Snapshot-Based Human Action Recognition using OpenPose and Deep Learning

Andi W. R. Emanuel*, Paulus Mudjihartono, and Joanna A. M. Nugraha, *Member, IAENG*

***Abstract*—This research builds a human action recognition system based on a single image or video capture snapshot. The TensorFlow Deep Learning models are developed using human keypoints generated by OpenPose. Four classifiers are considered: Neural Network, Random Forest, K-Nearest Neighbor (KNN), and Support Vector Machine (SVM) Classifiers. The models' input layer are 50 points from x and y coordinate of 25 keypoints from OpenPose, and the output layer is the numerical representation of 11 human action labels which are 'hand-wave', 'jump', 'leg-cross', 'plank', 'ride', 'run', 'sit', 'lay-down', 'squat', 'stand', 'walk'. A total of 2132 images dataset was used for model training and testing. The results show the two best classifier models: Neural Network Classifier with 512 hidden nodes with an accuracy of 0.7733, and Random Forest Classifier with 60 estimators with an accuracy of 0.7752. Both models are then used as inference engines to recognize human action from images and real-time video.**

***Index Terms*—snapshot-based, human action recognition, Neural Network Classifier, Random Forest Classifier**

I. Introduction

THE rise of Artificial Intelligence, especially in Deep Learning, has opened up many possibilities in solving many real-world problems, especially in computer vision. The ability of a computer to "see" objects in the image or video in almost real-time using algorithms such as Convolutional Neural Network, Yolo V4, Single Side Detection (SSD), etc. spark many types of research for the possible implementation in solving practical problems such as fall detection, accident avoidance, etc. [1][2][3][4]. After object detection beginning to mature in technology, the next plausible step is to recognize human action or activity called Human Activity Recognition (HAR). HAR is an emerging research area to determine the activity or movement of living or moving objects such as humans, animals, or other entities [3][5], and this research area still poses many challenges [6][7][8][9]. The combination of object detection and the activity related to these objects will improve the image or videos' overall comprehension without human intervention. This research presents a snapshot-based human action recognition system by utilizing OpenPose as a human keypoints generator dan Deep Learning Classifiers models.

Manuscript received January 1st, 2021; revised August 19th, 2021. This work was supported by internal research funding from Universitas Atma Jaya Yogyakarta assignment no. 113/In-Pen/LPPM/VII/2020.

Andi W. R. Emanuel is full-time lecturer at Faculty of Industrial Technology, Universitas Atma Jaya Yogyakarta, Indonesia (corresponding author, phone: (+62)274487711, email: andi.emanuel@uajy.ac.id)

Paulus Mudjihartono is full-time lecturer at Faculty of Industrial Technology, Universitas Atma Jaya Yogyakarta, Indonesia (email: paulus.mudjihartono@uajy.ac.id)

Joanna A. M. Nugraha is full-time lecturer at Faculty of Industrial Technology, Universitas Atma Jaya Yogyakarta, Indonesia (email: joanna.mita@uajy.ac.id)

II. Literature Reviews

Human Activity Recognition is emerging research that aims to identify humans' actions or activities using sensors, videos, or other means. This research may use an in-body sensor such as [10], [11][12], and [13]. This approach is impractical, and the result may be a non-natural movement of the object, especially for a human. The second approach for Human Activity Recognition uses non-body sensor approaches such as Thermal Infrared [14][15], Optical Flow [16], Spatio-Temporal Features [17], Part Affinity Fields [18], video [19][20], photonic computer [21], smartphone sensors, WiFi signals, or other devices. In video-based Human Action Recognition, the video or images captured are analyzed using a specific algorithm to identify the objects' actions, such as skeletal data [22][23]. Another video-based HAR is to estimate sign language as in [24]. In comparison, the non-vision-based approach may use devices and signals such as WiFi, a smartphone's accelerator, etc. [6][7].

Banjongkan *et al.* [25] offered a model to predict a job's status, especially at the final state of High-Performance Computer (HPC). They built three predictive models containing HPC-CNN, HPC-AlexNet, and HPC-VGG16. All these three models are founded from the HPC-work load dataset. Furthermore, the authors needed to compare those models with the baseline ones such as CART, ANN, and SVM. The results confirmed that the HPC-CNN model outperforms all the other models by achieving an accuracy of 76.48% compared to those of CART (75.60%) and SVM (66.80%).

Another research about human gestures is what [26] proposed. These researchers presented the recognition of student sitting posture. These student posters are captured in a classroom during the class. The tool they used is OpenPose, which is to extract the posture feature. Also, they used the Keras framework to build the convolutional neural network. The (x,y) position is saved in the dataset, and the authors made some adjustments to the images. Afterward, CNN is applied to the dataset. The result shows that this CNN achieves more than 90% of accuracy after 100 iteration training.

A new CNN architecture, so-called EfficientPose, is proposed [27]. This architecture is intended to reduce complexity and inefficiency due to the increasing computational burdens. EfficientPose controls the memory and computational load at the end devices. The authors adopt the EfficientPose from the EfficientNet by fine-tuning

the input resolution, network bandwidth, and network depth. Using the MPII dataset confirmed that EfficientPose consistently beats the OpenPose in terms of efficiency of 2.2 to 184 times fewer FLOPs and 4 to 56 times fewer parameters.

Similarly, another researcher investigated student gestures in the classroom. These gestures include listening, taking notes, playing cell phone, sleeping, raising the hand, and standing. This research used the COCO dataset, Caffe framework, CNN structure, and real-time multi-person OpenPose system. This model is fed the input of an image of hxwx3 and yields the key points' output of confidence maps. The result shows an accuracy of 75% [28].

Lastly, research of human gestures where computational intelligence takes place is run by [5]. It uses neural networks, fuzzy models, SVM, and long short-term memory (LSTM) networks to guess human gestures and activities. The input is based on 3D points of a human body model. The SVM classifies 95.97% correctly, while LSTM does 88%. Other models using Machine Learning and Deep Learning are used, such as the Hybrid Deep Learning model [29], two-stream CNN [30], Adaptive Neural Network [31], length control features fusion, and weighted entropy-variances [32].

This research is different from previous research since the final aim is to identify human actions in real-time based on a single snapshot. The recognition models are trained using labeled images of humans performing specific actions. The models are then implemented into inference engines to identify human actions in images, videos, and real-time videos. So, real-time identification is the core point of the research.

III. Methodology

The following is the methodology used for this research:

1. Prepare tools and libraries.
2. Collect image dataset using Google Image Search.
3. Build Deep Learning Classifier Models.
4. Build inference engines.

A. *Prepare Tools and Libraries*

This research utilizes several tools and libraries for the human detection process, creating the model and building an inference engine. These tools are:

--OpenPose: OpenPose is a real-time multi-person keypoint detection library for the body, face, hand, and foot estimation from CMU Perpetual Computing Lab [18]. OpenPose generates the 25-human body keypoints as the reference point of the human action in this research. OpenPose also uses other libraries such as Protobuf, Cmake, OpenCV, and Caffe. The detected body keypoints are the x and y coordinates along with their confidence values of 'Nose', 'Neck', 'RShoulder', 'RElbow', 'RWrist', 'LShoulder', 'LElbow', 'LWrist', 'MidHip', 'RHip', 'RKnee', 'RAnkle', 'LHip', 'LKnee', 'LAnkle', 'REye','LEye', 'REar', 'LEar', 'LBigToe', 'LSmallToe', 'LHeel', 'RBigToe', 'RSmallToe', 'RHeel'. Fig. 1 shows the keypoints generated from the model's image by using "--disable-blending" flag in OpenPose [18].

--CUDA 10.1 and CUDNN 7.5: libraries and utilities from NVIDIA for GPU optimized computation by OpenPose and Tensorflow. OpenPose will utilize 2647 MB of GPU memory during body keypoints detection using OpenPose.

--Tensorflow: Deep Learning library from Google to build the classifier model of action recognition. During the development using Tensorflow, other libraries integrated into the Tensorflow also utilized Keras, Numpy, Panda, etc.

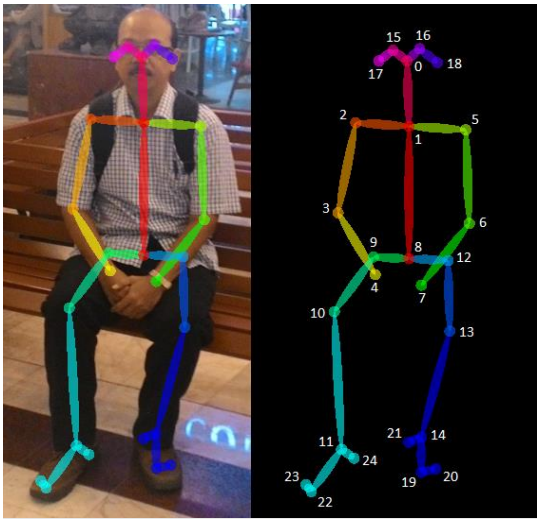

Fig. 1. Body Keypoints from OpenPose

--Ubuntu 18.04 LTS: Linux-based Operating System from Canonical in which OpenPose, CUDA, CUDNN, and other tools are installed and operated. This operating system is installed in Ryzen 7-based System using Dual GPU configuration for OpenPose keypoint detection and inference engine.

--Jupyter Notebook: interactive Python development environment installed in GPU Tesla-based server for building the classifier model.

--Python3: a scripting language for data collection and inference engine development.

B. *Collect Image Dataset using Google Image Search*

The second step is collecting images from Google Image Search. The collecting processes are:

--Collect images using Python script on Google Image Search Engine. The script collected 500 – 700 images in each run using a specific search string related to particular human actions and synonyms. The script utilizes selenium and chromedriver.

--Analyze collected images using OpenPose to find candidate images for model development. For images that can be used as data for training or testing, only images that OpenPose could identify all the 25 body keypoints are stored. Only 10% - 20% of the images are used during this phase, and the rest are deleted.

--Prepare data from the image by offsetting and normalizing body keypoints based on "body box," as shown in Fig. 2. The "body box" was created from the outer coordinates of body keypoints and then set the top left corner as (0,0) coordinates. Other keypoints were offset based on (0,0) coordinates and normalized to 128 pixels by 128 pixels for saving to CSV format.

Fig. 2. Body box of a person

--Classify the collected images manually into 11 actions, which are *'hand-wave'*, *'jump'*, *'leg-cross'*, *'plank'*, *'ride'*, *'run'*, *'sit'*, *'lay-down'*, *'squat'*, *'stand'*, and *'walk'*.

--Save the selected images' keypoints information into CSV format, including the filename, each keypoints (x and y in separate columns), and the action labels.

All collected images underwent second manual inspections to determine the accuracy of the images and their action labels. Total 2132 images are stored, and their keypoints are stored in CSV format. Three CSV files are being generated based on those images:

--The dataset of 1100 keypoints for 11 action labels: 100 images for each action label in this dataset. This dataset will be used for Neural Network Classifier model training and testing.

--The validation dataset: the remaining 1032 keypoints. The distribution of action labels in this dataset is not equal for each action label.

--The combined dataset: the combination of all 2132 images keypoints. This dataset's action labels are not distributed equally, but at least 104 keypoints or more for each action label.

C. Build Deep Learning Classifier Models

Deep Learning Classifier models are developed using the three CSV files. The processes of models building are:

--Determine the input layer: the input layer is 50 input from x and y of 25 keypoints detected by OpenPose and stored in a CSV file dataset. All keypoints are normalized to a value between 0 to 1 by dividing it by 128, as explained in sub-section 2.B.

--Determine the output layer, which integers representing each of 11 actions: 0:'hand-wave', 1: 'jump', 2: 'leg-cross', 3: 'plank', 4: 'ride', 5:'run', 6:'sit', 7:'lay-down', 8:'squat', 9:'stand', 10:'walk'

--Determine the Deep Learning Classifier model algorithm. There are four popular classifier algorithms used: Neural Network Classifier, Random Forrest Classifier, K-Nearest Neighbor (KNN) Classifier, and Support Vector Machine (SVM) Classifier. Models are built for all of these four classifier algorithms. The hyperparameters of each classifier are determined during the experiment, such as for Neural Network Classifier that are hidden nodes, test size, dropout, and epochs; Random Forest Classifier that are test size, estimators; KNN Classifier that is k value, test size; and SVM Classifier that are test size, the kernel function

--Each model is then analyzed for performance in terms of precision, recall, and f1-score. The best models are selected for inference engines. The best models are saved for the inference engines, and the rest of the models were dropped.

D. Build Inference Engines for Image and Video

The selected models are then being used for the inference engines. The inference engines were built to predict action from image and video (recorded video or live video from webcam). The inference engines are built using Ubuntu 18.04 LTS based system in dual GPU configuration: Geforce GXT 1650 4GB for OpenPose keypoints detection, and Geforce GTX 1660TI 6GB for Tensorflow models.

IV. Result and Discussion

A. Building Deep Learning Classifier Models

The Neural Network Classifier model was built by experimenting with 64, 128, 256, and 512 hidden nodes and setting a 0.2 test size, dropout rate of 0.2, and epochs from 300 to 600 (with epoch value interval from 10 to 50). The best NN Classifier model was found with 512 hidden nodes and 550 epochs with a training accuracy of 0.9398, test accuracy of 0.6636, and validation accuracy of 0.7733. Table I shows the detailed data for this model. This model was then saved for the inference engine. This model tells that the more hidden nodes, the higher accuracy. However, the epoch is not necessarily the most one. Furthermore, even though the training accuracy is high, but the test and validation accuracy is not.

TABLE I
Performance Data of Neural Network Classifier Model

action	precision	recall	f1-score	support
0	0.61	0.85	0.71	20
1	0.67	0.71	0.69	14
2	0.88	0.88	0.88	24
3	0.94	0.94	0.94	17
4	0.50	0.50	0.50	16
5	0.71	0.77	0.74	22
6	0.70	0.70	0.70	20
7	0.72	0.76	0.74	17
8	0.70	0.74	0.72	19
9	0.67	0.52	0.58	27
10	0.65	0.46	0.54	24

Note: 0:'hand-wave',1: 'jump',2: 'leg-cross',3: 'plank',4: 'ride', 5:'run', 6:'sit', 7:'lay-down', 8:'squat', 9:'stand', 10:'walk'

The Random Forest Classifier model was built using 0.2 test size and tried from 10 to 100 estimators with 10 intervals. The best model was found with 60 estimators with an accuracy of 0.7752. The model building of this model was trained using all of 2132 keypoints data. Table II shows the detailed data for this model. This model was then saved for the inference engine.

TABLE II
PERFORMANCE DATA OF RANDOM FORREST CLASSIFIER MODEL

action	precision	recall	f1-score	support
0	1.00	0.57	0.72	23
1	0.68	0.82	0.74	33
2	0.67	0.59	0.62	17
3	0.92	1.00	0.96	70
4	0.76	0.62	0.68	26
5	0.69	0.89	0.78	38
6	0.75	0.86	0.80	57
7	0.90	0.47	0.62	19
8	0.89	0.82	0.85	50
9	0.67	0.76	0.71	54
10	0.68	0.53	0.59	40

Note: 0:'hand-wave',1: 'jump',2: 'leg-cross',3: 'plank',4: 'ride', 5:'run', 6:'sit', 7:'lay-down', 8:'squat', 9:'stand', 10:'walk'

The KNN Classifier model was built by setting k = 11, corresponding to the 11 action labels. The resulting accuracy was lower than previously found NN Classifier and RF Classifier models with train accuracy 0.6511, test accuracy 0.5636, dan validation accuracy 0.6589. Setting k = 9 and k = 10 also tried but producing the same lower result. The KNN Classifier model was dropped and not used for the inference engine.

The SVM Classifier model was built by considering all kernel functions and showing reasonable results using a linear kernel function to predict the 11 action labels. The model had lower accuracy than previously found NN Classifier dan RF Classifier models with train accuracy 0.6648, test accuracy 0.5909, and validation accuracy 0.6860. There was no reasonable result using Poly, Sigmoid, and RBF kernel functions since some action labels were not predicted. The model was dropped and not used for the inference engine.

Based on the experiment, there are two best classifier models for inference engine models, which are:

--Neural Network Classifier model with 512 hidden nodes. This model has a training accuracy of 0.9398, test accuracy of 0.6636, and validation accuracy of 0.7733.

--Random Forest Classifier with 60 estimators. This model has an accuracy of 0.7752.

These models were saved for the inference engines to predict human actions from images and video.

B. Building Inference Engines

Imputers must be developed for inference engines since not all body keypoints are detected in images or videos from OpenPose. Feeding NaN or 0.0 in the missing keypoints were resulting in the inaccurate prediction of actions. Using readily available imputers such as datawig was impractical since the imputing outputs are too slow for real-time inference. On the other hand, using algorithms such as FastDTW was impractical since the imputing process requires about 24 seconds in each image or video frame. The imputers' current solution in this inference engine was using a lookup table in which the missing data will be automatically replaced with the closest non-NaN keypoints. This solution was not optimal but provided reasonable accuracy and speed in replacing the missing keypoints in the inference engines.

The inference engine for an image using Neural Network Classifier Model was implemented. The action prediction took 180 milliseconds for first-person prediction and 90 milliseconds for subsequent person action prediction, and the model consumes 5622 MB of GPU memory. This model gave each person's action prediction and confidence level detected in the image or video frame, as shown in Fig 3.

Fig. 3. Action Detection walk and hand-wave using Neural Network Classifier

Fig 3 shows two people's action recognition in the same image using the Neural Network Classifier model. Person 0 is detected with action 'walk' with the confidence of 57.58%, and person 1 is detected with action 'hand-wave' with the confidence of 92.63%. Whereas Fig 4 shows the action recognition of one person. Person 0 is detected with action 'run' with the confidence of 54.66%.

Fig. 4. Action Detection run using Neural Network Classifier

The inference engine for the image using the Random Forest Classifier Model was also implemented. By using the model, the prediction time is 5 milliseconds per person in the image. The model consumes 144 MB of GPU memory. The inference engine only predicted action without confidence value since the Random Forest Classifier model predicts majority voting of all decision trees.

Fig. 5. Action Detection ride using Random Forrest Classifier

Fig 5 shows the action recognition of one person using a Random Forrest Classifier model. Person 0 is detected with an action 'ride'. Whereas Fig 6 shows the action recognition of one person. Person 0 is detected with the action' jump'.

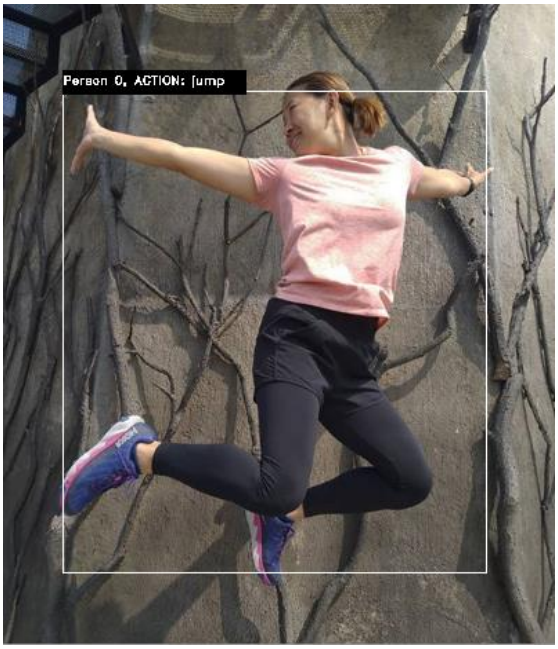

Fig. 6. Action Detection jump using RF Classifier

Inference engines for video using both classifier models are built using the same analogy as for image. Each frame from the video is sent to the inference engines for action detection. The video inference engine using Neural Network Classifier model resulted in slow frame rate inference engines since it took 180 milliseconds for first-person prediction and 90 milliseconds subsequent person action prediction, but after converting the model into the Tensorflow Lite model, the GPU memory consumption was reduced to 144 MB the prediction time was reduced to 0.1 milliseconds to enable faster prediction. Meanwhile, the video inference engine using the RFC model requires no modification since the prediction time and GPU consumption is acceptable.

V. Conclusions

It can be concluded that a snapshot-based human action recognition system based on Deep Learning models by using keypoints from OpenPose as the input layer can be built. There are two best classifier models: Neural Network Classifier with 512 hidden nodes performs with the accuracy of 0.7733, and Random Forest Classifier with 60 estimators performs with the accuracy of 0.7752. These models are then used as inference engines to detect human action from images and videos (files or live using a webcam). Neural Network Classifier's inference engine must be converted to Tensorflow Lite model to reduce GPU memory consumption to 144 MB of GPU memory and prediction time to 0.1 milliseconds. In comparison, Random Forest Classifier's inference engine can predict action from images and videos with 144 MB of GPU memory and a prediction time of 5 milliseconds.

There are some practical limitations of the research findings:

--Sometimes, it is challenging to determine actions by observing a single snapshot of video or a single image. Different actions may have a similar posture if seen in just a single snapshot.

--The prediction accuracy varies based on the image or video resolution and the person's 'body box' in the image or video.

This research's future work includes expanding the method to consider sequence-based action recognition so that inference engines can predict more complex actions. Hence, preventing something unexpected circumstances from happening is only a step away since predicting the actions is there to get. Furthermore, action recognition can be combined with object detection to increase the computer's image/video comprehension.

ACKNOWLEDGMENT

Authors would like to thank Departemen Informatika, Fakultas Teknologi Informasi, and LPPM Universitas Atma Jaya Yogyakarta, Indonesia in supporting this research.

REFERENCES

[1] A. Núñez-Marcos, G. Azkune, and I. Arganda-Carreras, "Vision-Based Fall Detection with Convolutional Neural Networks," Wireless Communications and Mobile Computing, vol. 2017, Article ID 9474806, pp1-16, 2017

[2] Y. Gu, H. Zhang, and S. Kamijo, "Multi-Person Pose Estimation Using an Orientation and Occlusion Aware Deep Learning Network," Sensors, vol. 20, issue 6, Article ID 1593, pp1-21, 2020

[3] J. Lee and B. Ahn, "Real-Time Human Action Recognition with a Low-Cost RGB Camera and Mobile Robot Platform," Sensors, vol. 20, issue 10, Article ID 2886, pp1-12, 2020

[4] Á. Morera, Á. Sánchez, A. B. Moreno, Á. D. Sappa and J. F. Vélez, "SSD vs. YOLO for Detection of Outdoor Urban Advertising Panels under Multiple Variabilities", Sensors, vol. 20, issue 16, Article ID 4587, pp1-23, 2020

[5] P. J. S. Gonçalves, B. Lourenço, S. Santos, R. Barlogis and A. Mission, "Computer Vision Intelligent Approaches to Extract Human Pose and Its Activity from Image Sequences," Electronics, vol. 9, issue 1, Article ID 159, pp1-20, 2020

[6] H. -B. Zhang, Y. -X. Zhang, B. Zhong, Q. Lei, L. Yang, J. -X. Du, and D. -S. Chen, "A Comprehensive Survey of Vision-Based Human Action Recognition," Sensors, vol. 19, issue 5, Article ID 1005, pp1-20, 2019

[7] I. Jengham, A. B. Khalifa, I. Alouani, and M. A. Mahjoub, "Vision-based Human Action Recognition: An Overview and Real World Challenges," Forensic Science International: Digital Investigation, vol. 32, Article ID 200901, pp1-17, 2020

[8] W. Gong, J. Gonzàles, and F. X. Roca, "Human Action Recognition Based on Estimated Weak Poses," EURASIP Journal on Advances in Signal Processing, vol. 2012, issue 1, Article ID 162, pp1-14, 2012

[9] Ó. G. Hernández, V. Morell, J. L. Ramon, and C. A. Jara, "Human Pose Detection for Robotic-Assisted and Rehabilitation Environments," Applied Sciences, vol. 11, issue 9, Article ID 4183, pp1-14, 2021

[10] C. Weich and M. M. Vieten, "The Gaitprint: Identifying Individuals by Their Running Style," Sensors, vol. 20, issue 14, Article ID 3810, pp1-13, 2020

[11] N. D. Nguyen, D. T. Bui, P. H. Truong, and G. M. Jeong, "Position-Based Feature Selection for Body Sensors regarding Daily Living Activity Recognition," Journal of Sensors, vol. 2018, Article ID 9762098, pp1-13, 2018

[12] N. Tufek, M. Yalcin, M. Altintas, F. Kalaoglu, Y. Li, and S. K. Bahadir, "Human Action Recognition Using Deep Learning Methods on Limited Sensory Data," IEEE Sensors Journal, vol. 20, issue 6, pp3101-3112, 2019

[13] Y. Jin, G. Chen, K. Lao, S. Li, Y. Lu, Y. Gan, Z. Li, J. Hu, J. Huang, J. Wen, H. Deng, M. Yang, Z. Chen, X. Hu, B. Liang, and J. Luo, "Identifying Human Body States by Using a Flexible Integrated Sensor," npj Flexible Electronics, vol. 4, no. 28, pp1-8, 2020

[14] Y. Tan, W. Yan, S. Huang, D. Du, and L. Xia, "A Motion Deviation Image-based Phase Feature for Recognition of Thermal Infrared Human Activities," Engineering Letters, vol. 28, issue 1, pp48-55, 2020

[15] A. Akula, A. K. Shah, and R. Gosh, "Deep Learning Approach for Human Action Recognition in Infrared Images," Cognitive System Research, vol. 50, pp146-154, 2018

[16] P. A. S. Mendes, M. Mendes, A. P. Coimbra, and M. M Crisostomo, "Movement Detection and Moving Object Distinction Based on Optical Flow," Proceedings of the World Congress on Engineering (WCE) 2019, pp3-5, 2019

[17] C. Chen, Y. Wang, K. Yi, T. Wang, and H. Xiang, "Semantic Analysis of Action with Spatio-Temporal Features Based on Object Detection," Engineering Letters, *v*ol. 28, issue 2, pp616-623, 2020.

[18] Z. Chao, G. Hidalgo, T. Simon, S. -E. Wei, and Y. Sheik, "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, 2017

[19] E. P. Ijjina and K. M. Chalavadi, "Human Action Recognition in RGB-D Videos using Motion Sequence Information and Deep Learning," Pattern Recognition, vol. 72, pp504-516, 2017

[20] M. Ma, N. Marturi, Y. Li, A. Leonardis, and R. Stolkin, "Region-sequence Based Six-Stream CNN Features for General and Fine-grained Human Action Recognition in Videos," Pattern Recognition, vol. 76, pp506-521, 2018

[21] P. Antonik, N. Marsal, D. Brunner, and D. Rontani, "Human Action Recognition with a Large Scale Brain-Inspired Photonic Computer," Nature Machine Intelligence, vol. 1(11), pp530-537, 2019

[22] J. Xiaopeng, C. Jun, F. Wei, and T. Dapeng, "Skeleton Embedded Motion Body Partition for Human Action Recognition using Depth Sequences," Signal Processing, vol. 143, pp56-68, 2018

[23] H. -H. Pham, L. Khoudour, A. Crouzil, P. Zegers and S. A. Velastin, "Exploiting Deep Residual Networks for Human Action Recognition from Skeletal Data," Computer Vision and Image Understanding, vol. 170, pp51-66, 2018

[24] J. Charles, T. Pfister, M. Everingham, and A. Zisserman, "Automatic and Efficient Human Pose Estimation for Sign Language Videos," International Journal of Computer Vision, vol. 110(1), pp70-90, 2014

[25] A. Banjongkan, W. Pongsena, N. Kerdprasop, and K. Kerdprasop, "A Comparative Study of Learning Techniques with Convolutional Neural Network Based on HPC-Workload Dataset," International Journal of Machine Learning and Computing, vol. 10, no. 1, pp10-17, 2020

[26] K. Chen, "Sitting Posture Recognition Based on OpenPose," Proceedings of 2019 4th International Conference on Insulating Materials, Material Application and Electrical Engineering (IMMAEE), pp1341-1346, 2019

[27] D. Groos, H. Ramampiaro and E. A. F. Ihlen, "EfficientPose Scalable single-person pose estimation," Applied Intelligence, vol. 51, pp2518-2533, 2021

[28] T. Zhang, "Gesture Recognition Based on Deep Learning," Journal of Physics: Conference Series, vol. 1449, Article ID 012081, pp1-7, 2020

[29] N. Jaouedi, N. Boujnah, and M. S. Bouhlel, "A New Hybrid Deep Learning Model for Human Action Recognition," Journal of King Saud University – Computer and Information Sciences, vol. 23, issue 4, pp447-453, 2020

[30] Q. Xiong, J. Zhang, P. Wang, D. Liu, and R. X. Gao, "Transferable Two-Stream Convolutional Neural Network for Human Action Recognition," Jurnal of Manufacturing Systems, vol. 56, pp605-614, 2020

[31] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View Adaptive Neural Network for High Performance Skeleton-Based Human Action Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, issue 8, pp1963-1978, 2019.

[32] F. Afza, M. A. Khan, M. Sharif, S. Kadry, G. Manogaran, T. Saba, I. Ashraf, and R. Damaševičius, "A Framework for Human Action Recognition using Length Control Features Fusion and Weighted Entropy-Variances based Feature Selection," Image and Vision Computing, vol. 106, article ID 104090, pp1-11, 2021.