Indonesian Journal of Electrical Engineering and Computer Science
Vol. 34, No. 1, April 2024, pp. 463~471
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v34.i1.pp463-471

Performance evaluation of adaptive offloading model using hybrid machine learning and statistic prediction

Siwoo Byun[1], Seok-Woo Jang[1], Joonho Byun[2]
[1]Department of Software Engineering, Anyang University, Anyang-shi, South Korea
[2]Department of Artificial Intelligence, Sogang University, Seoul, South Korea

Article Info

Article history:

Received Dec 11, 2023
Revised Dec 29, 2023
Accepted Jan 3, 2024

Keywords:

Adaptive offloading model
Compound sensor context
Fault diagnosis
Machine learning
Pattern matching

ABSTRACT

We introduce fast sensor diagnosis and focuses on intelligent offloading skills to enhance the sensor data screening efficiency. This study proposed the adaptive offloading model based on statistics-based prediction feedback and sensor candidate filtering. For the statistics-based filtering, sliding sensor grids and compounded sensor context were devised. This study also proposed hybrid prediction model using support vector machine (SVM) and k-nearest neighbors (KNN) machine training for the adaptive offloading. Therefore, the sensor information that is highly likely to be the cause of the actual device faults can be selected and transmitted, resulting in improved offloading performance. The test results through Google Colab show that the fault prediction accuracy of proposed models is 95%.

Corresponding Author:

Siwoo Byun
Department of Software Engineering, Anyang University
Samduk-ro, 37th Street, Anyang-shi, Kyongki, 430714, South Korea
Email: swbyun@anyang.ac.kr

1. INTRODUCTION

Fault diagnosis involves using data from various internet of things (IoT) sensors to identify the cause of a fault, and IoT data is required to analyze abnormal symptoms [1]–[4]. However, as IoT devices continue to become smaller and more complex, the process of fault diagnosis is becoming slower and slower [5]–[7]. Therefore, in order to cope with the explosion of IoT sensor data, the importance of fast fault diagnosis is gradually increasing [8]–[10], which should be preceded by efficient offloading.

Example 1 (Failure of a robot arm): industrial robot arms have problems such as motor failure, joint bearing failure, controller failure, and gear reducer failure due to long operation time and heavy workload as shown in Figure 1. For example, when the torque was increased to 30 degrees in the forward direction of the joint and then moved 10 degrees in the reverse direction, the accumulation of metal fatigues lead to a large vibration in the gear reducer. These vibrations cause serious errors in the welding or assembly process. The joint part is shaken heavily by forward and reverse torque and acceleration and deceleration loads, resulting in abnormal noise. If detection software responds quickly to the failure of the robot arm, it improves production accuracy.

Edge computing [11]–[13] is advanced skill that minimizes congestion and security issues in centralized data centers. It can provide effective IoT services to end users by processing tasks as locally as possible to reduce traffic and latency. By processing tasks as locally as possible, it can provide effective services to users.

Figure 1. Example of a robot arm failure

2. METHOD

Fault diagnosis takes a long time because of the long computation time required for offloading. For fast troubleshooting, error-indicating sensor data contributes to the pre-screening of worthless data. Reducing sensor data computation results in fast fault analysis [6], [10].

2.1. Statistics-based filtering

The time required to detect the error pattern is proportional to the type of sensors, so each error should be compared with the input data by fast filters. The statistics-based filtering scheme is based on sensor candidate filtering and fast sliding window [14]-[16]. For example, by using fast pattern matching and prediction scores to minimize the number of sensor candidates, the diagnosis of a fault can be carried out more quickly.

In this study, the fault prediction scores are periodically calculated by sliding grid window protocol. A scalable sensor grid was developed using the simple sliding window protocol. The grid scale of the cooling fan is one tenth as shown in Figure 2.

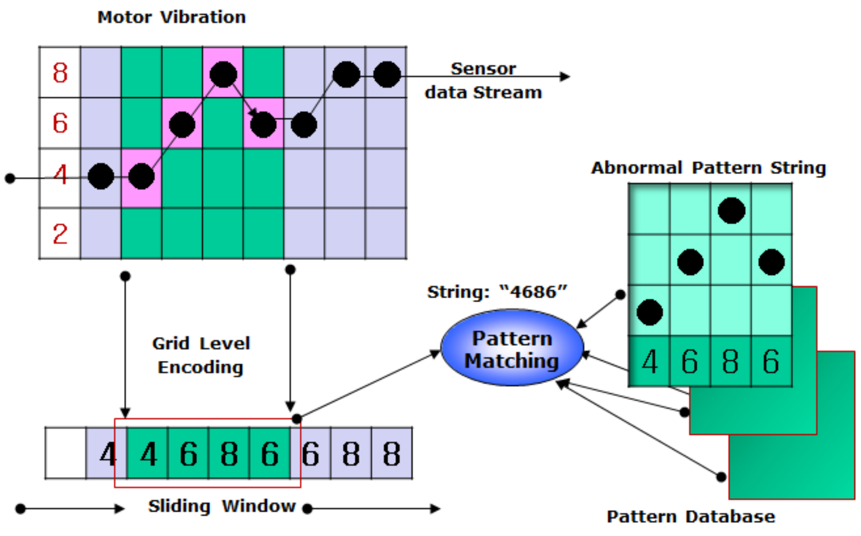


Figure 2. Example of scaled sensor grid

The followings are the pseudo codes of sensor-grid sliding window manager and adaptive offloading manager as shown in Code 1:

Code 1. Pseudo code for adaptive sensor offloading
```
Procedure Sensor-Grid Sliding Window Manager()
#Input snapshot of sensor data;
#Output fault prediction hit results;
Begin
Wait until next sliding interval;
Move sliding window;
For (all sensor item, sitem-i, in the sliding window;; i++) {
      hit-score=0;
      k=0;
      For (all sensor data, sdata-j in sitem-i;; j++) {
        if (sdata-j is not a faulty signature)
          continue;
        Grid-Scaling sdata-j to sd-j for pattern matching;
        if (sd-j is matched to the sensor value in labeled fault pattern)
          hit-score++;
        else
           hit-score--;
        k++;
      }
      fault-hit-ratio[i]=hit-score/k*100;
}
Make {Sensor-Context} with fault-hit-ratio[];
Send {Sensor-Context} to Adaptive Offloading Manager;
End;

Procedure Adaptive Offloading Manager() {
#Input stream of sensor data, Sensor-Context
#Output minimized stream of sensor data,
Begin
Read stream of sensor data;
For (all sensor item, sitem-i, in the input stream; i++) {
      if (sitem-i is not in Sensor-Context)
        continue;
      // Sensor Data Offloading
      For (all sensor data, sdata-j in sitem-i; j++) {
        Grid-Scaling sdata-j to sd-j for pattern matching;
        if (sd-j is matched to fault pattern) {
              insert sd-j into output stream of sensor data;
      }
}
End;
```

2.2. Machine learning-based filtering

The goal is to enhance the efficiency of offloading through the hybrid use of machine learning models such as support vector machine (SVM) [17], [18] and k-nearest neighbors (KNN) [19], [20], and a SVM and KNN-based machine learning (SKML) model was proposed in this study. The support vector classifier (SVC) algorithm is a member of the SVM group and is used for categorical variables. It is a classification algorithm that classifies two or more pieces of data and is expected to have high generalization performance [17], [21].

The KNN algorithm is a popular method for classification purposes because of its low training effort. The principle is to identify the *k* labels that are closest to the value you want to learn or predict, and then take the most frequent label as the value. In other words, it assumes a similarity between the new case data and the existing cases, and places the new case in the most similar category [19]-[24].

The SKML utilizes the filtering of candidates by relevance learning and can reduce the subsequent transmission and computation burden. It is effective in selecting candidates with high relevance, except for candidates with low relevance prior to decision tasks. The best prediction model among (Statistics, SVM, and KNN) is called as selected prediction model in this study. This selected model will be transmitted to the adaptive offloading manager at regular intervals. To make machine learning more effective, we considered a technique that uses an artificial intelligence boosting algorithm to learn and robustly extract useful fault signature features. This extraction task uses fault signature features from the input stream and an AdaBoost [21] learning algorithm. AdaBoost is a boosting algorithm that combines several weak classifiers with poor classification performance to build a strong classifier with good classification performance. Typically, a weak classifier refers to a classifier with a classification error greater than 50% and a strong classifier refers to a classifier with a small classification error. AdaBoost algorithms are used in image and pattern processing because of their ease of implementation and high performance.

The AdaBoost training algorithm generates a set from the original data using the error of the hypothesis, then uses it to learn a new hypothesis again and construct a data set from the learned hypotheses. In addition, the

AdaBoost algorithm focuses on data that is difficult to classify by reducing the weight of data correctly predicted by the hypothesis and increasing the weight of data incorrectly predicted according to the hypothesis errors. The AdaBoost algorithm has the advantage that learning errors are less than the upper bound of theoretical errors, and generalization errors are reduced along with learning errors. However, our experiments did not show any significant difference from the proposed technique, so we have excluded it from this experiment.

The following code 2 is for SKML training and testing. The load_data() is a function to load data from an Excel file and select columns specified by col. The KNN() and SVM() are functions to train classification models using X_train and Y_train. The Train_and_Test() is a function to split data into training and test sets. The main() is a function to iterate over different combinations of columns for training, load data, compute accuracy, and store the mean accuracy for each combination in a dictionary.

Code 2. SKML training and test codes
```
#Function to load data from an excel file and select columns specified by col
def load_data(col):
  data=pd.read_excel('data.xlsx',usecols=col,header=8)
  SensorDataBlock=data.loc[0:BlockSize]
  # Select sensor data block
  return SensorDataBlock

# Function to train an SVM model using X_train and y_train,
def KNN(Xtr,Xtst,ytr,ytst): #KNN-2
  clf = neighbors.KNeighborsClassifier(n_neighbors=3)
  clf.fit(Xtr, ytr)
  y_pr = clf.predict(Xtst)
  acc = accuracy_score(ytst, ypr)
  return acc

def SVM(Xtr,Xtst,ytr,ytst): #SVM-1
  svm=SVC()
  svm.fit(Xtr,ytr)
  # Train the model using X_train and y_train
  ypr=svm.predict(Xtst) #Predict labels for X_test
  return ytr,ypr

# Function to split data into training and test sets,
def Train_and_Test(data):
  data=np.array(data)
  X=data[:,:-1]
  y=data[:,-1].astype(int)

  Xtr,Xtst,ytr,ytst=train_test_split(X,y, test_size=0.2)
  #Train the ML model and predict labels for the test set
  ytr,ypr=SKML(Xtr,Xtst,ytr,ytst)
  acc = accuracy_score(ytst,ypr)
  return acc

# Function to iterate over different combinations of columns for training, load data,
# calculate accuracy, and store the mean accuracy for each combination in a dictionary
def main():
    columns = {
        1:  's1',  # x_Sensor data
        2:  's2',  # y_Sensor data
        3:  's3',  # z_Sensor data
        # ...
        10: 's4',  # h_Sensor data
        12: 'fault_sign',  # fault_sign ...
  }
    acc_dict = {}
   #loop through different combinations
    for cols in [ 'all combination of sensor_data' ]:
        col_names = [columns[col] for col in cols]
        data = load_data(cols)  # load data
        acc = []
        for i in range(100):  # repeant N times
            accuracy=Train_and_Test(data) #accuracy
           acc.append(accuracy)
        mean_acc = sum(acc) / len(acc) # mean accuracy
        acc_dict[tuple(col_names)] = mean_acc
     sorted_acc = sorted(acc_dict.items(), key=lambda x: x[1], reverse=True) #sort
    for cols, acc in sorted_acc:
        print(f"Variables: {cols}, Mean Accuracy: {acc:.4f}") # Print Analysis Results
#...
#End:=> Send (Analysis Results) with (Seleted Model) to (Offloading Manager)
```

3. RESULTS AND DISCUSSION

In this study, AI-Hub [25], [26] was used to verify the performance of the statistics and SKML-based. In Table 1, the vibrations of the motor are classified as normal up to 10 and as fault warning above 10. Failures are mainly caused by old bearings and mounting misalignment.

In Table 1, three sensor types denoted as s1, s2, and s3 mean the value of detailed directional vibration. The fault sign means the degree of fault and is classified as normal, warning, and danger for symptom analysis. The three types of vibration sensors are *x*-axis, *y*-axis, and *z*-axis, which are arranged in the Table 1.

Table 1. Sample values of sensor data

Sensor s1:x	Sensor s2:y	Sensor s3:z	Sensor s4:humidity
4.003906	1.641602	3.563477	51
3.803711	0.720703	4.324219	76
2.962891	0.360352	4.524414	50.9
2.522461	1.28125	5.084961	76.8
1.561523	2.522461	5.245117	50.9

3.1. Experimental software and dataset

The experimental system used MS-Windows64-I7 and CoLab with 16G memory and 1.5TB hdd. The experimental data consists of 100 pieces per block. The number of detailed iterations per block is 20 times. The original file configuration is (date, filename, data label, label_no, motor spec, period, sample rate, data length, {time, data} array). The extract data from the configuration is {time, data}.

– Experimental system: MS-Windows64-I7, 16G, 1.5TB, CoLab.
– Experimental data: 2 sets (100 pieces per block).
– Number of experiments: Number of detailed iterations per block: 20 times.
– Original file configuration: (date, file name, data label, label_no, motor spec, period, sample rate, data length, {time, data} array).
– Extract data: {time, data} values to the experiment.

3.2. Test results of statistcs-based prediction

Table 2 compares the fault prediction for each sensor type tracked by the proposed sliding window. Comparing each sensors, it is most effective to preferentially use s2 sensor for best fault prediction. Additionally, better performance can be obtained by adding s2 sensor in the experiment. That is, the fault prediction accuracy increased from 66% to 89% by compound context of (s2,s3). This also means that most of the unnecessary data among the original sensor data can be removed except missed data of 11%. As a result, effective filtering using compounded sensor contexts and adaptive offloading with sliding grids can significantly reduce the number of sensors involved in fault prediction and minimize the amount of sensor data to be processed.

Table 2. Comparison of fault prediction for each sensors

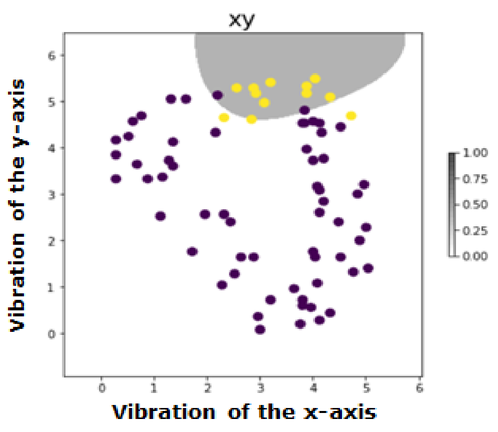
Sensors	Sensor s1:x	Sensor s2:y	Sensor s3:z	...	Sensor (y,z)
Hit count	135	296	234		397
Hit-rate	29.5	65.9	52.3		88.6
Prediction levle	low	high	mid		best

3.3. Test results of hybrid prediction model

3.3.1. SVM-based prediction part

To evaluate the machine learning models of SKML. Different combinations of the three sensors are used and correlated and uncorrelated sensors are classified by the hybrid prediction model. In the case of predicting by SVM-based model, the final filtering of the sensor data shows the test results using different sensor combinations, which are shown in Figure 3.

Finding the best between (xy, xz, yz) is the goal of the test. In this experiment, one sensor is not enough, but a combination of three sensors essentially requires a lot of resource consumption. In Figure 4, the sensor combination of x and z showed lower results in terms of accuracy, recall and F1-score compared to the combination of x and y. In particular, the accuracy of the combination of x and z was 84%, indicating that the combination of x and z was not a useful combination.

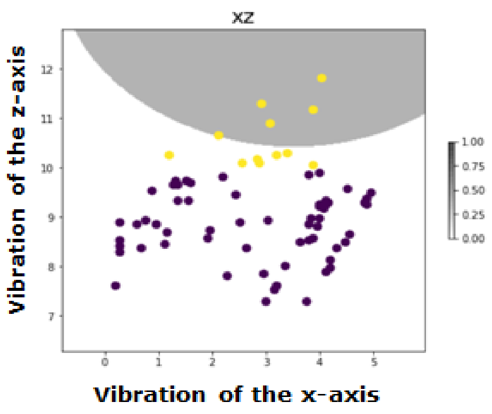


```
              precision    recall  f1-score   support

           0       0.88      1.00      0.94        15
           1       1.00      0.50      0.67         4

    accuracy                           0.89        19
   macro avg       0.94      0.75      0.80        19
weighted avg       0.91      0.89      0.88        19
```

Figure 3. Results of combining sensors x and y



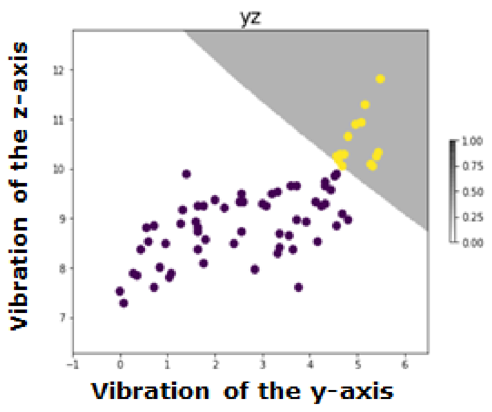
```
              precision    recall  f1-score   support

           0       0.83      1.00      0.91        15
           1       1.00      0.25      0.40         4

    accuracy                           0.84        19
   macro avg       0.92      0.62      0.65        19
weighted avg       0.87      0.84      0.80        19
```

Figure 4. Results of combining sensors x and z

In Figure 5, the sensor combination of y and z showed relatively higher performance in terms of recall and F1-score compared to other sensor combinations. In conclusion, the combination of y and z was found to be the most useful combination with 95% accuracy. Instead of (x,y,z), (y,z) is chosen as the best choice, as the inclusion of low-relevance data hinders AI learning and error detection, and storage resources can be reduced by 33%. It will need to be retrained using recent blocks if the optimal combination of (y,z) has changed.



```
              precision    recall  f1-score   support

           0       1.00      0.94      0.97        17
           1       0.67      1.00      0.80         2

    accuracy                           0.95        19
   macro avg       0.83      0.97      0.88        19
weighted avg       0.96      0.95      0.95        19
```

Figure 5. Results of combining sensors y and z

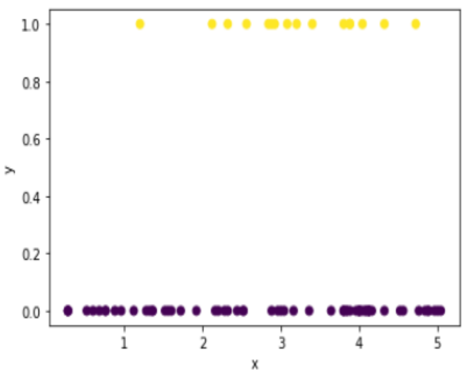
3.3.2. KNN-based prediction part

This part evaluates the performance of our KNN-based module on the same sensor data in the exactly the same way. The results showed that KNN had a slightly lower prediction than SVM as shown in summary Table 3. Consequently, proposed SKML scheme selects superior SVM-based model of 0.95 accuracy and sends its prediction information to the adaptive offloading manager.

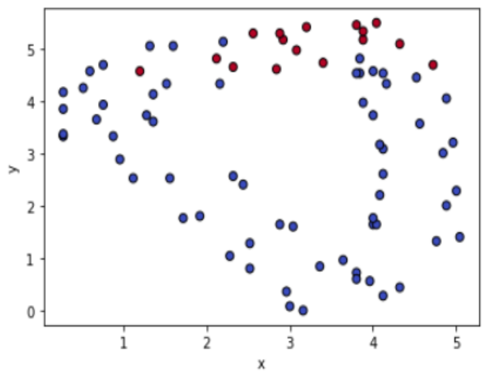
In addition, Figure 6 is the visualization graph of the KNN training model. Figure 6(a) is the classification graph using sensor x. Figure 6(b) is the classification graph using the combinations of sensor x and sensor y. Figure 6(c) is the classification graph using the combinations of sensor x, y, and z. The results show: (y,z) is the best choice with an accuracy of 0.94, while the combination of (*x*,*y*,*z*) showed an accuracy of 0.91. This means that adding less relevant sensor candidates leads to accuracy degradation and unnecessary waste of data spaces.

Table 3. Accuracy summary of proposed (SVM and KNN)-hybrid training model

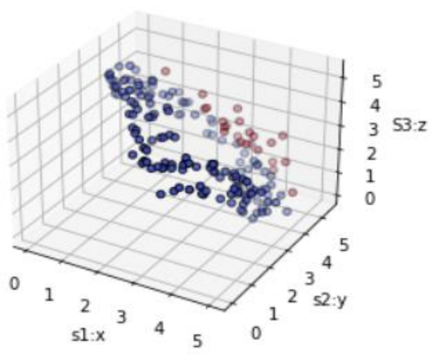
Combination	x+y	x+z	y+z
Accuracy mark of SVM	0.89	0.84	(0.95)
Accuracy mark of KNN	0.88	0.82	0.94



(a)



(b)



(c)

Figure 6. Sensor combinations of KNN part of SKML; (a) one sensor: 0.81, (b) two sensors: 0.94, and (c) three sensors: 0.91

4. CONCLUSION

We introduced sensor data offloading issues for IoT fault diagnosis. The fast offloading model used statistics-based prediction feedback and sensor candidate filtering. The statistics-based filtering exploits sliding sensor grids and compounded sensor context. We also proposed a hybrid adaptive model called SKML using SVM and KNN training feedback. The SKML model showed the better accuracy than the statistics-based model. However, compared to SKML which requires heavy data learning, the statistics-based model has the advantage of being able to quickly track optimal sensor combinations using light statistical computations on the edge gateways which have limited computational resources.

We evaluated the performance of SKML using CoLab and Python. In our experiments with public datasets, we were able to increase the prediction accuracy of SKML by up to 95%. We concluded that the proposed filtering model can reduce the overhead of sensor data computation and network transfer to remote cloud systems. Future studies include a lightweight offloading model for industrial sensor devices and sensor-aware intelligence for the extended SKML models.

ACKNOWLEDGEMENTS

This research was supported by NRF funded by the Ministry of Education, Science and Technology (2022R1F1A1063659).

REFERENCES

[1] L. Ashwini, H. Lee, M. Hwang, “Implementation of IoT application using geofencing technology for protecting crops from wild animals,” *Asia-pacific Journal of Convergent Research Interchange*, vol. 6, no. 6, pp. 13-23, 2019, doi: 10.21742/apjcri.2020.06.02.

[2] C. Sekhar and N.Suneetha, “A study on data categorization for data analytics,” *International Journal of Internet of Things and Big Data*, vol. 3, no. 1, pp. 1-6, 2018, doi: 10.21742/ijitbd.2018.3.1.01.

[3] J. Lee and H. Kim, “Design and implementation of IoT devices to foster emotions and prevent dementia for senior citizens,” *Journal of Next-generation Convergence Technology Association*, vol. 5, no. 3, pp. 347-353, 2021, doi: 10.33097/JNCTA.2021.05.03.347.

[4] S. Byun, “Sensor query control for IoT data monitoring,” *International Journal of Control and Automation*, vol. 11, no. 9, pp. 109-118, 2018, doi: 10.14257/ijca.2018.11.9.11.

[5] H. Lee, S. Kim, S. Kang, and Y. Lim, “A fault dropping technique with fault candidate ordering and test pattern ordering for fast fault diagnosis,” *Semiconductor and Devices*, vol. 46, no. 3, pp. 32-40, 2009.

[6] S. Mansfield, E. Vin, and K. Obraczka, “An IoT-based system for autonomous, continuous, real-time patient monitoring, and its application to pressure injury management,” in *Proc. International Conference on Distributed Computing in Sensor System*, pp. 66-68, 2021, doi: 10.1109/DCOSS52077.2021.00024.

[7] S. Byun, “Sensor data transmission control using selective dynamic compression for mobile IoT devices,” *International Journal of Control and Automation*, vol. 12, no. 8, pp. 87-96, Aug. 2019, doi: 10.33832/ijca.2019.12.8.08.

[8] Q. Li, J. Zhao, Y. Gong, and Q. Zhang, “Energy-efficient computation offloading and resource allocation in fog computing for Internet of Everything,” *China Communications*, vol. 16, no. 3, pp. 32-41, 2019.

[9] S. Koo and Y. Lim, “A multi-objective computation offloading algorithm for dependent tasks based on a mobile edge computing environment,” *KIISE Transactions on Computing Practices*, vol. 27, no. 2, pp. 122-127, 2021, doi: 10.5626/KTCP.2021.27.2.122.

[10] Y. Son, J. Jeong, and Y. Lee, “An adaptive offloading method for an IoT-cloud converged virtual machine system using a hybrid deep neural network,” *Sustainability*, vol. 10, no. 11, pp. 1-15, 2018, doi: 10.3390/su10113955.

[11] K. Xiao, Z. Gao, Q. Wang, and Y. Yang, “A heuristic algorithm based on resource requirements forecasting for server placement in edge computing,” in *Proc. IEEE/ACM Symposium on Edge Computing (SEC),* Seattle, WA, USA, pp. 354-355, 2018, doi: 10.1109/SEC.2018.00043.

[12] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, “A survey on the edge computing for the internet of things,” *IEEE Access,* vol. 6, pp. 6900-6919, 2017, doi: 10.1109/ACCESS.2017.2778504.

[13] J. Anitha, W. Godfrey, “A survey of edge computing in IoT devices,” in *Proc. of the International Conference on Innovative Computing and Communication (ICICC)*, 2022, doi: 10.2139/ssrn.4023176.

[14] K. Xiao, Z. Gao, Q. Wang, and Y. Yang, “A heuristic algorithm based on resource requirements forecasting for server placement in edge computing,” in *Proc. Of the IEEE/ACM Symposium on Edge Computing (SEC)*, Seattle, WA, USA, pp. 354-355, 2018, doi: 10.1109/SEC.2018.00043.

[15] I. Cho, H. Kwon, “Patient adaptive pattern matching method for premature ventricular contraction (PVC) classification,” *Journal of the Korea Institute of Information and Communication Engineering,* vol. 16, no. 9, pp. 2021-2030, 2012, doi: 10.6109/jkiice.2012.16.9.2021.

[16] S. Byun, “Composite context-based offloading scheme for IoT fault diagnosis,” *Journal of Next-generation Convergence Technology Association*, vol. 7, no. 5, pp. 762-773, 2023, doi: 10.33097/JNCTA.2023.07.05.762.

[17] JavaTpoint, “Support Vector Machine Algorithm”, 2024. [Online]. Available: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

[18] G. V. Gopal and G. R. M. Babu, “An ensemble feature selection approach using hybrid kernel based SVM for network intrusion detection system,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS),* vol. 23, no. 1, pp. 558-565, 2021, doi: 10.11591/ijeecs.v23.i1.

[19] M. Kasian and H. Kilavo, “A comparative study on performance of SVM and CNN in Tanzania sign language translation using image recognition,” *Applied Artificial Intelligence,* vol. 36, no. 1, pp. 452-466, 2022, doi: 10.1080/08839514.2021.2005297.

[20] R. Chivukula, T. J. Lakshmi, S. Uday, and S. T. Pavan, “Classifying clinically actionable genetic mutations using KNN and SVM,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS),* vol. 24, no. 3, pp. 167-1679, Dec. 2021, doi: 10.11591/ijeecs.v24.i3.

[21] F. Chollet, “Xception: deep learning with depthwise separable convolutions,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* Honolulu, HI, USA, 2017, pp. 1800-1807, Nov 2017, doi: 10.1109/CVPR.2017.195.

[22] E. H. Ali, H. A. Jaber, and N. N. Kadhim, “New algorithm for localization of iris recognition using deep learning neural networks,” *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS),* vol. 29, no. 1, pp. 110-119, 2023, doi: 10.11591/ijeecs.v29.i1.pp110-119.

[23] J. Byun and S. Byun, “Experiment of purpose-oriented correction filters using color weakness deep learning,” *Journal of Next-generation Convergence Technology Association,* vol. 7, no. 8, pp. 1190-1196, 2023, doi: 0.33097/JNCTA.2023.07.08.1190.

[24] P. Hyun-Ae and K. R. Park, “Iris recognition based on score level fusion by using SVM,” *Pattern Recognition Letters,* vol. 28, no. 15, pp. 2019-2028, 2007, doi: 10.1016/j.patrec.2007.05.017.

[25] AiHub, “Hardware failure prediction sensor data”, 2024. [Online]. Available: https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=238
[26] AiHub, “Water line leak detection data”, 2024. [Online]. Available: https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&dataSetSn=138

BIOGRAPHIES OF AUTHORS

Siwoo Byun received his M.S. and Ph.D. degrees in Computer Science from KAIST, South Korea, in 1991, and 1999, respectively. Since March 2000, he has been a professor in the Department of Software, Anyang University, South Korea. He Holds a Ph.D. degree with a specialization in database transaction analysis. His research areas are IoT data processing, IoT machine learning, embedded system, big database, flash memory database, medical healthcare and sensor pattern recognition. He is a recipient of best research scholar award and best paper award. His new research interests include enhanced image/signal processing, deep learning, and road facility recognition, IoT storage system, and compound sensor offloading. He can be contacted at email: swbyun@anyang.ac.kr.

Seok-Woo Jang received his B.S., M.S., and Ph.D. degrees in Computer Science from Soongsil University, Seoul, Korea, in 1995, 1997, and 2000, respectively. From October 2003 to January 2009, he was a senior researcher with the Construction Information Research Department at Korea Institute of Construction Technology (KICT), Ilsan, Korea. Since March 2009, he has been a professor at the Department of Software, Anyang University, Korea. His primary research interests include robot vision, augmented reality, video indexing and retrieval, biometrics and pattern recognition. He can be contacted at email: swjang@anyang.ac.kr.

Joonho Byun received his B.S. degree in Electronic Engineering from Kookmin University, Seoul, Korea, in 2022. From March 2023 to January 2024, he was an AI researcher with the Industrial AIoT research center at Anyang University, Anyang, Korea. He is in an M.S. at the Graduate School of Artificial Intelligence at Sogang University, Seoul, Korea. He is a recipient of best paper award in 2023. His main research interests include computer vision, IoT, machine learning, deep learning, and enhanced vision correction. He can be contacted at email: cdcd1115@gmail.com.