International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. #, No.# (Mon-20..)

Real-Time Human Action Recognition using OpenPose and Sequence-Based Classification

Paulus Mudjihartono [1*], Andi W. R. Emanuel [2], Joanna Ardhyanti Mita Nugraha[3], Fedelis Brian Putra Prakasa[4], Shuib Basri[5]

[1,2,3,4] Departemen Informatika,Universitas Atma Jaya Yogyakarta, Yogyakarta, Indonesia
[5]Department of Computer and Information Sciences, Universiti Teknologi Petronas, Bandar Seri Iskandar, 32610 Seri Iskandar Perak, Malaysia

Email: paulus.mudjihartono@uajy.ac.id [1], andi.emanuel@uajy.ac.id[2], joanna.mita@uajy.ac.id[3], fedelis.brian@uajy.ac.id[4], shuib_basri@utp.edu.my[5]*

Received ## Mon. 20##, Revised ## Mon. 20##, Accepted ## Mon. 20##, Published ## Mon. 20##

Abstract: Human Action Recognition is one important area of Artificial Intelligence that is still in development. The ability to recognize action in human objects will significantly increase the understanding of images or videos for many practical purposes. This research employs three sequence-based algorithms to detect human actions, which are LSTM, CNN-LSTM, and CONVLSTM, to predict human action sequences in videos. The steps taken are 1) Collect action videos from video clips of actions as the data source. Convert the video clips into data sets for model training and testing. 2) Build the model using the datasets and the selected sequence-based classification algorithms. The best model from each algorithm is then implemented to get the inference engines. 3) Build inference engines for each algorithm. Action videos are collected and extracted by their key points using OpenPose; these 30 frame key points data are used to train the models. The results are the ability to predict seven human actions with an accuracy of 83.1429% in the LSTM model, 83.7143% in the CNN-LSTM model, and 83% in the CONVLSTM model. Inference engines for these models converted in TFLite were built to demonstrate that the systems can detect real-time action in recorded or webcam. The TFLite versions of LSTM, CNN-LSTM, and CONVLSTM inference times are 0.5ms, 0.25ms, and 0.5ms, respectively.

Keywords: *Human Action Recognition, sequence-based algorithm, real-time inference, OpenPose.*

1. INTRODUCTION

Artificial Intelligence has developed significantly in the past decade, and almost all aspects of our lives have been influenced by Artificial Intelligence (AI). One of the aspects of AI that has evolved in maturity is the ability of computers to detect objects or object detection. The ability includes the capability to detect single or many objects in a very short time [1], the capability to recognize a human face with high accuracy [2], and the capability to detect human illness based on the image from medical images [3], and many more. This capability of detecting objects enables computers to detect the "subject" and the "object" of the image or video. But there's one thing missing from understanding an image or video: figuring out what the objects in the image or video are doing. Human action and emotion are detected based on object movement [4]. Also, it illustrates that are 25 joints generated using kinetics, and their connectivity helped identify human emotions.

Human Action Recognition is one area of computer science that needs to be improved to increase computer understanding of images or videos. Understanding images or videos requires the ability to detect the objects (can be the subject or objects) and their action so computers can better describe the image or videos. This condition is particularly important in images or videos involving humans as the subject or object; the capability to better comprehend the image or videos will improve many areas of human surveillance. This capability includes the capability of computers to give early warning and prevent harmful incidents to human operators in case of elderly falling, human drowning in the pool, and many

paulus.mudjihartono@uajy.ac.id, andi.emanuel@uajy.ac.id, joanna.mita@uajy.ac.id, fedelis.brian@uajy.ac.id, shuib_basri@utp.edu.my

more, which will improve our ability to protect and prevent further incidents [5] [6] [7].

This research continues our previous research [8], which attempts to identify human action based on a single snapshot of an image. The recognition of human action based only on a single snapshot has some potential weaknesses, such as predicting action based on a single snapshot will have many interpretations since many actions can have similar poses if inferred from a single image. The next logical step is to predict human action based on the sequence of poses in videos, and they will be inferred using a sequence-based algorithm, which is LSTM, CNN-LSTM, and CONVLSTM.

2. RECENT STUDIES

Many kinds of research on human action recognition emphasize using some types of networks or frameworks. These models/frameworks include skeleton edge motion networks [9], Depth Map Sequences [10], and deep view-invariant human action recognition framework [11]. Spatial-temporal dual-attention network (STDAN) [*12*], recognition of objects through hands and arms using Bi-GRU [*13*], Attention-based Hybrid 2D/3D CNN-LSTM [*14*], LSTM+YOLOv4 [*15*], Inception inspired CNN-GRU hybrid network [*16*], frames in Videos using double-feature double-motion (DDNet) [*17*], ResNet with Bi-LSTM [*18*].

Temporal Attention-Augmented Graph Convolutional Network [19], the fusion of skeletal joint dynamics and structural features [20], Self-Attention Network [21], multi-stream 3D CNN structure [22], Mixed 3D/2D Convolutional Tube [23], Bayesian Hierarchical Model [24], Semantic-Guided Neural Networks [25], SRNet: Structured Relevance Feature Learning Network [26], Spatio-Temporal Graph Deconvolutional Network [27], Attention Residual 3D Network [28], histogram of Oriented Gradient-Based Fusion of Features [29]. Some of those models emphasize skeletal framework, and others emphasize neural networks.

Furthermore, many human action recognition models have been developed based on specific algorithms. Examples of these models encompass CNN-Based Multistage Gated Average Fusion (MGAF) using depth and inertial sensors [30], Learning Graph Convolutional Network [31], Convolutional LSTM with Spatio-temporal networks [32]. As well as Bidirectional LSTM [33], Densely-connected Bi-directional LSTM (DB-LSTM) [34], Depthwise Spatio-Temporal STFT Convolutional Neural Networks for Human Action Recognition [35], deep ensemble learning in still images [36], Deep Learning Method in limited sensory data [37], Deep Belief Network [38], Geometric Deep Neural Network [39], the union of deep learning and swarm-based optimization [40].

Algorithms such as CNN and LSTM are among the favored ones. By using deep learning models such as CNN, LSTM, or two-stream networks, objects are detected and classified, but still, models lack contextual information during activity [*41*], and they are limited to standing, sitting, downstairs, upstairs, jogging, and walking [*42*], but objects and their skeleton based action were not focusing on many actions not reported

The majority of models are somewhat time-consuming to infer. Similarly, the study of the previous researchers is lacking by not implementing real-time inference. Even though the objects that the model identifies are only images of human poses, they, unfortunately, lead to an extensive inference. This condition exists because the method requires a substantial number of resources. The new study proposed by the authors focuses on how to infer categorization in real-time. This study differs from past studies on the inference engine's real-time nature. Even if the objects represent human movements, the model must be lightweight to permit the real-time inference of the action.

3. RESEARCH METHODOLOGY

To build the model as the main building block for the inference engines. These steps are implemented:

1. Collect action videos: Use video clips of actions as the data source. Convert the video clips into data sets for model training and testing.
2. Model building, evaluation, and selection: Build the model using the datasets and the selected sequence-based classification algorithms. The best model from each algorithm is then implemented to get the inference engines.
3. Building inference engines: Build inference engines for each algorithm.

A. Collect Action Videos

The videos for data sources are collected from YouTube with the following assumptions about which of the videos of action should be saved and collected for further processing:

1. A sequence of frames constitutes action.
2. The is only a single person in the video clip.
3. One sequence of action consists of 30 frames.

The collected videos are processed using OpenPose [43] to get the keypoints information, and then those keypoints are saved into a CVS file for further processing to be ready for the next phase, which is model building.

B. Model Building, Evaluation, and Selection

The next step is building the classification model. The model is built using a Personal Computer with Ubuntu 20.04 LTS Operating System, CUDA 11.5 with CUDNN version 7.5 and 8, Caffee and OpenCV support, Python3, OpenPose, and TensorFlow. The PC's hardware is CPU Ryzen 7 1700X (8 core, 16 threads) with single GPU configurations using NVIDIA GTX1660 Ti 6GB and NVIDIA RTX3060 12GB.

The python script used in the research is the adaptation from [45] with some modifications based on the collected dataset. Three sequence-based classification algorithm models are selected: LSTM, CNN-LSTM, and CONVLSTM. These three algorithms are selected since LSTM is the more advanced version of the Recurrent Neural Network algorithm, and the CNN-LSTM and CONVLSTM are slight variations of the algorithm, which may provide better performance and accuracy.

For the LSTM model, the model architecture is constructed using python code as shown below:

```
model = Sequential()
model.add(LSTM(120,
input_shape=(n_timesteps,n_features)))
model.add(Dropout(0.7))
model.add(Dense(80, activation='relu'))
model.add(Dense(n_outputs, activation='softmax'))
model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
```

Whereas the model fitting is:

```
model.fit(trainX, trainy, epochs=epochs,
batch_size=batch_size, verbose=verbose)
```

For the CNN-LSTM model, the model architecture is constructed using python code as shown below:

```
model = Sequential()
model.add(TimeDistributed(Conv1D(filters=64,
kernel_size=3, activation='relu'),
input_shape=(None,n_length,n_features)))
model.add(TimeDistributed(Conv1D(filters=64,
kernel_size=3, activation='relu')))
model.add(TimeDistributed(Dropout(0.5)))
model.add(TimeDistributed(MaxPooling1D(pool_siz
e=2)))
model.add(TimeDistributed(Flatten()))
model.add(LSTM(100))
model.add(Dropout(0.5))
model.add(Dense(120, activation='relu'))
model.add(Dense(n_outputs, activation='softmax'))
model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
```

Also, the model fitting is:

```
model.fit(trainX, trainy, epochs=epochs,
batch_size=batch_size, verbose=verbose)
```

Finally, for the CONVLSTM model, the model architecture is constructed using python code as shown below:

```
model = Sequential()
model.add(ConvLSTM2D(filters=64,
kernel_size=(1,3), activation='relu',
input_shape=(n_steps, 1, n_length, n_features)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(80, activation='relu'))
model.add(Dense(n_outputs, activation='softmax'))
model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
```

In this case, the model fitting is:

```
model.fit(trainX, trainy, epochs=epochs,
batch_size=batch_size, verbose=verbose)
```

The hyperparameters of all the LSTM, CNN-LSTM, and CONVLSTM models vary based on trial and error to find the best accuracy values. For the LSTM model, the hyperparameters modified during modeling trials were the memory unit, hidden layer, and dropout values. As for the CNN-LSTM model, modified hyperparameters are n_steps, n_length, memory unit, hidden layer, filter, and dropout values. Finally, for the CONVLSTM model, modified hyperparameters are n_steps, n_length, hidden layer, filters, kernel size, and dropout values. For all the models, the number of epochs and batch size also varied to find the best possible accuracy for the model. Each model is evaluated by observing the confusion matrix of each model, and the model with the best accuracy is saved. The fitting time during training is also observed and recorded for further analysis. Each model was also verified by testing using the unused dataset (dataset from Table I that is not being used for either training or testing). The best model from each algorithm was saved from being used for the inference engine.

C. Building Inference Engine

The last phase is building the inference engine for each model. The inference engines are built using TensorFlow and CUDA using single GPU and dual GPU configurations. The saved models and the TFLite

converted models are used to observe the performance and differences amongst these models. The inference engine was built to infer actions from recorded or live webcam video. The conversion from original models to TFLite models is an adaption from the script from Google Collaboration Research [44].

4. Result and Discussion

A. Collect Action Videos

Initially, 2771 video clips of actions were collected, consisting of 85772 frames. Each video clip is then truncated based on the action, with each truncated video clip consisting of 30 frames. After initial assumptions of 32 actions, the collection focuses only on seven pre-determined actions with sufficient datasets defined to be at least 500. The later data collection is focused on completing those seven actions to achieve the 500 datasets as the minimum requirement.

All the video clips are then processed with OpenPose [43] to get the sequence of keypoints during the actions. The OpenPose will extract 25 keypoints from each frame, and a single dataset is a sequence of keypoints (x and y coordinates) from 30 frames of action. The saved coordinates consists of 50 points which are x_Nose, y_Noze, x_Neck, y_Neck, x_RShoulder, y_RShoulder, x_RElbow, y_RElbow, x_RWrist, y_RWrist, x_LShoulder, y_LShoulder, x_LElbow, y_LElbow, x_LWrist, y_LWrist, x_MidHip, y_MidHip, x_RHip, y_RHip, x_RKnee, y_RKnee, x_RAnkle, y_RAnkle, x_LHip, y_LHip, x_LKnee, y_LKnee, x_LAnkle, y_LAnkle, x_REye, y_REye, x_LEye, y_LEye, x_REar, y_REar, x_LEar, y_LEar, x_LBigToe, y_LBigToe, x_LSmallToe, y_LSmallToe, x_LHeel, y_LHeel, x_RBigToe, y_RBigToe, x_RSmallToe, y_RSmallToe, x_RHeel, y_RHeel. Each coordinate is the offset at the top corner of the body box, as shown in Figure 1, and the x and y coordinates are then stored in CSV format for further processing, with the first column being the filename and the last column being the action name.

The next process is the data cleaning of the keypoints. The data cleaning process is conducted by removing bad datasets. Based on the data observations, the bad frame is determined by the following criteria: 1) frames that lost more than 11 keypoints (more than 44% of keypoints missing), 2) the dataset with more than two consecutive frames (missing more than 7% of the sequences), and 3) the dataset with less than 14 frames are also removed (missing more than 0.5 seconds of actions). These bad frames are removed from the dataset.

Figure 1. OpenPose Keypoints Coordinates and Body box

The next data cleaning process involves the imputation of the missing keypoints and missing frames. Like the previous approach from the research [8], the missing keypoints are then imputed with the closest keypoints. The missing frames consist of three conditions: missing first, middle, and last frames, with different treatments for each condition. The missing first frame is replaced with the keypoints from the second frame. Meanwhile, the missing last frame is replaced with the keypoints from the previous frame. Next, the missing midframes are replaced with the linear interpolation of keypoints from the adjacent frames. If the missing frame is only one, the interpolation is just replaced with the average values of adjacent frames, but if the missing frames are two, the interpolation replaces the missing dataset with the 0.33 and 0.67 summations of the adjacent frames. Finally, the last imputation is for the missing 16 or more frames; the imputation is conducted by taking the remaining frames from the previous adjacent sequence of actions (from the same video clip).

There are 5808 datasets due to the previous two data cleaning processes and data imputations. Table I summarizes 5808 datasets ready for the next research phase: model building.

TABLE I. DATASET FOR LSTM CLASSIFICATION

Action ID	Action Name	Dataset Total	Training Dataset	Testing Dataset	Alternate Testing Dataset
1	Dribble	1042	**400**	**100**	542
2	Kick	652	**400**	**100**	152
3	Run	1288	**400**	**100**	787
4	Sit	778	**400**	**100**	278
5	Squat	752	**400**	**100**	252
6	Stand	548	**400**	**100**	48
7	Walk	748	**400**	**100**	248
TOTAL		5808	**2800**	**700**	2307

The dataset is reduced to 500 datasets for each action. The reduction is to set equal values or a balanced dataset for each action to remove the possibility of a biased result. So, the total dataset is 3500 (500 datasets x 7 actions). For the model building, the data is split into 80% for training and 20% for testing (400 datasets for training and 100 datasets for testing). The remaining dataset is later used for further analysis to confirm the model's accuracy and is labeled as an alternate testing dataset.

B. Model Building, Evaluation, and Selection

***LSTM Model**:*

The LSTM model was trained with 150 epochs, batch size 64, and varying other hyperparameters to find the best model in terms of accuracy. The script is repeated more than 100 times to find the best possible hyperparameter, and it was then found that the best hyperparameters were LSTM 120, dense 80, and dropout 0.7. These hyperparameters were used to find the best possible model in terms of accuracy value.

Due to the heuristic nature of the LSTM algorithm, the script was repeated more than 2000 times using these selected hyperparameters to find the best model in accuracy. The training time of the LSTM model took an average of 38.232 to 45.637 seconds in each iteration. The model with the best accuracy of 83.1429% was discovered, with the training process consuming 931 MB of GPU memory. The Confusion Matrix for the best LSTM Model is shown in Table II.

Table III shows that LSTM models work best for predicting action 6 (Stand), followed by action 4 (Sit) and action 7 (Walk). The model has moderate prediction accuracy for action 3 (Run), Action 2 (Kick), and action 1 (Dribble) and the worst accuracy for action 5 (Squat). Table III shows the Classification Report for the best LSTM Model.

TABLE II. CONFUSION MATRIX FOR THE BEST LSTM MODEL

Predicted / Actual	1	2	3	4	5	6	7
1	**72**	10	16	0	0	1	1
2	11	**77**	11	0	0	1	0
3	12	5	**79**	0	1	0	3
4	0	3	0	**95**	2	0	0
5	2	14	6	0	**69**	6	3
6	0	1	0	0	0	**99**	0
7	3	3	2	0	0	1	**91**

Note: 1: Dribble, 2: Kick; 3: Run; 4: Sit, 5: Squat, 6: Stand, 7: Walk

TABLE III. CLASSIFICATION REPORT FOR THE BEST LSTM MODEL

Action ID	Precision	Recall	F1-Score	Support
1	0.72	0.72	0.72	100
2	0.68	0.77	0.72	100
3	0.69	0.79	0.74	100
4	**1.00**	**0.95**	**0.97**	**100**
5	**0.96**	0.69	0.80	100
6	**0.92**	**0.99**	**0.95**	**100**
7	**0.93**	**0.91**	**0.92**	**100**

Note: 1: Dribble, 2: Kick; 3: Run; 4: Sit, 5: Squat, 6: Stand, 7: Walk

Table III shows that the LSTM model has the best overall classification performance for action 4 (Dribble), action 6 (Stand), and action 7 (Walk). Action 5 (Squat) has good precision but moderate numbers for recall and F1-score values. The moderate recall value is due to some of the datasets for action 5 (Squat) being misclassified as other actions, and this also contributes to the moderate number of F1-score. The model was then further verified using the alternate testing dataset, which showed a similar value in accuracy.

CNN-LSTM Model:

The CNN-LSTM model was trained with 150 epochs, batch size 64, and varying other hyperparameters to find the best model in terms of accuracy. The script is repeated more than 100 times to find the best possible hyperparameter, and it was then found that the best hyperparameters were n_steps 3, n_length 10, LSTM 100, Dense 120, filters 64, and dropout 0.5. These hyperparameters were used to find the best possible model regarding accuracy value.

Due to the heuristic nature of the CNN-LSTM algorithm, the script was repeated more than 2000 times using these selected hyperparameters to find the best model in accuracy. The training time of the CNN-LSTM model took an average of 38.765 to 45.672 seconds in each iteration. The model with the best accuracy of 83.7143% was discovered, with the training process consuming 1219 MB of GPU memory. The Confusion Matrix for the best CNN-LSTM model is shown in Table IV.

TABLE IV. CONFUSION MATRIX FOR THE BEST CNN-LSTM MODEL

Predicted / Actual	1	2	3	4	5	6	7
1	**74**	7	18	0	1	0	0
2	9	**79**	12	0	0	0	0
3	8	5	**84**	0	0	0	3
4	0	3	0	**90**	7	0	0
5	4	11	5	3	**70**	2	5
6	0	0	0	0	2	**98**	0
7	3	1	5	0	0	0	**91**

Note: 1: Dribble, 2: Kick; 3: Run; 4: Sit, 5: Squat, 6: Stand, 7: Walk

It can be seen from Table IV that the CNN-LSTM Model works best for predicting action 6 (Stand), followed by action 7 (Walk) and action 4 (Dribble). The model has moderate prediction accuracy for actions 3 (Run), action 2 (Kick), action 1 (Dribble), and action 5 (Squat).

Table V shows the Classification Report for the best CNN-LSTM. In detail, Table V shows that the CNN LSTM model has the best overall classification performance for action 4 (Dribble), action 6 (Stand), and action 7 (Walk). Action 5 (Squat) has good precision but moderate numbers for recall and F1-score values. The moderate recall value is due to some of the datasets for action 5 (Squat) being misclassified as other actions, and this also contributes to the moderate number of F1-score. The model was then further verified using the alternate testing dataset, which showed a similar value in accuracy.

CONVLSTM Model:

The CONVLSTM model was trained with 150 epochs, batch size 64, and varying other hyperparameters to find the best model in terms of accuracy. The script is repeated more than 100 times to find the best possible hyperparameter, and it was then found that the best hyperparameters n_steps 5, n_length 6, filters 64, Dense 80, and dropout 0.4. These hyperparameters were used to find the best possible model in terms of accuracy value.

Due to the heuristic nature of the CONVLSTM algorithm, the script was repeated more than 2000 times using these selected hyperparameters to find the best model in accuracy. The training time of the CONVLSTM model took an average of 136.370 to 153.885 seconds in each iteration, which was more than three times longer than LSTM and CNN-LSTM. The model with the best accuracy of 83% was discovered, with the training process consuming 1113 MB of GPU memory. The Confusion Matrix for the best CONVLSTM Model is shown in Table VI.

TABLE V. CLASSIFICATION REPORT FOR THE BEST CNN-LSTM MODEL

Action ID	Precision	Recall	F1-Score	Support
1	0.76	0.74	0.75	100
2	0.75	0.79	0.77	100
3	0.68	0.84	0.75	100
4	**0.97**	**0.90**	**0.93**	**100**
5	**0.88**	0.70	0.78	100
6	**0.98**	**0.98**	**0.98**	**100**
7	**0.92**	**0.91**	**0.91**	**100**

Note: 1: Dribble, 2: Kick; 3: Run; 4: Sit, 5: Squat, 6: Stand, 7: Walk

TABLE VI. CONFUSION MATRIX FOR THE BEST CONVLSTM MODEL

Predicted / Actual	1	2	3	4	5	6	7
1	**78**	8	11	0	0	1	2
2	16	**76**	7	0	0	0	1
3	18	2	**75**	0	0	1	4
4	0	3	0	**92**	3	0	2
5	1	17	7	2	**69**	3	1
6	0	0	0	0	1	**99**	0
7	3	1	4	0	0	0	**92**

Note: 1: Dribble, 2: Kick; 3: Run; 4: Sit, 5: Squat, 6: Stand, 7: Walk

It can be seen from Table 6 that the CONVLSTM model works best for predicting action 6 (Stand), action 4 (Dribble), and action 7 (Sit). The model has moderate accuracy in predicting action 1 (Kick), action 2 (Kick), and action 3 (Run). The worst accuracy is for action 2 (Walk).

Table VII shows the Classification Report for the best CONVLSTM Model. In detail, Table VII shows that the CONVLSTM model has the best overall classification performance for action 4 (Dribble), action 6 (Stand), and action 7 (Walk). Action 5 (Squat) has good precision but moderate numbers for recall and F1-score values. The moderate recall value is due to some of the datasets for action 5 (Squat) being misclassified as other actions, and this also contributes to the moderate number of F1-score. The model was then further verified using the alternate testing dataset, which showed a similar value in accuracy.

FPS vs. Accuracy:

Since the models were built based on a dataset from 30 frames, the next step is to conduct sensitivity analysis based on the varying value of fps in the video. This analysis is conducted to anticipate the situation where the videos (recorded or from webcam) were not had an ideal framerate of 30 fps. This sensitivity analysis needs to be conducted to know the decrease in the accuracy for videos with less than 30 fps. The best models were then tested using videos with reduced framerates, assuming the acceptable accuracy was at least 70% as shown in Fig.2.

TABLE VII. CLASSIFICATION REPORT FOR THE BEST CONVLSTM MODEL

Action ID	Precision	Recall	F1-Score	Support
1	0.67	0.78	0.72	100
2	0.71	0.76	0.73	100
3	0.72	0.75	0.74	100
4	**0.98**	**0.92**	**0.95**	**100**
5	**0.95**	0.69	0.80	100
6	**0.95**	**0.99**	**0.97**	**100**
7	**0.90**	**0.92**	**0.91**	**100**

Note: 1: Dribble, 2: Kick; 3: Run; 4: Sit, 5: Squat, 6: Stand, 7: Walk

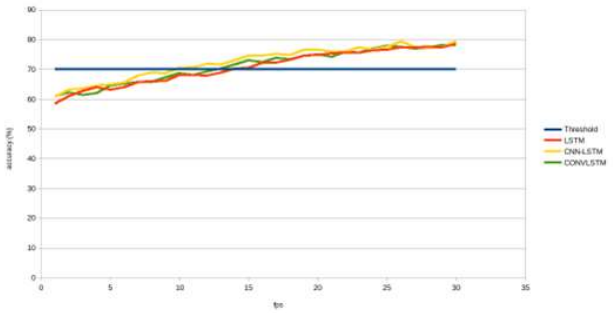

Figure 2. FPS vs. Accuracy of LSTM, CNN-LSTM, and CONVLSTM Models

Fig. 2 shows the accuracy values of LSTM, CNN-LSTM, and CONVLSTM for varying fps videos. It is shown that the CNN-LSTM Model reached the threshold values faster, starting from 10 fps, followed by the CONVLSTM model, starting from 13 fps, and the LSTM model reaching the threshold at 14 fps.

To conclude, during model building with all three models, the following observations are made:

1. The accuracy of LSTM, CNN-LSTM, and CONVLSTM models is similar and can reach 83% or more.
2. The CNN-LSTM model has the best accuracy, with 83.7143%, followed by the LSTM model, with 83.1429%, and the CONVLSTM model, with 83%. These accuracy values were discovered after more than 2000 iterations of model building.
3. The LSTM and CNN-LSTM models have similar training times, averaging 38 to 46 seconds in each model building. The CONVLSTM has the longest training time, averaging 136 to 154 seconds.
4. All models reached acceptable accuracy (accuracy value of 70% or more), with CNN-LSTM starting from 10 fps, CNVLSTM starting from 13 fps, and LSTM starting from 14 fps.

C. *Building inference Engines*

Inference time plays a crucial role in achieving real-time prediction. For 30 fps video, the maximum allowable time to compute time and infer is at most 33ms to make the video playback smoothness intact. However, the inference engines using the original models were found to be very slow since the inference times were 50ms for each frame in dual GPU configuration (GTX 1660Ti + GTX 1650) and 40ms for single GPU configuration (GTX 1660Ti). The inference engine based on these models cannot give a real-time action inference for videos with 30 FPS due to the slow inference times.

The saved models were then converted to the TFLite model resulting in faster inference times. The inference time of the LSTM model was 0.5ms, the CNN-LSTM model was 0.25ms, and the CONVLSTM model was 0.5ms. These inference times are acceptable for real-time inference, and the prediction can be conducted with a normal video speed of 30 frames per second. The inference engines can also predict the actions of up to 10 people in a single frame with limited tracking capabilities. The TFLite Model's accuracy is also compared to the original model, and it was found to have no difference in confusion matrices.

Figure 3. Inference Engine using TFLite version of LSTM Model

Fig. 3 is the LSTM inference engine snapshots using the TFLite version of the LSTM model. The inference shown in the frame is based on the last 30 frames of the videos and is continuously updated in every frame.

Figure 4. Inference Engine using TFLite version of CNN-LSTM Model

Fig. 4 is the snapshots of the CNN-LSTM inference engine using the TFLite version of the CNN-LSTM model. The inference shown in the frame is based on the last 30 frames of the videos and is continuously updated in every frame. The right figure shows the inference engine's capability to predict two people simultaneously.

Figure 5. Inference Engine using TFLite version of CONVLSTM Model

Fig. 5 is a snapshot of the CONVLSTM inference engine using the TFLite version of the CONVLSTM model. The inference shown in the frame is based on the last 30 frames of the videos and is continuously updated in every frame. The right figure shows the inference engine's capability to predict four people simultaneously.

5. CONCLUSION

This research can build the model using LSTM, CNN-LSTM, and CONVLSTM to predict seven real-time actions from video or webcam. The accuracies of the model are 80.26% for the LSTM model, 81.57% for the CNN-LSTM model, and 81.57% for the CONVLSTM model. The shortest model building time was the LSTM model, followed by the CNN-LSTM model, and the longest was the CONVLSTM model, which consumes four times longer. Inference engines for the three best models are successfully built by converting all saved models into the TFLite model. The inference times are 0.5ms for the TFLite version of the LSTM model, 0.25ms for the TFLite version of the CNN-LSTM model, and 0.5ms for the TFLite version of the CONVLSTM model. The short inference time makes it possible for real-time inference from streaming video or webcam.

The inference engines from LSTM, CNN-LSTM, and CONVLSTM show promising results in predicting action in real-time. The inference engines also design for inferring up to 10 people in a single frame, but the performance will be degraded significantly due to the limited computing capability. The inference engines can also trace the movements of each person in the image by using the shortest distance comparison. However, the tracing capability will suffer when two or more people overlap, confusing the tracking algorithm. More sophisticated algorithms for tracking and Multi-Stage Attention using GCN must be developed in further research.

ACKNOWLEDGMENT

The authors would like to thank Departemen Informatika, Fakultas Teknologi Informasi, and LPPM Universitas Atma Jaya Yogyakarta, Indonesia; also, Department of Computer and Information Sciences, Universiti Teknologi Petronas, Malaysia, for supporting this research.

REFERENCES

[1] Kumar, A., Zhang, Z. J., and Lyu, H. 2020. Object Detection in real time based on improved single shot multi-box detector algorithm. EURASIP Journal on Wireless Communications and Networking. Vol. 204. Pp 1-18

[2] Izhar, F., Ali, S., Ponum, M., Mahmood, M. T., Ilyas, H., and Iqbal, A. 2020. Detection & recognition of veiled and unveiled human face on the basis of eyes using transfer learning. Multimedia Tools and Applications.

[3] Rawal, K., and Sethi, G. 2020. Chapter 6: Medical Image Processing in Detection of Abdomen Diseases. Advancement of Machine Intelligence in Interactive Medical Image Analysis, Algorithms for Intelligent Systems. Springer Nature Singapore. Pp 153 – 166.

[4] Xu, S., Fang, J., Hu, X., Ngai, E., Wang, W., Guo, Y., & Leung, V. C. (2022). Emotion recognition from gait analyses: Current research and future directions. *IEEE Transactions on Computational Social Systems*.

[5] Afgari, A. P., Haque, M. M., and Washington, S. 2020. Applying a joint model of crash count and crash severity to identify road

segments with high risk of fatal and serious injury crashes. Accident Analysis and Prevention, Vol. 144. Pp 1 – 11

[6] Tian, J., Hu, N., Li, X., and Zhang, F. 2021. Research on a Computer Aid Risk Control System Using Classification Capabilities of Support Vector Machines. Proceeding of IPEC 2021 (Journal of Physics: Conference Series), Vol. 1952(2021) 042090

[7] Frasch, M. G., Strong, S. B., Nilosek, D., Leaverton, J., and Schifrin, B. S. 2021. Detection of Preventable Fetal Distress During Labor From Scanned Cardiotocogram Tracings Using Deep Learning. Frontiers in Paediatrics, Vol. 9, Article 736834

[8] Emanuel, A.W.R., Paulus, M., and Nugraha, J.A.M. 2021. Snapshot-Based Human Action Recognition using OpenPose and Deep Learning. IAENG International Journal of Computer Science, Vol. 48, Issue 4. Pp 862-867.

[9] Wang, H., Yu, B., Xia, K., Li, J., and Zuo, X. 2021. Skeleton edge motion networks for human action recognition. Neurocomputing, Vol. 423. Pp 1 – 12.

[10] Ji, X., Zhao, Q., Cheng, J., and Ma, C. 2021. Exploiting spatio-temporal representation for 3D human action recognition from depth map sequences. Knowledge-Based System, Vol. 227, 107040.

[11] Dhiman, C., and Vishwakarma, D. K. 2020. View-Invariant Deep Architecture for Human Action Recognition Using Two-Stream Motion and Shape Temporal Dynamics. IEEE Transactions on Image Processing, Vol. 29. Pp 3835 – 3844

[12] Zhang, Z., Lv, Z., Gan, C., & Zhu, Q. (2020). Human action recognition using convolutional LSTM and fully-connected LSTM with different attentions. *Neurocomputing*, *410*, 304-316.

[13] Tong, L., Ma, H., Lin, Q., He, J., & Peng, L. (2022). A novel deep learning bi-gru-i model for real-time human activity recognition using inertial sensors. *IEEE Sensors Journal*, *22*(6), 6164-6174.

[14] Bayoudh, K., Hamdaoui, F., & Mtibaa, A. (2022, January). An Attention-based Hybrid 2D/3D CNN-LSTM for Human Action Recognition. In *2022 2nd International Conference on Computing and Information Technology (ICCIT)* (pp. 97-103). IEEE.

[15] Lu, J., Nguyen, M., & Yan, W. Q. (2020, November). Deep learning methods for human behavior recognition. In *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)* (pp. 1-6). IEEE.

[16] Dua, N., Singh, S. N., Semwal, V. B., & Challa, S. K. (2022). Inception inspired CNN-GRU hybrid network for human activity recognition. *Multimedia Tools and Applications*, 1-35.

[17] Phan, H. H., Nguyen, T. T., Phuc, N. H., Nhan, N. H., Tran, C. T., & Vi, B. N. (2021, August). Key frame and skeleton extraction for deep learning-based human action recognition. In *2021 RIVF International Conference on Computing and Communication Technologies (RIVF)* (pp. 1-6). IEEE.

[18] Li, Y., & Wang, L. (2022). Human activity recognition based on residual network and BiLSTM. *Sensors*, *22*(2), 635.

[19] Heidari, N., and Iosifidis, A. Temporal Attention-Augmented Graph Convolutional Network for Efficient Skeleton-Based Human Action Recognition. Proceeding of 2020 25th International Conference on Pattern Recognition (ICPR).

[20] Muralikrishna, S. N., Muniyal, B., Acharya, U. D., and Holla, R. 2020. Enhanced Human Action Recognition Using Fusion of Skeletal Joint Dynamics and Structural Features. Hindawi Journal of Robotics, Vol. 2020, 16 pages

[21] Cho, S., Maqbool, M., Liu, F., and Foroosh, H. 2020. Self-Attention Network for Skeleton-based Human Action Recognition. Proceedings of the IEEE/CVF Winter on Applications of Computer Vision (WACV). Pp 635 – 644

[22] Chenarlogh, V. A., and Razzazi, F. 2018. Multi-stream 3D CNN structure for human action recognition trained by limited data. IET Computer Vision, Vol. 13, Issue 3. Pp 338 – 344

[23] Zhou, Y., Zha, Z. J., and Zen, W. 2018. MiCT: Mixed 3D/2D Convolutional Tube for Human Action Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018. Pp 449 – 458

[24] Zhao, R., Xu, W., Su, H., and Ji, Q. 2019. Bayesian Hierarchical Dynamic Model for Human Action Recognition. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2019. Pp 7733 – 7742

[25] Zhang, P., Lan, C., Zen, W., Xin. J., Xue, J., and Zheng, J. 2020. Semantics-Guided Neural Networks for Efficient Skeleton-Based Human Action Recognition. Proceedings of the IEEE/CVF Conference on Computer Vidio and Pattern Recognition (CVPR) 2020. Pp 1112 – 1121

[26] Nie, W., Wang, W., and Huang, X. 2019. SRNet: Structured Relevance Feature Learning Network From Skeleton Data for Human Action Recognition. IEEE Access, Vol. 7. Pp 132161 – 132172

[27] Peng, W., Shi, J., Zhao, G. 2021. Spatio Temporal Graph Deconvolutional Network for Skeleton-Based Human Action Recognition. IEEE Signal Processing Letter, Vol. 28. Pp 244 – 248

[28] Dong, M., Fang, Z., Li, Y., Bi, S., and Chen, J. 2021. AR3D: Attention Residual 3D Network for Human Action Recognition. MDPI Sensors, Vol. 21, 1656

[29] Patel, C., Labana, D., Pandya, S., Modi, K., Ghayvat, H., and Awais, M. 2021. Histogram of Oriented Gradient-Based Fusion of Features for Human Action Recognition in Action Video Sequences. MDPI Sensors, Vol. 20, 7299

[30] Ahmad, Z., and Khan, N. 2021. CNN-Based Multistage Gated Average Fusion (MGAF) for Human Action Recognition Using Depth and Inertial Sensors. IEEE Sensors Journal, Vol. 21, Issue 3. Pp 3623 – 3634

[31] Peng, W., Hong, X., Chen, H., and Zhao, G. 2020. Learning Graph Convolutional Neural Network for Skeleton-Based Human Action Recognition by Neural Searching. Proceeding of The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20). Pp 2669 – 2676

[32] Sarabu, A., and Santra, A. K. 2021. Human Action Recognition in Videos using Convolutional Long Short-Term Memory Network with Spatio-Temporal Networks. Emerging Science Journal, Vol. 5 No. 1. Pp 25 – 33

[33] Li, W., Nie, W., and Su, Y. 2018. Human Action Recognition Based on Selected Spatio-Temporal Features via Bidirectional LSTM. IEEE Access Special Section on Big Data Learning and Discovery, Vol. 6. Pp 44211 – 44220

[34] He, J.Y., Wu, X., Cheng, Z.Q., Yuan, Z., and Jiang, Y.G. 2021. DB-LSTM: Densely-connected Bi-directional LSTM for human action recognition. Neurocomputing, Vol. 444. Pp 319 – 331

[35] Kumawat, S., Verma, M., Nakashima, Y., and Raman, S. 2022. Depthwise Spatio-Temporal STFT Convolution Neural Network for Human Action Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 44, Issue 9. Pp 4839 – 4851

[36] Yu, X., Zhang, Z., Wu, L., Pang, W., Chen, H., Yu, Z., and Li, B. 2020. Deep Ensemble Learning for Human Action Recognition in Still Images. Hindawi Complexity, Vol. 20, 23 pages.

[37] Tufek, N., Yalcin, M., Altintas, M., Kalaoglu, F., Li, Y., and Bahadir, S. K. 2020. Human Action Recognition Using Deep Learning Methods on Limited Sensory Data. IEEE Sensors Journal, Vol. 20, Issue 6. Pp 3101 – 3112

[38] Abdellaoui, M., and Douik, A. 2020. Human Action Recognition in Video Sequences Using Deep Belief Network. Traitement du Signal, Vol. 37, No. 1. Pp 37 – 44

[39] Friji, R., Drira, H., Chaieb, F., Kchok, H., and Kurtek, S. 2021. Geometric Deep Neural Network using Rigid and Non-Rigid Transformations for Human Action Recognition. Proceedings of

the IEEE/CVF International Conference on Computer Vision (ICCV) 2021. Pp 12611 – 12620

[40] Basak, H., Kundu, R., Singh, P. K., Ijaz, M. F., Wozniak, M., and Sarkar, R. 2022. A union of deep learning and swarm-based optimization for 3D human action recognition. Scientific Reports, Vol. 12:5494

[41] Islam, M. M., Nooruddin, S., Karray, F., & Muhammad, G. (2022). Human activity recognition using tools of convolutional neural networks: A state of the art review, data sets, challenges, and future prospects. *Computers in Biology and Medicine*, 106060.

[42] Shiranthika, C., Premakumara, N., Chiu, H. L., Samani, H., Shyalika, C. & Yang, C. Y. (2020, December). Human Activity Recognition Using CNN & LSTM. In *2020 5th International Conference on Information Technology Research (ICITR)* (pp. 1-6). IEEE.

[43] Chao, Z., Hidalgo, G., Simon, T., Wei, S. -E., and Sheik, Y. 2017. Real-time Multi-Person 2D Pose Estimation using Part Affinity Fields. Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Pp7291-7299

[44] Google Collaboration, 2022. Keras LSTM fusion Codelab.ipny. Google Collaboration Research. Accessed: 1 May 2022. Available: https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/lite/examples/experimental_new_converter/Keras_LSTM_fusion_Codelab.ipynb#scrollTo=Z7gEg4DRBwbO

[45] Brownlee, J. 2018. LSTM for Human Activity Recognition Time Series Classification. Machine Learning Mastery: Making Developers Awesome at Machine Learning. 24 September 2018. Available: https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/

Paulus Mudjihartono is an Associate Professors (junior scale) in Informatics Department of Universitas Atma Jaya Yogyakarta, Indonesia. He has more than 20 years of experience in teaching and research. His area of interest includes computational intelligence, data science, and parallel computing.

Andi W.R. Emanuel is a Professor in Informatics Department, Universitas Atma Jaya Yogyakarta. He has more than 20 years of experience in teaching and research. His areas of interest include Software Engineering, Open-Source Software, Software Metrics, Software Quality, Information Systems, Knowledge Discovery, and Artificial Intelligence.

Joanna Ardhyanti Mita Nugraha is a lecturer at the Informatics, Faculty of Industrial Technology, Atma Jaya University, Yogyakarta. Completed undergraduate education in the Information Systems Department and continued Masters in the Information Systems Masters Department. The author specializes in data mining, machine learning and deep learning.

Fedelis Brian Putra Prakasa is a lecturer in the computer science study program at Universitas Atma Jaya Yogyakarta. He has four years of teaching and research experience. It focuses on major research areas with topics like UI/UX, gamification, and mobile development, while minor research areas focus on data visualization and machine learning.

Shuib Basri is currently a Senior Lecturer with the Department of Computer and Information Sciences Universiti Teknologi Petronas, Malaysia, lecturing undergraduate and post-graduates and supervising research for Master's and Doctoral students. In addition to the above, he is also a cluster leader for the Software engineering cluster mandated to manage the department's education program and any related research and innovation activities. He has been the external examiner of Master's and Doctoral theses and dissertations for various national and international universities. He is also a selected assessor and reviewer for various national research grants and program accreditation in Malaysia, namely MOSTI and MBOT national bodies.