# A Study of Human Fitness Pose Classification Using Artificial Neural Networks

Sijie Shang
*Department of Computer Science*
*California State University, Fullerton*
Fullerton, CA, USA
sshang@csu.fullerton.edu

Rong Jin
*Department of Computer Science*
*California State University, Fullerton*
Fullerton, CA, USA
rong.jin@fullerton.edu

Kevin Desai
*Department of Computer Science*
*University of Texas at San Antonio*
San Antonio, TX, USA
kevin.desai@utsa.edu

***Abstract*—Motivated by the increasing demand for detecting and classifying human poses in the realm of personalized fitness training by AI technologies, which provide feedback on the form to help users exercise more accurately, and the proven effectiveness of some deep learning models in achieving that, this study aims to investigate three different ensemble approaches for artificial neural network models for detecting and classifying human poses. The pre-trained MoveNet model was employed to extract the positions of 17 body keypoints, which were used as input data for the subsequent three classification models - a Feedforward Neural Network (FNN), LSTM, and GRU. The LSTM and GRU models have the ability to process time series data as input, resulting in improved accuracy compared to the FNN model. Specifically, the LSTM model achieved an accuracy of nearly 95%, while the GRU model outperformed with an accuracy exceeding 95% and the potential to reach 97.27%.**

***Index Terms*—Deep Learning, MoveNet, LSTM, GRU, Human Fitness Pose Classification**

2023 International Conference on Information Technology (ICIT) | 979-8-3503-2006-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICIT58056.2023.10225860

## I. Introduction

With the advancements in AI technologies, the demand for customized experiences is increasing. A prime example of this is the need for personalized training in human fitness. Multiple artificial neural network models have been utilized to solve fitness-related issues, including the Feedforward Neural Network, Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN).

The Feedforward Neural Network (FNN), as referenced in [1], [2], is a kind of artificial neural network where the flow of information is unidirectional, moving from the input layer through the hidden layers before eventually arriving at the output layer and the connections between neurons do not form any cycles or loops. This ”feedforward” architecture facilitates the efficient processing of input data and finds applications in several areas, including the recognition of fitness poses. For instance, [3] utilizes the FNN to identify tennis players, enabling them to apply the model in tennis coaching and technology.

The MoveNet is a CNN that is purpose-built for identifying 17 keypoints on the human body, including significant anatomical landmarks and joints [4], [5]. It is well-suited for tasks associated with motion analysis and human pose estimation. MoveNet has proven to be useful in developing applications for body exercises. For instance, [6] leverages MoveNet to create a body exercise app that can recognize human poses and provide audio feedback in real-time.

The LSTM network is a type of RNN architecture that addresses the vanishing gradient problem commonly encountered in traditional RNNs [7], [8]. This problem occurs when gradients, which are used to update the network's weights during backpropagation, become smaller with each iteration. As a result, minimal weight adjustments occur, limiting the network's ability to learn, particularly with long-term dependencies. However, the human fitness pose dataset is characterized by long-term dependencies, where earlier body positions significantly impact subsequent ones. The LSTM network overcomes this issue by employing specialized cell states and gating mechanisms that regulate the flow of information and selectively preserve or update the cell state as required. As a result, LSTMs can efficiently learn and retain long-term dependencies, making them a useful tool for human fitness pose classification. An example of the LSTM's efficiency is demonstrated in [9], which uses this model for real-time human action detection.

A simpler alternative type of RNN architecture to LSTM networks is the Gated Recurrent Unit (GRU) [10]. They are designed to effectively learn long-term dependencies in sequential data by addressing the vanishing gradient problem. Unlike LSTMs, GRUs have a simpler design with fewer parameters, which makes them easier and faster to train on a computer. This often results in the same or better performance for certain tasks, particularly when resources are limited, or the dataset is small. GRUs have demonstrated success in the sports field, and as an example, [11] propose a model for detecting sports-related actions using GRUs.

Motivated by the increasing demand for human fitness pose detection and classification in the realm of personalized training and the proven effectiveness of aforementioned deep learning models in achieving that, investigate three different ensemble learning models for the detection and classification of human fitness poses. In this study, the pre-trained MoveNet model was utilized to extract the positions of 17 body keypoints as input data for the subsequent classification models. The initial classification model employed is a Feedforward Neural Network, which predicts the human pose for each frame. The following two models, namely LSTM and GRU, can utilize time series data as input, resulting in improved accuracy compared to the first model. An LSTM model trained on 20-frame chunks of a video achieved an accuracy close to 95%, while a GRU model achieved an accuracy of 97.27%.

The organization of this paper is as follows: related works are discussed in Section II, followed by details on the dataset and model implementations in Section III and the evaluation results in Section IV. Lastly, a brief conclusion and future work are provided in Section V.

## II. Related Works

This section presents some related studies on the use of deep learning for the detection and classification of human fitness poses.

Jhen-Min Hung et al. [3] suggested a method that employs YOLO v5 multi-target bounding-box detection and Multi-Layer Perceptron (MLP) neural network, which is a type of FFNN that consists of one or more hidden layers between the input and output layers, to distinguish between forehand-swing and backhand-swing in tennis players. The work used video data from the 2020 and 2019 US Open Tennis Championships. The process involved extracting frames from the videos, detecting the players and balls using the YOLO v5 model, and classifying the poses using the MLP. The work used hyperparameters to optimize the result so maximum accuracy reached 93%. In another study, Rajdeep Chatterjee et al. [12] presented an intelligent system for detecting tennis players' poses. They used Detection2 to detect the keypoints of humans and used them as input to train a Random Forest (RF) model. This work also compared the results to different fine-tuned CNN models and found that the RF model had the highest accuracy and shortest training time. Detection2 is an object detection and segmentation framework developed by Meta AI [13].

Applications of human pose classification and recognition have been explored in other fields. For instance, Huu et al. [14] developed a posture recognition system called mobilenetV2 for medical surveillance. The system employs the LSTM network to analyze the temporal patterns of the features and classify them into different postures such as standing, sitting, lying down, bending, and squatting. With the use of LSTM, the system achieves high accuracy and efficiency on various datasets, and it can be used to monitor patients' activities and health conditions. Similarly, Meng Xu et al. [15] introduced a method that employs an LSTM module in conjunction with a rotation classification loss to estimate the camera pose (i.e., the position and orientation of a camera) from image sequences in order to estimate human poses. The LSTM module captures temporal information from the images, while the rotation classification loss helps minimize errors in determining camera orientation. The effectiveness of the proposed approach was evaluated on two publicly available datasets (KITTI and EuRoC), and the results were compared against those of several other advanced methods.

## III. Experimental Study

### A. Dataset

Our study employed a fitness video dataset supplied by [16]. The dataset contains 10 of the most commonly practiced exercises, such as armraise, legraise, bicyclecrunch, birddog, curl, fly, overhead press, pushup, squat, and superman, and each exercise consists of 100 videos filmed in diverse lighting conditions and backgrounds. For our experiment, we allocated 80% of the data to the training set and the remaining 20% to the validation set.

### B. Data Preprocessing

We performed the following two steps to preprocess the data for further analysis:

1) **Video Frame Extraction**: To analyze the dataset of fitness poses, the videos were first split into a sequence of images. This allowed the frames in a video to be processed using machine learning models. The desired temporal resolution can be adjusted by varying the number of frames extracted per second, which can have an impact on the accuracy and performance of the models. For this dataset, 24 frames per second were extracted to capture the crucial moments of each exercise.
2) **Person-focused Image Croppoing**: Once the frames were extracted, the subsequent step involved cropping the images to concentrate on the individuals performing the fitness poses. The dataset provided metadata about the person's position in each frame, which was utilized to achieve this. The images were cropped and centered on the person to ensure that the machine learning model focuses on the individual and their movements during the exercises.

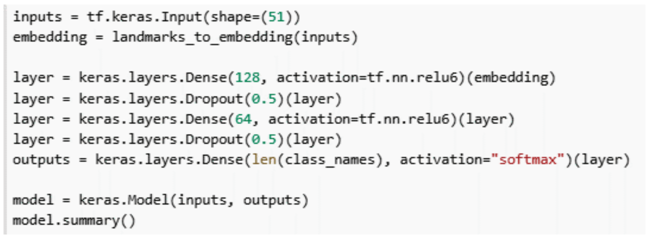
### C. Human Keypoints Detected by MoveNet

In this study, the pre-trained MoveNet model was used to detect 17 human keypoints in each video frame, including the nose, eyes, ears, shoulders, elbows, wrists, hips, knees, and ankles, which provides the keypoint locations in the form of $(x, y)$ coordinates and the confidence scores of each identified keypoint. These keypoints served as the input data for the subsequent three different classification neural networks. The MoveNet was chosen for its efficiency and real-time applicability. It is designed for human pose estimation and can run on mobile devices, making it a suitable choice for fitness pose classification applications.

### D. Artificial Neural Network Models

In this section, a comparative study of three different artificial neural network models is presented for the classification of human fitness poses. The positions of 17 human keypoints are obtained using the pre-trained MoveNet model in each video frame, which serves as input data for the subsequent classification task. Furthermore, the implementation details of each evaluated model are provided accordingly.

*1) Feedforward Neural Network:* Our FNN model takes a tensor of shape (batch_size, 51) as input, where '51' represents the number of features in the input data. The input tensor is passed through a custom function that maps the input tensor to a higher dimensional space, resulting in a tensor of shape (128). See Fig. 1.



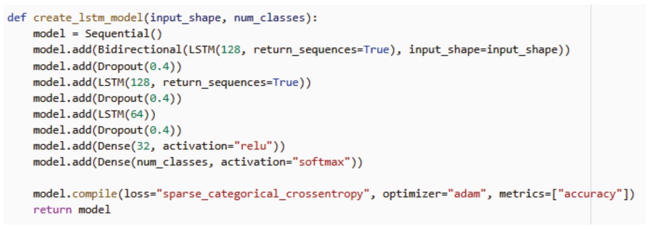
```
inputs = tf.keras.Input(shape=(51))
embedding = landmarks_to_embedding(inputs)

layer = keras.layers.Dense(128, activation=tf.nn.relu6)(embedding)
layer = keras.layers.Dropout(0.5)(layer)
layer = keras.layers.Dense(64, activation=tf.nn.relu6)(layer)
layer = keras.layers.Dropout(0.5)(layer)
outputs = keras.layers.Dense(len(class_names), activation="softmax")(layer)

model = keras.Model(inputs, outputs)
model.summary()
```

Fig. 1. The implementation details of FNN model.

The mapped tensor then passes through two dense layers with different numbers of neurons. The first dense layer has 128 neurons and applies the ReLU6 activation function, which is a variant of the Rectified Linear Unit (ReLU) activation function with a maximum output of 6. The second dense layer has 64 neurons and also applies the ReLU6 activation function. Both dense layers are followed by a dropout layer that randomly sets 50% of the input units to 0 at each update during training to prevent overfitting. Last, the output of the second dropout layer is fed into a dense layer with len(class_names) neurons, where class_names represents the number of exercises (which is equal to 10 in our study) and a softmax activation function is applied.

*2) Long Short-Term Memory Network:* Instead of classifying the names of fitness poses solely based on the raw pixel values of the video frames, our LSTM model was altered to handle a new type of input data by adjusting the input shape to match the quantity of 17 keypoints by MoveNet and confidence scores. In particular, the altered input data format consists of three features: the x and y coordinates and the confidence score of each detected keypoint by MoveNet for every frame in a video sequence and incorporates two additional features: the class index number (ranging from 0 to 9 for the 10 poses in our dataset) and the corresponding class names.

Moverover, LSTM is a recurrent model, and there is a need to decide the sequence length of each input. Therefore, two different LSTM models were experimented with in our study as follows.

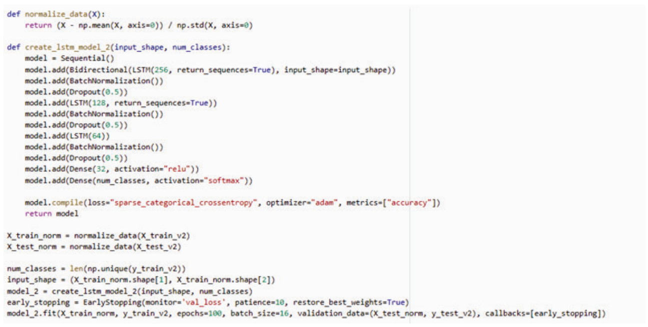


```
def create_lstm_model(input_shape, num_classes):
    model = Sequential()
    model.add(Bidirectional(LSTM(128, return_sequences=True), input_shape=input_shape))
    model.add(Dropout(0.4))
    model.add(LSTM(128, return_sequences=True))
    model.add(Dropout(0.4))
    model.add(LSTM(64))
    model.add(Dropout(0.4))
    model.add(Dense(32, activation="relu"))
    model.add(Dense(num_classes, activation="softmax"))

    model.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
    return model
```

Fig. 2. The implementation details of LSTM 1 model.

The first model, named LSTM 1 (see Fig. 2), involved taking the first 100 frames of each video as input for the first layer, which was a bidirectional LSTM layer with 128 units. Following that was a dropout layer, and then an LSTM layer with 128 units as the subsequent layer. Another dropout layer was added, followed by an LSTM layer with 64 units as the third layer, followed by a dropout layer, a dense layer having 32 units, and lastly, a dense softmax layer, which generated the class probabilities.

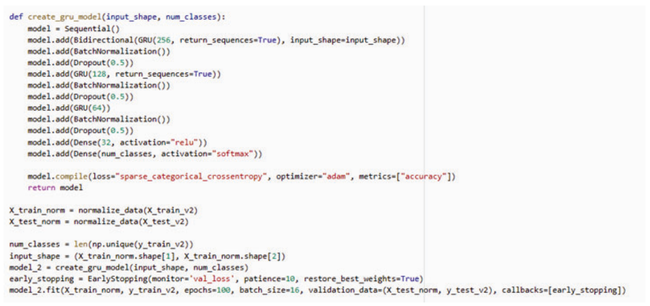
The second model, referred to as LSTM 2, involved segmenting each video into smaller 20-frame chunks. Redundant frames that contain repetitive movements are eliminated to improve the efficiency of the model. LSTM 2 adopted a similar architecture to LSTM 1. As shown in Fig. 3, the data for both training and testing were normalized. The model was constructed with a bidirectional LSTM layer consisting of 256 units, batch normalization, and dropout. This was followed by an additional LSTM layer with 128 units, dropout, and batch normalization. Subsequently, an LSTM layer with 64 units, batch normalization, and dropout, and then followed by two dense layers. To prevent overfitting and select the best model based on validation loss, early stopping was implemented as a callback during the training process. A batch size of 16 and a maximum of 100 epochs were employed to train this model.



```
def normalize_data(X):
    return (X - np.mean(X, axis=0)) / np.std(X, axis=0)

def create_lstm_model_2(input_shape, num_classes):
    model = Sequential()
    model.add(Bidirectional(LSTM(256, return_sequences=True), input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(LSTM(128, return_sequences=True))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(LSTM(64))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(32, activation="relu"))
    model.add(Dense(num_classes, activation="softmax"))

    model.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
    return model

X_train_norm = normalize_data(X_train_v2)
X_test_norm = normalize_data(X_test_v2)

num_classes = len(np.unique(y_train_v2))
input_shape = (X_train_norm.shape[1], X_train_norm.shape[2])
model_2 = create_lstm_model_2(input_shape, num_classes)
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
model_2.fit(X_train_norm, y_train_v2, epochs=100, batch_size=16, validation_data=(X_test_norm, y_test_v2), callbacks=[early_stopping])
```

Fig. 3. The implementation details of LSTM 2 model.

*3) Gated Recurrent Unit:* The GRU model was similar to the architecture of the LSTM model. It also involved segmenting each video into smaller 20-frame chunks. Redundant frames that contain repetitive movements are eliminated to improve the efficiency of the model. The data for both training and testing were also normalized. As shown in Fig. 4, the initial layer of our GRU model consisted of a bidirectional GRU layer with 256 units, batch normalization, and a dropout layer. Next, a second GRU layer with 128 units, batch normalization, and dropout were included. After that, a third GRU layer with 64 units was added, followed by batch normalization and dropout. Finally, two dense layers were added to output the class probabilities. Early stopping was included as a callback during training to avoid overfitting and saved the best model based on validation loss. A batch size of 16 and a maximum of 100 epochs are used to train the model. To prevent overfitting and select the best model based on validation loss, early stopping was also implemented as a callback during the training process. A batch size of 16 and a maximum of 100 epochs were employed to train this model.



```
def create_gru_model(input_shape, num_classes):
    model = Sequential()
    model.add(Bidirectional(GRU(256, return_sequences=True), input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(GRU(128, return_sequences=True))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(GRU(64))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(32, activation="relu"))
    model.add(Dense(num_classes, activation="softmax"))

    model.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
    return model

X_train_norm = normalize_data(X_train_v2)
X_test_norm = normalize_data(X_test_v2)

num_classes = len(np.unique(y_train_v2))
input_shape = (X_train_norm.shape[1], X_train_norm.shape[2])
model_2 = create_gru_model(input_shape, num_classes)
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
model_2.fit(X_train_norm, y_train_v2, epochs=100, batch_size=16, validation_data=(X_test_norm, y_test_v2), callbacks=[early_stopping])
```

Fig. 4. The implementation details of GRU model.

## IV. Evaluation Results

In this section, the training and validation accuracy of each experimented artificial neural network model is depicted as a function of the number of training epochs.

### A. MoveNet and Feedforward Neural Network

The model in this case was trained for a total of 128 epochs, and both the training and validation accuracy showed a positive trend as the number of epochs increased. As shown in Fig. 5, the training accuracy reached around 80%, while the validation accuracy achieved to 85%. It is worth noting that the validation accuracy surpassed the training accuracy, suggesting that the model generalized well to unseen data. This is desirable as it indicates that the model is less likely to overfit on the training data and can maintain good performance on new data samples. The model performed well overall, with a weighted accuracy of 76%.

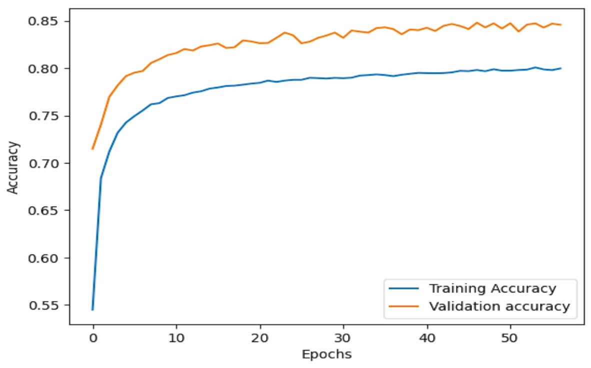


Fig. 5. Accuracy of MoveNet and Feedforward Neural Network Models for Each Frame per Video

### B. MoveNet and Long Short-Term Memory Network

At first, LSTM 1 model was trained using the first 100-frame sequences for each video, but the outcomes were unsatisfactory. The accuracy levels for both training and validation were below 22%, as depicted in Fig. 6. To improve the model's performance, for the LSTM 2 model, the videos were divided into 20-frame segments, with any additional frames being discarded. As a result, the training accuracy of LSTM 2 achieved an accuracy approaching 95% on both training and validation datasets, as displayed in Fig. 7, which was attributed to the implementation of normalized data input and an increased number of layers.

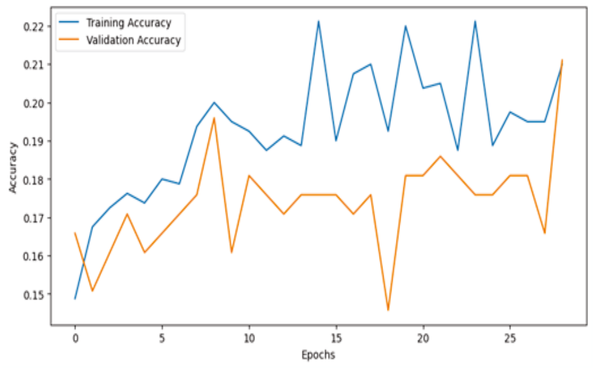


Fig. 6. Accuracy of MoveNet and LSTM 1 trained on the first 100-frame sequence per video

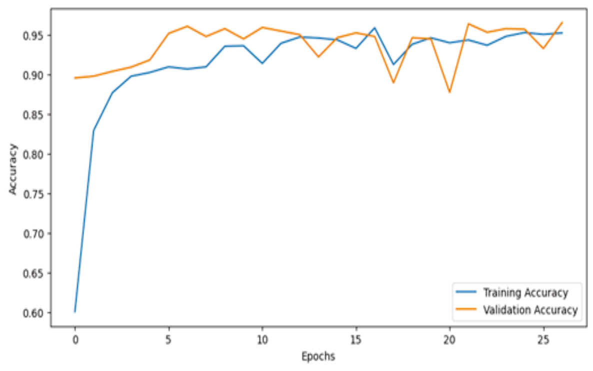


Fig. 7. Accuracy of MoveNet and LSTM 2 trained on 20-frame chunks per video with data normalization

### C. MoveNet and Gated Recurrent Unit

The GRUs model performs admirably, obtaining over 95% accuracy for both the training and validation sets after just five iterations, as shown in Fig. 8. The training and validation accuracies, however, occasionally switch due to variations in the plot. Despite the fluctuation, the model stops before its expected time at epoch 29, stopping with a weighted accuracy over 95%, with the possibility of reaching 97.27%.

### D. Comparison of Results

The model's performance was assessed using the precision score, which measures the proportion of true positive classifications relative to all positive classifications. The formula for precision is as follows:

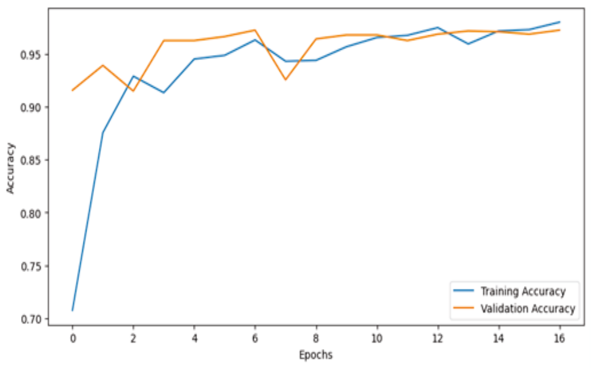


Fig. 8. Accuracy of MoveNet and GRU trained on 20-frame chunks per video with data normalization

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

As seen in Table I, the FNN model performs well on three classes (poses), such as “curl”, “fly”, and “overheadpress”, with precision scores of 0.86, 0.84, and 0.89, respectively. However, some classes, such as “basic_lagraise”, “bicyclecrunch”, and “superman” have low precision scores of 0.29, 0.17, and 0.28, respectively. These results indicate that the model can accurately classify certain exercises while maintaining a balance between precision and recall.

TABLE I
PRECISION SCORES OF THREE DIFFERENT ARTIFICIAL NEURAL NETWORKS MODELS FOR TEN HUMAN FITNESS POSES

| | FeedForward | LSTM 2 | GRU |
|---|---|---|---|
| armraise | 0.67 | 0.97 | 0.97 |
| basic legraise | 0.29 | 0.00 | 0.00 |
| bicyclecrunch | 0.17 | 0.00 | 0.25 |
| birddog | 0.59 | 0.77 | 0.93 |
| curl | 0.86 | 1.00 | 0.99 |
| fly | 0.84 | 0.99 | 0.99 |
| overheadpress | 0.89 | 0.96 | 1.00 |
| pushup | 0.74 | 0.53 | 0.91 |
| squat | 0.75 | 1.00 | 0.96 |
| superman | 0.28 | 0.08 | 0.00 |

LSTM 2 showed the best performance. Table I demonstrates that it demonstrated exceptional accuracy in recognizing five exercises, which are “armraise”, “curl”, “fly”, “overhead press”, and “squat”, with precision scores of 0.97, 1.00, 0.99, 0.96, and 1.00, respectively. However, the other exercises, such as “basic_legraise”, “bicyclecrunch”, “pushup”, and “superman”, were more challenging to classify correctly. Clearly, compared to the Feedforward Neural Network, the MoveNet framework and the LSTM model showed robust performance in identifying specific human poses.

Based on the results in Table I, the GRU model shows outstanding performance in recognizing most workout poses with high precision scores for seven exercises, namely “armraise”, “birddog”, “curl”, “fly”, “overheadpress”, “pushup”, and “squat” with precision scores close to 1.00. However, some exercises, such as “basic_legraise”, “bicyclecrunch”, and “superman”, present difficulties for the model and have low precision scores.

## V. Conclusion and Future Work

In this paper, a comparative study of three artificial neural network models is presented: a FNN model, LSTM models in different configurations, and a GRU model combined with the pre-trained MoveNet framework for classifying human fitness poses. The experimental results reveal promising achievements in accurately classifying human exercise poses. The GRU model exhibited the highest accuracy, surpassing 95% and potentially reaching 97.27% in the study. Additionally, one tested LSTM model also demonstrated good performance with an accuracy score approaching 95%.

In this study, we acknowledge a limitation related to the challenges encountered in classifying certain exercises. Despite the overall promising and high accuracy results, we faced difficulties in effectively distinguishing between a few similar poses. These poses exhibited similarities in terms of body movements or positions, posing a challenge for our models to accurately classify them.

To address this limitation and improve model performance, several approaches can be explored. Increasing the quantity and quality of training data using some data augmentation techniques, or adjusting the model architecture are potential strategies. Furthermore, future research could involve integrating alternative pre-trained human pose estimation frameworks such as OpenPose [17], PoseNet [18], or Detectron2 [19] to detect human keypoints. These frameworks can then be incorporated as inputs for our evaluated neural network models, potentially leading to improved accuracy and performance.

## References

[1] DeepAI, “Feed Forward Neural Network,” DeepAI, Jun. 2020, [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network.

[2] V. Kurama, “Feedforward Neural Networks: A Quick Primer for Deep Learning,” Built In, Sep. 2019, [Online]. Available: https://builtin.com/data-science/feedforward-neural-network-intro.

[3] J. -M. Hung, J. -Y. Chiang and K. Wang, ”Tennis Player Pose Classification using YOLO and MLP Neural Networks,” 2021 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Hualien City, Taiwan, 2021, pp. 1-2, doi: 10.1109/ISPACS51563.2021.9650925.

[4] “MoveNet: Ultra fast and accurate pose detection model.,” TensorFlow, [Online]. Available: https://www.tensorflow.org/hub/tutorials/movenet.

[5] “Next-Generation Pose Detection with MoveNet and TensorFlow.js.” [Online]. Available: https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html.

[6] Cai, Zixin. ”PoseBuddy: workout platform with MoveNet.” (2022). Available: https://hdl.handle.net/10356/162846

[7] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[8] “Recurrent Neural Network (RNN) – Part 5: Custom Cells,” The Neural Perspective, Nov. 20, 2016. [Online]. Available: https://web.archive.org/web/20180416083110/https://theneuralperspective.com/2016/11/17/recurrent-neural-network-rnn-part-4-custom-cells/.

[9] Amin Nasiri, Jonathan Yoder, Yang Zhao, Shawn Hawkins, Maria Prado, Hao Gan, Pose estimation-based lameness recognition in broiler using CNN-LSTM network, Computers and Electronics in Agriculture, Volume 197, 2022, 106931, ISSN 0168-1699, https://doi.org/10.1016/j.compag.2022.106931.

[10] K. Cho et al., “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation,” arXiv (Cornell University), Jan. 2014, doi: 10.3115/v1/d14-1179.

[11] Mekruksavanich, Sakorn, and Anuchit Jitpattanakul. ”Sport-Related Activity Recognition from Wearable Sensors Using Bidirectional GRU Network.” Intelligent Automation Soft Computing 34.3 (2022).

[12] R. Chatterjee, S. Roy, S. H. Islam and D. Samanta, ”An AI Approach to Pose-based Sports Activity Classification,” 2021 8th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 2021, pp. 156-161, doi: 10.1109/SPIN52536.2021.9565996.

[13] Analytics Vidhya, “Introduction to Artificial Neural Networks,” Analytics Vidhya. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/09/introduction-to-artificial-neural-networks/. [Accessed: 07-Jan-2023].

[14] P. Nguyen Huu, N. Nguyen Thi and T. P. Ngoc, ”Proposing Posture Recognition System Combining MobilenetV2 and LSTM for Medical Surveillance,” in IEEE Access, vol. 10, pp. 1839-1849, 2022, doi: 10.1109/ACCESS.2021.3138778.

[15] M. Xu, L. Wang, J. Ren and S. Poslad, ”Use of LSTM Regression and Rotation Classification to Improve Camera Pose Localization Estimation,” 2020 IEEE 14th International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, China, 2020, pp. 6-10, doi: 10.1109/ASID50160.2020.9271762.

[16] “Papers with Code - InfiniteRep Dataset.” [Online]. Available: https://paperswithcode.com/dataset/infiniterep.

[17] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, ‘OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields’, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019.

[18] “Real-time Human Pose Estimation in the Browser with TensorFlow.js.” https://blog.tensorflow.org/2018/05/real-time-human-pose-estimation-in.html

[19] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, ‘Detectron2’, 2019. [Online]. Available: https://github.com/facebookresearch/detectron2.