SUMMARY With the development of cameras and sensors and the spread of cloud computing, life logs can be easily acquired and stored in general households for the various services that utilize the logs. However, it is difficult to analyze moving images that are acquired by home sensors in real time using machine learning because the data size is too large and the computational complexity is too high. Moreover, collecting and accumulating in the cloud moving images that are captured at home and can be used to identify individuals may invoke the privacy of application users. We propose a method of distributed processing over the edge and cloud that addresses the processing latency and the privacy concerns. On the edge (sensor) side, we extract feature vectors of human key points from moving images using OpenPose, which is a pose estimation library. On the cloud side, we recognize actions by machine learning using only the feature vectors. In this study, we compare the action recognition accuracies of multiple machine learning methods. In addition, we measure the analysis processing time at the sensor and the cloud to investigate the feasibility of recognizing actions in real time. Then, we evaluate the proposed system by comparing it with the 3D ResNet model in recognition experiments. The experimental results demonstrate that the action recognition accuracy is the highest when using LSTM and that the introduction of dropout in action recognition using 100 categories alleviates overfitting because the models can learn more generic human actions by increasing the variety of actions. In addition, it is demonstrated that preprocessing using OpenPose on the sensor side can substantially reduce the transfer quantity from the sensor to the cloud.

1. Introduction

The development of cameras and sensors and the spread of cloud computing have enabled people to acquire and store life logs in ordinary homes. These data are used for services such as monitoring children, elderly people, and pets and for security systems. M. Mohammadi et al. [2] introduced applications in various fields that analyze a large amount of stream data that are generated from Internet of things (IoT) devices (sensors) using machine learning, e.g., in smart homes, smart cities, healthcare, and agriculture. In addition, according to the authors, the use of machine learning to analyze large-scale stream data and realize application objectives is promising. We aim at supporting security and monitoring systems using videos that are captured in ordinary homes. In these services, remote users other than the target person must be able to receive alerts for emergencies, such as injuries and illness. However, it is difficult to analyze moving images that are acquired by a home sensor with machine learning in real time because the data size and the computational complexity of the analysis are large and high-performance servers and storage are not installed in ordinary homes.

Although we can use abundant computing resources in a cloud, the network bandwidth between a sensor and the cloud is limited, and it is highly difficult to conduct analyses with the processing latency that is expected by each application. In addition, with the development of IoT devices with GPU such as NVIDIA® Jetson™ [3], small devices with GPU can be installed and used in ordinary homes. Hence, distributed processing, which is called edge computing [4] or fog computing [5] and refers to the processing of part of the calculation on the sensor side or with an edge device near the sensor side, is effective.

The advantage of preprocessing on the sensor side is not only the reduction in the processing latency but also privacy protection and the reduction in communication costs. Collecting and accumulating in the cloud moving images that have been captured at home and can be used to identify individuals may invoke the privacy of application users. In addition, when a mobile network is used for communication between a sensor and the cloud, the pay-per-use model is typically applied according to the transfer amount. It is necessary to minimize the transfer data size as much as possible to reduce not only the communication latency but also the costs. Therefore, we address these issues by collecting and analyzing only feature vectors in the cloud. The feature vectors are extracted from moving images by preprocessing on the sensor side, and only small feature vectors are sent to the cloud. However, since the amount of information that is contained in the original video data is reduced significantly by preprocessing, the accuracy with which we can recognize actions using only the feature vectors in learning and inference is unknown.

We propose a method of distributed processing over the edge and the cloud to address the processing latency and the privacy concerns [1]. On the edge (sensor) side, we extract...
feature vectors of human key points from moving images using OpenPose [6]–[9], which is a pose estimation library. On the cloud side, we recognize actions by machine learning using only the feature vectors.

We investigate the following 4 aspects:

1. Comparison of action recognition accuracies using only the feature vectors with various machine learning methods;
2. Measurement of the analysis processing time at the sensor and the cloud;
3. Investigation of the amount of data that are transferred from the sensor to the cloud;
4. Evaluation of the proposed system by comparing it with the 3D ResNet model.

First, we compare the action recognition accuracies of various machine learning methods using the feature vectors that are extracted from moving images from the STAIR Actions [10] dataset. We employ logistic regression, random forest, support vector machine (SVM), a fully connected neural network (NN), and long short-term memory (LSTM) as the machine learning methods. We build NN models and LSTM models with the Keras deep learning framework [11]. In these experiments, we recognize 3 categories with 2 similar actions and 1 action that is relatively easy to distinguish from the other 2 actions. We also conduct experiments using all 100 categories in the STAIR Actions dataset since we aim for real applications using a wider variety of action recognition procedures. Second, we measure the analysis processing times at the sensor and the cloud to investigate the feasibility of real-time action recognition. Third, we investigate the reduction that can be realized in the amount of data that must be transferred via the conversion of moving images to feature vectors. Finally, we evaluate the proposed system by comparing it in recognition experiments with the 3D ResNet [12] model after it has been fine-tuned on the STAIR Actions dataset. ResNet is a model that enables the use of multiple convolutional neural network (CNN) layers and is commonly used for image analysis, and 3D ResNet is a model that applies ResNet to moving image analysis.

The earlier version of this manuscript that appeared in [1] showed that the proposed model can recognize 3 actions with more than 80% accuracy using only key points and that the amount of data that were transferred from sensors to a cloud is significantly reduced. However, the issue of overfitting remains because the recognition of 3 actions is not sufficiently versatile. In this version, we conduct action recognition using 100 categories of the STAIR Actions dataset. The results of our approach is compared with the best results that are obtained using 3D ResNet, although 3D ResNet cannot be applied for distributed processing over sensors and a cloud like our proposal. Then, we evaluate the performance of the proposed system based on this comparison. The experimental results demonstrate that multiple machine learning methods can recognize 3 actions with more than 80% accuracy but that the alleviation of overfitting remains unresolved. According to the actions recognitions results that were obtained using the 100 categories, LSTM can recognize the actions with the highest accuracies, and increasing the variety of actions can alleviate overfitting. In addition, preprocessing at the sensor can drastically reduce the amount of data that are transferred from the sensor to the cloud. We also discuss a method for further reducing the analysis processing time and examine the recognition accuracy based on the experimental results.

2. Background

We consider an action recognition system, which is illustrated in Fig. 1. Feature vectors are extracted from moving images that are acquired by a camera that has been installed in each ordinary home. The feature vector data are transferred from the sensor side to the cloud and the data are collected in the cloud. Then, the system conducts action recognition via a machine learning process. We investigate whether actions can be sufficiently identified using only the feature vectors that were extracted on the sensor side and without using moving images or still images. In addition, we determine which machine learning method can realize high accuracy. We will explain the related technologies that are used in this study below.

2.1 OpenPose

OpenPose is a pose estimation library that extracts key points such as human joints in real time using deep learning. It has been developed by Carnegie Mellon University and other institutions. It is possible to detect 135 key points, not only of a person’s body but also of the face and hands, that are included in a video or image. It is also possible to conduct the analysis with only the camera’s image and video and without using a special sensor such as an acceleration sensor. Additionally, by using the GPU, OpenPose can analyze in real time even if the image or video contains multiple people. The developers evaluated the performance of keypoint detection using MPII [13] and COCO [14], which are datasets of images of multiple people. In the evaluation using MPII, they measured mean average precision (mAP).
of all body parts using the “PCKh” metric, and 75.6% mAP was realized. In the evaluation using COCO, they measured mAP using the object keypoint similarity (OKS), and 65.3% was realized. In addition, OpenPose adopts the bottom-up method, in which all keypoints are extracted and grouped by person, and it is possible to extract keypoints at almost constant speed regardless of the number of people, as described in [6].

2.2 Keras

Keras is a library for implementing a neural network, which was developed as part of the research for the Open-ended Neuro-Electronic Intelligent Robot Operating System (ONEIROS) project. It supports TensorFlow, Theano and the Microsoft Cognitive Toolkit as a backend, and it focuses on enabling the description of a network model very easily and fast experimentation. Three features render prototyping easy and fast: modularity, extensibility, and user-friendliness. Since Keras operates seamlessly on CPUs and GPUs, it is capable of high-speed operation and is compatible with convolution and recurrent neural networks and their combination.

3. Machine Learning Methods That are Applied in This Study

In this paper, we compare the action recognition accuracies of various machine learning methods using the coordinate data of keypoints that are extracted from images using OpenPose. To investigate the action recognition accuracies using only the feature vectors, we conduct 2 types of experiments: (1) action recognition using 3 categories and (2) action recognition using 100 categories. For the datasets, we use images that were acquired from the STAIR Actions [10] dataset, which is a collection of approximately 1000 videos of 100 daily actions, such as walking and cooking. This dataset is a collection of indoor actions at home and in an office. It has action categories of a finer granularity than other datasets that are used for human action recognition such as HMDB51 [15] and UCF101 [16], which include a variety of categories such as outdoor actions. Also, the backgrounds of the videos are composed of a few people and indoor objects such as furniture and windows.

3.1 Machine Learning Methods

We measure the action recognition accuracies of the following 5 machine learning methods:

1) Logistic regression;
2) Random forest;
3) SVM;
4) Neural network (NN) model built with Keras;
5) LSTM model built with Keras.

1) Logistic regression is a model that regresses to a logistic function and outputs the probability of belonging to a class. 2) Random forest is a model that determines the result from the majority of the prediction results of multiple decision trees. 3) SVM is a model that is optimized to maximize the margin of the projected high-dimensional space using a kernel function; we used RBF as the kernel function in this experiment. 4) An NN is a model that simulates human neurons; we used a fully connected NN. In addition, we adjusted the parameters to improve the action recognition accuracy that was realized using the NN in the following configurations:

4a) NN + Dropout;
4b) NN + Batch Normalization (BN);
4c) NN + Dropout and BN.

Dropout is a method that is used to prevent overfitting by invalidating and learning a subset of the nodes in each layer and forcibly reducing the degrees of freedom of the network to improve the general performance. Batch normalization is a method that is used to increase the accuracy and speed of learning via normalization by calculating the mean and variance of the input batch and adjusting the scale and shift.

In the experiments, on time series data, we also conduct an experiment using 5) LSTM, which is an extension of the recurrent neural network (RNN), to consider the context of feature vectors for a longer time. RNN is a model that is related to the recursive neural network, which has the advantage of being able to use continuous information such as sentences. RNN cannot learn long-term data dependencies, although the information that has been calculated in the previous time step can be stored and used in subsequent calculations. LSTM overcomes this disadvantage and enables the learning of long-term data dependencies.

RNN might be able to consider sufficient dependencies in the experiments with a small number of steps. However, we will use LSTM, which can learn longer-term dependencies, to consider the many images that are acquired from videos. Figure 2 illustrates the configuration of the LSTM model. After learning the features that are acquired from each image by a fully connected NN (MLP) of 2 layers, we input the output from the NN into the LSTM for each time
step and classify the actions using the output of the previous step. Additionally, we introduce dropout to prevent overfitting. Both dropout and recurrent dropout can be used for LSTM; the dropout of a network refers to the proportion of invalidated nodes in the linear transformation of the input, whereas the recurrent dropout represents the proportion of invalidated nodes in the linear transformation of recursion. We set the node invalidation rate to 20% and measure the action recognition accuracies with the following three configurations:

5a) LSTM + Dropout;
5b) LSTM + Recurrent dropout;
5c) LSTM + Dropout and Recurrent dropout.

In experiment (1), namely, action recognition using 3 categories, we measure accuracies using the above 5 methods. In experiment (2), namely, action recognition using 100 categories, we measure accuracies using the 4) an NN and 5) LSTM.

### 3.2 Datasets

We extracted multiple still images from each moving image in the STAIR Actions dataset at equal intervals. For each still image, we obtained the x and y coordinates of 25 key points on the image using OpenPose and created data for 50 feature vectors. Examples of keypoints that were extracted by OpenPose are presented in Fig. 3, and a subset of the feature vector data from this image is shown in Fig. 4. The resolutions of the videos in this dataset vary from a minimum of 128 \times 224 pixels to a maximum of 1920 \times 1080 pixels. Our dataset does not include videos that are of too low a resolution or do not show a large portion of the body because we selected videos from which OpenPose can extract a series of keypoints. We also utilized videos that show only one person for the keypoint extraction. In the dataset for experiment (1), namely, action recognition using 3 categories, we extracted still images at 0.1 second intervals from each 1 to 3 second moving image. For the experiment that involved 1) logistic regression, 2) the random forest method, 3) SVM, and 4) an NN, we used 500 feature data that corresponded to 50 features of 10 images that were acquired from each video in chronological order starting from the first image as the input data. For the experiment that involved 5) LSTM, we acquired 10, 20, and 30 images according to the number of LSTM time steps from each video, and we input 50 features of each image as the input at each LSTM time step. The amount of data in each category is presented in Table 1. We used 70% of the data for training and 30% for validation.

Next, we created datasets using all the categories in the STAIR Actions dataset for experiment (2), namely, action recognition using 100 categories, and we recognized various additional actions. For the experiment that involved 4) the NN, we extracted 10 still images at 0.1 and 0.3 second intervals. We used 500 feature data that corresponded to 50 features of 10 images that were acquired from each video in chronological order starting from the first image as the input data. For the experiment that involved 5) LSTM, we created datasets by acquiring 10, 20, and 30 images according to the number of LSTM time steps from each video, and we input 50 features of each image as the input at each LSTM time step.
time step. We acquired 10 images at intervals of 0.1 and 0.3 seconds and 20 and 30 images at 0.1 second intervals. The number of images, the interval for image acquisition and the number of data in each dataset are presented in Table 2. We used 70% of the data for training and 30% for validation.

4. Action Recognition Using Only the Feature Vectors

We measure the accuracies that were realized in experiment (1), namely, action recognition using 3 categories, and in experiment (2), namely, action recognition using 100 categories, by multiple machine learning methods. In experiment (1), we use the top 1 accuracy, namely, the accuracy with which the highest probability is assigned to the correct category. In experiment (2), we use the top 5 accuracy, namely, the accuracy with which any of the 5 highest probabilities are assigned to the correct answer. We also introduced dropout and BN to investigate their effectiveness in alleviating overfitting.

4.1 Action Recognition Using 3 Categories

Table 3 presents the action recognition accuracies of each method, and Table 4 presents the parameters of logistic regression, the random forest method and SVM. In the NN, the values in Table 3 are the accuracies that are realized when learning to use the model with 3 middle layers of 500 nodes and 1600 epochs; a dropout of the invalidation ratio of 20% and batch normalization are introduced to prevent overfitting. In LSTM, the values are the accuracies that are realized when learning using the model with 50 nodes in the LSTM layer, 30 time steps and 1600 epochs; dropout and recurrent dropout of the invalidation ratio of 20% are introduced to prevent overfitting. The accuracy that was realized using the random forest is the highest, namely, 0.828, and we were able to improve the NN and LSTM accuracies by introducing dropout and BN.

We measured the action recognition accuracies by changing the number of middle layers and the number of nodes in the middle layers to optimize the parameters of the NN model. We conducted a grid search using cross-validation when the number of middle layers was changed from 3 to 6 and the number of nodes was changed from 500 to 600, 700 and 800 in a heatmap. When the number of middle layers was set to 4 and the number of nodes was set to 700, the accuracy was the highest. In addition, we conducted a grid searches with and without dropout and batch normalization using this NN model. The model with 4 middle layers and 700 nodes and with dropout with an invalidation ratio of 0.2 realized the highest accuracy, and the accuracy tended to improve as the node invalidation ratio of dropout was increased in the range of 0.0 to 0.2 without introducing batch normalization. Hence, the accuracy may be improved by more finely adjusting the parameters in this range.

Next, we measured the action recognition accuracies that were realized by changing the numbers of time steps and nodes in the LSTM layers. The accuracies that were realized when the number of time steps was set to 10, 20 and 30 and the number of nodes was set to 50 and 100 are presented in Table 5. The result with 30 steps and 100 nodes shows the best performance, while the accuracies that were realized with 20 steps were the lowest. The accuracies do not always improve as the number of time steps is increased even though the number of feature vectors increases in proportion to the number of time steps.

We introduced dropout because there is a tendency to overfit when learning using the fully connected NN model. Figure 5 and 6 show the accuracies for each time step setting with dropout only, recurrent dropout only, and both dropout and recurrent dropout. In the experiments, the invalidation rate was set to 20%, and the number of nodes in the LSTM

| Table 3 | Action recognition accuracies that were realized by each method in experiment (1), namely, action recognition using 3 categories. |
|---|---|---|---|
| Method | Training | Validation |
| 1) Logistic Regression | 0.869 | 0.580 |
| 2) Random Forest | 1.000 | 0.828 |
| 3) SVM | 1.000 | 0.440 |
| 4) NN | 0.976 | 0.748 |
| 4a) NN w/ Dropout | 0.999 | 0.800 |
| 4b) NN w/ BN | 0.999 | 0.813 |
| 4c) NN w/ Dropout and BN | 0.987 | 0.765 |
| 5) LSTM | 1.0 | 0.748 |
| 5a) LSTM w/ Dropout | 1.0 | 0.802 |
| 5b) LSTM w/ Recurrent Dropout | 1.0 | 0.773 |
| 5c) LSTM w/ Dropout and Recurrent Dropout | 1.0 | 0.819 |

| Table 4 | Parameters that are used in learning by each method in experiment (1), namely, action recognition using 3 categories. |
|---|---|---|---|
| Method | Parameter | Value |
| 1) Logistic Regression | C | 0.001 |
| 2) Random Forest | bootstrap, criterion, max_depth, max_features, min_samples_leaf, min_samples_split, estimators | false, entropy, none, 10, 2, 300 |
| 3) SVM | C, gamma | 1, 0.0001 |
| 4) NN | Number of layers in the middle layer, Number of nodes in the middle layer, Epoch | 3, 500, 1600 |
| 5) LSTM | Number of nodes in the LSTM layer, Epoch, Number of time steps | 50, 1600, 30 |

| Table 5 | Action recognition accuracies that were realized by 5) LSTM in experiment (1), namely, action recognition using 3 categories [1]. |
|---|---|---|---|
| Number of time steps | Number of nodes | Training | Validation |
| 10 | 50 | 1.0 | 0.802 |
| 20 | 50 | 1.0 | 0.780 |
| 30 | 50 | 1.0 | 0.765 |
| 30 | 50 | 1.0 | 0.819 |
| 30 | 50 | 1.0 | 0.829 |
network was set to 50 and 100. Since the convergence was not sufficient after 1600 epochs of training, we set the number of epochs to 3000. According to the results, the accuracy tended to improve when only dropout was used for the LSTM model with both 50 and 100 nodes. Nevertheless, the suppression of overfitting by the introduction of dropout was insufficient.

From the results of learning using the fully connected NN model and the LSTM model, we normalized and augmented the feature vector data because the introduction of dropout was not sufficient to suppress overfitting. In the normalization process, we used the maximum and minimum values of the coordinate values of all the images that were acquired from the same moving image. In the augmentation process, we added the horizontally inverted and the ±5° and ±10° rotated data of each moving image. Figure 7 presents (a) the losses that were incurred using data without normalization and augmentation, (b) the losses that were incurred using normalized data, and (c) the losses incurred using augmented data during learning with the LSTM with 50 nodes and 10 steps. According to the figures, the learning results that were obtained using the normalized and augmented data stabilize and their convergences accelerate, but the overfitting problem is not resolved.

4.2 Action Recognition Using 100 Categories

Figure 8 and 9 present the action recognition accuracies that are realized by 4) the NN with 500 epochs and by 5) LSTM with 1200 epochs, respectively. In these figures, the horizontal axis represents the types of dropout and BN that were introduced into the training model, and the vertical axis represents the dataset that was used in training. From the figures, the introduction of BN reduced the recognition accuracy of the NN, but the introduction of dropout increased the accuracies that were realized by both the NN and LSTM. The accuracy that was realized by LSTM tended to increase when the time step was set to 20 and was the highest when using dataset (2-2d) and introducing only dropout. It is assumed that the recognition accuracy that was realized using dataset (2-2c) tended to become high because the number of data was the largest. Comparing the accuracies of action recognition that were realized by the NN and LSTM, LSTM
is suitable for the learning of time series data because learning with LSTM results in higher accuracy.

Next, we introduce dropout to suppress the overfitting and investigate the losses and accuracies that were realized during the learning by the LSTM model with 20 time steps, which tended to result in high accuracy. The losses and accuracies that were realized during learning by 5) LSTM, 5a) LSTM + Dropout, 5b) LSTM + Recurrent Dropout, and 5c) LSTM + Dropout and Recurrent Dropout are presented in Figs. 10, 11, 12 and 13, respectively. In each of these figures, the left figure shows the amount of loss at each epoch, and the right figure shows the accuracy at each epoch. The vertical axis represents the amount of loss or the accuracy, and the horizontal axis represents the number of epochs. The blue lines represent the training values, and the orange lines represent the validation values. In this experiment, we choose 1200 epochs because the convergence with 600 epochs was not sufficient. Figure 10 shows over-
fitting, where the loss of validation increases as the number of epochs increases while the loss of training converges to 0. In contrast, Figs. 11, 12 and 13 show the suppression of overfitting because the difference in losses between training and validation is small. In addition, we demonstrate that the introduction of dropout increases the accuracy because the validation accuracies that were realized after learning for 1200 epochs by 5a), 5b) and 5c) were 0.638, 0.642, and 0.635, respectively, while the accuracy that was realized by 5) was 0.618. The suppression of overfitting is an issue in action recognition using 3 categories. In contrast, it is assumed that the models could learn more generic human actions if the variety of actions is increased to 100 categories.

5. Discussion

5.1 Improvement of the Action Recognition Accuracy

We conducted experiment (1), namely, action recognition using 3 categories, and experiment (2), namely, action recognition using 100 categories. In experiment (1), we realized greater than 80% accuracy in the experiments using random forest, an NN and LSTM. However, based on the experimental results that were obtained using the NN and LSTM, we conducted experiments using the normalized and augmented data because the introduction of dropout was not sufficient to suppress overfitting. These experiments showed that the learning results that were obtained using normalized and augmented data stabilized and their convergences accelerated, but overfitting was not suppressed. Then, we conducted experiment (2), namely, action recognition using 100 categories, and the results demonstrated that the accuracy improved after the introduction of dropout. It is assumed that the models could learn more generic human actions if the variety of actions that are used for action recognition is increased.

Next, we investigate the moving image misclassification when learning with the NN and LSTM. Figure 14 shows the classification rate when the fully connected NN model with 3 layers and 500 nodes in the middle layer is used, and Fig. 15 shows the classification rate when the LSTM model with 50 nodes and 10 steps is used. The horizontal axis represents the categories that are inferred by the model, and the vertical axis represents the correct categories. The identification rate differs among the categories, and the bowing category is the easiest to identify. Table 6 represents the match rate of misclassified moving images between the NN and LSTM experiments, namely, the rate at which images are misclassified by both methods. The match rates of misclassified moving images in all three categories are only 50% to 60%, and approximately half of these images are correctly classified by either model. Therefore, ensemble learning with the LSTM and NN can increase the accuracy.

5.2 Distributed Processing over the Sensor and Cloud

Figure 16 compares the data sizes of raw video, images that have been converted from the video at a frame rate of 10 and JSON-formatted feature vectors of the images for each category. The amount of data has been reduced to less than half by converting from video to images and to less than 1/100 by converting to the feature vectors. The figure shows that we could substantially reduce the transfer quantity by preprocessing on the sensor side.

We also investigate the analysis processing times for extracting feature vectors and machine leaning. Table 7 presents the specifications of the computation nodes that are used in the experiments. Table 8 presents the analysis processing time for machine learning inference per 1000 data.

![Fig. 14](classification_rate_with_NN.png)

**Fig. 14** Classification rate when learning with the NN (3 middle layers and 500 nodes) model [1].

![Fig. 15](classification_rate_with_LSTM.png)

**Fig. 15** Classification rate when learning with the LSTM (10 steps and 50 nodes) model [1].

| Category          | Match rate |
|-------------------|------------|
| writing           | 39.28%     |
| reading newspaper | 56.82%     |
| bowing            | 51.72%     |
Fig. 16 Comparison of the data volume before versus after the pre-procesing of moving images [1].

Table 7 Specifications of the computation nodes that were used in the experiment.

| OS               | CentOS Linux release 7.5.1804 (Core) |
|------------------|--------------------------------------|
| CPU              | Intel Xeon Gold 6148 CPU 2.6 GHz     |
|                  | 14 Cores (28 Threads)                |
| GPU              | NVIDIA Tesla V100 for NVLink 16GiB HBM2 |
| Memory           | 384 GiB DDR4 2666 MHz RDIMM (ECC)    |

Table 8 Comparison in terms of the inference time at the cloud.

| Method                | Processing time per 1000 data |
|-----------------------|------------------------------|
| (2-1a) NN             | 0.031 sec                    |
| (2-1b) NN             | 0.029 sec                    |
| (2-2a) LSTM w/ 10 time steps | 0.091 sec                |
| (2-2b) LSTM w/ 20 time steps | 0.086 sec                |
| (2-2c) LSTM w/ 30 time steps | 0.143 sec                |
| (2-2d) LSTM w/ 30 time steps | 0.202 sec                |

Table 9 Specifications of the computation nodes that are used in the experiment with OpenPose and Lightweight OpenPose.

| OS              | Ubuntu 18.04.4 LTS |
|-----------------|--------------------|
| CPU             | Intel(R) Xeon(R) Gold 5117 CPU @ 2.00GHz |
|                  | 28 Cores (56 Threads) |
| GPU             | NVIDIA Corporation GV100GL [Tesla V100 PCIe 32GB] |
| Memory          | 128GiB             |

Table 10 Comparison of keypoint extraction times using OpenPose and Lightweight OpenPose.

| Method                | OpenPose (C++) | Lightweigh OpenPose (C++) | Lightweigh OpenPose (Python) |
|-----------------------|----------------|--------------------------|-------------------------------|
| CPU                   | 0.014 fps      | 26 fps                   | 1.1 fps                       |
| GPU                   | 19 fps         | N/A                      | 5.0 fps                       |

using each of the datasets that are utilized in experiment (2), namely, action recognition on 100 categories. The inference time per 1000 data is approximately 0.03 to 0.20 seconds. In contrast, the average processing time for extracting feature vectors using OpenPose from all frames in each moving image is 11.13 seconds, and the average time per frame is 0.089 seconds. According to the result, the computational complexity at the sensor side is too high because extracting feature vectors using OpenPose takes longer than machine learning inference.

We conduct experiments in which we compare our approach to Lightweight OpenPose [17]. Lightweight OpenPose is a much lighter version of OpenPose and enables real-time inference on CPU. It provides a pre-trained COCO model with 18 keypoints and C++ and Python demos. The mean average precision (mAP) using the COCO dataset is 65.3% for OpenPose, compared to 40% for Lightweight OpenPose, as described in [6] and [17]. We compare the processing times for extracting keypoints from videos using OpenPose and two versions of Lightweight OpenPose. Table 9 presents the specifications of the computation nodes that are used in the experiments. Table 10 presents the number of processed frames per second (FPS). The C++ demo of lightweight OpenPose is provided using Intel® OpenVINO™ Toolkit [18]. We did not conduct an experiment using the C+ version of Lightweight OpenPose with GPU because the GPU of the node that was used in the experiment is not supported. The results demonstrate that the C++ version of Lightweight OpenPose with CPU can extract keypoints faster than OpenPose with GPU. However, the AP of Lightweight OpenPose is more than 20% lower than that of OpenPose due to the reduction in weight. Consequently, Lightweight OpenPose may not be able to extract accurate keypoints, and the accuracy of action recognition using it will be significantly reduced. In contrast, OpenPose provides a total of 135 keypoints provided from its own BODY25 keypoints extraction model and face and hand models. Hence, we adopt OpenPose in this study to increase the accuracy of action recognition.

5.3 Evaluation of the Proposed System by Comparing It with the 3D ResNet Model

Finally, in an action recognition experiment, we compare the proposed system with a 3D ResNet model that has been fine-tuned on the STAIR Actions dataset to evaluate the accuracy and processing time of the proposed system. The 3D ResNet model showed the best accuracy for action recognition using the STAIR Actions dataset [10]. We use the 3D ResNet model to evaluate our model because we must evaluate the performance of our action recognition model using only keypoints by comparing it with the model that realizes the best accuracy.

A CNN is useful for image analysis because it can extract features in a region by performing convolution, but it suffers from the loss of the gradient by stacking multiple layers. ResNet enables the use of multiple CNN layers, and 3D ResNet applies ResNet to moving image analysis by adding a timeline. In this model, all frames of each moving image are divided into sets of 16 frames, and we recognize actions using the mean of the inference results for each set. The average processing time for the recognition of 1 moving image by the 3D ResNet model is 7.93 seconds, and the action recognition accuracy is 84.9%. The 3D ResNet model has
the advantages that it has the highest accuracy for action recognition using the STAIR Actions data set and its inference is faster than that of the proposed method. The disadvantages are that it takes a substantial amount of time to train the model and that it cannot be applied for distributed processing over sensors and a cloud because videos must be sent to the cloud as input data. The proposed system has the advantages that the keypoint information is reusable for tasks other than motion identification and that the amount of information can be significantly reduced by converting videos to keypoints. The disadvantages are that the action recognition accuracy is lower than that of 3D ResNet and that the keypoint extraction using OpenPose takes a substantial amount of time. Therefore, we claim that the proposed model is useful for the services that we aim to provide because the stored feature vector data of human keypoints can be reused for not only recognizing action but also analyzing further actions, e.g., detecting abnormal actions before a person falls down.

6. Related Work

Several studies have been conducted on human action recognition using deep learning. It has become possible to recognize more complex motions with high accuracy by using various deep learning methods such as convolutional neural networks (CNNs) and LSTM. Hara et al. [19] investigated behavior identification from video input using various 3D CNN-based methods that perform convolution in a three-dimensional space that is formed by adding a one-dimensional time space to a two-dimensional space. They realized improved action identification performance using a 3D CNN that was based on a residual network (ResNet) [12]. Pigou et al. [20] pooled features to consider temporal information in a video and realized performance improvements by combining recursion and temporal convolution. Li et al. [21] conducted multi-class human action classification with CNN using data that were obtained by RFID. Fragkiadaki et al. [22] recognized and predicted human actions by generating human pose motion captures using a model with an encoder and decoder before and after LSTM. Ordóñez et al. [23] conducted recognition using data from an accelerometer, a gyroscope and a magnetometer with a mode that combined CNN and LSTM. However, the computational complexity of such an action recognition process is too large for analysis using deep learning in ordinary homes.

Various distributed processing methods that use edge computing or fog computing to analyze using data that are obtained from IoT devices have been investigated to address these issues. Li et al. [24] proposed a method for analyzing data that were obtained from IoT devices using deep learning. Since the processing capacity in an IoT device is limited, they used edge computing to construct a deep learning application for the IoT devices, and they designed and evaluated an offload strategy for performance optimization. Tang et al. [25] proposed a hierarchical distributed fog computing architecture that supports massive infrastructure integration for future smart cities. Yang [26] investigated the models and architectures of four typical fog computing systems and analyzed the design space for four dimensions: system, data, human, and optimization.

In addition, users’ privacy must be protected in the collection of multimedia data in a fog and a cloud. Ma et al. [27] proposed a scheme for collecting multimedia data from devices such as smartphones. They balanced collecting multimedia data and protecting personal data by creating virtual aggregation areas that offload the collection of sensor data from multiple devices and their encryption and decryption. Sood [28] proposed a method for protecting data from inside attacks, such as data snooping from a cloud service provider, and from outside attacks when collecting and storing enterprise multimedia data (EMD). Thus, multimedia data security in edge computing is usually based on storing encrypted data in the cloud and storing the information that is necessary for decryption in the fog or at the edge.

In this study, we investigate whether we can sufficiently recognize actions using only the coordinate values of human key points that are included in a moving image and not the original video data. In the proposed system, we can store data in a fog or a cloud without invading users’ privacy because we cannot reconstruct videos from these keypoint data. In addition, we can significantly reduce the amount of data by converting from video to keypoints and, thus, reduce the communication cost and the amount of data that are stored in a fog or a cloud. The objective is to ensure sufficient recognition accuracy even after reducing the quantity of data and separating the processing between the edge and cloud to analyze the moving images in real time.

7. Conclusions

We proposed a method of distributed processing over the edge and the cloud to address the processing delay and the problem of privacy. We used feature vectors that were generated from a video of the STAIR Actions dataset using OpenPose and measured the action recognition accuracy with multiple machine learning methods. It was found from action recognition using 3 categories that multiple machine learning methods can recognize 3 actions with more than 80% accuracy, but the overfitting issue remains unresolved. In contrast, we observed that the introduction of dropout in action recognition using 100 categories alleviates overfitting because the models can learn more generic human actions by increasing the variety of actions. We can also substantially reduce the transfer quantity from the sensor to the cloud by preprocessing on the sensor side. In addition, we measured the analysis processing times at the sensor and the cloud to investigate the feasibility of recognizing actions in real time, and we evaluated the proposed system by comparing it with the 3D ResNet model in recognition experiments.

We observed that the 3D ResNet model can recognize actions faster and more accurately than the proposed system because the original moving images are analyzed directly.
However, this method cannot be applied for the scenario of distributed processing over the sensor and cloud that is discussed in this paper. We claim that the feature vectors of human key points are more reusable than moving and still images; thus, the proposed method is beneficial.

In the future, we will conduct an evaluation in a more realistic environment using home sensors, and then we aim at recognizing actions in real time.

Acknowledgments

This paper is partially supported by JSPS KAKENHI Grant Number JP19H04089, ROIS NII Open Collaborative Research 2020-20S0501, and the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] C. Takasaki, A. Takefusa, H. Nakada, and M. Oguchi, “A study of action recognition using pose data toward distributed processing over edge and cloud,” Proc. 11th IEEE International Conference on Cloud Computing Technology and Science (CloudCom2019), pp.111–118, 2019.

[2] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, “Deep learning for IoT big data and streaming analytics: A survey,” IEEE Communications Surveys & Tutorials, vol.20, no.4, pp.2923–2960, 2018.

[3] “Nvidia Jetson.” https://developer.nvidia.com/embedded-computing

[4] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, “Edge-centric computing: Vision and challenges,” ACM SIGCOMM Computer Communication Review, vol.45, no.5, pp.37–42, 2015.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp.13–16, ACM, 2012.

[6] Z. Cao, G. Hidalgo Martinez, T. Simon, S.-E. Wei, and Y. Sheikh, “Openpose: Realtime multi-person 2d pose estimation using part affinity fields,” IEEE Transactions on Pattern Analysis and Machine Intelligence, pp.172–186, 2019.

[7] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, “Hand keypoint detection in single images using multiview bootstraping,” Proc. IEEE conference on Computer Vision and Pattern Recognition, pp.1145–1153, 2017.

[8] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields,” Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.7291–7299, 2017.

[9] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, “Convolutional pose machines,” Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.4724–4732, 2016.

[10] Y. Yoshikawa, J. Lin, and A. Takeuchi, “Stair actions: A video dataset of everyday home actions,” arXiv preprint arXiv:1804.04326, 2018.

[11] “Keras: The Python Deep Learning library.” https://keras.io/

[12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” Proc. IEEE conference on computer vision and pattern recognition, pp.770–778, 2016.

[13] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, “2d human pose estimation: New benchmark and state of the art analysis,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014.

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” Computer Vision – ECCV 2014, ed. D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Cham, pp.740–755, Springer International Publishing, 2014.

[15] H. Kuehne, H. Huang, E. Garrote, T. Poggio, and T. Serre, “Hmdb: A large video database for human motion recognition,” 2011 International Conference on Computer Vision, pp.2556–2563, 2011.

[16] K. Soomro, A. Zamir, and M. Shah, “Ucf101: A dataset of 101 human actions classes from videos in the wild,” CRCVT-TR-12-01, Nov. 2012.

[17] D. Osokin, “Real-time 2d multi-person pose estimation on cpu: Lightweight openpose,” arXiv preprint arXiv:1811.12004, 2018.

[18] “OpenVINO Toolkit.” https://software.intel.com/en-us/openvino-toolkit

[19] K. Harra, H. Kataoka, and Y. Satoh, “Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?”, Proc. IEEE conference on Computer Vision and Pattern Recognition, pp.6546–6555, 2018.

[20] L. Pigou, A. Van Den Oord, S. Dieleman, M. Van Herreweghe, and J. Dambre, “Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video,” International Journal of Computer Vision, vol.126, no.2-4, pp.430–439, 2018.

[21] X. Li, Y. Zhang, I. Marsic, A. Sarcevic, and R.S. Burd, “Deep learning for rfid-based activity recognition,” Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM, pp.164–175, ACM, 2016.

[22] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, “Recurrent network models for human dynamics,” 2015 IEEE International Conference on Computer Vision (ICCV), pp.4346–4354, 2015.

[23] F. Ordu Native and D. Roggen, “Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition,” Sensors, vol.16, no.1, p.115, 2016.

[24] H. Li, K. Ota, and M. Dong, “Learning iot in edge: Deep learning for the internet of things with edge computing,” IEEE Network, vol.32, no.1, pp.96–101, 2018.

[25] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, “A hierarchical distributed fog computing architecture for big data analysis in smart cities,” Proc. ASE BigData & SocialInformatics 2015, p.28, ACM, 2015.

[26] S. Yang, “Iot stream processing and analytics in the fog,” IEEE Communications Magazine, vol.55, no.8, pp.21–27, 2017.

[27] S. Ma, T. Zhang, A. Wu, and X. Zhao, “Lightweight and privacy-preserving data aggregation for mobile multimedia security,” IEEE Access, vol.7, pp.111431–111440, 2019.

[28] S.K. Sood, “Mobile fog based secure cloud-iot framework for enterprise multimedia security,” Multimedia Tools and Applications, vol.79, no.15, pp.10717–10732, 2020.

Chikako Takasaki received a B.S. from Ochanomizu University in 2019. She is currently a student in the master’s program in the Department of Computer Science at Ochanomizu University. Her research field is parallel and distributed computing. She is a student member of IPSJ.
Atsuko Takefusa received B.S., M.S. and Ph.D. (Sci.) degrees from Ochanomizu University in 1996, 1998, and 2000, respectively. She previously worked at Ochanomizu University and the National Institute of Advanced Industrial Science and Technology (AIST), and she joined the National Institute of Informatics as an associate professor in 2016. Her research field is parallel and distributed computing including Grid, Cloud and HPC. She is a member of ACM, IEEE, IPSJ and IEICE.

Hidemoto Nakada earned a Ph.D. (Eng.) degree from the University of Tokyo in 1995 and joined the Electrotechnical Laboratory. The lab was merged into AIST (National Institute of Advanced Industrial Science and Technology) in 2001. Since 2015, he has also been working for the University of Tsukuba as an Adjunct Professor of the Cooperative Graduate School. His research fields are parallel and distributed computing, including Grid, Cloud and HPC, and machine-learning-related technologies. He is a member of ACM, IPSJ, JSAI and IEICE.

Masato Oguchi received a Ph.D. from the University of Tokyo in 1995. He worked at NACSIS, which is currently known as the National Institute of Informatics (NII), and at the Institute of Industrial Science in University of Tokyo and the Research and Development Initiative in Chuo University. He joined Ochanomizu University in 2003 as an associate professor. Since 2006, he has been a professor at the Department of Information Sciences, Ochanomizu University. His research fields are network computing and data engineering, including high-performance computing, mobile networking, network security, and information management. He is a member of IEEE, ACM, IEICE, IPSJ, and DBSJ.