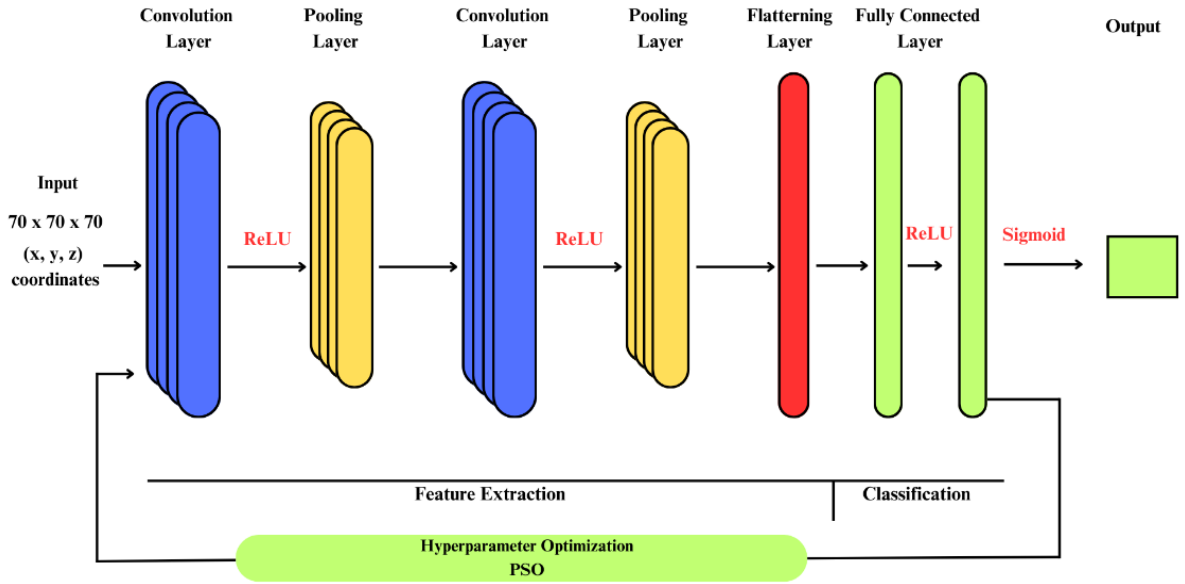


Fig 3. 3D LeNet with PSO process

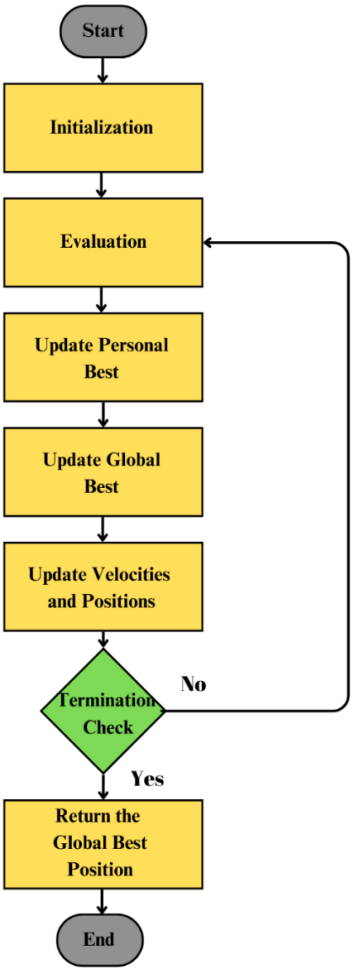


Fig 4. Steps of the PSO Procedure

- Initialization: Begin by setting up a group of particles, where each particle symbolizes a potential solution. These particles have positions representing specific values in the search space and velocities determining their changes in each step.
- Fitness Assessment: Evaluate the fitness of each particle. This fitness could be measured by validation accuracy or as its inverse if minimizing.
- Update Personal Best: If a particle's new fitness score surpasses its previous best, update its personal best to this new score.
- Update Global Best: Identify the particle with the best fitness in the entire swarm and mark its fitness score and values as the global best.
- Adjust Velocities and Positions: Update particles' velocities and positions based on their current velocities, distances from their personal best, and the global best. This update involves a mix of randomization to maintain diversity and specific parameters (cognitive and social) controlling the influence of personal and global bests on new velocities.
- Termination Check: Check if termination conditions are met, such as reaching a maximum iteration limit or achieving a satisfactory fitness score. If met, stop the algorithm and return the global best fitness score; otherwise, loop back to the fitness evaluation step.[12].

The process can be defined as follows as Shami et al defined [13]: Each particle i possesses a current velocity vector $V_i = [V_{i1}, V_{i2}, \ldots, V_{id}]$ and a current position vector $x_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$, where D is the number of dimensions. The PSO process starts by randomly initializing V_i and x_i . Then, in each iteration, the best position that has been found by particle i$Pbest_{id} = [Pbest_{i1}, Pbest_{i2}, \ldots, Pbest_{id}]$ and the best position that has been found by the whole swarm $G_{best} = [G_{best_1}, G_{best_2}, \ldots, G_{best_d}]$ guide particle i to update its velocity and position by Equation (2) and Equation (3):

$$V_{id}(t+1) = V_{id}(t) + c_1 r_1 \left(Pbest_{id}(t) - x_{id}(t)\right) + c_2 r_2 \left(G_{best_d}(t) - x_{id}(t)\right)$$

Equation (2)

$$x_{id}(t+1) = x_{id}(t) + V_{id}(t+1)$$

Equation (3)

where c_1 and c_2 are the cognitive and social acceleration coefficients, and r_1 and r_2 are two uniform random values generated within [0, 1] interval [14].

The hyperparameter optimization process briefly is as follows as Kilichev et al implemented [15] : PSO is implemented with a swarm of 20 particles over 10 cycles, using cognitive and social constants set to 2. Hyperparameter boundaries are defined, and particles are randomly initialized within this space with zero velocities. Personal and global best positions are established, and particle fitness is evaluated by training a CNN model. Fitness values are compared with personal and global bests, updating them if surpassed. Particle velocities and positions are updated using PSO equations with an inertia weight of 0.5. The new positions are constrained within defined bounds. After a set number of iterations, the PSO outputs hyperparameters corresponding to the global best position, achieving the highest validation accuracy. [16][14].

After the PSO process, the final 3D LeNet 5 model using these parameters is constructed. The model is trained using a training dataset with a specified number of epochs with the best batch sizes. The dataset is obtained through a game engine and

split into training and test sets. After the training phase, finally the model is evaluated using the test dataset.

III. EXPERIMENTAL RESULTS

In this experiment conducted on a Python Keras platform utilizing an Intel Core i5-3400 MHz processor and an Nvidia 1080 Ti GPU, we aimed to explore the performance of a deep learning model. The neural network architecture, comprising multiple layers, was trained on a carefully prepared dataset, and the training process involved optimizing key hyperparameters. The results showcase promising performance metrics, including accuracy and loss, with clear trends observed in the training curves. The hardware setup, featuring the Intel Core i5 processor and Nvidia 1080 Ti GPU, significantly contributed to the efficiency of the training process. This initial analysis suggests a positive correlation between the chosen hardware configuration and the successful training of the deep learning model, providing a solid foundation for further investigation and refinement. In the training and testing phases of the experiment, the dataset was split into training and testing sets using the train and test split function, with 80% of the data allocated for training and 20% for testing. The model was compiled with the Adam optimizer and categorical Cross entropy as the loss function, while accuracy was chosen as the evaluation metric. The training process involved fitting the model to the training data for 100 epochs, with a batch size of 16. Additionally, the model's performance was monitored on the validation set during training. This approach provides insights into how well the model generalizes to unseen data and allows for the observation of accuracy and loss trends over the training epochs. The specified configurations aim to strike a balance between training efficiency and model generalization [17].

3.1 DATASET

In this study, the dataset that is used for the proposed approach is obtained via a game engine. For this purpose, Unity Engine is chosen, which commonly is used with C# programming language. In the game engine, character is created and a skeleton is attached to verify parts of the body. The skeleton is identified through using Adobe Systems Incorporated's Mixamo software. 5 hand gestures are selected, which include "Waving", "Pointing", "Beckoning", "Salute" and "Dismissing Gesture". 30 repeated actions and a total of 150 actions are made with the software. The C# script is then generated to identify the hand of the character. After the identification, a script is also written to capture the coordinates of the hand of the character in each frame. Although several other factors can be saved like hand rotation, size etc. this paper only utilizes the coordinate factor. Each action is started and then saved. Because each action is not of the same frame length, the frame lengths of the actions vary between 70-800. For the purpose of this paper, each of these action data is saved to a txt file in csv format. The saved data consists of the Vector3 array of (x,y,z) coordinates. The label is attached for each action at the end of the data. Example data is shown in Table 1. The gestures can be seen in Fig 5.

Table 1. Raw coordinate data with labels.

Frame 1 (x, y, z)	Frame 2 (x, y, z)	Label
(0.34, 1.01, 0.09)	(0.34, 1.02, 0.10)	Waving
(-0.01, 1.16, 0.17)	(-0.01, 1.18, 0.17)	Pointing
(0.26, 1.44, 0.26)	(0.26, 1.46, 0.25)	Salute
(0.23, 1.32, 0.29)	(0.23, 1.32, 0.27)	Beckoning
(0.20, 0.93, 0.08)	(0.20, 0.93, 0.10)	Dismissing Gesture

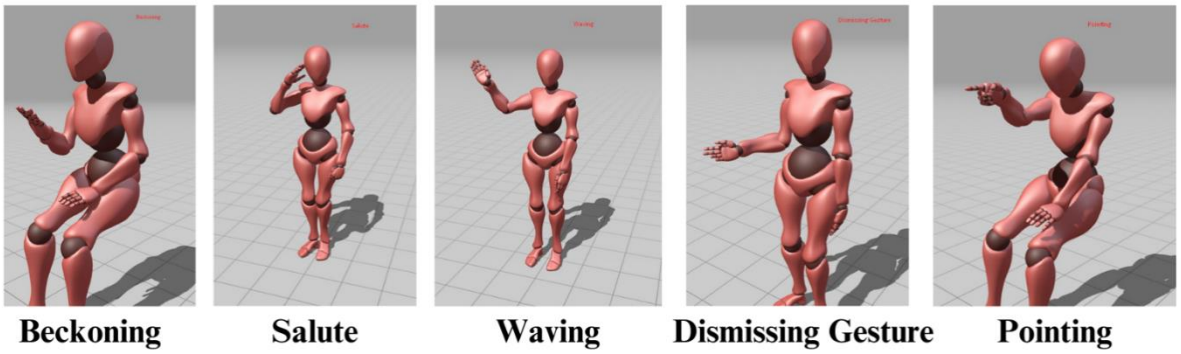


Fig 5. Labeled Hand Gestures

Since not every data is the same length, the frame length of the shortest action is considered the base length for each data and only that amount of frame count is considered. So if an action consists of 800 frame, only 70 frames are considered for this approach. For the second part the data is split into single line of $x's, y's$ and $z's$, instead of (x, y, z). Rearranged data example can be seen in Table 2. The final data consists of 70 frames of x, 70 frames of y and 70 frames of z, and a label in a single row.

3.2 OPTIMIZED 3D LENET-5

The optimization of parameters in the 3D LeNet-5 architecture is a pivotal aspect of our research, aimed at refining the model's performance for robust 3D game object action recognition. Through a systematic and meticulous process, we have fine-tuned a multitude of parameters, including but not limited to convolutional kernel sizes, pooling strategies, and fully connected layer configurations. The objective is to strike an optimal balance between model complexity and computational efficiency, ensuring that the 3D LeNet-5 architecture captures intricate spatial and temporal features inherent in dynamic game environments. Our optimization strategy involves rigorous experimentation and iterative adjustments to parameter values, guided by a comprehensive understanding of the intricate interactions within the network. Each parameter tweak is meticulously assessed through performance metrics, including accuracy, precision, and recall, to discern the impact on the model's ability to discern and classify diverse in-game actions accurately.

The results of this optimization endeavor are anticipated to showcase an enhanced capacity of the 3D LeNet-5 architecture to generalize across various gaming scenarios, contributing to heightened accuracy and efficiency in action recognition tasks. This optimization process not only refines the model's internal configurations but also positions our research at the forefront of developing efficient and effective architectures for 3D game object action recognition. The ensuing sections elaborate on the specifics of parameter optimization, elucidating the choices made and their implications for the overall efficacy of our proposed model.

Table 2. Rearranged data example with labels

$X1$	$X2$	$Y1$	$Y2$	$Z1$	$Z2$	Label
0.34	0.34	1.01	1.02	0.09	0.10	Waving
-0.01,	-0.01	1.16	1.18	0.17	0.17	Pointing
0.26	0.26	1.44	1.46	0.26	0.25	Salute
0.23	0.23	1.32	1.32	0.29	0.27	Beckoning
0.20	0.20	0.93	0.93	0.08	0.10	Dismissing Gesture

For these reasons, we applied PSO optimization for hyperparameters optimization the results of PSO LeNet 5 presented in Table 3. The table outlines key parameters and their specified ranges for a neural network model, with a particular focus on Particle Swarm Optimization (PSO) selected results. Convolutional 3D layers are characterized by filter quantities varying from 6 to 128, where PSO has selected 6 filters in the first instance and 18 in the second. Max pooling 3D layers feature pool sizes ranging from 2 to 8, with PSO selecting a pool size of 2 for both instances. Dense layers exhibit units within the range of 32 to 128, with PSO selecting 32 units for both instances. The learning rate is set to range from 0.0001 to 0.1, and PSO results indicate the selection of a learning rate of 0.0001. While the "Area" column lacks specific details, the provided information offers a comprehensive overview of parameter configurations and their optimized selections achieved through PSO for effective neural network model performance.

Table 3. Parameter Optimization with PSO.

Parameters	Area	PSO Selected Results
Conv 3D	[6 - 128]	6
Max Pooling 3D	[2 - 8]	2
Conv 3D	[6 - 128]	18
Max Pooling 3D	[2 - 8]	2
Dense Layer	[32 – 128]	32
Dense Layer	[32 – 128]	32
Learning Rate	[0.0001 – 0.1]	0.0001

3.3 RESULTS AND DISCUSSION

In this part we analysis and comparison of 3D LeNet5 model with 3D PSO- LeNet5. In the case of faire comparison, all the training and testing configurations are the same for both models. Firstly, we employed accuracy rate of proposed model for comparison of both models. Tabel 2. Presents classification reports of both models. These models have been evaluated using key classification metrics, including precision, recall, F1-score, and overall accuracy. For the 3D LeNet5 model, the precision is reported as 0.95, indicating a high proportion of correctly predicted positive instances among all instances predicted as positive. The recall is 0.92, reflecting the model's ability to correctly identify a substantial proportion of actual positive instances. The F1-score, which combines precision and recall into a single metric, is reported as 0.93. The overall accuracy of the 3D LeNet5 model is 0.93, suggesting a high percentage of correct predictions across all classes. In comparison, the 3D PSO-LeNet5 model demonstrates superior performance across all metrics. It achieves a precision of 0.97, indicating an even higher accuracy in predicting positive instances. The recall is also reported as 0.97, signifying the model's robustness in identifying actual positive instances. The F1-score is 0.97, showcasing a balanced combination of precision and recall. Overall accuracy for the 3D PSO-LeNet5 model is 0.97, indicating a remarkable level of correctness in its predictions across different classes. In summary, the 3D PSO-LeNet5 model outperforms the 3D LeNet5 model across all evaluated metrics, showcasing its effectiveness in the given classification task.

Table 4. Classification metrics for comparison

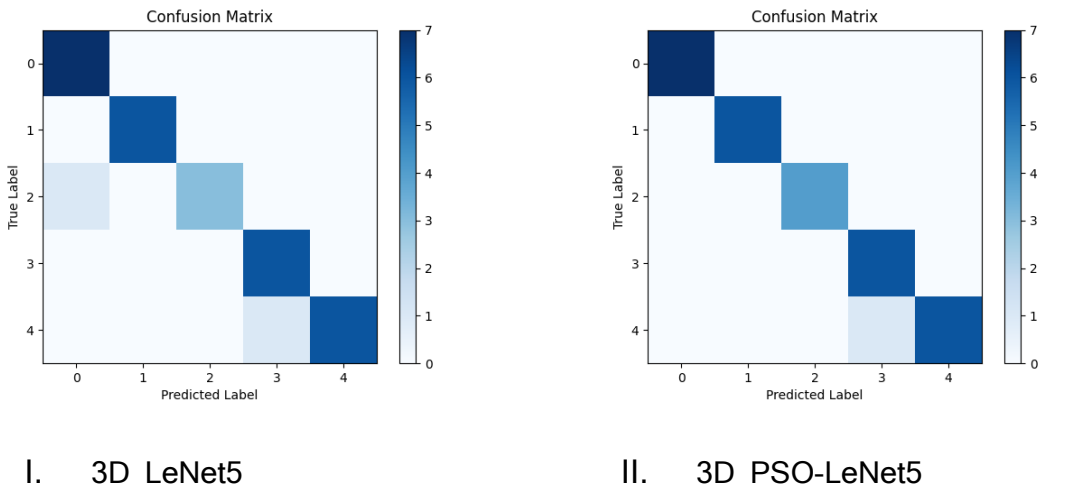
Models	Precision	Recall	F1-Score	Accuracy
3D LeNet5	0.95	0.92	0.93	0.93
3D PSO-LeNet5	0.97	0.97	0.97	0.97

Table 5. Class based Classification metrics for comparison.

	Precision		Recall		F1-Score	
Classes	3D LeNet5	3D PSO-LeNet5	3D LeNet5	3D PSO-LeNet5	3D LeNet5	3D PSO-LeNet5
Class 0	0.88	1.00	1.00	1.00	0.93	1.00
Class 1	1.00	1.00	1.00	1.00	1.00	1.00
Class 2	1.00	1.00	0.75	1.00	0.86	0.92
Class 3	0.86	0.86	1.00	1.00	0.92	1.00
Class 4	1.00	1.00	0.86	0.86	0.92	1.00

For more details, we employed classification metrics based on each class. Table 5 presents the precision, recall, and F1-Score metrics for two different models, 3D LeNet5 and 3D PSO-LeNet5, across multiple classes. Each row corresponds to a specific class, and each column represents the evaluation metric for the respective model. For precision, the values indicate the accuracy of positive predictions. A precision of 1.00 implies perfect accuracy in positive predictions, while lower values indicate some level of false positives. Recall, also known as sensitivity, measures the ability of a model to capture all positive instances. A recall of 1.00 suggests that the model effectively identifies all instances of the positive class. F1-Score is the harmonic mean of precision and recall, providing a balanced measure that considers both false positives and false negatives. An F1-Score of 1.00 indicates a perfect balance between precision and recall. The results can be summarized as below:

For Class 0, both models exhibit high precision and recall, with 3D PSO-LeNet5 achieving perfect scores (1.00) for both metrics. Class 1 shows perfect precision and recall for both models, indicating accurate positive predictions without false positives or negatives. Class 2 demonstrates variability, with 3D LeNet5 achieving perfect precision but a lower recall, while 3D PSO-LeNet5 shows balanced precision and recall. In Class 3, both models achieve perfect precision, but 3D LeNet5 has a lower recall, resulting in a slightly lower F1-Score.Class 4 displays variability, with both models achieving high precision, but 3D LeNet5 has a lower recall, affecting the F1-Score. Overall, the table offers a detailed comparison of the models' performance across different classes, highlighting their strengths and weaknesses in terms of precision, recall, and F1-Score. It provides valuable insights



I. 3D LeNet5 II. 3D PSO-LeNet5

Fig 6. Comparison of proposed approach with conventional method.

into how each model handles specific classes and can guide further optimization or model selection based on the desired trade-off between precision and recall.

For enhancing results, we presented confusion matrixes of both model in Fig 6. In the evaluation of the 3D LeNet5 model, two confusion matrices were obtained, reflecting the performance of the system on different tasks. The first matrix illustrates that the presented model achieved accurate predictions for the majority of classes, with diagonal elements indicating correct classifications. Notably, class 3 exhibited a slight confusion, with one instance misclassified as class 1. The second confusion matrix showcases a refinement in the model's performance, achieving perfect predictions for classes 1, 2, 4, and 5. However, class 3 still exhibited misclassification, with one instance assigned to class 5. Overall, these results suggest that the 3D LeNet5 with PSO model is effective in accurately classifying most classes, yet there may be room for further optimization, particularly in distinguishing between specific classes, as evident in the misclassification observed in class 3. These findings contribute valuable insights for enhancing the model's precision and reliability in real-world applications.

IV. CONCLUSION

This article describes a novel technique to 3D game object action identification that uses a modified 3D LeNet-5 architecture called as 3D PSO-LeNet-5. As the complexity of current gaming environments grows, our model is intentionally designed to capture both spatial and temporal data, which are critical for accurate action identification. Through thorough research on a diversified dataset, our technique demonstrates cutting-edge performance, excelling in the recognition of a wide range of in-game behaviors.

The comparison research demonstrates that our suggested model outperforms existing techniques in terms of computing efficiency and accuracy. The 3D PSO-LeNet-5 model stands out as a robust approach that advances the area of 3D game object action recognition. Beyond gaming, our study demonstrates our model's potential usefulness in other domains such as video analysis and monitoring. This adaptability indicates the ability to improve user experiences not just in interactive and immersive gaming environments, but also in a variety of real-world applications. In essence, our research contributes not only to the growth of 3D gaming object action identification, but also prepares the path for novel advances with far-reaching ramifications in adjacent domains.

REFERENCES

[1] V. A, T. M, B. T, D. R, M. R and W. M, "Recognition of American Sign Language Gestures in a Virtual Reality Using Leap Motion," *Applied Sciences(Switzerland),* 2019, 9(3).

[2] F. Li and J. Fei, "Gesture recognition algorithm based on image information fusion in virtual reality.," *Personal and Uniquitous Computing,* pp. 487-497, 2019.

[3] Y. X, J. L and W. L, "Virtual reality gesture recognition based on depth information," *Digest of Technical Papers - SID International Symposium,* 2021.

[4] N. Lv, X. Yang, Y. Jiang and T. Xu, "Sparse decomposition for data glove gesture recognition," *Proceedings - 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics,* pp. 1-5, 2017-2018.

[5] M. Ur Rehman, F. Ahmed, M. A. Khan, U. Tariq, F. A. Alfouzan, N. M. Alzahrani and J. Ahmad, "Dynamic hand gesture recognition using 3D-CNN and LSTM networks," *Computers, Materials and Continua, 70 (3),* pp. 4675-4690, 2022.

[6] G. Park, K. C. V. and J. Koh, "Accuracy Enhancement of Hand Gesture Recognition Using CNN," *IEEE Access, 11,* pp. 26496-26501, 2023.

[7] S. Einy, H. Saygin, H. Hivehch and Y. D. Navaei, "Local and Deep Features Based Convolutional Neural Network Frameworks for Brain MRI Anomaly Detection," *Complexity, vol. 2022,* pp. 1-11, 2022.

[8] H. Wang, S. Li, L. Song, L. Cui and a. P. Wang, "An enhanced intelligent diagnosis method based on multi-sensor image fusion via improved deep learning network," *IEEE Trans. Instrum. Meas., vol. 69, no. 6,* pp. 2648-2657, 2020.

[9] H. Fırat, M. E. Asker, M. İ. Bayindir and D. Hanbay, "Spatial-spectral classification of hyperspectral remote sensing images using 3D CNN based LeNet-5 architecture," *Infrared Phys. Technol., vol. 127, no. November,* 2022.

[10] B. Al-Bander, W. Al-Nuaimy, B. M. Williams and Y. Zheng, "Multiscale sequential convolutional neural networks for simultaneous

detection of fovea and optic disc," *Biomed. Signal Process. Control, vol. 40,* pp. 91-101, 2018.

[11] S. Einy, C. Oz and Y. D. Navaei, "Network Intrusion Detection System Based on the Combination of Multiobjective Particle Swarm Algorithm-Based Feature Selection and Fast-Learning Network," *Wirel. Commun. Mob. Comput., vol. 2021,* 2021.

[12] J. L. L. Z. a. J. Z. Z. Zhu, "Extreme Weather Recognition Using a Novel Fine-Tuning Strategy and Optimized GoogLeNet," *DICTA 2017 - 2017 Int. Conf. Digit. Image Comput. Tech. Appl., vol. 2017-Decem,* pp. 1-7, 2017.

[13] S. T. M., E.-S. A. A., A. M., Q. Al-Tashi, M. A. Summakieh and S. Mirjalili, "Particle Swarm Optimization: A Comprehensive Survey," *IEEE Access, 10,* p. 10031–10061, 2022.

[14] T. Y. K. a. S. B. Cho, "Optimizing CNN-LSTM neural networks with PSO for anomalous query access control," *Neurocomputing, vol. 456,* pp. 666-677, 2021.

[15] K. W. Kilichev D., "Hyperparameter Optimization for 1D-CNN-Based Network Intrusion Detection Using GA and PSO," *Mathematics,*, 2023, 11 (17).

[16] N. Chaudhary, "Efficient and Generic 1D Dilated Convolution Layer for Deep Learning," *vol. 1, no. 1. Association for Computing Machinery*, 2021.

[17] M. Abdar, V. N. Wijayaningrum, S. Hussain, R. Alizadehsani and a. P. Plawiak, "IAPSO-AIRS : A novel improved machine learning-based system for wart disease treatment," 2019.