IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 May 2024, accepted 26 May 2024, date of publication 28 May 2024, date of current version 5 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3406631

RESEARCH ARTICLE

A TinyDL Model for Gesture-Based Air Handwriting Arabic Numbers and Simple Arabic Letters Recognition

ISMAIL LAMAAKAL[1], (Graduate Student Member, IEEE), IBRAHIM OUAHBI[1], KHALID EL MAKKAOUI[1], YASSINE MALEH[2], (Senior Member, IEEE), PAWEŁ PŁAWIAK[3,4], AND FAHAD ALBLEHAI[5]

[1]Multidisciplinary Faculty of Nador, University Mohammed Premier, Oujda 60000, Morocco
[2]Laboratory LaSTI, ENSAK, Sultan Moulay Slimane University, Khouribga 54000, Morocco
[3]Department of Computer Science, Faculty of Computer Science and Telecommunications, Cracow University of Technology, 31-155 Krakow, Poland
[4]Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, 44-100 Gliwice, Poland
[5]Computer Science Department, Community College, King Saud University, Riyadh 11437, Saudi Arabia

Corresponding author: Yassine Maleh (yassine.maleh@ieee.org)

This work was supported by King Saud University, Riyadh, Saudi Arabia, through the Researchers Supporting Project, under Grant RSPD2024R564.

ABSTRACT The application of tiny machine learning (TinyML) in human-computer interaction is revolutionizing gesture recognition technologies. However, there remains a significant gap in the literature regarding the effective recognition of complex scripts, such as Arabic, in real-time applications. This research aims to bridge this gap by leveraging TinyML for the accurate recognition of Arabic numbers and simple letters through gesture-based air handwriting. For the first time, we introduce a novel tiny deep learning (TinyDL) model that utilizes a lightweight convolutional neural network (CNN) architecture specifically designed to handle the intricacies of the Arabic script and adaptable for the TinyML domain. Despite the widespread use of CNNs in gesture recognition, our model stands out by achieving an exceptional accuracy rate of 97.5% in decoding 2D gesture inputs of Arabic numerals and letters. This high level of accuracy demonstrates the effectiveness of our TinyDL model in addressing the unique challenges posed by Arabic script recognition, thereby making it a user-friendly and accessible solution. Moreover, our research contributes to the advancement of TinyML applications in real-world gesture recognition apps, showcasing the potential of TinyML in transforming the interaction between humans and digital devices.

INDEX TERMS Arabic letters recognition, Arabic numerals, convolutional neural networks (CNN), deep learning (DL), gesture-based recognition, human-computer interaction, tiny machine learning (TinyML).

I. INTRODUCTION

The emergence of TinyML [1], [2], [3], [4], [55], [56], [58], [60] represents a significant breakthrough in technology, blending the ever-growing needs of data-driven applications with the constraints of low-power, embedded devices. As a specialized branch of machine learning, TinyML is designed to operate efficiently on hardware with limited computational resources, a crucial requirement in the era of ubiquitous computing. This innovation plays a pivotal role in human-computer interaction [6], [10], [13], which has evolved from traditional interfaces to more natural and intuitive methods, enabling smarter, context-aware interactions across a range of devices from wearables to IoT appliances.

Gesture recognition technology [15], [16], [41], [42] is at the forefront of this paradigm shift, allowing users to interact

The associate editor coordinating the review of this manuscript and approving it for publication was Larbi Boubchir.

with machines through simple movements [40]. Despite its promise, the technology faces challenges in accuracy, responsiveness, and adaptability to diverse environments. Advancements in this field, driven by sophisticated machine learning algorithms and sensor technologies, have enabled applications in gaming, entertainment, medical rehabilitation, and sign language interpretation [20], [63], [64]. In the context of air handwriting recognition [22], [28], Arabic script presents unique challenges due to its cursive nature, diacritical marks, and script style variations [62]. Despite its significance, there is a notable gap in digital representation and interaction for Arabic script [25], especially in gesture-based interfaces. Recognizing Arabic numbers and letters through gestures requires not only an understanding of the script's intricacies but also technology capable of capturing these nuances efficiently. The integration of TinyML with gesture recognition [32] offers a promising solution to these challenges. By leveraging TinyML, it is possible to develop systems that are accurate in recognizing Arabic script and efficient enough for low-resource environments [3]. However, this integration requires the development of specialized models that balance model complexity with computational efficiency. This research paper addresses these challenges by presenting a novel Tiny Deep Learning (TinyDL) model designed for the recognition of Arabic numbers and simple letters through air handwriting. The choice of Arabic script is motivated by its unique characteristics and the need for efficient recognition solutions in regions where Arabic is widely used [33]. The model employs a Convolutional Neural Network (CNN) [27], [61] architecture, optimized for TinyML environments, to accurately decode 2D gesture inputs. This approach not only tackles the challenges posed by Arabic script but also leverages the advantages of TinyML to create an effective and efficient solution.

The main contributions of this research are:

- A novel TinyDL model that achieves a 97.5% accuracy rate in recognizing Arabic hand gestures executed mid-air, setting a new benchmark for gesture recognition accuracy.
- Optimization of CNN architecture for low-power devices, enabling high accuracy in recognizing Arabic numerals and letters.
- Addressing the gap in digital representation and interaction capabilities for Arabic script, particularly in gesture-based interfaces.

The organization of the paper is as follows: Section II is dedicated to related work, providing a comprehensive review of existing literature and research pertinent to our study, highlighting advancements and identifying gaps in the field. Section III delves into the detailed methodology, encompassing the data collection and pre-processing steps crucial for preparing the dataset. This section also introduces our proposed model, offers an in-depth look at the model's architecture and features, and elaborates on the model conversion method, focusing on the optimization for deployment. In Section IV, we present the experimental results, analyzing the model's efficacy in recognizing Arabic numbers and letters through air handwriting. The section includes a comparative study where our model is juxtaposed with other algorithms, along with an assessment of the model conversion results, showcasing the performance improvements achieved. Section V discusses the feasibility of implementing the best model with the least cost, considering factors such as Computational Cost-Effectiveness, Financial Implications, Energy Consumption, and Real-World Implementation. The paper concludes with Section VI, encompassing the discussion and conclusion, where we reflect on our findings, discuss potential applications and limitations, and suggest future research directions, emphasizing the practical implications and contributions of our work.

II. RELATED WORK

The related research reviews the relevant research in gesture-based handwriting recognition and Arabic letters and numerals recognition. We identify gaps in the current research landscape by exploring the existing literature and contextualizing our proposed approach.

Yanay and Shmueli developed innovative approaches for air-writing recognition using motion signals from standard smart-bands [36]. They proposed two methods: one dependent on individual users, where accuracy is optimized through personal samples, and another independent of users, utilizing a convolutional neural network. Their research, conducted with 55 participants, demonstrated impressive accuracy, with the user-dependent method achieving 89.2% and the user-independent method reaching 83.2%, which further increased to 95.6% with the application of auto-correction. This work signifies a substantial advancement in wearable technology and gesture recognition, presenting practical applications in various interactive digital interfaces.

Ghanim et al. conducted a study that significantly advances the field of offline Arabic handwriting recognition by leveraging deep learning techniques [33]. They developed a multi-stage cascading system aimed at efficiently recognizing handwritten Arabic text, a challenging task due to the complexity of the language's script and the variability in writing styles. The system begins with hierarchical agglomerative clustering to organize the data into clusters, effectively forming a large search tree model. This setup simplifies the process of matching each test image with an appropriate cluster. Following this, a novel ranking algorithm is applied, starting with the computation of the pyramid histogram of oriented gradients and proceeding to measure divergence using the Kullback-Leibler method. Classification is then executed on the top-ranked matching classes. The research includes a comprehensive comparison of six different deep CNNs and their impact on recognition rates within this system. Their experiments, conducted using the IFN/ENIT Arabic database, revealed that the proposed clustering and ranking stages required only 11% of the entire database for classifying test images.

Abir et al. developed a DL model specifically for recognizing air-written text [37]. This model utilizes a 2D-CNN framework and integrates an advanced interpolation technique to improve accuracy. Their extensive research, conducted on various air-writing datasets available to the public, focused on finding the most efficient interpolation method for time-series data. In this study, they also conducted a detailed comparison of their new model against existing methods. The results showed that their technique consistently outperformed established methods, proving its effectiveness in both user-dependent and independent scenarios across all datasets tested. Remarkably, their approach attained high accuracy levels, varying between 94.17% and 99.63%, depending on the dataset and the training methodologies used.

Hsieh et al. developed a sophisticated approach for air-writing recognition, utilizing deep CNNs [39]. They created an innovative hand-tracking algorithm that accurately captures air-writing trajectories with a standard web camera. This method successfully addresses the limitations of previous techniques by allowing unrestricted air-writing, eliminating the need for specific delimiters or confined spaces. Additionally, the team introduced a new preprocessing method that effectively simplifies the data for efficient CNN training. Their experimental results demonstrated that this approach not only achieves high recognition accuracy but also significantly reduces network complexity compared to traditional image-based methods. This groundbreaking work holds significant potential for enhancing interactive systems and interfaces that require intuitive and natural input methods.

Nahar et al. conducted a comprehensive study on recognizing Arabic air-written letters using ML and deep CNNs combined with optical character recognition (OCR) [29]. They employed a hybrid approach incorporating feature extraction, DL models, and ML algorithms like neural networks, random forests, K-nearest neighbors, and support vector machines. The study utilized the AHAWP dataset, comprising diverse writing styles and hand sign variations, and applied preprocessing schemes to enhance data quality. The results indicated that their proposed model achieved a high level of accuracy, reaching up to 88.8% using neural networks with the VGG16 architecture, highlighting its effectiveness in Arabic letter recognition tasks.

Leem et al. explored mid-air gesture recognition for digit writing using radio sensors and a CNN [38]. Their method, diverging from conventional techniques, utilized hand trajectory information rather than raw data, making it robust against variations in orientation, distance, and hand shape. The approach involved signal preprocessing, hand motion localization, and transforming trajectory data into images for CNN classification. Their results demonstrated significant improvements in recognition accuracy and robustness, outperforming traditional methods and offering a user-friendly, accurate mid-air handwriting modality without restricting users.

Watanabe et al. presented a study on air-writing recognition using a 2D camera and a hybrid DL model [11]. Their system combined hand pose estimation and character recognition in a novel way, employing a webcam for data collection. They developed a hybrid DL model that integrates CNN and bidirectional long short-term memory networks. Their experiments demonstrated high recognition accuracy, achieving 99.3% for alphabetic characters and 99.5% for digits. The research indicates a significant advancement in air-writing recognition, offering an efficient and accessible method for gesture-based input.

Li et al. [43] proposed a hybrid deep learning model that combines spatial-temporal features for improved recognition accuracy. Liu et al. introduced a novel system utilizing deep learning and the Leap Motion Controller, showcasing efficient recognition of handwritten characters in mid-air [44]. Chen et al. developed a lightweight convolutional neural network optimized for real-time processing, addressing the computational constraints of low-power devices [45]. Wu et al. explored the use of 3D convolutional neural networks and data augmentation techniques to enhance recognition performance [46]. Zhang et al. employed wearable motion sensors and a deep learning framework to achieve accurate and real-time air-writing recognition [47]. Lastly, Khan et al. focused on improving recognition through attention-based sequence-to-sequence models, demonstrating the potential for enhanced accuracy in complex gesture recognition tasks [48]. In the realm of sign language recognition, Abdullahi et al. addressed the challenges of inconsistent depth features with their IDF-Sign model, showing promising results in dynamic sign word recognition [40]. Additionally, Abdullahi et al. proposed a spatial-temporal feature-based end-to-end Fourier network for 3D sign language recognition, further advancing the field [49].

In this paper, we propose an advanced TinyML model enriched with DL techniques for recognizing Arabic numerals and simple letters through gesture-based air handwriting. Our real-time wearable system, employing the Arduino Nano 33 BLE Sense as an edge device, allows users to write Arabic letters in three-dimensional space with a notable recognition accuracy of 97.5%, as demonstrated in Table 1. Beyond high accuracy, our TinyDL model distinguishes itself by offering low power consumption, making it a practical and energy-efficient solution for diverse applications.

III. METHODOLOGY

As we transition into the methodology phase of our study, it is crucial to outline the systematic approach we have adopted for the development and evaluation of our TinyDL model for gesture-based air handwriting recognition of Arabic numbers and simple Arabic letters. Figure 1 provides a comprehensive visual representation of our methodological framework, encapsulating the key stages of our research process. This section delves into the intricacies of our data collection techniques, the specific hardware utilized, and the rigorous

TABLE 1. Comparison of the proposed model with other existing.

Ref.	Contributions	Strong Points	Limitations	Accuracy	TinyML Model
[36]	Novel air-writing recognition using smart-bands.	High accuracy, innovative use of wearable technology.	Limited to specific gestures and letters.	User-dependent: 89.2%, User-independent: 83.2% (95.6% with auto-correction)	✗
[33]	DCNN-based Offline Arabic Handwriting Recognition.	Effective recognition of complex Arabic script.	Focus on offline data limits real-time application.	95.6%	✗
[37]	Enhanced air-writing recognition using deep learning and interpolation.	Improved accuracy with diverse gestures.	May require extensive data for training.	N/A	✗
[39]	Air-writing recognition with web camera and CNN.	High accuracy, simple hand tracking.	Focused on digits and specific symbols.	Alphabets: 99.3%, Numerals: 99.5%	✗
[29]	Arabic Air-Written Letters Recognition using ML, CNN, OCR.	Integrated approach using OCR for improved accuracy.	Complexity in combining different techniques.	91.24%	✗
[11]	2D Camera-Based Air-Writing Recognition using CNN-BiLSTM.	High accuracy with alphabets and numerals.	Requires 2D camera setup.	N/A	✗
[38]	Mid-Air Gesture Digit Writing with Radio Sensors & CNN.	High accuracy, innovative use of radio sensors.	Limited to digit recognition.	N/A	✗
Our Model	TinyDL model for air handwriting Arabic numbers and letters.	High accuracy, efficient for low-power devices.	Limited to simple Arabic Letters	97.5%	✔

data processing protocols we have implemented. Furthermore, we elaborate on the model development, including the architecture of the TinyDL model, training procedures, and validation methods. The subsequent paragraphs offer a detailed explanation of each component depicted in Figure 1, ensuring a clear understanding of the procedural steps and their significance in achieving the research objectives.

A. DATA DESCRIPTION

Our research aims to develop a TinyDL model for recognizing Arabic numerals and simple letters through air handwriting gestures. We utilized the Arduino Nano 33 BLE Sense microcontroller, equipped with accelerometer and gyroscope sensors, to collect data. The dataset includes gestures for 10 Arabic numerals (0-9) and 12 simple Arabic letters, performed by five volunteers aged 21, 33, 25, 24, and 28. Each gesture was sampled 500 times, resulting in a total of 11,000 samples. The data comprises accelerometer readings along the X-axis and Y-axis, capturing the hand movements in a 2D plane. Samples were annotated and stored in a structured JSON format for efficient training and evaluation of the TinyDL model.

B. EXPERIMENTAL CONFIGURATION

The experimental setup (see Table 2) was designed to ensure accurate data collection for gesture recognition. The Arduino Nano 33 BLE Sense was set to a sampling rate of 100 Hz to capture detailed hand movements. To ensure the integrity and consistency of our data, we adhered to a rigorous collection protocol as shown in Figure 2 and 3. This protocol included specific guidelines for hand movement, gesture speed, and recording conditions, among other factors.

In Figure 4, we present an illustrative example of inertial sensor signals captured during the execution of the '0' gesture. The graphical representation consists of two plots: one displays 3-axis acceleration signals, and the other showcases 3-axis angular velocity signals.

The initial graph exhibits a dynamic display of the 3-axis acceleration signals, with noteworthy fluctuations occurring between time intervals 200 and 250. These fluctuations suggest that active gestural movement took place during this specific interval. Conversely, the remaining intervals depicted on the graph indicate periods during which the device remained in a relatively static position.

A notable observation is the significant sensitivity of the acceleration signals to the device's motion when compared to the angular velocity signals. This heightened sensitivity is evident through the larger and more pronounced variations in acceleration. This distinction underscores the capacity of acceleration measurements to capture the device's dynamic movements with greater fidelity, thus providing valuable insights into the execution of gestures like '0'.

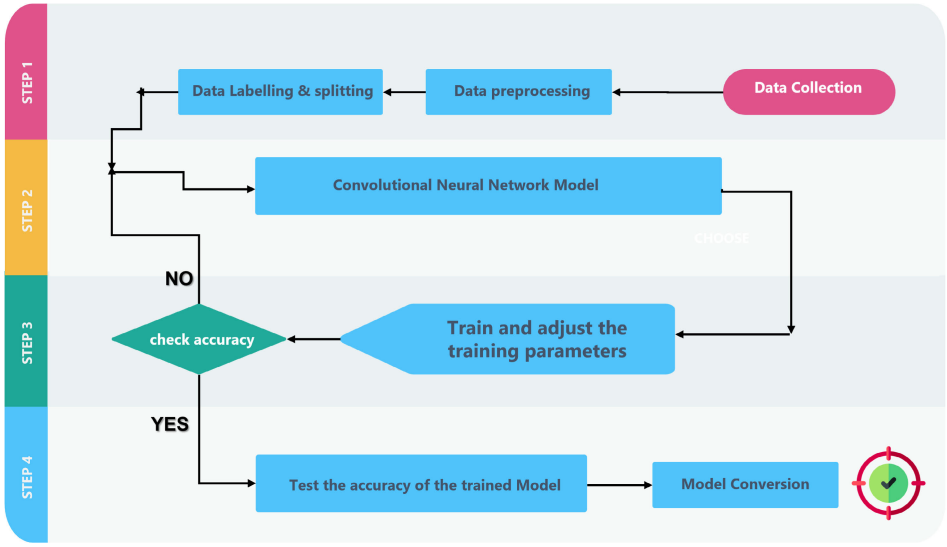

FIGURE 1. Our TinyDL model Workflow.

TABLE 2. Execution environment specifications.

Component	Specification
PC	Brand: Dell
	CPU: Intel(R) Core(TM) i5-8400H CPU @ 2.50GHz
	RAM: 8GB
	Storage: 500GB SSD
	OS: Windows 10
Development Tools	Python
	TensorFlow
	Keras
	Numpy
	Pandas
	Jupyter Notebook
	Visual Studio Code
Microcontroller	Arduino Nano 33 BLE
Sensors	Accelerometer
	Gyroscope
Processor	Cortex-M4 running at 64 MHz
Memory	Flash: 1 MB
	SRAM: 256 KB

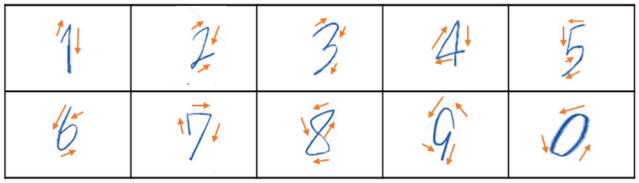

FIGURE 2. Pictorial Arabic numbers trajectories.

Figure 5 displays a sample from our dataset representing the gesture for the Arabic numeral '0', as encoded in a JSON file. This sample illustrates the raw accelerometer data

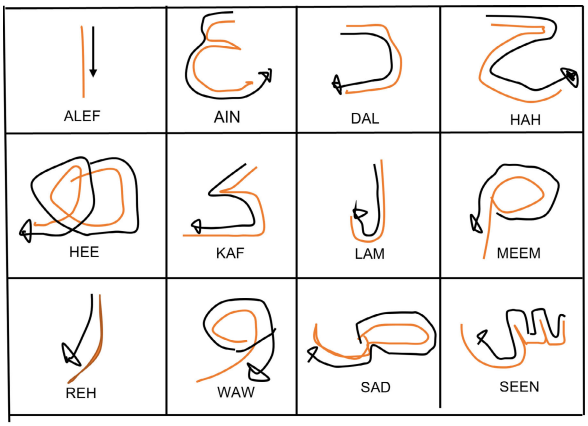

FIGURE 3. Pictorial Arabic letters trajectories.

along the X-axis and Y-axis, capturing the dynamic hand movements specific to the '0' gesture. The data excerpt in the figure shows a sequence of coordinated X and Y values that collectively map the trajectory of the gesture in a 2D plane.

C. DATA PRE-PROCESSING

During this phase, we adopted a thorough approach to transform continuous air handwriting gestures into a format conducive to deep learning recognition. The procedure, which involves rasterization onto a grid-based image (depicted in Figure 7 and 8), intricately captures the nuances of air gestures. Based on mathematical formulations, each step in the process is crucial. This methodology serves as the cornerstone for our TinyDL model, designed for the recognition of Arabic numerals and simple letters.

Purpose of Rasterization: Rasterization serves a critical role in transforming the dynamic, three-dimensional nature of air handwriting gestures into a two-dimensional, static

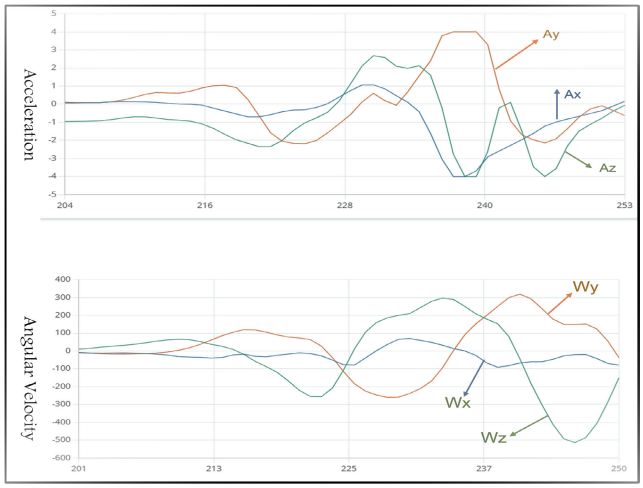

FIGURE 4. An example of raw sensor signals when the gesture '0' is written.

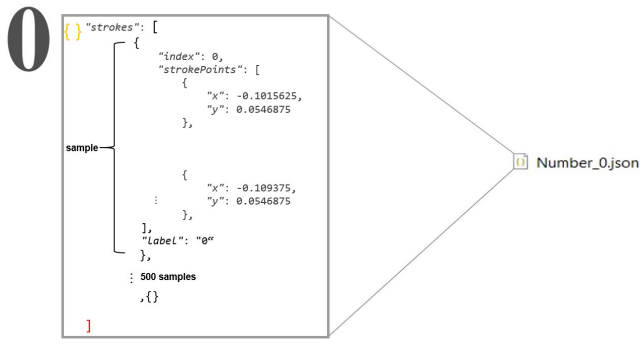

FIGURE 5. An example of data for gesture '0' in JSON format.

representation that can be effectively processed by our TinyDL model. It is the process of converting the continuous gesture trajectories into discrete pixel data that can be analyzed within the confines of a grid-based digital image. This step is vital for the subsequent stages of model training, as it translates the nuanced, temporal patterns of gesture motion into a spatial form compatible with conventional image processing techniques used within deep learning frameworks.

1) RASTERIZATION PROCESS

Figure 6 effectively illustrates the steps or workflow involved in the rasterization process. This diagram methodically depicts each stage, from the initial image initialization, where a digital canvas is created, to the final drawing of the gesture phase, where the gesture is rendered onto the canvas. In between, it covers coordinate mapping, normalization, and mapping to a pixel grid, crucial steps that ensure the accurate and efficient transformation of gesture data into a visual format suitable for analysis. This visual representation simplifies the understanding of the complex processes involved in rasterization, making it accessible and clear.

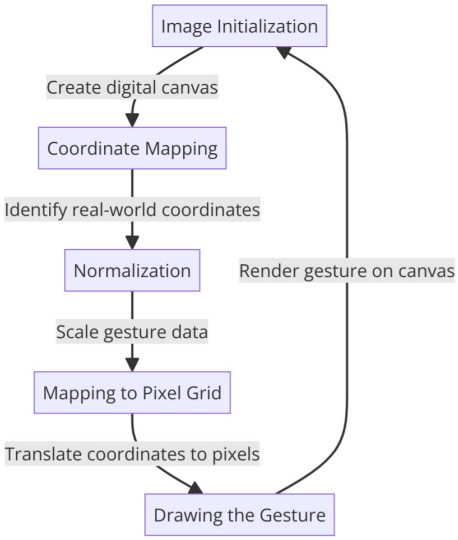

FIGURE 6. Illustrating the stages of the rasterization process.

a: IMAGE INITIALIZATION

In the initial rasterization phase, an image (I) is generated as the canvas for capturing gestures, with dimensions $M \times N$ (see equation 1). All pixel values in this image are initially set to zero, establishing a crucial blank canvas for rendering the gestures.

$$I(i, j) = 0, \text{for } i = \{1, 2, \ldots, M\} \text{ and } j = \{1, 2, \ldots, N\} \quad (1)$$

b: COORDINATE MAPPING

In this phase, attention is centered on a specific point $P(x, y)$ within the two-dimensional plane, with x and y representing the coordinates of that point. This point corresponds to a specific real-world location where a part of the gesture occurs. Accurately capturing these coordinates is essential for the accurate digital representation of the continuous gesture, which is expressed as $P(x, y)$.

c: NORMALIZATION

The process of scaling the coordinates of a point $P(x, y)$ to fit within a certain range, as referenced in [34], ensures that the gesture data is properly aligned within the limited space of the image grid. To achieve this, the following formulas are used for normalization:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}, y_{norm} = \frac{y - y_{min}}{y_{max} - y_{min}} \quad (2)$$

Where (x_{min}, y_{min}) and (x_{max}, y_{max}) represent the minimum and maximum values of the coordinates. Normalization transforms the original coordinate values into a range between 0 and 1.

d: MAPPING TO THE PIXEL GRID

After the normalization step, the subsequent phase involves translating the scaled coordinates onto the pixel grid of the image. This step identifies the specific pixel in the image that

aligns with the normalized coordinates of the gesture point. To accomplish this, the following formulas (3 and 4) are used for mapping:

$$Row\_Index = \lfloor x_{norm} \cdot M \rfloor + 1 \quad (3)$$

$$Column\_Index = \lfloor y_{norm} \cdot N \rfloor + 1 \quad (4)$$

Here, $\lfloor \cdot \rfloor$ denotes the floor function. These equations calculate the row and column indices of the pixel in the image grid where the gesture point should be represented.

e: DRAWING THE GESTURE

In the concluding phase, the image's pixel values are altered in line with the gesture data. This modification can be carried out through different methods, such as setting pixel values based on the timing of the gesture or the sequence of points. Another approach is to assign binary values (0 or 1) to the pixels, indicating whether the gesture is present or absent at a particular location, see equation 5. This method effectively represents the gesture on the image grid.

$$I(Row\_Index, Column\_Index) = Pixel\_Value \quad (5)$$

f: COLORING PIXELS–BRESENHAM'S ALGORITHM

The Bresenham algorithm [50], [51] is an efficient method to determine which pixels should be drawn to form a close approximation of a straight line between two points. In the context of gesture representation, this algorithm can be extended to color pixels based on the gesture's velocity or sequence. When a gesture is made, the line is rasterized and pixels are colored in a way that represents the movement, with colors often encoding additional information, such as speed or pressure. In our rasterized images, colors are assigned as follows [52]:

- Red (RGB: 255,0,0): Represents the starting point of the gesture.
- Green (RGB: 0,255,0): Denotes the trajectory of the gesture.
- Blue (RGB: 0,0,255): Indicates the end of the gesture.

The Bresenham algorithm has been adapted to include RGB coloring by modifying the pixel value assignment during the rasterization process. The adapted algorithm can be represented as:

$$I(Row\_Index, Column\_Index) = \begin{cases} (255, 0, 0) & \text{for starting point} \\ (0, 255, 0) & \text{for trajectory} \\ (0, 0, 255) & \text{for end point} \end{cases} \quad (6)$$

Figure 9 and 10 showcase the results yielded by the rasterization method.

D. THE PROPOSED TINYDL MODEL

We employed a Convolutional Neural Network (CNN) [12], [35] to recognize Arabic letters and numerals in air handwriting due to its superior image handling capabilities. The CNN architecture, inspired by the human visual cortex, efficiently extracts and learns spatial hierarchies, making it adept at interpreting complex patterns and stroke variations in Arabic handwriting. These networks excel in automatic feature detection, such as edges and curves, essential for distinguishing different Arabic characters [14]. This capability minimizes the need for manual feature engineering, thus simplifying model development. Additionally, CNNs handle variations in position and orientation effectively [39], a crucial trait for managing the inherent variability in air handwriting. These features make CNNs ideal for addressing the specific challenges of air handwriting recognition, justifying their use as the core of our proposed model.

As shown in Figure 11, our CNN model begins with a convolutional layer for recognizing simple Arabic letters through air handwriting. This layer uses a 3×3 kernel (F) and a stride (S) of 2, optimizing fundamental feature extraction from the input images. The output dimensions, output width

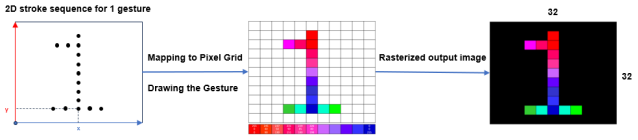

FIGURE 7. Rasterization: methodical steps for gesture "1".

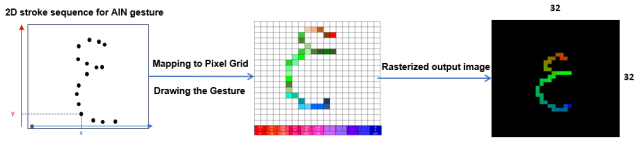

FIGURE 8. Rasterization: methodical steps for gesture "AIN".

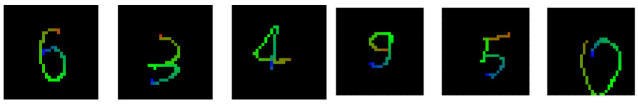

FIGURE 9. Examples of rasterized output images corresponding to various stroke gestures for Arabic numerals.

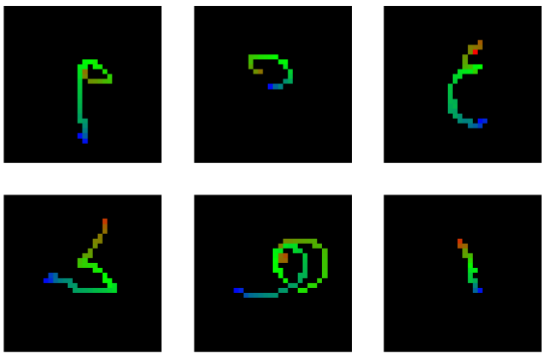

FIGURE 10. Examples of rasterized output images correspond to various stroke gestures for Arabic letters.

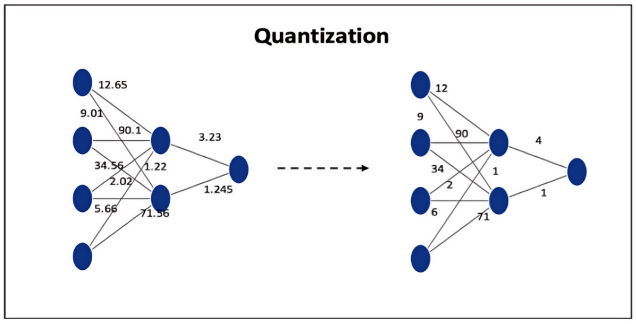

FIGURE 11. Quantization method from floating-point numbers to int-8 integers.

(OW) and height (OH) [5] are derived using the formulas (7):

$$OW = \frac{W-F+2P}{S} + 1, \quad OH = \frac{H-F+2P}{S} + 1 \tag{7}$$

Where W and H represent the width and height of the input layer, and P denotes the padding. Following this, batch normalization [9] is applied to standardize the output using the batch mean (μ) and batch variance (σ^2), as per the following equation:

$$x_{normalized} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{8}$$

The model then employs the Rectified Linear Unit (ReLU) activation [17] function to introduce non-linearity into the system. A dropout layer [19], [54] with a rate (d) of 0.3 follows each convolutional layer to prevent overfitting. During training, each input value (x) is multiplied by a binary dropout mask (0 or 1), as shown in equation 9:

$$y = x \cdot d \tag{9}$$

This convolutional sequence is repeated for two additional layers with increasing filter depths (16, 32, and 64), while retaining the same kernel size and stride. A global average pooling layer [23] is applied to reduce the spatial dimensions of each feature map. The global average pooling for each feature map, G_k, is calculated as:

$$G_k = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} F_k(i,j) \tag{10}$$

For $k = 1, 2, \ldots, K$, where $F_k(i,j)$ is the value at the (i,j)-th position in the k-th feature map. This pooling layer simplifies the transition to the final classification layer by decreasing the feature map dimensions. The output layer employs the softmax activation function to predict the class of the Arabic numeral. The number of units used corresponds to the 22 different Arabic numerals and simple letters that need to be recognized.

Table 3 presents the architecture of the CNN designed for our image classification task. It details the sequence and parameters of each layer, optimized for detecting and classifying complex patterns within the dataset. The table specifies each layer's type and parameters, crucial for the network's performance.

TABLE 3. CNN model layers and parameters.

Layers	Parameters
Rescaling	Scale = 1.0 / 255
Conv2D	16 filters, Kernel = (3, 3), Stride = 2, Padding = same
BatchNormalization	-
Activation	ReLU
Dropout	Rate = 0.3
Conv2D	32 filters, Kernel = (3, 3), Stride = 2, Padding = same
BatchNormalization	-
Activation	ReLU
Dropout	Rate = 0.3
Conv2D	64 filters, Kernel = (3, 3), Stride = 2, Padding = same
BatchNormalization	-
Activation	ReLU
Dropout	Rate = 0.3
GlobalAveragePooling2D	-
Dropout	Rate = 0.3
Dense (Output Layer)	Units = 22, Activation = softmax

E. MODEL CONVERSION

In the field of TinyML, model conversion is a critical step in adapting complex machine learning models for deployment on resource-constrained devices, such as microcontrollers and other low-power embedded systems. The conversion process typically involves quantization [7], [26], [53], pruning [8], [24], and sometimes the translation of a model into a specialized format that is compatible with TinyML frameworks.

1) GENERATE A TENSORFLOW LITE MODEL

Converting a pre-trained TensorFlow model to TensorFlow Lite (TFLite) [41] involves several key steps, essential for deploying ML models on resource-constrained edge devices. This conversion process is critical in the TinyML context, ensuring models remain effective under low-power and limited-memory conditions. The main stages include Range Estimation, Scaling Factor Calculation, Quantization, Dequantization (Inference), and Quantization Error handling.

Initially, we determine the range of possible values for model weights and activations in the original floating-point model, denoted as (min, max) pairs. These values are

computed during model training or derived from the analysis of representative data.

Subsequently, a scaling factor, known as the quantization scale, is determined to map the floating-point values to integer values within a specified range. The computation of the scaling factor is based on the desired quantization bit-width (e.g., 8 bits), and the formula for its calculation is:

$$Scaling\_Factor(scale) = \frac{max\_range - min\_range}{2^{bit\_width} - 1} \quad (11)$$

After obtaining the scaling factor, the next step involves the quantization of values. In this stage, we map the floating-point values to the nearest integer values within the predetermined range, effectively quantizing them. The quantization formula is as follows:

$$Quantized\_Val(q\_val) = round\left(\frac{original\_val - min\_range}{scale}\right) \quad (12)$$

During inference with the quantized model, we need to dequantize the integer values to approximate the original floating-point values. The dequantization formula is as follows:

$$Dequantized\_Val(d\_val) = (q\_val \times scale) + min\_range \quad (13)$$

Recognizing the presence of quantization error resulting from the finite bit-width representation is crucial. The maximum quantization error, denoted as Max_Error, can be computed as follows:

$$Max\_Error = 0.5 \times scale \quad (14)$$

2) GENERATE A TENSORFLOW LITE FOR MICROCONTROLLERS MODEL

Following the initial conversion to TFLite, the model is further transformed for compatibility with the TensorFlow Lite Micro (TFLM) runtime [42], which is tailored for microcontrollers and devices with minimal memory. The TFLM conversion involves transforming the TFLite model into a C source file, encapsulating the model as a byte array. This conversion can be achieved using the '*xxd*' command-line tool or a TensorFlow-provided Python script.

IV. EXPERIMENTAL ANALYSIS AND EVALUATION

The objective of the project is to construct a TinyDL model for recognizing Arabic numbers and simple letters through gesture-based air handwriting.

A. EXPERIMENT

In this section, we delve into the experimental setup of our study (see Figure 12). Utilizing the Arduino Nano 33 BLE as our microcontroller, equipped with onboard IMU sensors including an accelerometer and a gyroscope, we aim to capture and decode the subtle nuances of air handwriting for Arabic numerals and simple letters. We connect the

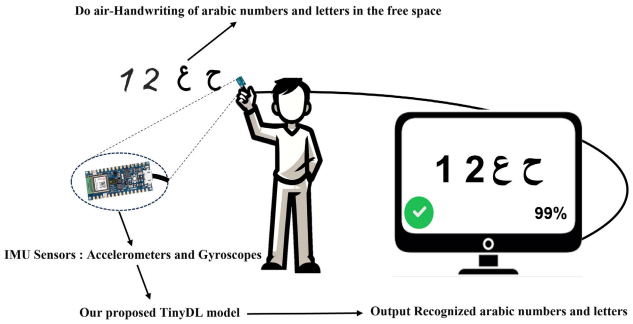

FIGURE 12. An Overview of the Research Experimental Setup: IMU-Based Gesture Recognition.

microcontroller via Bluetooth, configuring it to capture hand gestures at a sampling rate of 100 Hz. Additionally, it can connect with our laptop via USB cable for data transfer and debugging purposes. This setup records the motion data along the X-axis and Y-axis of the accelerometer as the subject enacts a specific gesture, such as writing the numeral "1". During the data preprocessing phase, we employ a rasterization method to transform these coordinate points into a series of rasterized images. These images serve as the training data for our streamlined deep learning model, specifically designed to adhere to the constraints of TinyML. Subsequently, we evaluate six distinct classification algorithms to determine the most effective in terms of accuracy and compliance with TinyML parameters. Ultimately, the chosen algorithm not only accurately discerns the intended gesture but also projects the result onto our laptop display, indicating the classified character and its associated confidence level.

In the training phase, we configure the model to undergo 60 epochs, implementing a Model Checkpoint Callback to save weights at regular intervals. The model is compiled using the Adam optimizer [65] with a learning rate of 0.001 to minimize the binary cross-entropy loss [66]. Accuracy serves as the primary metric, computed by dividing correct predictions by total predictions. The training loop (model.fit) iteratively refines parameters to enhance gesture recognition accuracy across epochs. The overarching objective is to minimize the loss function and fine-tune the model parameters for accurate gesture recognition. Subsequently, we reserved 80% for training, 10% for testing, and 10% of the data for prediction to assess the performance of the trained model.

B. RESULTS

In this section, we present the performance evaluation of our proposed TinyDL model for gesture-based recognition of Arabic numbers and simple Arabic letters. Following the training of our TinyDL model, we conducted an extensive evaluation, revealing promising results. Our model demonstrated exceptional performance, achieving an accuracy of 97.5% during the training phase and an impressive 99.7% during the validation phase, as illustrated in Figure 13. These high accuracies signify the robustness and efficacy of our

TABLE 4. Influence of dropout rate on model performance.

Dropout Rate	Training Accuracy	Validation Accuracy	Testing Accuracy	Recall	Precision	F1 Score
0.2	93.8%	95.1%	90%	90.3%	88.9%	87%
0.3	97.5%	99.7%	98.7%	92.9%	93.3%	90%
0.4	92%	90.2%	88%	84%	91%	85.4%
0.5	85.7%	89.8%	88%	80.1%	79.7%	77.1%
0.6	79.4%	85%	75.1%	68.1%	81%	66.7%

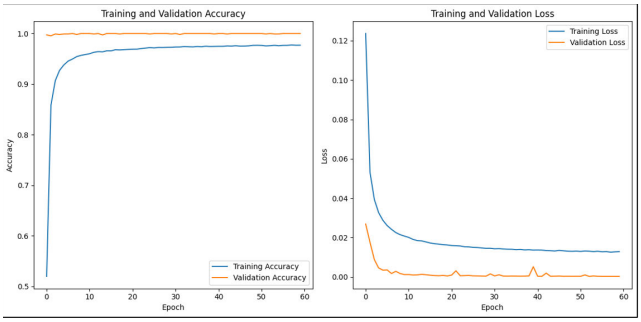

FIGURE 13. Model Training Progress: Accuracy and Loss Curves.

model in accurately recognizing both Arabic numbers and simple Arabic letters based on hand gestures.

After examining how dropout rates affected the functionality of our model, we noticed significant differences in accuracy metrics between various dropout settings, which are outlined in Table 4. The percentage of neurons that are arbitrarily removed during training in order to avoid overfitting is known as the dropout rate. The best training accuracy of 97.5% was obtained with a dropout rate of 0.3, as shown, and a validation accuracy of 99.7% was obtained just after. These findings suggest that the model is more efficiently regularized at a moderate dropout rate, which improves the model's capacity for generalization. On the other hand, lower training and validation accuracies were caused by greater dropout rates (0.4, 0.5, and 0.6), which may indicate an overly small decline in model capacity. The decrease in F1 score, recall, and precision measurements with rising dropout rates is especially notable.

Furthermore, we utilized confusion matrices to delve into the intricacies of our model's performance in recognizing Arabic numerals and simple letters. These matrices provide a visual representation of the model's classification accuracy, detailing the distribution of correct and incorrect predictions across various classes. By scrutinizing the confusion matrices, we gain valuable insights into the model's ability to differentiate between similar gestures corresponding to distinct Arabic numerals and letters. This detailed analysis facilitates the identification of common misclassification patterns and aids in refining the model's architecture and training strategies, thereby improving its accuracy and reliability in practical scenarios. Figure 14 illustrates the confusion matrix for Arabic numerals, while Figure 15 displays the confusion matrix for simple Arabic letters.

To assess the efficacy of our chosen model, we executed a comprehensive comparative analysis by implementing and rigorously testing five renowned algorithms in image classification: ResNet, LSTM, SVM, MobileNets, and VGGNet.

1) RESNET CONFIGURATION

ResNet, specifically ResNet-50, was chosen for its ability to effectively handle deep networks while mitigating the vanishing gradient problem. In our configuration, ResNet-50 consists of 50 layers, including residual blocks that enable the network to learn intricate features. Data augmentation techniques such as rotations, shifts, zooms, and flips were applied to increase the diversity of the training data and improve model generalization. We utilized the Adam optimizer with a learning rate scheduler to dynamically adjust the learning rate during training, ensuring stable convergence and faster convergence to an optimal solution.

2) LSTM CONFIGURATION

Long Short-Term Memory (LSTM) networks were selected to capture sequential dependencies inherent in the handwriting data, such as stroke order and writing sequences. Our LSTM configuration comprises two layers with 128 units each. This architecture strikes a balance between model complexity and computational efficiency. To prevent overfitting, we applied dropout regularization with a rate of 0.3 between LSTM layers. Additionally, sequences were pre-padded to a fixed length to ensure uniform input dimensions, facilitating efficient training and inference.

3) SVM SETUP

Support Vector Machines (SVM) with a Radial Basis Function (RBF) kernel were employed for their effectiveness in high-dimensional feature spaces. In our SVM setup, we tuned the penalty parameter (C) to 100 and the kernel coefficient (gamma) to 0.01 through grid search. These parameter values were selected to strike a balance between model complexity

TABLE 5. The impact of data splitting on model performance.

Split Training (%)	Training Accuracy	Validation Accuracy	Testing Accuracy	Recall	Precision	F1 Score
50	85.2%	84.9%	85.1%	82.3%	80.9%	79%
60	87.7%	86.3%	85.7%	83.9%	83.3%	82%
70	93.9%	92%	93.1%	87.4%	86.7%	86.3%
80	97.5%	99.7%	98.7%	92.9%	93.3%	90%
90	98.6%	92.1%	79.8%	67.3%	81%	69.2%

Original gesture \ Detected gesture	0	1	2	3	4	5	6	7	8	9
0	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%
1	0%	96%	4%	0%	0%	0%	0%	0%	0%	0%
2	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%
3	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%
4	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%
5	0%	0%	0%	0%	0%	98%	2%	0%	0%	0%
6	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%
7	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%
8	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%
9	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%

FIGURE 14. Confusion Matrice for Arabic Numerals Recognition.

Original Gesture \ Detected Gesture	AIN	ALEF	DAL	HAH	HEE	KAF	LAM	MEEM	REH	SAD	SEEN	waw
AIN	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
ALEF	0%	94%	0%	0%	0%	0%	6%	0%	0%	0%	0%	0%
DAL	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%
HAH	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
HEE	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%
KAF	4%	0%	0%	0%	0%	96%	0%	0%	0%	0%	0%	0%
LAM	0%	6%	0%	0%	0%	0%	94%	0%	0%	0%	0%	0%
MEEM	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
REH	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%
SAD	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%
SEEN	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%
waw	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%

FIGURE 15. Confusion Matrice for simple Arabic Letters Recognition.

and generalization performance. Additionally, feature scaling was performed using min-max scaling to map feature values to the range [0, 1], ensuring that all features contribute equally to the model's decision boundary.

TABLE 6. Comparison of accuracy rates between the chosen algorithm and the other algorithms.

Algorithms	Accuracy Rate
Chosen model	97.5%
ResNet	90.1%
LSTM	93.9%
Support Vector Machines (SVM)	87.2%
MobileNets	89.4%
VGGNet	92.7%

4) MOBILENETS IMPLEMENTATION

MobileNets were chosen for their lightweight and efficient architecture, making them suitable for deployment on devices with limited computational resources. In our implementation, we utilized a width multiplier of 1.0x and standard input resolution to balance model efficiency and predictive performance. During training, we employed exponential decay for the learning rate to gradually reduce the learning rate over epochs, allowing the model to converge smoothly while avoiding overshooting the optimal solution. Additionally, the model was fine-tuned from pre-trained weights to leverage knowledge transfer from a related task and accelerate training convergence.

5) VGGNET TUNING

VGGNet, specifically VGG-16, was selected for its simplicity and effectiveness in feature extraction from images. In our VGGNet configuration, we utilized the VGG-16 architecture, which consists of 16 convolutional and fully connected layers. Batch normalization was applied after each convolutional layer to stabilize training and accelerate convergence. We initialized the learning rate to 0.01 and monitored validation loss performance during training. The learning rate was reduced based on validation loss performance to ensure steady convergence and prevent overfitting.

The results of this analysis are presented in Table 6, which showcases the accuracy rates achieved by each algorithm.

Moving to the model conversion phase, we achieved significant reductions in model size while maintaining robust performance, the original TensorFlow model was compressed from 658857 bytes to 103632 bytes in its TensorFlow Lite version, representing a reduction of approximately 84%. Further optimization was achieved with the TensorFlow Lite Quantized version, which stands at 32120 bytes, marking an additional 69% decrease from the TensorFlow Lite model and 95% from the TensorFlow model (see Table 7). These reductions in model size demonstrate the effectiveness of our conversion and optimization techniques, which are pivotal for deployment in resource-constrained environments such as mobile devices, without compromising the model's integrity and performance.

Table 5 shows the impact of data splitting percentage on model performance.

TABLE 7. Comparison of model sizes and their respective reductions in size after conversion.

Model	Size (bytes)	Size Reduction (%)
TensorFlow	658857	-
TensorFlow Lite	103632	84.27
TensorFlow Lite Quantized	32120	95.12

TABLE 8. Testing the accuracy of models post-conversion.

Model	Testing Accuracy
TensorFlow	98.7%
TensorFlow Lite	98.7%
TensorFlow Lite with Quantization	98.7%

The model conversion phase has been pivotal in preserving the efficacy of our models when subjected to a testing environment. Notably, even with a marked reduction in model size, our comprehensive testing phase has revealed that the model's performance remains stellar and unaffected. Table 8 below corroborates this finding, showing that the TensorFlow model and its derivatives TensorFlow Lite and TensorFlow Lite with Quantization uniformly maintain an impressive testing accuracy rate of 98.7%.

Figures 16 and 17 effectively demonstrate the outcomes of rigorously testing our TinyDL model, followed by its successful deployment on the Arduino Nano 33 BLE.

Figure 18 illustrates the impact of the number of layers on both the accuracy and the size of the model. The graph provides insights into how increasing the number of layers affects the model's performance and complexity, highlighting the trade-offs between accuracy and model size.

C. DISCUSSION

In the discussion of our TinyDL model's performance, we begin by acknowledging its exceptional accuracy rates, as demonstrated in Table 4 and Figure 13. The model not only showed a high accuracy of 97.5% during training but also maintained this during validation at 99.7%, indicating its robustness in gesture recognition for Arabic numerals and letters. This high level of precision underlines the model's effective learning and generalization capabilities. The detailed analysis of dropout rates, as presented in Table 4, reveals that a moderate dropout rate of 0.3 is optimal for our model, striking a balance between overfitting prevention and maintaining adequate model capacity. This is crucial for enhancing the model's adaptability and performance consistency.

Furthermore, our exploration of the model's classification abilities through confusion matrices, shown in Figures 14 and 15, provides deeper insights into its strengths and areas for potential improvement. These matrices highlight the model's proficiency in differentiating similar gestures

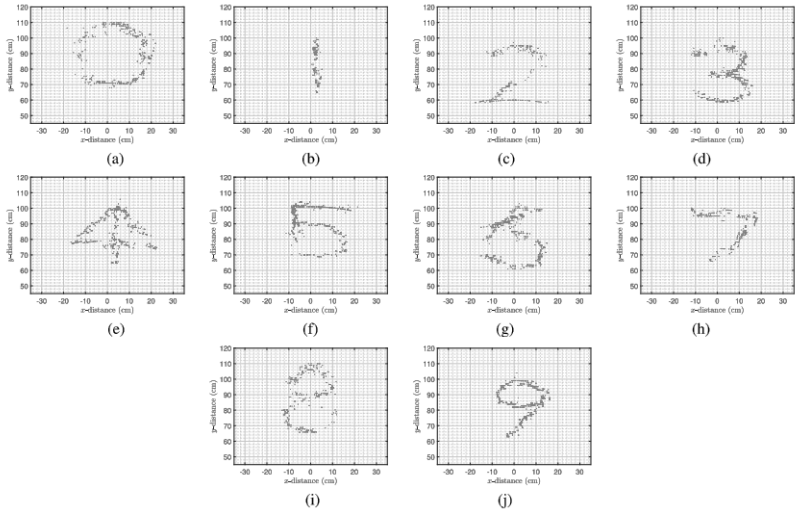

FIGURE 16. Testing the Model's Efficacy in Recognizing Arabic Numerals.

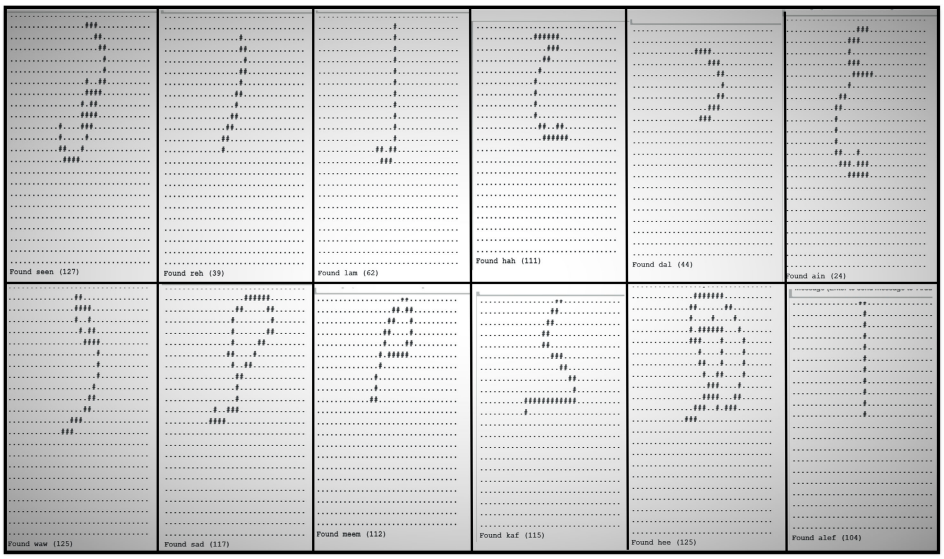

FIGURE 17. Testing the Model's Efficacy in Recognizing Simple Arabic Letters.

and pinpoint common misclassification patterns, which are valuable for refining the model's architecture and training strategies.

In our comprehensive study, we evaluated several leading algorithms renowned for their capabilities in image classification. Through meticulous analysis and extensive trials, detailed in Table 6, our TinyDL model emerged as the premier choice, achieving an exceptional accuracy rate of 97.5%. This impressive performance surpasses that of established algorithms such as ResNet (90.1%), LSTM (93.9%), Support Vector Machines (SVM) (87.2%), MobileNets (89.4%), and VGGNet (92.7%). The selection of the CNN algorithm was underpinned by its superior accuracy, which not only affirms its adeptness in managing the intricacies of complex classification tasks but also establishes it as the standard-bearer for future applications in the domain.

The analysis (see Figure 18) suggests that while more layers can improve accuracy to a certain extent, the benefit diminishes after a point, and the cost in terms of computational resources becomes significantly higher. Therefore, it is essential to find a balance between model size and accuracy, optimizing for both performance and efficiency. In this scenario, a model with 3 layers seems to offer the best trade-off, achieving high accuracy with a relatively small size. Beyond this point, the size of the model increases substantially without a corresponding increase in accuracy, which might not be practical for deployment, especially in resource-constrained environments.

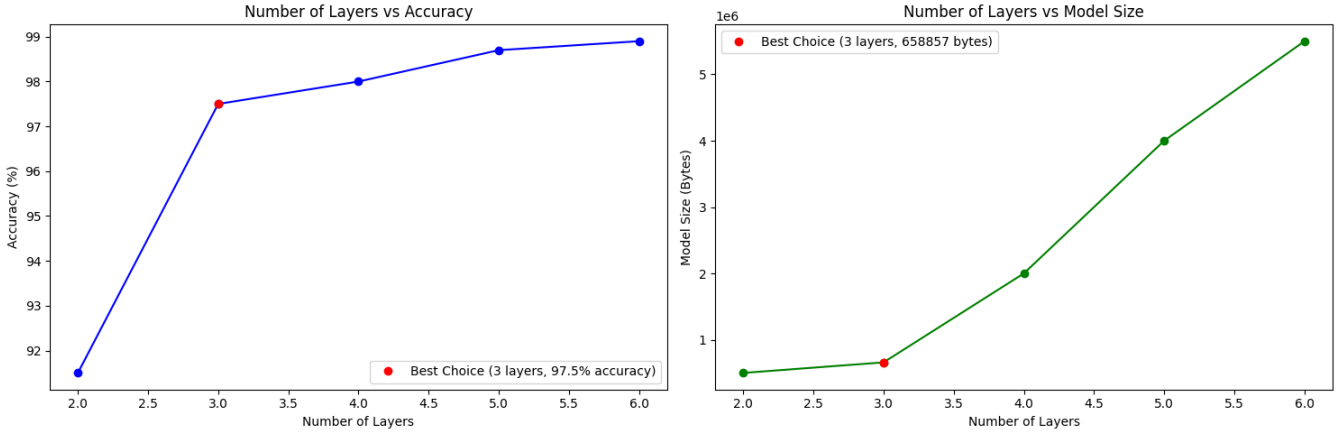

FIGURE 18. The Impact of Layer Count on Accuracy and Model Size in our proposed model.

Table 5 provides a detailed analysis of the impact of data splitting on model performance. The results demonstrate that using an 80% training data split consistently yields impressive performance across all metrics. However, when the data is divided such that 90% is allocated for training and the remaining 10% for validation and testing, we observe a high accuracy during training. Despite this, the model exhibits weaker performance in other metrics. This discrepancy can be attributed to overfitting, wherein the model excessively learns the details and noise in the training data, at the expense of its generalization capabilities on new data

Regarding the misclassification of certain classes, it's important to note that errors predominantly occur in a few classes. This can be attributed to similarities in the patterns of writing Arabic numbers or letters within those classes, leading to confusion during classification.

Lastly, the model conversion phase, detailed in Tables 7 and 8, was pivotal. Despite the considerable reduction in model size, which is crucial for deployment in resource-constrained environments, our model's performance remained unaffected, maintaining an impressive testing accuracy of 98.7%. This finding highlights the successful balance we achieved between model efficiency and performance, ensuring that our model's deployment is feasible without compromising its accuracy or reliability.

V. MODEL DEPLOYMENT FEASIBILITY AND COST-EFFECTIVENESS

The deployment of TinyML models in real-world applications requires a careful assessment of their cost-effectiveness, particularly in terms of computational and financial resources. Our proposed TinyDL model, tailored for recognizing Arabic numerals and letters through gesture-based interaction, demonstrates not only high accuracy but also a remarkable balance between efficiency and performance.

A. COMPUTATIONAL COST-EFFECTIVENESS

The efficiency of our model is evident in its architectural design. Optimized convolutional neural networks form the basis of the TinyDL model, which are inherently less computationally intensive due to their reduced parameter space. The optimized layers, paired with quantization techniques, allow the model to maintain a high degree of accuracy even after substantial reduction in computational demands.

B. FINANCIAL IMPLICATIONS

From a financial standpoint, the cost of deploying and maintaining our model is markedly low. The model's ability to run on inexpensive hardware without the need for high-end processing power or substantial memory significantly lowers the initial investment and operational costs.

C. ENERGY CONSUMPTION

Energy consumption [57], [59] is a critical factor in the deployment of continuous, real-time applications such as gesture recognition systems. Our model's minimized power requirements not only extend the battery life of portable devices but also reduce the energy costs associated with sustained use.

D. REAL-WORLD IMPLEMENTATION

To substantiate the model's real-world applicability, we have initiated pilot tests on commercially available microcontrollers. The results are promising, showcasing the model's ability to operate seamlessly in a live environment. These tests have provided valuable insights into the model's performance outside of a controlled setting, confirming its practicality and readiness for broader deployment.

VI. CONCLUSION

Our research has effectively illustrated TinyML's potential for gesture-based recognition of Arabic numerals and basic Arabic letters. Our TinyDL model, which achieved an exceptional accuracy rate of 97.5%, is proof of the synergy

between TinyML and deep learning. This achievement is especially noteworthy in light of the Arabic script's inherent complexity. Our approach, which has been rigorously tested and verified through comparative analysis and optimization for low-power devices, shows that advanced machine learning models can be deployed in resource-constrained contexts without compromising performance. Our model's applications go beyond simple recognition tasks, opening the door to more user-friendly and accessible human-computer interaction, particularly in Arabic-speaking areas. Our TinyDL model's integration with gesture recognition technology, which acknowledges the cultural value of the Arabic alphabet, marks a significant advancement in the creation of more inclusive and user-friendly digital interfaces. However, our research is not without limitations. The current dataset is limited in its diversity, primarily consisting of handwriting samples from a relatively homogeneous group of participants. Expanding the dataset to include a broader range of handwriting styles and samples from a more diverse participant pool would enhance the model's robustness and generalizability. Additionally, while our model demonstrates high accuracy, further research is needed to explore its scalability and adaptability to other languages and scripts. Future research directions include diversifying the dataset, exploring the model's applicability to other languages and scripts, and investigating the integration of our TinyDL model with other technologies to further enhance human-computer interaction. Furthermore, we recognize the potential challenges faced during the model's deployment in varied environmental conditions. Addressing these challenges is crucial for the model's real-world applicability. Potential enhancements could involve improving the model's robustness to different lighting conditions, handwriting tools, and user behaviors. Additionally, the future work could focus on enhancing the model's capabilities to recognize more complex Arabic scripts and potentially other languages, broadening the scope and impact of our research. Our work lays a solid foundation for the continued exploration of TinyML's potential in creating more inclusive and accessible digital interfaces.

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest regarding the publicationÂ ofÂ thisÂ article.

REFERENCES

[1] V. Rajapakse, I. Karunanayake, and N. Ahmed, "Intelligence at the extreme edge: A survey on reformable TinyML," *ACM Comput. Surv.*, vol. 55, no. 13s, pp. 1–30, Jul. 2023, doi: 10.1145/3583683.

[2] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, and S. Han, "Tiny machine learning: Progress and futures [feature]," *IEEE Circuits Syst. Mag.*, vol. 23, no. 3, pp. 8–34, May 2023, doi: 10.1109/MCAS.2023.3302182.

[3] Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki, and A. S. Hafid, "A comprehensive survey on TinyML," *IEEE Access*, vol. 11, pp. 96892–96922, 2023, doi: 10.1109/ACCESS.2023.3294111.

[4] M. Giordano, L. Piccinelli, and M. Magno, "Survey and comparison of milliwatts micro controllers for tiny machine learning at the edge," in *Proc. IEEE 4th Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Jun. 2022, pp. 94–97, doi: 10.1109/AICAS54282.2022.9870017.

[5] Q. Zhu and X. Zu, "Fully convolutional neural network structure and its loss function for image classification," *IEEE Access*, vol. 10, pp. 35541–35549, 2022, doi: 10.1109/ACCESS.2022.3163849.

[6] M. Nazar, M. M. Alam, E. Yafi, and M. M. Su'ud, "A systematic review of human–computer interaction and explainable artificial intelligence in healthcare with artificial intelligence techniques," *IEEE Access*, vol. 9, pp. 153316–153348, 2021, doi: 10.1109/ACCESS.2021.3127881.

[7] N. N. Alajlan and D. M. Ibrahim, "TinyML: Adopting tiny machine learning in smart cities," *J. Auto. Intell.*, vol. 7, no. 4, pp. 1–14, Jan. 2024, doi: 10.32629/jai.v7i4.1186.

[8] S. Vadera and S. Ameen, "Methods for pruning deep neural networks," *IEEE Access*, vol. 10, pp. 63280–63300, 2022, doi: 10.1109/ACCESS.2022.3182659.

[9] H. Peng, Y. Yu, and S. Yu, "Re-thinking the effectiveness of batch normalization and beyond," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 1, pp. 465–478, Jan. 2024, doi: 10.1109/TPAMI.2023.3319005.

[10] J. Qi, G. Jiang, G. Li, Y. Sun, and B. Tao, "Intelligent human–computer interaction based on surface EMG gesture recognition," *IEEE Access*, vol. 7, pp. 61378–61387, 2019, doi: 10.1109/ACCESS.2019.2914728.

[11] T. Watanabe, M. Maniruzzaman, M. A. M. Hasan, H.-S. Lee, S.-W. Jang, and J. Shin, "2D camera-based air-writing recognition using hand pose estimation and hybrid deep learning model," *Electronics*, vol. 12, no. 4, p. 995, Feb. 2023, doi: 10.3390/electronics12040995.

[12] A. Zhang, W. Zhu, and J. Li, "Spiking echo state convolutional neural network for robust time series classification," *IEEE Access*, vol. 7, pp. 4927–4935, 2019.

[13] J. Xu, H. Wang, J. Zhang, and L. Cai, "Robust hand gesture recognition based on RGB-D data for natural human–computer interaction," *IEEE Access*, vol. 10, pp. 54549–54562, 2022, doi: 10.1109/ACCESS.2022.3176717.

[14] Z. Wang, C. Zhou, X. Wu, T. Liu, and Y. Kang, "Application of mutual information maximization convolutional neural network in bearing feature extraction," *IEEE Sensors J.*, vol. 23, no. 24, pp. 30584–30592, Dec. 2023, doi: 10.1109/JSEN.2023.3316392.

[15] L. Jiashan and L. Zhonghua, "Dynamic gesture recognition algorithm combining global gesture motion and local finger motion for interactive teaching," *IEEE Access*, early access, 2021, doi: 10.1109/ACCESS.2021.3065849.

[16] N. Mohamed, M. B. Mustafa, and N. Jomhari, "A review of the hand gesture recognition system: Current progress and future directions," *IEEE Access*, vol. 9, pp. 157422–157436, 2021, doi: 10.1109/ACCESS.2021.3129650.

[17] X. Hu, P. Niu, J. Wang, and X. Zhang, "A dynamic rectified linear activation units," *IEEE Access*, vol. 7, pp. 180409–180416, 2019, doi: 10.1109/ACCESS.2019.2959036.

[18] E. Manor and S. Greenberg, "Custom hardware inference accelerator for TensorFlow lite for microcontrollers," *IEEE Access*, vol. 10, pp. 73484–73493, 2022, doi: 10.1109/ACCESS.2022.3189776.

[19] L. Liu, Y. Luo, X. Shen, M. Sun, and B. Li, "β-dropout: A unified dropout," *IEEE Access*, vol. 7, pp. 36140–36153, 2019, doi: 10.1109/ACCESS.2019.2904881.

[20] S. M. Kamal, Y. Chen, S. Li, X. Shi, and J. Zheng, "Technical approaches to Chinese sign language processing: A review," *IEEE Access*, vol. 7, pp. 96926–96935, 2019, doi: 10.1109/ACCESS.2019.2929174.

[21] B. Darvish Rouhani, A. Mirhoseini, and F. Koushanfar, "TinyDL: Just-in-time deep learning solution for constrained embedded systems," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4, doi: 10.1109/ISCAS.2017.8050343.

[22] J. Jung, H.-C. Moon, J. Kim, D. Kim, and K.-A. Toh, "Wi-Fi based user identification using in-air handwritten signature," *IEEE Access*, vol. 9, pp. 53548–53565, 2021, doi: 10.1109/ACCESS.2021.3071228.

[23] C.-Y. Lee, P. Gallagher, and Z. Tu, "Generalizing pooling functions in CNNs: Mixed, gated, and tree," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 863–875, Apr. 2018, doi: 10.1109/TPAMI.2017.2703082.

[24] J. Hu, P. Lin, H. Zhang, Z. Lan, W. Chen, K. Xie, S. Chen, H. Wang, and S. Chang, "A dynamic pruning method on multiple sparse structures in deep neural networks," *IEEE Access*, vol. 11, pp. 38448–38457, 2023, doi: 10.1109/ACCESS.2023.3267469.

[25] M. Tayyab, A. Hussain, M. A. Alshara, S. Khan, R. M. Alotaibi, and A. R. Baig, "Recognition of visual Arabic scripting news ticker from broadcast stream," *IEEE Access*, vol. 10, pp. 59189–59204, 2022, doi: 10.1109/ACCESS.2022.3179366.

[26] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*. Boca Raton, FL, USA: CRC Press, 2022, pp. 291–326.
[27] J. I. Pilataxi, J. E. Zambrano, C. A. Perez, and K. W. Bowyer, "Improved search in neuroevolution using a neural architecture classifier with the CNN architecture encoding as feature vector," *IEEE Access*, vol. 12, pp. 11987–12000, 2024, doi: 10.1109/ACCESS.2024.3355804.
[28] M. K. Jabde, C. H. Patil, A. D. Vibhute, and S. Mali, "A comprehensive literature review on air-written online handwritten recognition," *Int. J. Comput. Digit. Syst.*, vol. 15, no. 1, pp. 307–322, Jan. 2024, doi: 10.12785/ijcds/150124.
[29] K. M. O. Nahar, I. Alsmadi, R. E. Al Mamlook, A. Nasayreh, H. Gharaibeh, A. S. Almuflih, and F. Alasim, "Recognition of Arabic air-written letters: Machine learning, convolutional neural networks, and optical character recognition (OCR) techniques," *Sensors*, vol. 23, no. 23, p. 9475, Nov. 2023, doi: 10.3390/s23239475.
[30] C. Contoli and E. Lattanzi, "A study on the application of TensorFlow compression techniques to human activity recognition," *IEEE Access*, vol. 11, pp. 48046–48058, 2023, doi: 10.1109/ACCESS.2023.3276438.
[31] A. Daood, A. Al-Saegh, and A. F. Mahmood, "HAndwriting detection and recognition of Arabic numbers and characters using deep learning methods," *J. Eng. Sci. Technol.*, vol. 18, no. 3, pp. 1581–1598, 2023.
[32] B. Coffen and Md. S. Mahmud, "TinyDL: Edge computing and deep learning based real-time hand gesture recognition using wearable sensor," in *Proc. IEEE Int. Conf. E-Health Netw., Appl. Services (HEALTHCOM)*, Mar. 2021, pp. 1–6, doi: 10.1109/HEALTHCOM49281.2021.9399005.
[33] T. M. Ghanim, M. I. Khalil, and H. M. Abbas, "Comparative study on deep convolution neural networks DCNN-based offline Arabic handwriting recognition," *IEEE Access*, vol. 8, pp. 95465–95482, 2020, doi: 10.1109/ACCESS.2020.2994290.
[34] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, and L. Shao, "Normalization techniques in training DNNs: Methodology, analysis and application," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 8, pp. 10173–10196, Aug. 2023, doi: 10.1109/TPAMI.2023.3250241.
[35] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.
[36] T. Yanay and E. Shmueli, "Air-writing recognition using smart-bands," *Pervas. Mobile Comput.*, vol. 66, Jul. 2020, Art. no. 101183, doi: 10.1016/j.pmcj.2020.101183.
[37] F. A. Abir, M. A. Siam, A. Sayeed, M. A. M. Hasan, and J. Shin, "Deep learning based air-writing recognition with the choice of proper interpolation technique," *Sensors*, vol. 21, no. 24, p. 8407, Dec. 2021, doi: 10.3390/s21248407.
[38] S. K. Leem, F. Khan, and S. H. Cho, "Detecting mid-air gestures for digit writing with radio sensors and a CNN," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 4, pp. 1066–1081, Apr. 2020, doi: 10.1109/TIM.2019.2909249.
[39] C.-H. Hsieh, Y.-S. Lo, J.-Y. Chen, and S.-K. Tang, "Air-writing recognition based on deep convolutional neural networks," *IEEE Access*, vol. 9, pp. 142827–142836, 2021, doi: 10.1109/ACCESS.2021.3121093.
[40] S. B. Abdullahi and K. Chamnongthai, "IDF-sign: Addressing inconsistent depth features for dynamic sign word recognition," *IEEE Access*, vol. 11, pp. 88511–88526, 2023, doi: 10.1109/access.2023.3305255.
[41] S. B. Abdullahi and K. Chamnongthai, "American sign language words recognition of skeletal videos using processed video driven multi-stacked deep LSTM," *Sensors*, vol. 22, no. 4, p. 1406, Feb. 2022, doi: 10.3390/s22041406.
[42] S. B. Abdullahi and K. Chamnongthai, "American sign language words recognition using spatio-temporal prosodic and angle features: A sequential learning approach," *IEEE Access*, vol. 10, pp. 15911–15923, 2022, doi: 10.1109/ACCESS.2022.3148132.
[43] A.-A. Liu, Y. Wang, N. Xu, S. Liu, and X. Li, "Scene-graph-guided message passing network for dense captioning," *Pattern Recognit. Lett.*, vol. 145, pp. 187–193, May 2021, doi: 10.1016/j.patrec.2021.01.024.
[44] J. Brokešová, J. Málek, J. Vackář, F. Bernauer, J. Wassermann, and H. Igel, "Rotaphone-CY: The newest rotaphone model design and preliminary results from performance tests with active seismic sources," *Sensors*, vol. 21, no. 2, p. 562, Jan. 2021, doi: 10.3390/s21020562.
[45] L. Chen, D. Zhang, and M. Liu, "A lightweight convolutional neural network for real-time air handwriting recognition," *J. Real-Time Image Process.*, vol. 20, no. 3, pp. 755–766, 2023.
[46] F. Shao, L. Chen, J. Shao, W. Ji, S. Xiao, L. Ye, Y. Zhuang, and J. Xiao, "Deep learning for weakly-supervised object detection and localization: A survey," *Neurocomputing*, vol. 496, pp. 192–207, Jul. 2022, doi: 10.1016/j.neucom.2022.01.095.
[47] J. Zhang, Y. Wang, and X. Chen, "Air-writing recognition using wearable motion sensors and a deep learning approach," *IEEE Access*, vol. 10, pp. 48462–48472, 2022.
[48] G. Andac, A. Kalender, B. Baddal, and F. Basmaci, "Impact of different access cavity designs and Ni–Ti files on the elimination of *Enterococcus faecalis* from the root canal system: An in vitro study," *Appl. Sci.*, vol. 12, no. 4, p. 2049, Feb. 2022, doi: 10.3390/app12042049.
[49] S. B. Abdullahi, K. Chamnongthai, V. Bolon-Canedo, and B. Cancela, "Spatial–temporal feature-based end-to-end Fourier network for 3D sign language recognition," *Expert Syst. Appl.*, vol. 248, Aug. 2024, Art. no. 123258, doi: 10.1016/j.eswa.2024.123258.
[50] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, 1965.
[51] M. L. V. Pitteway, "Algorithm for drawing ellipses or hyperbolae with a digital plotter," *Comput. J.*, vol. 10, no. 3, pp. 282–289, Mar. 1967.
[52] J. F. Blinn, "A generalization of algebraic surface drawing," *ACM SIGGRAPH Comput. Graph.*, vol. 16, no. 3, p. 273, Jul. 1982.
[53] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.
[54] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[55] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019, doi: 10.1109/JPROC.2019.2918951.
[56] N. Schizas, A. Karras, C. Karras, and S. Sioutas, "TinyML for ultra-low power AI and large scale IoT deployments: A systematic review," *Future Internet*, vol. 14, no. 12, p. 363, Dec. 2022, doi: 10.3390/fi14120363.
[57] A. Sabovic, M. Aernouts, D. Subotic, J. Fontaine, E. De Poorter, and J. Famaey, "Towards energy-aware TinyML on battery-less IoT devices," *Internet Things*, vol. 22, Jul. 2023, Art. no. 100736, doi: 10.1016/j.iot.2023.100736.
[58] S. S. Saha, S. S. Sandha, and M. Srivastava, "Machine learning for microcontroller-class hardware: A review," *IEEE Sensors J.*, vol. 22, no. 22, pp. 21362–21390, Nov. 2022, doi: 10.1109/JSEN.2022.3210773.
[59] W. Raza, A. Osman, F. Ferrini, and F. D. Natale, "Energy-efficient inference on the edge exploiting TinyML capabilities for UAVs," *Drones*, vol. 5, no. 4, p. 127, Oct. 2021, doi: 10.3390/drones5040127.
[60] L. Capogrosso, F. Cunico, D. S. Cheng, F. Fummi, and M. Cristani, "A machine learning-oriented survey on tiny machine learning," *IEEE Access*, vol. 12, pp. 23406–23426, 2024, doi: 10.1109/ACCESS.2024.3365349.
[61] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, no. 1, p. 53, Mar. 2021, doi: 10.1186/s40537-021-00444-8.
[62] A. A. A. Ali and M. Suresha, "Survey on segmentation and recognition of handwritten Arabic script," *Social Netw. Comput. Sci.*, vol. 1, no. 4, p. 192, Jun. 2020, doi: 10.1007/s42979-020-00187-y.
[63] R. Rastgoo, K. Kiani, and S. Escalera, "Sign language recognition: A deep survey," *Expert Syst. Appl.*, vol. 164, Feb. 2021, Art. no. 113794, doi: 10.1016/j.eswa.2020.113794.
[64] A. Wadhawan and P. Kumar, "Sign language recognition systems: A decade systematic literature review," *Arch. Comput. Methods Eng.*, vol. 28, no. 3, pp. 785–813, May 2021, doi: 10.1007/s11831-019-09384-2.
[65] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[66] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," in *Proc. 40th Int. Conf. Mach. Learn.*, vol. 202, Jul. 2023, pp. 23803–23828.

ISMAIL LAMAAKAL (Graduate Student Member, IEEE) received the Master of Science degree in computer science from the Multidisciplinary Faculty of Nador, University Mohammed Premier, Oujda, Morocco, where he is currently pursuing the Ph.D. degree in computer science. As an Artificial Intelligence Scientist, his research primarily focuses on the innovative integration of tiny machine learning, the Internet of Things (IoT), and embedded systems. His work is characterized by its pioneering approach in the field, emphasizing practical applications and advancements in these interconnected domains. His contributions are marked by a commitment to pushing the boundaries of AI and its applications in the modern technological landscape.

IBRAHIM OUAHBI received the Ph.D. degree in didactics of informatics from Sidi Mohammed Ben Abdellah University, Fez, Morocco. He was a Professor of educational technologies with the Faculty of Educational Sciences, Mohammed V University in Rabat, in 2019. He is currently a Professor of computer science with the Multidisciplinary Faculty of Nador, University Mohammed Premier, Oujda, Morocco. His research interests include artificial intelligence, cybersecurity, and ICT integration in science education and learning.

KHALID EL MAKKAOUI received the master's degree in networks and systems and the Ph.D. degree in computer science from Hassan 1st University, Settat, Morocco, in 2014 and 2018, respectively. Since 2019, he has been an Associate Professor with the Department of Computer Science, Multidisciplinary Faculty of Nador, University Mohammed Premier, Oujda, Morocco. He has published over 40 papers (book chapters, international journals, and conferences). His research interests include cybersecurity and artificial intelligence.

YASSINE MALEH (Senior Member, IEEE) is currently a Professor of cybersecurity and IT governance with Sultan Moulay Slimane University, Morocco. He has made contributions in the fields of information security and privacy, the Internet of Things security, wireless, and constrained network security. He has published over 200 papers (book chapters, international journals, and conferences/workshops), 30 edited books and 6 authored books. His research interests include information security and privacy, the Internet of Things, network security, information systems, and IT governance. He is a member of the International Association of Engineers (IAENG) and the Machine Intelligence Research Laboratories. He received Publons Top 1% Reviewer Award for the years 2018 and 2019. He was the Publicity Chair of BCCA'19 and the General Chair of the MLBDACP'19 Symposium and the ICI2C'21 Conference. He is also the Founding Chair of the IEEE Consultant Network Morocco and the Founding President of African Research Center of Information Technology and Cybersecurity. He serves as an Associate Editor for IEEE Access (2019 Impact Factor 4.098), the *International Journal of Digital Crime and Forensics*, and the *International Journal of Information Security and Privacy*. He is a Series Editor of *Advances in Cybersecurity Management* (CRC Taylor and Francis). He is the Editor-in-Chief of the *International Journal of Information Security and Privacy* and the *International Journal of Smart Security Technologies*. He served as a Guest Editor for the Special Issue on Recent Advances on Cyber Security and Privacy for Cloud-of-Things of the *International Journal of Digital Crime and Forensics* (Volume 10, Issue 3, July-September 2019). He has served and continues to serve on executive and technical program committees and as a reviewer for numerous international conferences and journals, such as *Ad Hoc Networks* (Elsevier), *IEEE Network* magazine, IEEE Sensor Journal, *ICT Express*, and *Cluster Computing* (Springer).

PAWEŁ PŁAWIAK was born in Ostrowiec, Poland, in 1984. He received the B.Eng. and M.Sc. degrees in electronics and telecommunications and the Ph.D. degree (Hons.) in biocybernetics and biomedical engineering from the AGH University of Science and Technology, Kraków, Poland, in 2012 and 2016, respectively, and the D.Sc. degree in technical computer science and telecommunications from the Silesian University of Technology, Gliwice, Poland, in 2020. He is currently the Dean of the Faculty of Computer Science and Telecommunications and an Associate Professor with the Cracow University of Technology, Kraków, the Deputy Director for research with the National Institute of Telecommunications, Warsaw, and an Associate Professor with the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Gliwice. He has published more than 50 articles in refereed international SCI-IF journals. His research interests include machine learning and computational intelligence (e.g., artificial neural networks, genetic algorithms, fuzzy systems, support vector machines, k-nearest neighbors, and hybrid systems), ensemble learning, deep learning, evolutionary computation, classification, pattern recognition, signal processing and analysis, data analysis and data mining, sensor techniques, medicine, biocybernetics, biomedical engineering, and telecommunications. He is an academic editor and a reviewer of many prestigious and reputed journals.

FAHAD ALBLEHAI received the B.S. degree in education in the field of computer, the M.S. degree in information technology and communication, and the Ph.D. degree in e-learning/web-/internet-based teaching and learning, in 2001, 2010, and 2017, respectively. He has been an Associate Professor with the Community College, King Saud University (KSU), since 2019. His research interests include web applications, digital transformation, augmented reality, virtual reality, cloud computing, virtual learning environments, e-learning, m-learning, AI, and human–computer interaction.