TEKNIK

Tersedia online di: http://ejournal.undip.ac.id/index.php/teknik

TEKNIK, 43 (2), 2022, 190-201

Human Action Recognition (HAR) Classification Using MediaPipe and Long Short-Term Memory (LSTM)

Ichsan Arsyi Putra *, Oky Dwi Nurhayati , Dania Eridani

Departemen Teknik Komputer Fakultas Teknik, Universitas Diponegoro,
Jl. Prof. Soedarto, S.H., Kampus UNDIP Tembalang, Semarang, Indonesia 50275

Abstract

Human Action Recognition is an important research topic in Machine Learning and Computer Vision domains. One of the proposed methods is a combination of MediaPipe library and Long Short-Term Memory concerning the testing accuracy and training duration as indicators to evaluate the model performance. This research tried to adapt proposed LSTM models to implement HAR with image features extracted by MediaPipe library. There would be a comparison between LSTM models based on their testing accuracy and training duration. This research was conducted under OSEMN methods (Obtain, Scrub, Explore, Model, and iNterpret). The dataset was preprocessed Weizmann dataset with data preprocessing and data augmentation implementations. Video features extracted by MediaPipe: Pose was used in training and validation processes on neural network models focusing on Long Short-Term Memory layers. The processes were finished by model performance evaluation based on confusion matrices interpretation and calculations of accuracy, error rate, precision, recall, and F1score. This research yielded seven LSTM model variants with the highest testing accuracy at 82%, taking 10 minutes and 50 seconds of training duration.

Keywords*: classification; deep learning; human action recognition; mediapipe; long short-term memory*

1. Introduction

The development of computer technology in machine learning and computer vision domain is still far from being done. The uniqueness of various imagery data from various sources becomes interesting research material. Information gathering from images can be done through Human Action Recognition (HAR). HAR is an important issue due to its various implementations, e.g., surveillance videos, human-machine interaction, and other ways of information-gathering from videos (Cheng et al., 2015). A proposed method used a combination of MediaPipe as image features detector and extractor along with Long Short-Term Memory (LSTM) as an identifier or classifier. This combination can be found in Hand Gesture Recognition (HGR) research (Agrawal et al., 2022; Ghosh, 2021; Lakkapragada et al., 2022; Moetia Putri & Fuadi, 2022). HAR can also be conducted with similar methods (Daniel Tanugraha et al., 2022; Zhang et al., 2017).

*) Corresponding author.
E-mail: ichsan.x.an@outlook.co.id

Zhang had done previous research in HAR (Zhang et al. 2017) used NTU, SBU, and SYSU datasets consisting of sequences of skeleton 3D data. These data were used as input for LSTM model constructed by 3 LSTM layers with a Fully-connected layer as a classifier. This implementation results in accuracies of 87.6%, 97.2%, and 77.5% for NTU, SBU, and SYSU datasets, respectively.

Ghosh conducted research with a classification in 5 classes from a dataset of 126 videos (Ghosh, 2021). The video features were extracted by MediaPipe: Hands with ignored z-axis. The LSTM architecture consisted of 2 LSTM layers, 2 dropout layers, 1 flatten layer, and 1 dense layer. This research results in an accuracy of 94%.

With a combination of one layer for each LSTM layer, dropout layer, and dense layer, this research (Lakkapragada et al., 2022) could obtain a model with a testing accuracy of 69.55%. The input data was gathered from the extraction of the Self-Stimulatory Behavior Dataset (SSBD) with MediaPipe.

Research in Sports Action Recognition Based on Long Short-Term Memory Using MediaPipe (Daniel Tanugraha et al., 2022) showed that their LSTM model

doi: 10.14710/teknik.v43i2.46439

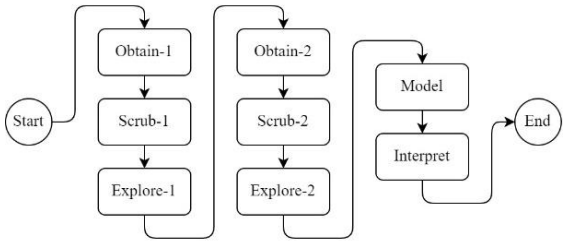


Fig. 1. Research workflow.

needed a training time of 10 to 12 minutes. This model used RNN for Human Activity Recognition-2D dataset for its training phase. The validation accuracies in T-Pose, Warrior II Pose, and Tree-Pose were obtained at 100%, 85%, and 80%, respectively.

Agrawal et al. (2022) did research in HGR consisting of 10 gestures using MediaPipe: Holistic. The model which implemented 4 LSTM layers and 3 Dense layers could result in testing and validation accuracies of 90%.

Another HGR research using LSTM was conducted by Putri et al. (2022). They used the BISINDO gesture dataset consisting of 30 vocabularies. The gesture was extracted with MediaPipe: Holistic in advance before it was set as input data for 3 variants of LSTM model, i.e., 1 layer LSTM, 2 layers LSTM, and Bidirectional LSTM. The highest accuracy reached 94%, 97%, and 96% for 1 layer LSTM, 2 layers LSTM, and Bidirectional LSTM, respectively.

Those researches showed that detection accuracy was the primary indicator in model evaluation. However, the training time as an essential factor in Deep Learning (Sarker, 2021) model construction was shown only in (Daniel Tanugraha et al., 2022). Other research in Deep Learning showed that the accuracy value and training time could be used to determine the best model (An et al., 2019; Codreanu et al., 2017; Tan & Le, 2021).

Based on this condition, this research tried to adapt the LSTM model (Ghosh, 2021; Lakkapragada et al., 2022; Zhang et al., 2017) for HAR implementation with landmarks data extracted with MediaPipe library (Google LLC, 2020). We also made an accuracy and training time comparison between those models and our self-constructed models based on parameters recommended by research (Reimers & Gurevych, 2017).

2. Research Method

2.1 Research Tools Specification

This research was conducted in a Lenovo Ideapad 100-14IBD laptop series for hardware powered with Intel Core i35005U processor, Intel HD Graphics 5500 VGA, and 6 GB of RAM. In the software aspect, this research implemented in Microsoft Windows 10 Pro 64-bit operating system, Jupyter Notebook 6.4.5 Anaconda 3 for IDE, python 3.9 programming language, and Kdenlive 21.12.3 as video editing application.

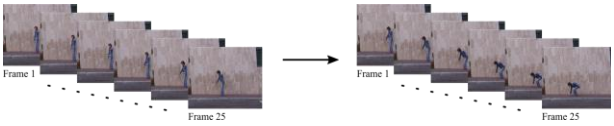


Fig. 2. The first second (25 frames) of "ira_bend.avi" video before (left figure) and after (right figure) duration cutting.

2.2 Research Workflow

The workflow for this research adopted the OSEMN model. OSEMN (read 'awesome') is a model or a method in data science introduced by Masson and Wiggins (Mason & Wiggins, 2010). OSEMN consisted of chronological steps called Obtain, Scrub, Explore, Model, and iNterpret. Since the model scheme of OSEMN used to be implemented with a customized sequence (Janssens, 2021), the research method followed the workflow as shown in Figure 1.

2.2.1 Obtain-1

The dataset used in this research was Weizmann dataset (Gorelick et al., 2007) which was directly downloaded from its official website at https://www.wisdom.weizmann.ac.il/~vision/SpaceTimeActions.html, in the "Classification Database" section. It has 336 MB of compressed size or 454 MB in uncompressed form.

The weizmann dataset consists of 10 classes with a total of 93 videos. Each class has 6 videos (jack, jump, pjump, side, wave2, wave1, and bend class) or 10 videos (run, walk, and skip class). Those videos are formatted in AVI with 1 to 3 seconds of duration, 25 fps of framerate, 180×144 pixels of frame size, and 9 persons of actors.

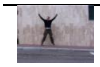
2.2.2 Scrub-1

As a prerequisite, the dataset used in this research was gathered from 25 former frames of each video. However, there was a video titled "ira_bend.avi" unfulfilling mentioned prerequisite, hench the preprocessing technique (Minh et al., 2018) called features selection (Beniwal et al., 2012) are applied to that video. This technique included a duration cutting implementation on 20 former frames, so a representative video was obtained compared to its class. Figure 2 shows an example of the duration cutting technique.

The Weizmann dataset was known for having 10 classes with 9 to 10 videos each. Due to the small-sized dataset (Wang et al., 2017) for 60:20:20 division ratio and the imbalanced data distribution, this research also implemented data augmentation techniques to the dataset through Kdenlive video editor. These techniques were

doi: 10.14710/teknik.v43i2.46439

Table 1. An example of data augmentation processes.

Original Video	Data Augmentation Process			
	Video #1	Video #2	Video #3	Video #4
	 Zooming: 150% Translation: (x,y) = (0,+20)	 Zooming: 150% Translation: (x,y) = (+45,+20)	 Zooming: 150% Translation: (x,y) = (+45,+20) Mirroring: horizontal	 Zooming: horizontal

mirroring, zooming, translation, and their combinations (Verdhan, 2021). All videos were augmented 1, 2, or 4 times. Table 1 shows an example of data augmentation processes.

2.2.3 Explore-1

After Scrub-1 was done, the dataset had 450 videos (45 videos per class) with 461 MB of data size. Since the data ratio was divided into 60 : 20 : 20 for training : validation : testing, the training data consisted of 270 videos (27 videos per class), the validation data consisted of 90 videos (9 videos per class), and the testing data had the same number of videos as the validation data that was 90 videos (9 videos per class).

2.2.4 Obtain-2

In Obtain-2, there were detection processes and video feature extraction using MediaPipe: Pose library. The detection phase was done by first converting video frames from BGR to RGB format using OpenCV (Bradski, 2000). These frames were then used for landmarks detection by MediaPipe.

The gathered data were 33 points of landmarks or key points with x, y, and z values in each point multiplied by the number of videos. Those data were saved in 25 NumPy array formatted files (.npy) for each video. Figure 3 illustrates the process of Obtain-2.

Table 3. The array dimension of data y

Data	Dimension	
	Video	Class
y_train	270	10
y_valid	90	10
y_test	90	10

Table 2. The array dimension of data X.

Data	Dimension		
	Video	Frame	Key points
X_train	270	25	132
X_valid	90	25	132
X_test	90	25	132

2.2.5 Scrub-2

In this step, the data obtained from Obtain-2 were labeled with a class code for every 25 frames of data (Amershi et al., 2019). The class codes mentioned ranged from 0 to 9 for the run, walk, skip, jack, jump, pjump, side, wave2, wave1, and bend, respectively.

2.2.6 Explore-2

The data obtained from Scrub-2 were multidimensional arrays for each training, validation, and testing data. Those data were distinguished into 2 groups, namely X and y. Table 2 shows the array dimension of data X. Meanwhile, data y had the array dimension, as shown in Table 3 below.

2.2.7 Model

The neural network model design was adapted from 4 LSTM models used in research (Ghosh, 2021; Lakkapragada et al., 2022; Zhang et al., 2017). We determined those models as VA-LSTTM-SYSU (Zhang et al., 2017), VA-LSTM-SBU (Zhang et al., 2017), LSTM-PASL (Ghosh, 2021), and LSTM-AHM (Lakkapragada et al., 2022).

a. VA-LSTM-SYSU

2 LSTM layers constructed this model with 100 units of neurons and 3 Dropout layers with a 50% rate. The Adam optimizer was used with a 0.005 learning rate and Gradient Clipping techniques by clipnorm parameter setting (Keras, n.d.-a). This model was trained with 64 batches size in 200 epochs. The architecture of VA-LSTM-SYSU can be seen in Figure 4.

b. VA-LSTM-SBU

The model of VA-LSTM-SBU had a similar structure to VA-LSTM-SYSU with neuron number adjustment to 50 and batch size to 8. VA-LSTM-SBU architecture is shown in Figure 5.

c. LSTM-PASL

LSTM-PASL model constructed by 2 LSTM layers with 256 and 128 neurons for each layer. This model implemented 2 dropout layers with a 20% rate. The training process was done by Adam optimizer with a 0.0001

Fig. 3. An illustration of the features extraction process using MediaPipe: Pose library.

doi: 10.14710/teknik.v43i2.46439

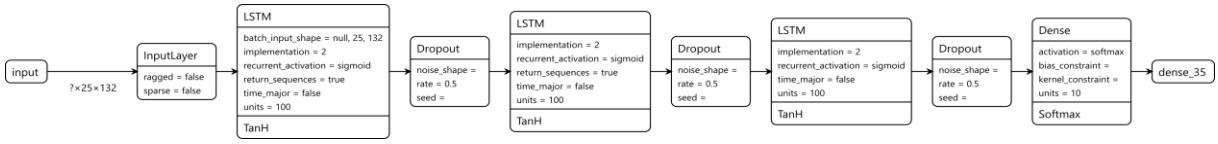


Fig. 4. VA-LSTM-SYSU architecture.

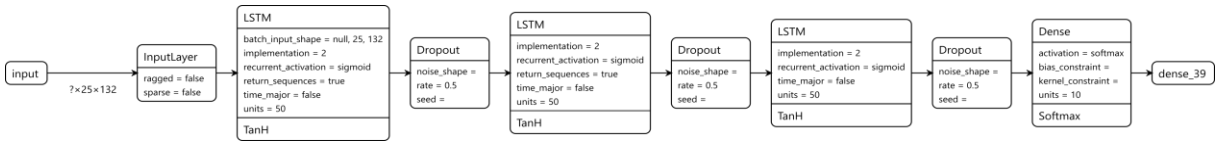


Fig. 5. VA-LSTM-SBU architecture.

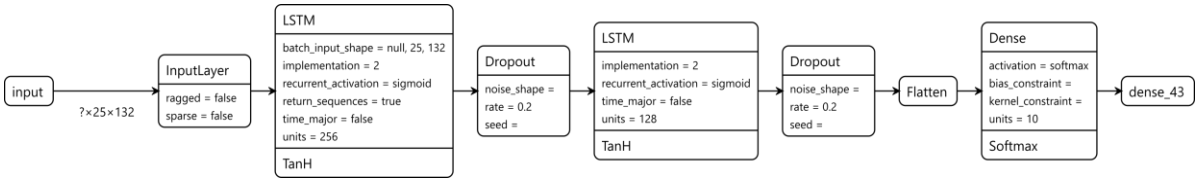


Fig. 6. LSTM-PASL architecture.

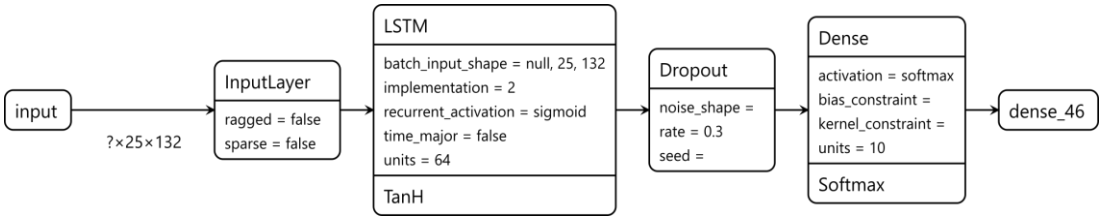


Fig. 7. LSTM-AHM architecture.

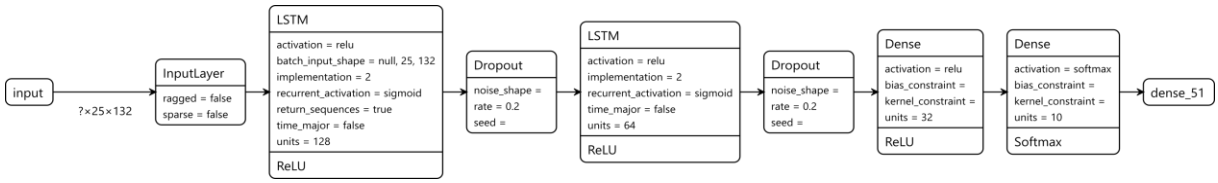


Fig. 8. Model 1 architecture.

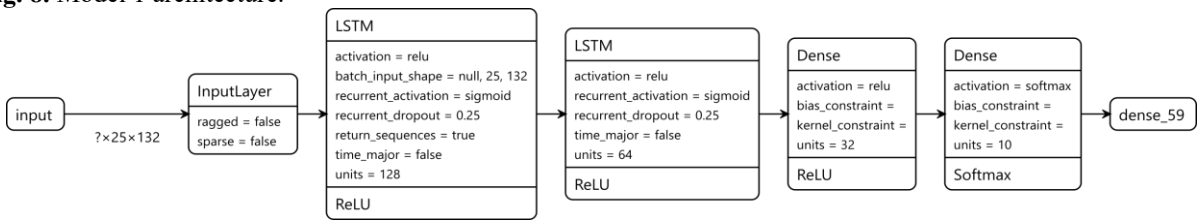


Fig. 9. Model 2 architecture.

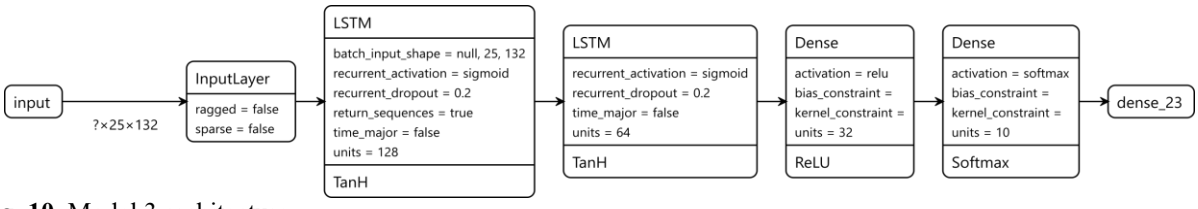


Fig. 10. Model 3 architecture.

learning rate, 32 batches size, and 200 epochs. Figure 6 shows the architecture of LSTM-PASL.

d. LSTM-AHM

LSTM-AHM had only an LSTM layer with 64 units of neurons pairing with a 30% rated Dropout layer. This model would be trained with Adam optimizer,

doi: 10.14710/teknik.v43i2.46439

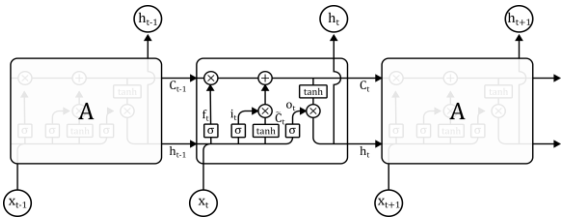


Fig. 11. An illustration of the LSTM algorithm (Olah, 2015).

learning rate = 0.01, batch size = 32, and 200 epochs. This model architecture can be seen in Figure 7.

Furthermore, we also designed 3 variants of models as a comparison to the former models. Those models had several same hyperparameters configuration, namely, (25, 132) for input shape, Nadam optimizer, categorical crossentropy for the loss function, categorical accuracy metrics, 200 epochs, and True value for the shuffle.

In addition to initialized hyperparameters above, the models were constructed with some recommendations, such as keeping LSTM layers in minimum number, the usage of dropout (especially a variational dropout), and small batch size (Reimers & Gurevych, 2017). To optimize the models, this research used Model Checkpoint (Keras, n.d.-b) function in the training process to save the best model weight for each epoch based on the value of validation categorical accuracy.

Referring to the recommendations above, three neural network models were designed.

a. Model 1

This model contained 2 LSTM layers, 2 Dropout layers, and 2 Dense layers. The dropout layers were positioned after each LSTM layer to prevent the overfitting condition probability (McCullum, 2020). Figure 8 shows the configurations of Model 1 architecture with its hyperparameters. Other hyperparameter configurations in Model 1 were 0.0001 for learning rate and 4 for batch size.

b. Model 2

Figure 9 shows Model 2 constructed of 2 LSTM layers and 2 Dense layers. The dropout values were configured as variational dropout by the recurrent_dropout variable. Other hyperparameters configured in Model 2 were 0.000075 for learning rate and 2 for batch size.

c. Model 3

In Model 3, the architecture was constructed similar to Model 2's yet used a different activation function in its LSTM layers, that was TanH (default value). The recurrent_dropout value also was reduced to 0.2. Figure 10 shows the architecture of Model 3 followed by its hyperparameters. Other hyperparameters were left unchanged with the same value as Model 2's.

2.2.8 Interpret

The interpretation was conducted by comparing the model performance based on accuracy and loss value from the training, validation, and testing process. A confusion matrix method was used to evaluate the testing process (Xu et al., 2020), followed by the calculation of accuracy, error rate, precision, and recall F_1score. Each calculation implemented micro-averaging and macro-averaging methods for multi-class classification problems (Chinchor, 1992; Sokolova & Lapalme, 2009).

After those processes, this research was continued with a simple prediction implementation in a demo video. The demo video had 640 x 360 pixels of resolution and 25 fps of framerate. An actor acted the actions in the demo video with 2 variations for each action. Each variation of action was performed in 2 seconds.

Each trained model would be implemented sequentially, and then the results were predicted based on key points values in the latest 25 frames of video. The changes in detection would be recorded and interpreted in the form of narrational text.

2.3 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a variety of Recurrent Neural Networks (RNN) designed for a temporal-dependent model with better accuracy than traditional RRN (Sak et al., 2014). LSTM was first introduced by Hochreiter and Schmidhuber (Hochreiter & Schmidhuber, 1997) to address error back-flow problems, i.e., blow up or vanish on the backpropagation method.

The visual of the LSTM algorithm is illustrated in Figure 11. In mathematic form, LSTM has a calculation sequence involving forget gate (f_t), input gate (i_t), new value that can be added to the cell state ($\tilde{C}_t$), cell state (C_t), output gate (o_t), and output order-t (h_t). Firstly, f_t was calculated with Equation 1.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

with σ is a sigmoid function, W_f is weight value for f_t, h_{t-1} is output value before order-t, x_t is input value in order-t, and b_f is bias value for f_t.

After f_t calculation, the data are processed with i_t through Equation 2 and $\tilde{C}_t$ in Equation 3.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3)$$

with W_i is weight value for i_t, W_C is weight value for $\tilde{C}_t$, b_i is bias in i_t, dan b_C is bias in $\tilde{C}_t$.

After f_t, i_t, dan $\tilde{C}_t$ are obtained, the C_t can be calculated through Equation 4.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (4)$$

with C_{t-1} is cell state value before order-t.

The o_t value is obtained with Equation 5, then the

(a) (b)

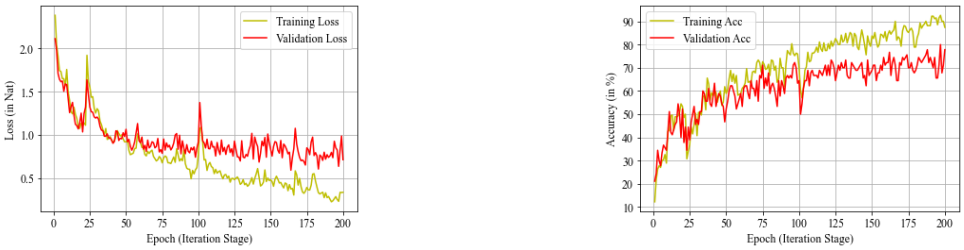
Fig.12. Loss (a) and accuracy (b) charts in training and validation processes of VA-LSTM-SYSU

(a) (b)

Fig.13. Loss (a) and accuracy (b) charts in training and validation processes of VA-LSTM-SBU.

(a) (b)

Fig.14. Loss (a) and accuracy (b) charts in training and validation processes of LSTM-PASL.



(a) (b)

Fig.15. Loss (a) and accuracy (b) charts in training and validation processes of LSTM-AHM

(a) (b)

Fig.16. Loss (a) and accuracy (b) charts in training and validation processes of Model 1.

output value is calculated with Equation 6 to obtain an output value for order-t.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * \tanh(C_t) \quad (6)$$

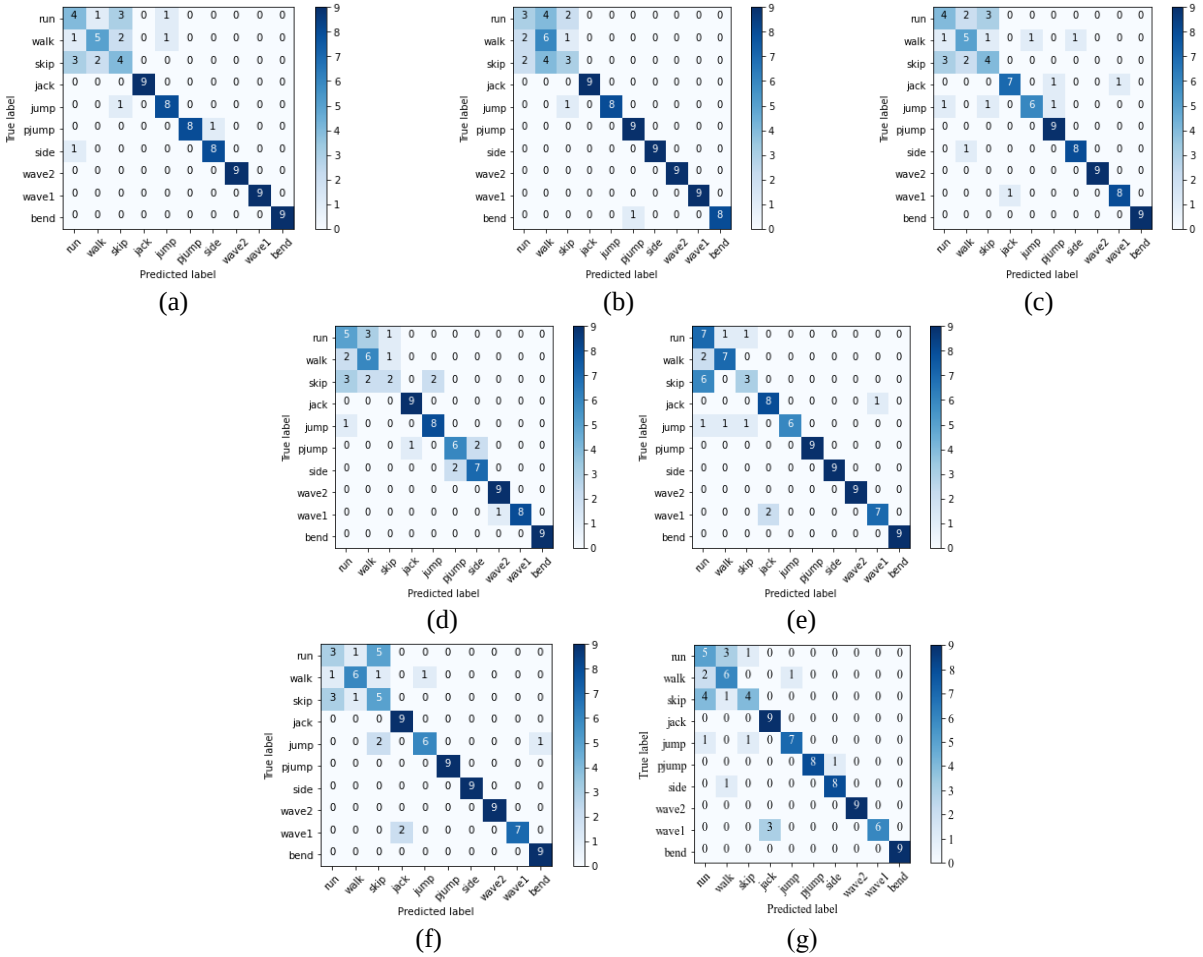
doi: 10.14710/teknik.v43i2.46439

(a) (b)

Fig. 17. Loss (a) and accuracy (b) charts in training and validation processes of Model 2.

(a) (b)

Fig.18. Loss (a) and accuracy (b) charts in training and validation processes of Model 3.



(a) (b) (c)

(d) (e)

(f) (g)

Fig.19. The confusion matrices resulted in the testing process on VA-LSTM-SYSU (a), VA-LSTM-SBU (b), LSTM-PASL (c), LSTM-AHM (d), Model 1 (e), Model 2 (f), and Model 3 (g).

doi: 10.14710/teknik.v43i2.46439

Table 4. TP, FP, TN, and FN values of VA-LSTM-SYSU, VA-LSTM-SBU, LSTM-PASL, and LSTM-AHM testing results.

Class Name	VA-LSTM-SYSU				VA-LSTM-SBU				LSTM-PASL				LSTM-AHM			
	TP	FP	TN	FN	TP	FP	TN	FN	TP	FP	TN	FN	TP	FP	TN	FN
run	4	5	76	5	3	4	77	6	4	5	76	5	5	6	75	4
walk	5	3	78	4	6	8	73	3	5	5	76	4	6	5	76	3
skip	4	6	75	5	3	4	77	6	4	5	76	5	2	2	79	7
jack	9	0	81	0	9	0	81	0	7	1	80	2	9	1	80	0
jump	8	2	79	1	8	0	81	1	6	1	80	3	8	2	79	1
pjump	8	0	81	1	9	1	80	0	9	2	79	0	6	2	79	3
side	8	1	80	1	9	0	81	0	8	1	80	1	7	2	79	2
wave2	9	0	81	0	9	0	81	0	9	0	81	0	9	1	80	0
wave1	9	0	81	0	9	0	81	0	8	1	80	1	8	0	81	1
bend	9	0	81	0	8	0	81	1	9	0	81	0	9	0	81	0

Table 5. TP, FP, TN, and FN values of our proposed LSTM models (Model 1, Model 2, and Model 3) testing results.

Class Name	Model 1				Model 2				Model 3			
	TP	FP	TN	FN	TP	FP	TN	FN	TP	FP	TN	FN
run	7	9	72	2	3	4	77	6	5	7	74	4
walk	7	2	79	2	6	2	79	3	6	5	76	3
skip	3	2	79	6	5	8	73	4	4	2	79	5
jack	8	2	79	1	9	2	79	0	9	3	78	0
jump	6	0	81	3	6	1	80	3	7	1	80	2
pjump	9	0	81	0	9	0	81	0	8	0	81	1
side	9	0	81	0	9	0	81	0	8	1	80	1
wave2	9	0	81	0	9	0	81	0	9	0	81	0
wave1	7	1	80	2	7	0	81	2	6	0	81	3
bend	9	0	81	0	9	1	80	0	9	0	81	0

Table 6. Measurement results of classification quality by each model.

Measure	VA-LSTM-SYSU	VA-LSTM-SBU	LSTM-PASL	LSTM-AHM	Our Proposed LSTM Models		
					Model 1	Model 2	Model 3
Average Accuracy	81%	81%	77%	77%	82%	80%	79%
Error Rate	19%	19%	23%	23%	18%	20%	21%
Precision_{μ}	81%	81%	77%	77%	82%	80%	79%
Recall_{μ}	81%	81%	77%	77%	82%	80%	79%
$\text{F}_1\text{score}_{\mu}$	81%	81%	77%	77%	82%	80%	79%
$\text{Precision}_{\text{M}}$	82%	82%	77%	76%	85%	81%	81%
Recall_{M}	82%	81%	77%	77%	82%	80%	79%
$\text{F}_1\text{score}_{\text{M}}$	82%	81%	77%	76%	82%	80%	79%
Training Time	2′ 33″	5′ 35″	5′ 52″	1′ 4″	10′ 50″	54′ 13″	24′ 52″

is a TanH function.

Meanwhile, the weight (W) is calculated through Equation 7.

$$W = (-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}) \quad (7)$$

with d is the amount of data.

3. Results and Discussion

3.1 Results Analysis

This research presented reports from our seven models' training, validation, and testing phase. For training and validation, some charts of loss and accuracy rate were compared to their epoch stages. The trained and weighted models were saved locally as .h5 files by considering the highest validation categorical accuracy in

doi: 10.14710/teknik.v43i2.46439

Table 7. Detection changing sequence by each model.

Action	VA-LSTM-SYSU	VA-LSTM-SBU	LSTM-PASL	LSTM-AHM	Our Proposed LSTM Models		
					Model 1	Model 2	Model 3
run	walk,run, wave1,run, wave1,run	wave1,skip, run	walk,run, walk,run	walk,run, walk,skip, run	walk,run, wave1,run, side	run,wave1, run	run
walk	walk,run, walk,pjump, run,pjump	wave1,walk, wave1	bend,walk	walk,run, walk,run, walk	walk,wave1, walk,run	skip,run, skip,run, wave1	walk,run
skip	wave1,walk, run,bend, run,pjump	wave1,side, run	walk,run, walk,run	walk,run, walk,run	jack,wave1, run,wave1, run	run,skip, bend	wave1,walk, run
jack	jack	wave1, jack	jack,wave2, jack,wave2, jack	wave2,jack,run, jack,wave2,jack, run	jack	wave1,jack, wave2,jack	wave1,jack, wave2
jump	walk,run, walk,run, bend	run,wave1, walk	run,jump	walk,run, walk,run, skip	wave1,run, walk,run, jump,run	jack,run, walk,run, skip,jump	jack,run, skip,run
pjump	pjump, wave1, pjump	bend,side, pjump, wave1,pjump	bend,pjump, wave1,jack	walk,wave1, run,wave1, run	bend,pjump, wave1,jack, wave1,jack,wave1	pjump, wave1, pjump	pjump, wave1, pjump
side	pjump	pjump, wave1	wave1, pjump, walk,jack	wave1,run, walk,wave1	wave1,jack, pjump,wave1, pjump,side	wave1,side, pjump,side, wave1	wave1,pjum, side,pjump
wave2	jack, wave2, jack	jack	wave1,jack, wave2,jack	wave2,jack, wave2,run, wave2,run	jack, wave2, jack	wave2,jack, wave2,jack	wave1,wave, jack,wave2, jack
wave1	wave1,jack, wave1,jack, wave1,jack, wave1	jack,wave1, wave2,wave, jack,wave1	wave1	wave2,wave1, wave2,wave1	jack, wave1	wave1	wave1
bend	bend	pjump, bend, pjump	bend	bend	bend,pjump, bend	bend, pjump	pjump,bend, pjump,bend

200 epochs. Testing and implementation are also reported in the form of confusion matrices and a detection results table.

3.1.1 Training and Validation Analysis

The training and validation process of VA-LSTM-SYSU model can be seen in Figure 12. This model reached its best weight at 189th epoch. It had the value of training loss = 0.1923, training categorical accuracy = 0.9407, validation loss = 0.5696, and validation categorical accuracy = 0.8778. The training duration of this model was 2 minutes 33 seconds.

Figure 13 shows the training and validation chart of VA-LSTM-SBU model. The best weight was 186th epoch. The loss and accuracy values were training loss = 0.3745, training categorical accuracy = 0.8704, validation loss = 0.5259, and validation categorical accuracy = 0.8556 with 5 minutes 35 seconds of training time.

LSTM-PASL got the best weight at 92nd epoch of the training process. As shown in Figure 14, the values were training loss = 0.0576, training categorical accuracy = 0.9963, validation loss = 0.5801, and validation categorical accuracy = 0.8444. The model training took 5 minutes and 52 seconds of duration.

LSTM-AHM model acquired its best epoch at 186th with loss and accuracy values were training loss = 0.2337, training categorical accuracy = 0.9295, validation loss = 0.6382, and validation categorical accuracy = 0.8000. This training and validation process can be seen in Figure 15 and were done in 1 minute 4 seconds.

Our Model 1 showed its best training and validation at 153rd epoch, as shown in Figure 16. The evaluation values were known as training loss = 0.1287, training categorical accuracy = 0.9519, validation loss = 0.8884, validation categorical accuracy = 0.8444, and training time = 10 minutes 50 seconds.

Model 2 has the best values of training loss = 0.0590, training categorical accuracy = 0.9815, validation loss = 0.7505, and validation categorical accuracy = 0.8556. It was acquired at 194th epoch in 200 epochs of training and validation with 54 minutes 13 seconds of duration. The charts of these processes are shown in Figure 17.

doi: 10.14710/teknik.v43i2.46439

(a) (b)

Fig. 20. Jack detection by VA-LSTM-SBU (a) and wave1 detection by Model 2 (b).

(a) (b)

Fig. 21. Run detection by Model 1 (a) and wave2 by Model 2 (b).

(a) (b)

Fig. 22. Detection results of side by LSTM-PASL (a) and jump by Model 3 (b).

The charts of training and validation of Model 3 are depicted in Figure 18. In 24 minutes 52 seconds, this process yielded the best weight at 103rd epoch with training loss = 0.0682, training categorical accuracy = 0.9852, validation loss = 0.5674, and validation categorical accuracy = 0.8556.

The seven models with respective weights obtained from training and validation processes were then tested in the testing process. The results from the testing process were evaluated using a confusion matrix along with its calculations of average accuracy, error rate, precision, recall, and F_1score, in terms of micro and macro variants.

All seven matrices in Figure 19 show pretty good results of jack, jump, pjump, side, wave2, wave1, and bend actions prediction. This was shown by the True Positive (TP), which had values ranging from 6 to 9. In contrast, the prediction of the run, walk and skip actions was not good enough. The matrices show this condition from the TP values of the individual actions that vary from 2 to 7.

To summarize the True Positive (TP), False Positive (FP), True Negative (TN), and False Positive (FP), we presented Table 4 and Table 5. Then, the average accuracy, error rate, precision, recall, and F_1score were evaluated in Table 6, including the training time comparison of each model. The times were formatted in a unit of minute (′) and second (″).

3.1 Model Implementation on a Demo Video

The model passing through the training and validation processes yielded some weight values that could be used as a matrix in action classification. With a self-recorded video, this research used a simple python program through a Jupyter Notebook to implement classification testing for each model. This implementation resulted in some action classifications, as shown in Table 7.

Models implementation yielded different action detection on the video. This results of VA-LSTM-SYSU, VA-LSTM-SBU, LSTM-PASL, LSTM-AHM, Model 1, Model 2, and Model 3 models showed a consistent detection of action “jack” and “bend”; “jack”; “wave1”

doi: 10.14710/teknik.v43i2.46439

and “bend”; “jack”; “wave1”; and also “run” and “wave1” respectively. Figure 20 shows several samples of these consistent detections.

Some poor detection (there were many detection inconsistencies) were known in “wave1” action for VA-LSTM-SYSU and VA-LSTM-SBU models. In LSTM-PASL, LSTM-AHM, Model 1, Model 2, and Model 3, this poor detection happened in “jack”; “jack” and “wave2”; “run”, “pjump”, and “side”; “jump” and “side”; and “wave2” as shown in Figure 21.

Meanwhile, the detection of “skip”, “jump”, and “side”; “skip”, “jump”, “side”, and “wave2”; “skip” and “side”; “skip”, “jump”, “pjump”, and “side”; “skip”; “walk”, and also “skip” and “jump” actions were hard to detect by VA-LSTM-SYSU, VA-LSTM-SBU, LSTM-PASL, LSTM-AHM, Model 1, Model 2, and Model 3 respectively that can be seen in Figure 22.

4. Conclusion

According to this research, there were several points to conclude. They were performance, evaluation results, and model implementation results.

Artificial Neural Network modeling using MediaPipe and Long Short-Term Memory Architecture could be done with various combinations of the input layer, hidden layer, and output layer. The number of input neurons should be customized upon input data, whereas the output neurons fitted to the number of classification classes. In the hidden layer section, customization consisted of the number of LSTM layers, Dropout layers, Dense layers, and/or Flatten layers as well as its hyperparameters.

The classification accuracies of all seven LSTM models were in the range of 77% to 82%. The highest accuracy was obtained by Model 1, whereas LSTM-AHM model obtained the lowest. From the training time aspect, LSTM-AHM model had the fastest duration, namely 1 minute 4 seconds. In contrast, Model 2 had the longest duration, 54 minutes 13 seconds.

The detections yielded some fluctuated classifications in the demo video implementation phase compared to the testing results. This condition indicated differences in model behavior when classifying static data (from training, validation, and testing dataset) compared to dynamic data (from demo video). Therefore, the generalization and detection consistency of the seven models were not good enough.

References

Agrawal, A. S., Chakraborty, A., & Rajalakshmi, C. M. (2022). Real-Time Hand Gesture Recognition System Using MediaPipe and LSTM. *International Journal of Research Publication and Reviews*, *3*(4), 2509–2515.

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., & Zimmermann, T. (2019). Software Engineering for Machine Learning: A Case Study. *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019*, 291–300. https://doi.org/10.1109/ICSE-SEIP.2019.00042

An, J., Li, W., Li, M., Cui, S., & Yue, H. (2019). Identification and classification of maize drought stress using deep convolutional neural network. *Symmetry*, *11*(2). https://doi.org/10.3390/sym11020256

Beniwal, S., Jambheshwar, G., & Arora, J. (2012). Classification and Feature Selection Techniques in Data Mining. *International Journal of Engineering Research & Technology (IJERT)*, *1*(6). https://www.researchgate.net/publication/263662705

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*, *25*(11), 120–123.

Cheng, G., Wan, Y., Saudagar, A. N., Namuduri, K., & Buckles, B. P. (2015). Advances in Human Action Recognition: A Survey. *ArXiv*, *abs/1501.05964*. http://arxiv.org/abs/1501.05964

Chinchor, N. (1992). MUC-4 Evaluation Metrics. *Proceedings of the 4th Conference on Message Understanding - MUC4 ’92*, 22–29. https://doi.org/10.3115/1072064.1072067

Codreanu, V., Podareanu, D., & Saletore, V. A. (2017). Scale out for large minibatch SGD: Residual network training on ImageNet-1K with improved accuracy and reduced time to train. *ArXiv*, *abs/1711.04291*.

Daniel Tanugraha, F., Pratikno, H., Musayanah, M., & Indah Kusumawati, W. (2022). Pengenalan Gerakan Olahraga Berbasis (Long Short-Term Memory) Menggunakan Mediapipe. *Journal of Advances in Information and Industrial Technology*, *4*(1), 37–45. https://doi.org/10.52435/jaiit.v4i1.182

Ghosh, S. (2021). Proposal of a Real-time American Sign Language Detector using MediaPipe and Recurrent Neural Network. *International Journal of Computer Sciences and Engineering*, *9*(7), 46–52. https://doi.org/10.26438/ijcse/v9i7.4652

Google LLC. (2020). *MediaPipe Pose*. https://google.github.io/mediapipe/solutions/pose.html

Gorelick, L., Blank, M., Shechtman, E., Irani, M., & Basri, R. (2007). Actions as Space-Time Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(12), 2247–2253. https://doi.org/10.1109/TPAMI.2007.70711

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Janssens, J. (2021, December 17). *Data Science at the Command Line, 2e: Obtain, Scrub, Explore, and Model Data with Unix Power Tools*. https://datascienceatthecommandline.com/2e/

Keras. (n.d.-a). *Adam*. Retrieved July 3, 2022, from https://keras.io/api/optimizers/adam/

Keras. (n.d.-b). *ModelCheckpoint*. Retrieved April 5, 2022, from https://keras.io/api/callbacks/model_checkpoint/

Lakkapragada, A., Kline, A., Mutlu, O. C., Paskov, K., Chrisman, B., Stockham, N., Washington, P., & Wall, D. (2022). Classification of Abnormal Hand Movement for Aiding in Autism Detection: Machine Learning Study. *JMIR Biomedical Engineering (JBME)*, *7*(1). https://doi.org/10.2196/33771

Mason, H., & Wiggins, C. (2010). *A Taxonomy of Data Science*. https://sites.google.com/a/isim.net.in/datascience_isim/taxonomy

McCullum, N. (2020, July 13). *The Ultimate Guide to Recurrent Neural Networks in Python*. https://www.freecodecamp.org/news/the-ultimate-guide-to-recurrent-neural-networks-in-python/

Minh, T. N., Sinn, M., Lam, H. T., & Wistuba, M. (2018). Automated Image Data Preprocessing with Deep Reinforcement Learning. *ArXiv*, *abs/1806.05886*. https://doi.org/10.48550/ARXIV.1806.05886

Moetia Putri, H., & Fuadi, W. (2022). Pendeteksian Bahasa Isyarat Indonesia Secara Real-Time Menggunakan Long Short-Term Memory (LSTM). *Jurnal Teknologi Terapan and Sains 4.0*, *3*(1).

Olah, C. (2015, August 27). *Understanding LSTM Networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Reimers, N., & Gurevych, I. (2017). Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 338–348. https://doi.org/10.48550/arxiv.1707.09861

Sak, H., Senior, A., & Beaufays, F. (2014). Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition. *ArXiv*, *abs/1402.1128*. http://arxiv.org/abs/1402.1128

Sarker, I. H. (2021). Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Computer Science*, *2*(6). https://doi.org/10.1007/s42979-021-00815-1

Sokolova, M., & Lapalme, G. (2009). A Systematic Analysis of Performance Measures for Classification Tasks. *Information Processing and Management*, *45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Tan, M., & Le, Q. v. (2021). EfficientNetV2: Smaller Models and Faster Training. *ArXiv*, *abs/2104.00298*. https://doi.org/10.48550/arxiv.2104.00298

Verdhan, V. (2021). *Computer Vision Using Deep Learning* (1st ed.). Apress. https://doi.org/10.1007/978-1-4842-6616-8

Wang, T., Chen, Y., Zhang, M., Chen, J., & Snoussi, H. (2017). Internal Transfer Learning for Improving Performance in Human Action Recognition for Small Datasets. *IEEE Access*, *5*, 17627–17633. https://doi.org/10.1109/ACCESS.2017.2746095

Xu, J., Zhang, Y., & Miao, D. (2020). Three-Way Confusion Matrix for Classification: A Measure Driven View. *Information Sciences*, *507*, 772–794. https://doi.org/10.1016/j.ins.2019.06.064

Zhang, P., Lan, C., Xing, J., Zeng, W., Xue, J., & Zheng, N. (2017). View Adaptive Recurrent Neural Networks for High Performance Human Action Recognition from Skeleton Data. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-October*, 2136–2145. https://doi.org/10.48550/arxiv.1703.08274

doi: 10.14710/teknik.v43i2.46439