Comparison of Cloud Computing and TinyML Methods for Brain-Computer Interface in Motor Imagery Problems

Jonathan Daniel S. Ong
Department of Manufacturing Engineering and Management
De La Salle University
Manila, Philippines
jonathan_daniel_s_ong@dlsu.edu.ph

Robert Kerwin C. Billones
Department of Manufacturing Engineering and Management
De La Salle University
Manila, Philippines
robert.billones@dlsu.edu.ph

Ronnie Concepcion II
Department of Manufacturing Engineering and Management
De La Salle University
Manila, Philippines
ronnie.concepcion@dlsu.edu.ph

Nilo T. Bugtai
Department of Manufacturing Engineering and Management
De La Salle University
Manila, Philippines
nilo.bugtai@dlsu.edu.ph

2023 8th International Conference on Business and Industrial Research (ICBIR) | 979-8-3503-9964-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICBIR57571.2023.10147441

***Abstract*—In the modern day, several different innovations have been created with the sole purpose of improving existing models, products, and systems. One such innovation would be the improvement of brain-computer interfaces (BCI). BCIs tend to require high processing capabilities for the accurate gathering, processing, and classification of different biosignals such as the electroencephalography (EEG) signals. As such, cloud computing and TinyML methods are suggested as a means to solve this issue. In this study, the two methods are compared by simulating two cases, one for cloud computing and another for TinyML implementations. The latencies in which these two implementations can provide a classification for a data point is used as the main metric for comparison. It was observed that TinyML implementations had significantly lower latencies at an average of 0.0009 s, while cloud computing implementation had significantly higher latencies at an average of 1.5459 s. This difference can be attributed to the fact that cloud computing systems are reliant on connectivity between the local machine and the cloud computing services while TinyML implementations do not have the same restrictions and are instead capable of providing a classification as soon as it receives a data point.**

Keywords—brain-computer interface, cloud computing, edge computing, electroencephalography, TinyML

I. Introduction

The increasing awareness surrounding the capabilities and uses of brain-computer interfaces (BCIs) has brought about the emergence of several non-invasive BCI-based technologies. These technologies can be seen in a variety of fields and applications. Still, these technologies are mostly seen and noted in the clinical or biomedical field, particularly in patient care applications [1]. Other applications of BCI-based technologies would be in the field of assistive devices, where BCIs are utilized to control or command prosthetics [2] and mobility aiding devices, such as wheelchairs [3]. Due to the nature of these applications, signals are expected to be read, processed, and classified accurately [4] with little to no burden on the physical device. As such, computing methods such as cloud computing or edge computing were put forth as means to solve this problem.

Cloud computing has been regarded as a solution due to its ability to scale and provide services as demanded by the problems presented [5]. For BCI applications, cloud computing alleviates the burden that is normally involved in BCI-related physical computing systems, which are computationally heavy [6]. Aside from this, the usage of cloud computing applications or hosted services would require little to no resources from the users, which would make BCI technologies and applications more accessible to the general population.

Despite these upsides, cloud computing technologies are not as capable as physical computing systems with regard to realtime applications, such as BCI applications. This is caused by the fact that cloud computing has a reliance on the networking and connectivity capabilities of the accessing device. As such, having slow internet or networking capabilities would hinder and delay transactions between the local device and the cloud system [7]. As such, a means to aid in alleviating physical computing would be the utilization of Tiny AI.

Tiny AI focuses on the application of machine learning methods and minimizing its computational costs by embedding the machine learning models onto tiny or embedded devices [8]. For the case of BCI applications, this would alleviate the computational cost of constantly reading, processing, and classification of biosignals by providing an embedded classification system for faster processing. At the same time, these models tend to be optimized for lower storage requirements, lower computational power requirements, and low latencies.

This study developed an analytical comparison and contrast of the difference in classification capabilities between cloud computing applications and Tiny AI applications by simulating an Electroencephalography (EEG) Motor Imagery classification problem. It is important to note that a real-time database system from Google Firebase will serve as the primary means of processing data through cloud applications, while Tiny AI applications will focus on the training and saving of a signal classification model for use.

II. Review of Related Literature

A. Overview of Brain-Control Interfaces (BCIs)

Brain-computer interfaces allow for direct interaction and communication between an individual's brain and a specified output device. This is done by gathering

electroencephalography signals, signals generated from brain activity, processing them, and classifying these signals based on the user's intention [9]. Brain-computer interface technology and its corresponding applications are normally regarded as a means to aid in providing individuals with impaired movement a solution for their everyday activities [10]. There have been different implementations and approaches for BCI-based applications, but the use of Electroencephalography signals is normally preferred due to its capacity to be applied to a wider range of individuals affected by impaired movement [11]. BCIbased applications and systems with assistive devices in mind tend to require real-time signal processing and real-time classification to perform their intended function [4].

Due to the nature and characteristics of EEG signals, artifacts and interference within the collected signals must first be removed before the signal may be accurately processed and classified [12]. Currently, there are several methods that may be utilized to identify and classify EEG signals into their corresponding outputs. Machine learning methods, particularly the use of Convolutional Neural Networks and Deep Learning are popular methods, as seen in [13][14]. Other methods would include Support Vector Machines [15], Long short-term memory (LSTM) [16], Spectral regression discriminant analysis (SRDA) [17], and Naïve Bayesian Classifiers (NBC) [18]. It is also important to note that transfer learning is a highly regarded solution since EEG signals tend to express nonstationary characteristics and are subject to inter-subject variability, which transfer learning aims to solve [19].

B. Overview of Cloud Computing

Cloud computing technology is defined by its ability to provide on-demand, cloud-based computing services for the user, specifically in the efficient storage, computing, and networking of data [20]. Cloud computing is structured in a way such that resources are used efficiently and are only provided as needed, providing the necessary services as required by the user and his desired application [21]. Key characteristics for the uniqueness of cloud computing would be its ability for portability, cost efficiency, and flexibility, specifically for the user and his target outlets, since computations, storage, and accessibility can be performed from any location as long as the user has access to the cloud service provider [22]. Cloud computing is generally used in applications that deal with cloud storage, data security, web platform hosting, data warehousing, and data recovery, rather than real-time application of Artificial Intelligence or Machine Learning.

C. Overview of TinyAI

TinyAI (Tiny Artificial Intelligence), or TinyML (Tiny Machine Learning), is defined by its ability to provide machine learning models without the computational power requirements that modern machine learning models would entail for a system [8]. TinyAI models would try to optimize and compact these machine learning models such that they are able to run efficiently and can be embedded onto tiny, or embedded, devices and systems to achieve the goal of the application [23]. At the same time, due to its embedded nature, there is no longer a need for data to be transported from other sources, as seen in cloud computing, which makes it more reliable and stable when it comes to certain types of applications, such as real-time classification [24]. Aside from this, TinyAI model is portable, requires less resources, and incurs less burden, similar to its cloud computing counterpart, which makes it a great solution, especially in creating interfaces and systems that is accessible by a larger population while still being able to perform its task well. Currently, TinyML is utilized in several healthcare-based applications [25][26][27], applications that require accurate, prompt, and efficient results. These attributes are especially important for BCI applications since BCI applications tend to utilize, process, and classify complex biosignals, which are normally computationally heavy. In implementing TinyAi into the BCI paradigm, BCI applications become more accessible to the end users while still maintaining the advantages of cloud computing regarding lower computational costs, lower resource introduction, and portability.

III. Methodology

The study aims to compare and contrast the difference between the utilization of cloud computing technologies and the utilization of TinyML methodologies in the real-time classification of EEG Motor Imagery Signals. As such, two cases will be utilized to compare and contrast the two technologies.

The first case will determine the capabilities of cloud computing technologies by simulating the sending and receiving of data from a local machine to a "cloud machine" through a real-time cloud database. On the other hand, the second case will demonstrate the capabilities of TinyML methodologies by simulating the direct classification of a data sample from a stored classification model. The flowchart for both cases is seen in the figure below.

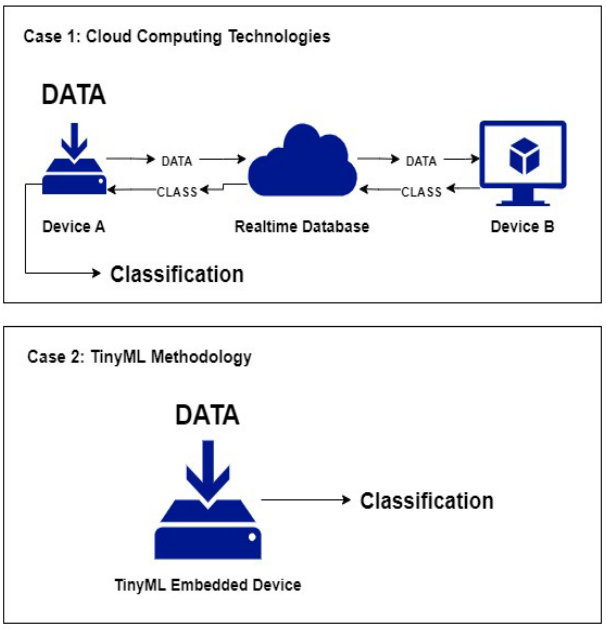

Fig. 1. Case 1: Cloud Computing Simulation Flowchart (Top) and Case 2: TinyML Methodology Simulation Flowchart (Bottom)

A. Dataset

As for the dataset to be used for the simulation of both cases, the BCI Competition IV-2a dataset [28] was chosen. The BCI Competition IV-2a dataset features EEG 4-class, left hand, right hand, foot, and tongue, motor imagery data from nine different subjects. The EEG signals are gathered from over twenty-two different electrode channels and three electrooculography channels. For the purposes of this study, only the EEG channels will be utilized while the EOG

channels will be deleted and taken out of consideration. At the same time, only two classes will be utilized, namely the left hand motor imagery data and the right hand motor imagery data. Last but not the least, only subject 1's data will be utilized for the purposes of this study.

With these considerations in mind, there are one hundred forty-four (144) data points for both the left and right motor imagery signals. A training-test split ratio of 80-20 was utilized, which means that there were 116 data points for the training data and 28 points for the testing data. It is important to note that prior to the segregation of the training and testing data, all of the data gathered were first treated and filtered, particularly using the Filter-Bank Common Spatial Pattern (FBCSP) method.

Once the data was treated and processed, the classification model was trained and modeled with an accuracy of 78.57%. It is important to note that for the case of this study, the accuracy of the model is not the main focus but rather serves as a means of providing a classification model for the simulations of both case 1 and case 2 (Fig. 1) of the comparison.

B. Metrics

Due to the nature of the study, the main metric that will be utilized would be the latency, or the delay in which the collected data is classified. This is done by instantiating a datetime object as 'startping' that takes the current timestamp at that point in time then comparing it to the datetime object instantiated as 'endping' at the end of the classification task. The difference between these two values are then taken into account as the measure of latency for the completion of the task.The formula for the latency metric is presented in (1) below.

$$latency = endping - startping \tag{1}$$

It is important to note that for the case of the study, the average latency of each case will be taken into account. The formula for the average latency metric is presented in (2) below.

$$ave.\ latency = \frac{\text{endping} - \text{startping}}{no.of\ trials} \tag{2}$$

```
# ============================================================================

# As an admin, the app has access to read and write all data, regradless of Security Rules
ref = db.reference('/unclassified')

# Converting datetime object to string
dateTimeObj = datetime.now()
global timestampStr
timestampStr = dateTimeObj.strftime("%d-%b-%Y (%H:%M:%S)")
print('Current Timestamp : ', timestampStr)

startping = dateTimeObj#.strftime("%M:%S.%f")
print(startping)

# ============================================================================
```

Fig. 2. (Case 1) Instantiation of 'startping' datetime object at the start of task.

```
    if classification == 1:
        print("Classification: Left")

    elif classification == 0:
        print("Classification: Right")

    dateTimeObj = datetime.now()
    endping = dateTimeObj#.strftime("%M:%S.%f")

    latency = endping-startping

    print ('latency:', latency)

    #print ('latency:', minutes,'minutes', 'and', seconds, 'seconds' )

    print("ALL DONE!")

ref.listen(listener)
```

Fig. 3. (Case 1) instantiation of 'endping' datetime object at the end of classification task.

```
print(sig1, sig2, sig3, sig4)

dateTimeObj = datetime.now()
startping = dateTimeObj#.strftime("%M:%S.%f")

sample = sample.reshape(1,-1)

test = loaded_model.predict(sample)

if test == 1:
    classification = test
    print("Classification: Left")

elif test == 0:
    classification = test
    print("Classification: Right")

dateTimeObj = datetime.now()
endping = dateTimeObj#.strftime("%M:%S.%f")

latency = endping-startping

print ('latency:', latency)
```

Fig. 4. (Case 2) Instantiation of 'startping' datetime object at the start of task and instantiation of 'endping' datetime object at the end of classification task.

C. Simulation (Case 1)

The simulations for this study are split into two cases, namely Case 1: Cloud Computing Technologies and Case 2: TinyML Methodology.

For Case 1, a Google Firebase Realtime Database was utilized as the cloud storage application that will be utilized to simulate the cloud-computing end of the flowchart. At the same time, two different programs were created, one of which will simulate Machine A which sends the signals to be classified while the other will simulate Machine B which classifies the received signals.

The first program, which simulates Machine A, randomly selects a data point from the training data used for the training of the classification model and sends it to the Google Firebase Realtime Database as a simulation of projecting data onto the cloud platform for the classification of said data.

Fig. 5. Selection and transmittal of random data point - Python Implementation (left) and appearance and reception of random data point onto real-time database (right).

Machine B then retrieves the data point from the Google Firebase Realtime Database and classifies the data utilizing the trained classification model. Once a classification is provided, Machine B then updates the data within the real-time database with the classification of the data point.

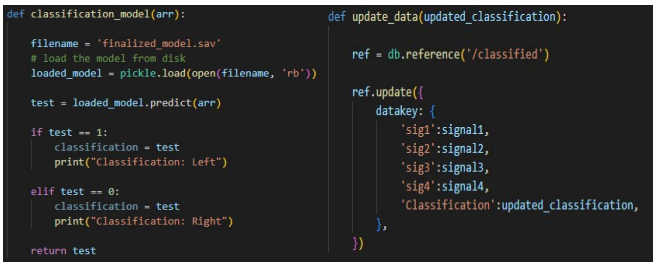

Fig. 6. Classification Function of Machine B (top) and Update_data Function of Machine B(bottom)

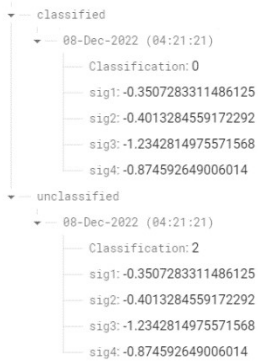

Fig. 7. Update of the real-time database with proper classification of the received data poin

Machine A is then able to retrieve the classification provided by Machine B over the real-time database and output said classification, thus ending the simulation for the classification of a single data point with the use of cloud computing technologies.

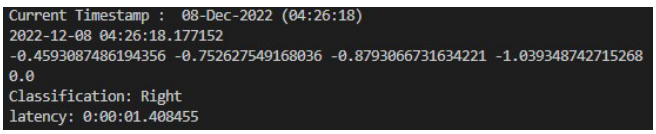

Fig. 8. Update of the real-time database with proper classification of the received data poin

D. Simulation (Case 2)

For Case 2, the trained classification model was stored onto the device as a separate file that can be loaded and called upon. Case 2 will utilize the same data points that are used in Case 1 to ensure that only the computing methods are being compared and contrasted. Case 2 will only utilize one program since it is supposed to simulate the embedding of a TinyML model onto a local machine. As such, the selection of the data point, the classification, and the output of the device will all be made on a local machine through a single program

Similar to Case 1, a data point will be randomly selected from the training data used for the training of the classification model. The classification model is to be loaded onto the workspace of the program and tasked to classify the data point that was randomly selected. Classification is then provided and outputted onto the terminal of the program.

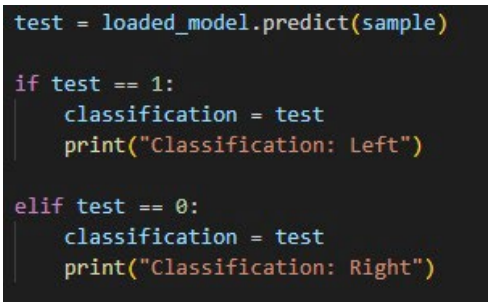

Fig. 9. Classification task for the embedded model and output commands.

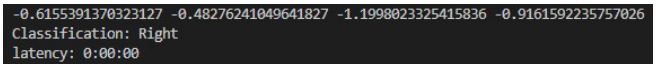

Fig. 10. Classification from embedded classification model with latency value from transactions.

IV. Results and Discussion

Following the implementation of the simulations, test cases were performed to compare and contrast the difference in latency between Case 1 and Case 2. For these test cases, thirty runs were made for EEG signals that would be classified as “Left”, while another thirty runs would be made for EEG signals that would be classified as “Right”. A summarized table of these test cases can be seen in (Fig. 11) and (Fig. 12) below.

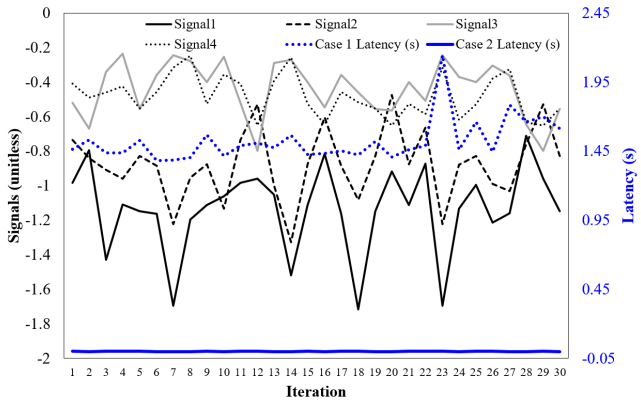

Fig. 11. Classification from embedded classification model with latency value from transactions.

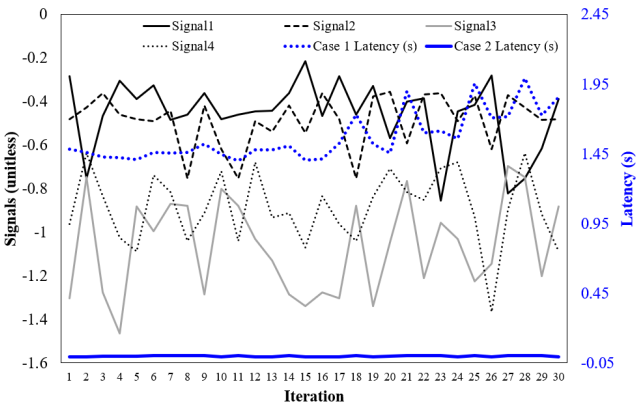

Fig. 12. Classification from embedded classification model with latency value from transactions.

From the figures shown, it can be seen that similar latencies can be observed for the TinyML-embedded device for both the "Left" and the "Right" classification. This means that there are no classification biases for either of these classifications, while at the same time, these classifications are done at a speed that is befitting of a real-time use case such as BCIs. The average latency for both cases is also computed for further comparison between the two cases. The values for the computed average latency of each case can be seen in Table I.

TABLE I. SUMMARIZED TABLE FOR MEAN AVERAGE LATENCIES FOR CASE 1 AND CASE 2 (DIFFERENT CLASSIFICATION)

Type of Average Classification	Case 1	Case 2
Ave. Left Classification	1.521775 s	0.000898 s
Ave. Right Classification	1.570046 s	0.000909 s
Ave. Classification (Total)	1.545910 s	0.000904 s

From the collected data, it can be seen that Case 1, which utilizes cloud computing technologies, has significantly higher values for its latency. This is due to the fact that cloud computing technologies are reliant on the connection that is to be made between the local machine and the cloud computing service. On the other hand, Case 2, which utilized an embedded machine learning model from the local machine, had lower values for its latency. This is because the embedded model is immediately able to classify the input signals without needing to relay the data to a different service or needing to receive the data from the cloud service.

These results are in line with the average latencies that were computed from the different classifications done for the comparison of both cases. The average latency for all the classifications done for Case 1 was seen to be 1.5459 s, while the average latency for all the classifications done for Case 2 was seen to be 0.000904 s. A comparison between this work and other works with similar applications can be found in the Table 2. Introducing in-depth computational models in the network might increase the speed of data transmission [29-33].

TABLE II. COMPARISON OF LATENCIES FOR DIFFERENT CLOUD AND EDGE-BASED MODELS FOR BCI APPLICATIONS

Methods	Latency (s)	Application	Reference
ncCCA	0.1755	SSVEP-based	[34]
wcCCA	0.0875		
CS-ncCCA	0.0910	EEG Controlled Wheelchair	
CS-wcCCA	0.0435		
Q-EEGNET	5.8200	Classification of EEG - Motor Imagery Signals	[35]
DL-DFCM (Cloud)	3.8098	Spike-Sorting System for BCI Applications	[36]
DL-DFCM (Edge)	75.5211		
FBCSP (Cloud)	1.5459	Classification of EEG - Motor Imagery Signals	This work
FBCSP (TinyML)	0.0009		

V. CONCLUSION

The results of the study were able to highlight the idea that there is a stark difference between cloud computing models and TinyML methods when it comes to real-time applications, such as the classification of EEG signals for BCI paradigms. For the case of this study, the average latencies were compared between the two methods since these two methods are the leading trends and models within the Artificial Intelligence Internet of Things (AI IoT) technologies. The average latency for cloud computing methods, specifically for the classification of EEG signals was seen to be at 1.545910 s, which is 1.545007 s slower than the average latency of TinyML methods which is seen to be at 0.000904 s. This significant difference in latency can be attributed to the fact that cloud computing methods rely on the transmission of data over to cloud computing platforms and the speed of the connection between the local machine and the cloud computing platforms. These problems, on the other hand, do not exist for TinyML methods, which utilize an embedded machine learning model. It is important to note that for this comparison, simulations were created for each case rather than having an actual comparison between different implementations of each methodology. With this in mind, TinyML embedded devices are much more reliable in real-time applications, such as Brain-controlled Wheelchairs (BCWs), which would require fast, accurate classification of EEG signals into motor output. On the other hand, Brain-controlled IoT technologies may look to utilize cloud computing technologies, which would require data on-demand, rather than real-time applications and classifications.

Future studies may consider comparing actual implementations for each methodology rather than utilizing simulations. At the same time, future studies should also investigate the comparison of different machine learning models and their effects on the latencies for each of the methodologies. Despite these, the results of the study can serve as a benchmark for future studies that would look into comparisons between cloud computing methods and TinyML methods. At the same time, these results may also serve as a reference for future studies that are in the related fields of cloud computing and TinyML methods, specifically for Brain-Computer Interface (BCI) applications.

REFERENCES

[1] C. Stevenson, Y. Chang, C. He, C.-R. Phang, C.-H. Su, R.-W. Lin, and L.-W. Ko, "Emerging non-invasive brain–computer interface technologies and their clinical applications," *Lecture Notes in Networks and Systems*, pp. 269–290, 2022.

[2] Ü. Hayta, D. C. Irimia, C. Guger, İ. Erkutlu, and İ. H. Güzelbey, "Optimizing motor imagery parameters for robotic arm control by Brain Computer interface," *Brain Sciences*, vol. 12, no. 7, p. 833, 2022.
[3] K. Liu, Y. Yu, Y. Liu, J. Tang, X. Liang, X. Chu, and Z. Zhou, "A novel brain-controlled wheelchair combined with computer vision and augmented reality," *BioMedical Engineering OnLine*, vol. 21, no. 1, 2022.
[4] X. Ding, X. Yue, R. Zheng, C. Bi, D. Li, G. Yao, "Classifying major depression patients and healthy controls using EEG, eye tracking and galvanic skin response data," *J. Affect. Disord.*, vol. 251, pp. 156–161, 2019. doi: 10.1016/j.jad.2019.03.058.
[5] J. Singh and G. Dhiman, "A survey on cloud computing approaches", *Materials Today: Proceedings*, 2021.
[6] N. Arizumi and T. Aksenova, "Fast continuous wavelet transform for brain computer interface using piecewise polynomials," *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2019.
[7] M. Shahbaz and R. Zahid, "Probing the factors influencing cloud computing adoption in healthcare organizations: A Three-way interaction model," *Technology in Society*, vol. 71, p. 102139, 2022.
[8] M. Z. Shamim, "TinyML model for classifying hazardous volatile organic compounds using low-power embedded edge sensors: Perfecting factory 5.0 using edge ai," *IEEE Sensors Letters*, vol. 6, no. 9, pp. 1–4, 2022.
[9] V. Jadhav, N. Tiwari, and M. Chawla, "A review on EEG data classification methods for Brain–computer interface," *International Conference on Innovative Computing and Communications*, pp. 747–760, 2022.
[10] S. P. Zavala, S. G. Yoo, and D. E. Tituana, "Controlling a wheelchair using a brain computer interface based on User Controlled Eye Blinks," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, 2021.
[11] K. O. Chicaiza and M. E. Benalcazar, "A brain-computer interface for controlling IOT devices using EEG signals," *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM)*, 2021.
[12] Z. Dokur and T. Olmez, "Classification of motor imagery electroencephalogram signals by using a divergence based convolutional neural network," *Applied Soft Computing*, vol. 113, part A, 2021.
[13] M. Rashid, N. Sulaiman, A. P. P. Abdul Majeed, R. M. Musa, A. F. Ab. Nasir, B. S. Bari, and S. Khatun, "Current status, challenges, and possible solutions of EEG-based brain-computer interface: A comprehensive review," *Frontiers in Neurorobotics*, vol. 14, 2020.
[14] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. H. Falk, and J. Faubert, "Deep learning-based Electroencephalography Analysis: A systematic review," *Journal of Neural Engineering*, vol. 16, no. 5, p. 051001, 2019.
[15] D. Rathee, H. Raza, G. Prasad, and H. Cecotti, "Current source density estimation enhances the performance of motor-imagery-related brain–computer interface," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 12, pp. 2461–2471, 2017.
[16] R. Zhang, Q. Zong, L. Dou, and X. Zhao, "A novel hybrid deep learning scheme for four-class motor imagery classification," *Journal of Neural Engineering*, vol. 16, no. 6, p. 066004, 2019.
[17] Q. Ai, A. Chen, K. Chen, Q. Liu, T. Zhou, S. Xin, and Z. Ji, "Feature extraction of four-class motor imagery EEG signals based on Functional Brain Network," *Journal of Neural Engineering*, vol. 16, no. 2, p. 026032, 2019.
[18] J. Wang, Z. Feng, X. Ren, N. Lu, J. Luo, and L. Sun, "Feature subset and time segment selection for the classification of EEG data-based motor imagery," *Biomedical Signal Processing and Control*, vol. 61, p. 102026, 2020.
[19] O. George, S. Dabas, A. Sikder, R. Smith, P. Madiraju, N. Yahyasoltani, and S. I. Ahamed, "Enhancing motor imagery decoding via transfer learning," *Smart Health*, vol. 26, pp. 100339, 2022.
[20] A. Khayer, Y. Bao, and B. Nguyen, "Understanding cloud computing success and its impact on firm performance: An integrated approach," *Industrial Management and Data Systems*, vol. 120, no. 5, pp. 963–985, 2020.
[21] C. Wu, Q. Peng, Y. Xia, Y. Jin, and Z. Hu, "Towards cost-effective and robust AI microservice deployment in edge computing environments," *Future Generation Computer Systems*, vol. 141, pp. 129-142, 2023.
[22] O. Rodríguez-Espíndola, S. Chowdhury, P. K. Dey, P. Albores, and A. Emrouznejad, "Analysis of the adoption of emergent technologies for risk management in the era of digital manufacturing," *Technological Forecasting and Social Change*, vol. 178, pp. 121562, 2022.
[23] J. I.-Z. Chen, P.-F. Huang, and C. S. Pi, "The implementation and performance evaluation for a smart robot with edge computing algorithms," *Industrial Robot*, 2022.
[24] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J*, vol. 5, no. 1, pp. 450–465, 2017.
[25] A.-T. Shumba, T. Montanaro, I. Sergi, L. Fachechi, M. De Vittorio, and L. Patrono, "Leveraging IOT-aware technologies and AI techniques for real-time critical healthcare applications," *Sensors*, vol. 22, no. 19, pp. 7675, 2022.
[26] F. Sakr, H. Younes, J. Doyle, F. Bellotti, A. De Gloria, and R. Berta, "A tiny CNN for Embedded Electronic Skin Systems," *Lecture Notes in Networks and Systems*, pp. 564–573, 2022.
[27] P. Ruiz-Barroso, F. M. Castro, R. Delgado-Escaño, J. Ramos-Cózar, and N. Guil, "High performance inference of gait recognition models on embedded systems," *Sustainable Computing: Informatics and Systems*, vol. 36, p. 100814, 2022.
[28] BCI competition IV. [Online]. Available: https://www.bbci.de/competition/iv/.
[29] J. Alejandrino et al., "Congestion Detection in Wireless Sensor Networks Based on Artificial Neural Network and Support Vector Machine," *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 2020.
[30] C. J. Pornillos et al., "Smart Irrigation Control System Using Wireless Sensor Network Via Internet-of-Things," *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 2020.
[31] R. A. Bedruz et al., "Analysis of Big Data Technologies for Policy Building in the Philippines," *2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 2019.
[32] R. Concepcion II et al., "The Technology Adoption and Governance of Artificial Intelligence in the Philippines," *2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 2019.
[33] J. Alejandrino et al., "A Hybrid Data Acquisition Model using Artificial Intelligence and IoT Messaging Protocol for Precision Farming," *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 2020.
[34] H. Rivera-Flor, D. Gurve, A. Floriano, D. Delisle-Rodriguez, R. Mello, and T. Bastos-Filho, "CCA-based compressive sensing for SSVEP-based brain-computer interfaces to command a robotic wheelchair," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–10, 2022.
[35] T. Schneider, X. Wang, M. Hersche, L. Cavigelli, and L. Benini, "Qeegnet: An energy-efficient 8-bit quantized parallel eegnet implementation for edge motor-imagery brain-machine interfaces," *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2020.
[36] S. Qian, H. Hong, Z. Zhu, K. Chen, N. Zheng, and Y. Qi, "A highefficiency spike sorting cloud-edge computing system with DL-DFCM," *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2019.