An Amygdala-Inspired Classical Conditioning Model Implemented on an FPGA for Home Service Robots

IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received October 27, 2020, accepted November 8, 2020, date of publication November 16, 2020, date of current version December 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3038161

An Amygdala-Inspired Classical Conditioning Model Implemented on an FPGA for Home Service Robots

YUICHIRO TANAKA[1,2], (Graduate Student Member, IEEE), TAKASHI MORIE[1,3], (Member, IEEE), AND HAKARU TAMUKOH[1,3], (Member, IEEE)

[1]Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, Fukuoka 808-0196, Japan
[2]Japan Society for Promotion of Science, Tokyo 102-0083, Japan
[3]Research Center for Neuromorphic AI Hardware, Kyushu Institute of Technology, Fukuoka 808-0196, Japan

Corresponding author: Yuichiro Tanaka (tanaka.yuichiro483@mail.kyutech.jp)

This work was supported by the JSPS KAKENHI under Grant 19J11524.

ABSTRACT This study develops an intelligent system for home service robots mimicking human brain function that can manage common knowledge applicable to any environment and local knowledge reflecting its specific environment. Deep learning is effective for acquiring common knowledge because the performance of deep learning relies on the amounts of training and big training data that can be accessed for such knowledge; however, deep learning is ineffective for acquiring local knowledge because no big training data for such knowledge exist. Thus, we propose a brain-inspired learning model and system for acquiring local knowledge using small training data. We focus on the amygdala because its classical fear conditioning is effective for training using small training data. We propose an amygdala-inspired classical conditioning model comprising multiple self-organizing maps (lateral nucleus) and a fully connected neural network (central nucleus), imitating the function and structure of the amygdala. The proposed model is applied to a task of a waiter robot in a restaurant, and the model can learn customers' preferences after only a few human-robot interactions. We accelerate the computation of the model and reduce its power consumption by proposing a hardware-oriented algorithm for the model and its digital hardware design and implement it in an XCZU9EG field programmable gate array. The hardware-oriented algorithm reduces the multiplication operations and exponential functions requiring huge hardware resources. The performance of the hardware operated at 150 MHz is 1,273 times faster than the software implementation on Arm Cortex-A53, and the power consumption of the chip is 5.009 W.

INDEX TERMS Amygdala, classical conditioning, field programmable gate array (FPGA), hardware, home service robot.

I. INTRODUCTION

Recently, due to an aging population and a shrinking workforce, home service robots [1] have attracted considerable attention. Such robots should support human memories and assist human decisions while working and living with their owners in domestic environments; therefore, a brain-like intelligent system is necessary. Thus, we developed a system that mimics human brain functions. This study focuses on knowledge acquisition functions in such an intelligent system. In general, home service robots must acquire two types of knowledge, namely, common knowledge, one that holds in all environments, and local knowledge, one that holds only in a specific environment.

For example, to respond to an owner's request to bring "green tea," the robot must possess common knowledge of what "green tea" is. Deep learning (DL) [2] is a powerful tool for acquiring common knowledge because copious amounts of training data on common knowledge can be prepared. Image recognition is a typical example of a task where common knowledge is acquired. Deep neural networks have been widely used in image recognition tasks where

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

substantial training data are available to furnish state-of-the-art results [3]–[5]. Thus, previous studies have proposed the use of DL for home service robots [6]–[8].

Local knowledge, however, is qualitatively different. For example, if an owner asks a robot to bring "that," the robot must understand the local meaning of "that." DL performs poorly in the acquisition of local knowledge because its performance relies on the size of the training dataset, and large datasets of local knowledge cannot be obtained. In contrast, human beings easily acquire local knowledge (e.g., the family's preferences and customs), requiring minimal contact with the environment to do so. Similarly, home service robots should be able to quickly obtain local knowledge through only a small number of human-robot interactions. Thus, we propose a brain-inspired system that can acquire local knowledge, aside from DL, from a few human-robot interactions. We specifically focus on the amygdala, which is an area of the brain associated with classical fear conditioning [9]–[13], which is effective for training with little data.

To construct an intelligent system into home service robots, we have to consider computational limitations and power consumption. In the home service robot system, various software programs run simultaneously; thus, offloading and accelerating programs are effective. Although graphics processing units (GPUs) are often used to accelerate the computations, they tend to have high power consumption, severely impairing the use of home service robots, which should consume little power. Thus, modern GPUs are unsuitable for home service robots.

We provide a system with fast processing and low power consumption by designing and implementing an intelligent system into a field programmable gate array (FPGA). The hardware on FPGAs runs at a lower frequency than software on central processing units (CPUs) and GPUs. For example, a convolutional neural network [3] implemented on an FPGA is faster and consumes less power compared with that implemented on an embedded GPU [14]. Therefore, such hardware can implement a system with fast processing and low power consumption.

In this study, we propose a classical conditioning model imitating the structure and functions of the biological amygdala. Moreover, we propose a very large-scale integration (VLSI) implementation of our model using an FPGA. The paper is organized as follows. Section II describes the theory of the biological amygdala and related studies of amygdala models, and Section III describes our proposed classical conditioning model and its hardware implementation. Section IV describes experiments evaluating the proposed model and the hardware. Section V provides the discussion, and Section VI concludes this paper.

II. AMYGDALA

A. CLASSICAL CONDITIONING

The amygdala is in the cerebral system of the brain and is well known as a part processing emotion, particularly associated with fear conditioning [9]–[13]. Fear conditioning is a type of classical conditioning, which is famous for the Pavlovian dog [15] where a dog learns from repeated experience; the dog receives food whenever it hears the sound of a bell, and therefore, it associates the sound of the bell with food. The classical conditioning approach is effective for acquiring local knowledge from experience.

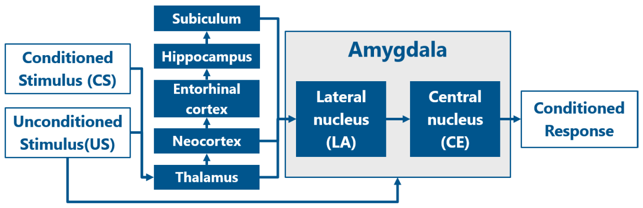


FIGURE 1. Mechanism of fear conditioning.

Fig. 1 shows the mechanism of fear conditioning. During fear conditioning, input sensory stimuli are classified as either unconditioned stimuli (US) or conditioned stimuli (CS). US cause unconditioned responses, whereas CS are neutral and pass through various areas of the brain: thalamus, neocortex, and hippocampus. These stimuli are then fed into a part of the amygdala called the lateral nucleus (LA) via two pathways [11]. One is a direct pathway from the thalamus where sensory information is rapidly delivered, and the other is an indirect pathway through cortexes where information is slowly delivered. In the direct pathway, the sensory stimulus features are fed into the LA. In the indirect pathway, the recognition of objects, concepts, and contexts are fed into the LA. Therefore, the LA is a part that integrates CS from various brain regions. Moreover, the neurons that react to certain stimuli are located near each other [11], [16]. The LA conditions CS to US [17]. Once integrated, the stimuli are fed into a part of the amygdala called the central nucleus (CE). After the conditioning, the CE outputs conditioned responses (CRs) even if CS is fed into the amygdala without US.

In summary, the processes of fear conditioning in the amygdala involve sensory stimulus recognition and conditioning CS to US. In sensory stimulus recognition, CS are classified based on the features of stimuli without supervised signals. Therefore, the amygdala can classify unknown sensory stimuli using its self-organizing recognition system. In the conditioning process, the amygdala learns the relations between CS and US. As a result of the process, the amygdala can predict emotional values of sensory stimuli that are originally neutral.

B. MODELS BASED ON THE BIOLOGICAL AMYGDALA

Several studies on fear and classical conditioning models based on the biological amygdala have been proposed [18]–[21]. Armony *et al.* [18] proposed an amygdala model for fear conditioning by considering the connections between the amygdala, auditory thalamus, and auditory cortex. The model performed fear conditioning between the auditory inputs as CS and US.

Morén and Balkenius [19] proposed a model of connections between the amygdala and orbitofrontal cortex that controls the extinction [22]. The orbitofrontal cortex part of the model inhibits the amygdala part and can achieve extinction effects in fear conditioning.

Izhikevich [20] proposed a spiking neuron model for classical and instrumental conditioning. The model implemented by Izhikevich neuron model [23] uses a spike-timing-dependent plasticity learning rule modulated by dopamine. This study simulated a model comprising 1,000 neurons and revealed that the model could solve the distal reward problem [24] where reward was given after reward-triggering actions.

Li *et al.* [21] proposed a spiking neuron model for acquisition and extinction of fear memory. The model is based on fear, persistent, and extinction neurons in the amygdala and medial prefrontal cortex. This study elucidated that the network involving the amygdala and prefrontal cortex showed the partial reinforcement extinction effect [25]–[27], in which fear memory acquired through the partial pairings of CS and US was more resistant to extinction than that acquired through full pairings.

Although these studies proposed fear and classical conditioning models, the purposes of these studies are different from that of ours. They aimed to implement biologically plausible models, whereas we aim to develop a local knowledge acquisition system for home service robots. From the viewpoint of robot applications, we need a sensory stimulus recognition, which is an important process in fear and classical conditioning. Moreover, we have to consider computational costs of the system because computational resources and power in robot systems are limited.

C. MODELS FOR ROBOT APPLICATIONS

Several studies involving amygdala models applied to robots have been proposed [28]–[30]. Sonoh *et al.* [28] proposed an emotional expression model of the amygdala (EMA), comprising a self-organizing map (SOM) [31] and a fully connected neural network (FCNN). The SOM is a model of the LA, while the FCNN is a model of the CE. Neurons in the SOM with similar reference vectors are proximally located. The SOM feature is intended to imitate the LA feature. Experiments with the EMA implemented in an FPGA achieved classical conditioning behavior between signals from a color sensor (as CS) and signals from a tactile sensor (as US). For example, when the EMA simultaneously receives "red" and "slapped" from the sensors, it conditions these stimuli. After the conditioning, the EMA outputs the "fear" response even if the EMA receives a "red" stimulus without being "slapped." This model includes the concept of the amygdala where the LA classifies sensory stimuli and then learns the relation between CS and US. Owing to the introduction of the SOM as the LA, the model can predict an emotional response, even if it receives unknown CS. However, the task is still simplistic compared to the complex demands placed on home service robots.

Rizzi *et al.* [29], [30] proposed a situation-aware fear learning (SAFEL) model. To predict threatening situations, the SAFEL model learns whether situations are aversive. The model comprises the amygdala, hippocampus, and working memory modules. The amygdala module is used as a trigger for learning aversive situations; it receives a stimulus from the environment and judges whether the stimulus is aversive. If the amygdala module judges that the stimulus is aversive, the module outputs an adrenaline signal into the hippocampus module that buffers sensory stimuli to determine whether the buffered stimuli are aversive. When the hippocampus module receives the adrenaline signal from the amygdala module, the module regards the time-series stimulus (defined as a situation) right before receiving the adrenaline signal, as an aversive situation. The aversive situation is then memorized in the working memory module. Thereafter, the SAFEL model can predict aversive situations by referring to the working memory module. In this model, the amygdala model is implemented by a type of FCNN. Estimation of aversiveness is based on the classical conditioning of aversive stimuli (as US) and neutral stimuli (as CS). Classical conditioning is implemented by associative learning. Although the concept of the amygdala module is based on LeDoux's research, this study does not consider the structure of the amygdala shown in EMA [28]; hence, predicting aversiveness from unknown stimuli is challenging. Moreover, they do not implement the model on the hardware.

III. PROPOSED MODEL

In this study, to apply a local knowledge acquisition system to home service robots, we focus on the amygdala concept where LA and CE are based on classical conditioning rather than completely mimicking the biological amygdala. We follow the concept of EMA [28] because it has the sensory stimulus recognition system implemented by the SOM that can accept several types of CS. Moreover, we extend the functions of the model to adapt complex tasks for home service robots.

A. STRUCTURE

We proposed a novel amygdala-inspired classical conditioning model [32], [32]. Fig. 2 shows the structure of the model (right side of the figure), comprising multiple SOMs as the LA (two SOMs are shown in Fig. 2) and an FCNN as the CE. We replaced the single SOM of EMA with multiple SOMs that can be regarded as a layer of deep SOM networks [33].

The model simultaneously receives a US and multiple CS. These CS are fed into and classified by the SOMs, and the classification results are converted as vectors and concatenated. The vectors are then fed into the FCNN, learning the relation between US and the vectors (the training mode). Thereafter, the model outputs a CR even if the model only receives CS (the inference mode).

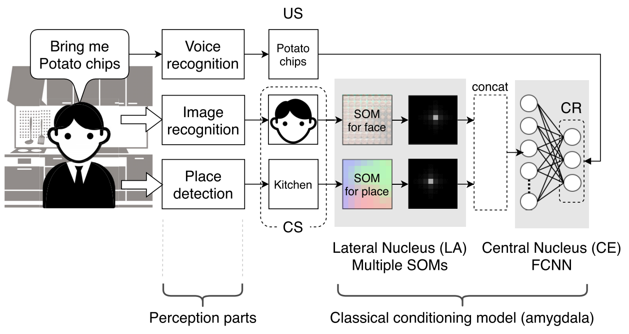


FIGURE 2. Classical conditioning model and perception parts.

As a test case, we applied the model to a robot working in a restaurant. For this, we added perception parts for human-robot interactions (left side of Fig. 2). When the robot takes an order, it attempts to retrieve the information of the ordered object via perception parts. The object's information is US. Simultaneously, the robot retrieves information about the customer's face and the robot's location via the perception parts as CS. The model learns the relation between US and CS in the training mode. Following this training mode, the model can predict what the customer wants using only CS in the inference mode.

Although, in this study, we use the structure shown in Fig. 2, which includes two SOMs and three perception parts, the SOMs and the perception parts can be extended. For example, in addition to the customer's face and place, the time when the customer places the order can be used as CS. If the model receives the face, place, and time as CS, it can learn and predict the customer's preference depending on the place and time. Compared with EMA [28], the proposed model is advantageous because it can consider combinations of multiple factors in the environment. In particular, the model shown in Fig. 2 predicts what the customer wants based on the combinations of the face and place. Furthermore, by assigning various factors in the environment as CS, we can use the model even if we do not know which factor triggers US.

B. ALGORITHM

In this study, the outputs of the perception parts are fed into the LA and CE models on the hardware.

The inference mode is implemented as described here. If several vectors of CS $\boldsymbol{x}_k \in \mathbb{R}^N$ $(k = 1, \ldots, K)$ are given, where K is the number of CS ($K = 2$ in the case of the classical conditioning model shown in Fig. 2) and N is a dimension of the vector, then CS are fed into the SOMs. This LA module comprises K SOMs. Thus, $\boldsymbol{x}_k$ is fed into the k^{th} SOM. In each SOM, a winning neuron with a reference vector most like the input vector is chosen according to the following equation:

$$(i_c, j_c) = \arg\min_{i,j} \|\boldsymbol{x}_k - \boldsymbol{w}_{i,j}\|, \tag{1}$$

where $\boldsymbol{w}_{i,j} \in \mathbb{R}^N$ is one of the reference vectors of the SOM. The neurons in each SOM are located on a two-dimensional grid ($I \times J$ grid), and the positions of the neurons in each SOM are represented using indices (i, j) $(i = 1, \ldots, I; j = 1, \ldots, J)$. The indices (i_c, j_c) represent the position of the winning neuron. Then, the SOM can output a vector that depends on the position of the winning neuron. The output vector of the SOM $\boldsymbol{h}_k \in \mathbb{R}^{I \times J}$ is computed as

$$h_{i,j} = \exp(-d_{i,j}^2/2\sigma_{\text{prop}}^2), \tag{2}$$

$$d_{i,j} = \sqrt{(i - i_c)^2 + (j - j_c)^2}, \tag{3}$$

where $h_{i,j}$ is an element of the output vector $\boldsymbol{h}_k$. In other words, one neuron at position (i, j) in the SOM outputs a scalar value $h_{i,j}$. The variable $d_{i,j}$ is the distance from the winning neuron to a position (i, j). In addition, σ_{prop} indicates the width of the Gaussian function. All the output vectors of the SOMs are concatenated as

$$\boldsymbol{H} = \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_K\}. \tag{4}$$

The concatenated vector $\boldsymbol{H} \in \mathbb{R}^M$ $(M = I \times J \times K)$ is fed into the FCNN. The FCNN outputs a vector representing a likelihood of what the customer wants. The output vector of the FCNN, $\boldsymbol{y} \in \mathbb{R}^L$, is computed via the following equation:

$$y_l = f(\boldsymbol{H} \cdot \boldsymbol{v}_l^\top), \tag{5}$$

where y_l is an element of the output vector $\boldsymbol{y}$. $\boldsymbol{v}_l \in \mathbb{R}^M$ is a weight vector corresponding to a neuron l $(l = 1, \ldots, L)$, and f is an activation function. As the activation function, the identity, Sigmoid, and Softmax functions can be used.

In the training mode, the reference vectors of SOMs are updated via unsupervised competitive learning, which can be represented as

$$\Delta \boldsymbol{w}_{i,j} = \alpha(\boldsymbol{x}_k - \boldsymbol{w}_{i,j})\exp(-d_{i,j}^2/2\sigma_{\text{learn}}^2), \tag{6}$$

$$\boldsymbol{w}_{i,j}^{\text{new}} = \boldsymbol{w}_{i,j}^{\text{old}} + \Delta \boldsymbol{w}_{i,j}, \tag{7}$$

where α is a learning rate and σ_{learn} is the width of the Gaussian function; the weight vectors of the FCNN are updated by the delta rules.

$$\Delta \boldsymbol{v}_l = -\beta(y_l - t_l)f'(\boldsymbol{H} \cdot \boldsymbol{v}_l^\top)\boldsymbol{H}, \tag{8}$$

$$\boldsymbol{v}_l^{\text{new}} = \boldsymbol{v}_l^{\text{old}} + \Delta \boldsymbol{v}_l, \tag{9}$$

where t_l is a supervised signal. In other words, t_l is an element of the US vector, $\boldsymbol{t} \in \mathbb{R}^L$. β is a learning rate.

C. HARDWARE-ORIENTED ALGORITHM

To implement the classical conditioning model into an FPGA, a hardware-oriented algorithm that reduces hardware resources is required because of limited FPGA resources. In particular, a reduction in the multiplication operations of the algorithm is required because of limited multipliers in the FPGA (digital signal processor (DSP) slices). In the case of an XCZU9EG FPGA [34], the target device in this study, 2,520 DSP slices, 274,080 lookup tables (LUTs), and 548,160 flip flops (FFs) are present. Moreover, exponential functions should be avoided because they require huge hardware resources. Herein, we developed a hardware-oriented algorithm for the classical conditioning model and implemented it using a fixed-point or integer number, because hardware implementation using a floating-point number requires larger hardware resources than that using the fixed-point and integer number.

In (1), the distance of vectors $\|\boldsymbol{x}_k - \boldsymbol{w}_{i,j}\|$ is computed. In the case of an original SOM algorithm, L2 norm, $\|\boldsymbol{x}_k - \boldsymbol{w}_{i,j}\|_2 = \sum_{n=1}^{N}(x_{k,n} - w_{i,j,n})^2$, is often used for distance. However, the L2 norm includes multiplication operations that should be reduced for hardware implementation. Thus, instead of the L2 norm, we use the L1 norm, $\|\boldsymbol{x}_k - \boldsymbol{w}_{i,j}\|_1 = \sum_{n=1}^{N}|x_{k,n} - w_{i,j,n}|$, which does not require any multiplications. In the algorithm, we use eight-bit integer numbers as variables for $\boldsymbol{x}_k$ and $\boldsymbol{w}_{i,j}$.

Equations (2), (3), and (6) use various multiplication operations and exponential functions. We replace these multiplication operations and exponential functions with bit-shift operations. The algorithm is operated according to the following equations:

$$h_{i,j} = 2^{-D_{i,j}}, \tag{10}$$

$$D_{i,j} = |i - i_c| + |j - j_c|, \tag{11}$$

$$\Delta\boldsymbol{w}_{i,j} = 2^{-(A+D_{i,j})}(\boldsymbol{x}_k - \boldsymbol{w}_{i,j}), \tag{12}$$

$$\boldsymbol{w}_{i,j}^{\text{new}} = \boldsymbol{w}_{i,j}^{\text{old}} + \Delta\boldsymbol{w}_{i,j}, \tag{13}$$

where A is a positive integer and 2^{-A} is the learning rate corresponding to α in (6). In (10) and (12), we use the power of two instead of exponential functions. The power of two can be represented as an arithmetic bit-shift operation. In (10), we use an eight-bit fixed-point number with two bits of the integer part and six bits of the decimal part as a variable for $h_{i,j}$. In (3), the Euclidean distance $d_{i,j}$ is used. Alternatively, we use the Manhattan distance $D_{i,j}$. The critical feature of these equations is that they do not require multiplication operations.

Besides the LA part, we do not change the algorithm of the CE part shown in (5) and (8). We avoid using the floating-point numbers by using an eight-bit fixed-point number with two bits of the integer part and six bits of the decimal part as a variable for $\boldsymbol{v}_l$. In addition, we use an identity function as the activation function f to avoid exponential functions, such as in the Sigmoid and Softmax functions.

D. HARDWARE IMPLEMENTATION

Using the proposed hardware-oriented algorithm, we designed a dedicated hardware for the classical conditioning model. Fig. 3 shows the entire configuration of the hardware of the model, comprising K LA modules and a CE module. Vectors $\boldsymbol{x}_k$ as CS are fed into the LA modules. Computations of the LA modules are simultaneously executed in parallel. Each LA module outputs $\boldsymbol{h}_k$, and all output vectors are concatenated as $\boldsymbol{H}$. The concatenated vector $\boldsymbol{H}$ is fed into the CE module. The CE module outputs a vector $\boldsymbol{y}$ as CR. In the training mode, vector $\boldsymbol{t}$, as US, is fed into the CE module. Then, the CE module updates its weight vector, and the LA modules update the reference vectors.

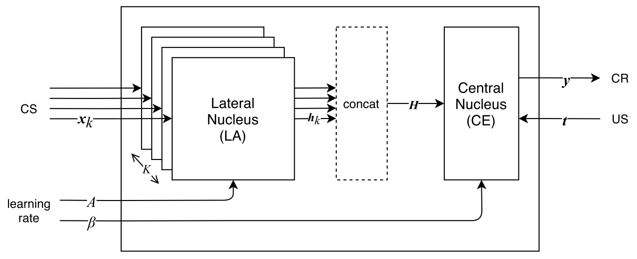


FIGURE 3. Hardware of the classical conditioning model.

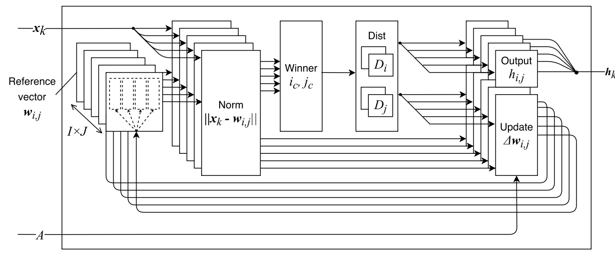


FIGURE 4. Hardware of the LA module.

1) LA MODULE

Fig. 4 shows a configuration of the hardware of the LA module. The LA module has $I \times J$ neurons, and all computations of the neurons are executed in parallel. The LA module includes random access memories (RAMs) for reference vectors $\boldsymbol{w}_{i,j}$ and several computation units, including norm computation units (Norm), a winner selection unit (Winner), distance computation units (Dist), output units (Output), and update units (Update). The norm computation units compute the L1 norm $\|\boldsymbol{x}_k - \boldsymbol{w}_{i,j}\|_1$ loading $\boldsymbol{x}_k$ from an input port, and $\boldsymbol{w}_{i,j}$ from the RAMs. The computation results are fed into the winner selection unit. The winner selection unit selects a winner neuron with the smallest $\|\boldsymbol{x}_k - \boldsymbol{w}_{i,j}\|_1$ and outputs indices of the winner (i_c, j_c). The indices of the winner are fed into the distance computation units. The distance computation units compute the distances between own neurons and the winner neuron $D_{i,j} = D_i + D_j$. The distances are simultaneously fed into the output and update units. The output units compute a vector $\boldsymbol{h}_k$ using the distances $D_{i,j}$. In the case of the training mode, the update units compute updated reference vectors. The update units can simultaneously compute several elements of the vector. The computed results of the update units are stored in the RAMs for the reference vectors.

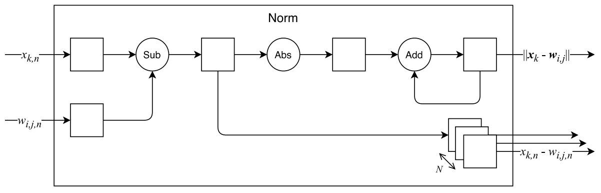


FIGURE 5. Hardware of the norm computation unit.

Fig. 5 shows the norm computation unit. Sub in the figure shows a processing element for subtraction, Abs is a processing element for absolution, and Add is a processing element for addition. Rectangles in the figure show registers. The norm computation unit sequentially receives elements from the input vector $\boldsymbol{x}_k$ and the reference vector $\boldsymbol{w}_{i,j}$, represented as $x_{k,n}$ and $w_{i,j,n}$, respectively, in the figure. First, a subtraction of those elements $(x_{k,n} - w_{i,j,n})$ is computed. Then, an absolute value of the subtraction $|x_{k,n} - w_{i,j,n}|$ is computed and accumulated to compute the L1 norm $\|\boldsymbol{x}_k - \boldsymbol{w}_{i,j}\|_1 = \sum_{n=1}^{N} |x_{k,n} - w_{i,j,n}|$. Simultaneously, the results of the subtraction $(x_{k,n} - w_{i,j,n})$ are stored in registers, and the values in the registers are fed into the update unit to compute updated reference vectors.

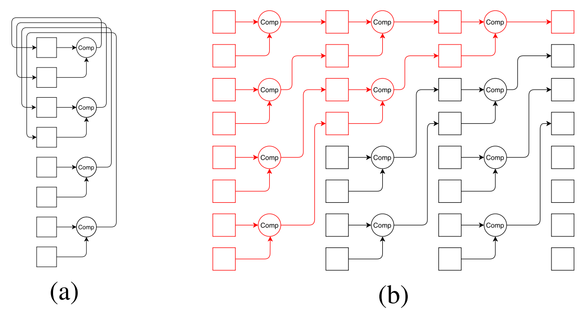


FIGURE 6. Hardware of the winner selection unit.

Fig. 6-(a) shows the winner selection unit. Comp in the figure is a comparator that receives the L1 norms with the indices from two norm computation units and returns the smaller L1 norm with its indices. The winner selection unit includes $I \times J$ registers, illustrated as rectangles in the figure, and $(I \times J) \div 2$ comparators. The outputs of the comparators are connected to the first half of the registers. To select a winner, the registers are first initialized with outputs (L1 norms) of the norm computation units and their indices. Then, computations, using the comparators, are repeated $\log_2(I \times J)$ times. Note that the number of neurons $I \times J$ must be in the power of two. Otherwise, dummy registers initialized with the maximum values of the variables and comparators are added. Fig. 6-(b) shows the winner selection unit unfolded through time. As indicated by red in the figure, the smallest L1 norm and its indices are stored in the first element of the registers.

Fig. 7 shows the distance computation unit that receives the winner indices (i_c, j_c). First, subtractions of winner indices and own indices, $i_c - i$, $i - i_c$, $j_c - j$, and $j - j_c$ are computed simultaneously. To compute the absolute values $D_i = |i_c - i|$ and $D_j = |j_c - j|$, either $i_c - i$ or $i - i_c$ and either $j_c - j$ or $j - j_c$ are selected based on the comparators. Then, the distance computation unit outputs the distance from the winner neuron D_i and D_j.

Fig. 8 shows the output unit; Sat depicts a processing element for saturation, and Shift depicts a processing element for the bit-shift operation. The output unit receives the distance from the winner D_i and D_j and adds these values. The distance is saturated in a range from 0 to *BIT_WIDTH_H*, which is a bit width of variable $h_{i,j}$. In this study, *BIT_WIDTH_H* is 8. Then, 1.0 is right bit shifted by the saturated value to compute $h_{i,j}$.

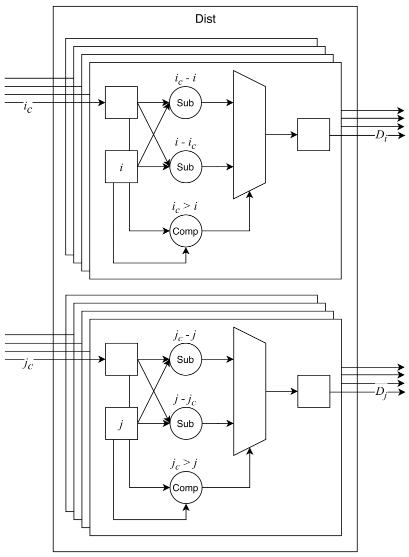


FIGURE 7. Hardware of the distance computation unit.

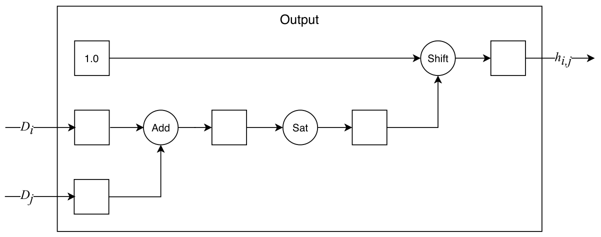


FIGURE 8. Hardware of the output unit.

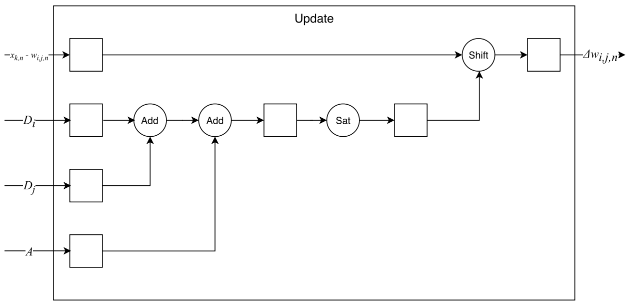


FIGURE 9. Hardware of the update unit of LA.

Fig. 9 shows the update unit of the LA module. The update unit receives the distance D_i, D_j, and a learning coefficient A and adds these values. Then, the added value is saturated in a range from 0 to *BIT_WIDTH_W*, which is a bit width of variable $\boldsymbol{w}_{i,j}$. In this study, *BIT_WIDTH_W* is 8. The update unit also receives $(x_{k,n} - w_{i,j,n})$ from the norm computation unit, and the value is right bit shifted. The shifted value $\Delta w_{i,j,n}$ is fed into the RAM for the reference vector and added to $w_{i,j,n}$.

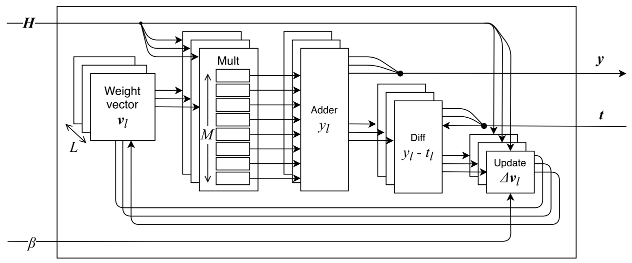


FIGURE 10. **Hardware of the CE module.**

2) CE MODULE

Fig. 10 shows a configuration of the hardware of the CE module. The CE module has L neurons (L is the dimensions of US), and all computations of the neurons are executed in parallel. The CE module includes RAMs for weight vectors $\boldsymbol{v}_l$ and several computation units, such as multiplication operation units (Mult), adder-tree units (Adder), difference computation units (Diff), and update units (Update). The multiplication operation units compute elementwise multiplications of $\boldsymbol{H}$ and $\boldsymbol{v}_l$. Thus, the computation results are $M(= I \times J \times K)$ elements. The computation results are fed into the adder-tree units that add the outputs from the multiplication operation units to compute $y_l = \boldsymbol{H} \cdot \boldsymbol{v}_l^\top$. In the case of the training mode, the computation results of the adder-tree units are fed into the difference computation units that receive y_l and supervised signals t_l and compute the differences $y_l - t_l$. Finally, the differences are fed into the update units to compute the updated weight vectors. The computation results of the update units are stored in the RAMs for the weight vectors.

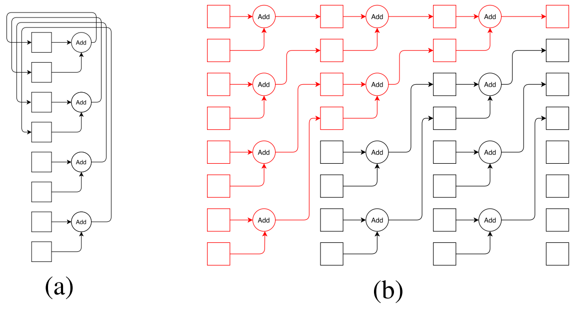


FIGURE 11. **Hardware of the adder-tree unit.**

Fig. 11 shows the adder-tree unit. The adder-tree unit includes $M(= I \times J \times K)$ registers and $M \div 2$ adders. Each adder receives two values and outputs the addition of the two values. The outputs of the adders are connected to the first half of the registers. To compute additional operations of all values stored in the registers, first, the registers are initialized with the outputs of the multiplication operation units. Then, computations using the adders are repeated $\log_2 M$ times. Note that the number of outputs of the multiplication operation units M must be in the power of two; otherwise, dummy registers initialized with zero and dummy adders will be added. Fig. 11-(b) shows the adder-tree unit unfolded through time. As indicated by red in the figure, the results of the additional operation of all values are stored in the first element of the registers.

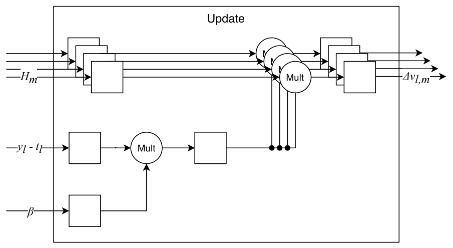


FIGURE 12. **Hardware of the update unit of CE.**

Fig. 12 shows the update unit of the CE module, where Mult in the figure is a multiplier. The multiplier receives two values and outputs the multiplication of the two values. The update unit receives $y_l - t_l$ from the difference computation unit, and β is the learning rate. Those values are fed into a multiplier. The update unit also receives H_m and computes the multiplication operations of H_m and $\beta(y_l - t_l)$. The outputs of the update unit $\Delta v_{l,m}$ are fed into the RAM for the weight vector to update.

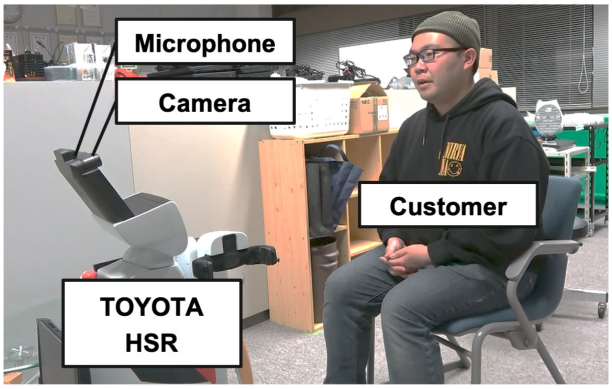


FIGURE 13. **Human-robot interaction in the experiment.**

IV. EXPERIMENTS

A. LEARNING PREFERENCES FROM A FEW HUMAN-ROBOT INTERACTIONS

We verified that the proposed classical conditioning model could learn customers' preferences through a few human-robot interactions. We implemented the model on a home service robot (TOYOTA HSR [7], [35]) for this experiment. The perception parts of the model were implemented via Web Speech API [36] as voice recognition, YOLOv2 [37] as object visual recognition, and simultaneous localization and mapping as place detection. Using a camera and microphone installed on the robot, the model fetched CS and US (Fig. 13). We prepared two types of vectors for each CS, namely, face images (64 × 64 pixel RGB channel) representing customers A and B and two vectors (three-dimensional

TABLE 1. Defined situations.

Situation	Face (CS)	Place (CS)	Ordered object (US)
A	Customer A	Place A	Object A
B	Customer A	Place B	Object B
C	Customer B	Place A	Object B
D	Customer B	Place B	Object C

one-hot vectors) representing places A and B. We also defined four situations (Table 1). Using situation A as an example, whenever customer A is in place A, object A is ordered (showing that the preference in situation A is object A). The model received US and CS from the defined situation and learned the relation between US and CS. First, the model received the US and CS of situation A ten times, after which the situation is changed. Then, similarly, the model received the US and CS of situation B ten times, followed by the same procedure for situations C and D.

We used software of the model for this experiment. In this experiment, the hardware-oriented algorithm was not applied, and all variables were 32-bit floating-point numbers. The activation function in (5) was Softmax function. Thus, the outputs of the CE were probabilities.

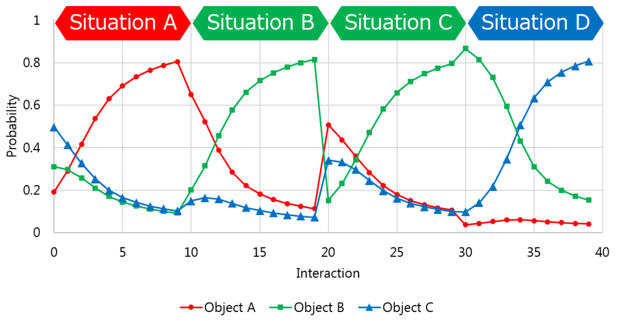


FIGURE 14. Probabilities of object prediction during human-robot interactions.

Fig. 14 shows the probabilities of objects estimated by the model from the situation during human-robot interactions. In each new situation, three or four interactions were required by the model to favor the correct ordered object, demonstrating the capability of the model to adapt to a new situation through only a few human-robot interactions.

Moreover, to verify the generalization performance of the model, we prepared 1,000 face images per customer representing the faces captured from various directions, which were divided into 750 images for training and 250 images for validation. At a human-robot interaction in the training mode, an image was randomly selected from the 750 training images. The image and a vector representing the place were fed into the model. After each interaction, we fed all 250 validation images and place vectors of the situation into the model to verify its accuracy rate with respect to the validation images. Figure 15 shows the accuracy rate of the

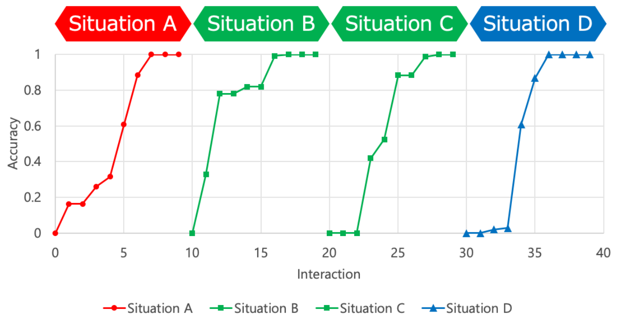


FIGURE 15. Accuracy rate of the classical conditioning model with respect to the validation images during the human-robot interactions.

model during the interactions and indicates that the model adapted all situations within ten human-robot interactions.

B. HARDWARE IMPLEMENTATION

We designed and implemented a dedicated hardware of the classical conditioning model into an FPGA. The target device was an XCZU9EG FPGA [34], and the developing environment was Xilinx SDSoC 2018.3 [38].

We set the hyperparameters of the structure of the model as follows: the number of SOMs in the LA was $K = 2$, the dimension of reference vectors of SOMs was $N = 12,288 (= 64 \times 64 \times 3)$, the size of the SOMs in the LA was $I \times J = 8 \times 8$, and the number of neurons in the CE was $L = 3$.

We designed the hardware using Vivado HLS [39] and described the computations of the model as a hardware function and synthesized the hardware function. We measured the logic utilization of the hardware operated at 150 MHz. Table 2 shows the logic utilization of the hardware; the utilization was less than 100 %. Thus, the proposed hardware of the model could be implemented on the XCZU9EG FPGA. The utilization of the BRAM was the maximum in some types of hardware resources. The worst negative slack and the worst hold slack of the hardware were 0.568 ns and 0.01 ns, respectively.

TABLE 2. Logic utilization of the proposed hardware.

	Used	Available	Utilization [%]
LUT	67,198	274,080	24.52
LUTRAM	1,472	144,000	1.02
FF	48,398	548,160	8.83
BRAM	672	912	73.68
DSP	675	2,520	26.79

Furthermore, we verified the effect of the proposed hardware-oriented algorithm by designing and synthesizing hardware functions using the original algorithm of the model and floating-point numbers. We designed two types of hardware implementations: sequential and parallel implementation. In sequential implementation, hardware computations were sequentially executed. In parallel implementation, hardware computations were executed in parallel. We set the clock frequency as 150 MHz in both implementations.

TABLE 3. Logic utilization and latencies of three types of hardware implementations.

	Sequential (floating-point)	Parallel (floating-point)	Proposed (fixed-point)
LUT	36,174 (13.20%)	1,057,222 (385.73%)	67,198 (24.52%)
LUTRAM	2,047 (1.42%)	484,387 (336.38%)	1,472 (1.02%)
FF	46,517 (8.49%)	401,409 (73.23%)	48,398 (8.83%)
BRAM	784.50 (86.02%)	912 (100.00%)	672 (73.68%)
DSP	329 (13.06%)	4,866 (193.10%)	675 (26.79%)
Latency	62,159,985 [CLK]	95,426 [CLK]	15,395 [CLK]

Table 3 shows the logic utilization and latencies of the computation. The logic utilization and latency of the parallel implementation were estimated values by Xilinx Vivado and SDSoC because the required hardware resources exceeded the resources on the target device and the hardware did not satisfy timing constraints. Therefore, the hardware could not be implemented on the FPGA. In sequential implementation, logic utilization, except LUTRAM and BRAM, was smaller than that for the proposed hardware because a single processing unit was implemented on the hardware and sequentially executed to compute multiple elements of vectors. However, SDSoC estimated the latency of this hardware as 62,159,985 CLK, whereas the latency of the proposed hardware was 15,395 CLK. Thus, sequential implementation was ineffective. In parallel implementation, logic utilization was larger than that of the proposed hardware. The latency of this hardware was estimated as 95,426 CLK, which was also higher than the latency of the proposed hardware.

C. ABILITY OF THE HARDWARE

We executed the proposed hardware on the FPGA. First, we executed a pre-training mode, feeding CS into the LA for several iterations to obtain organized reference vectors of the SOMs. In the pre-training mode, face images used in experiment A were fed into the LA for faces. A three-dimensional vector, whose elements were random values of eight-bit integers, was fed into the LA for places. We set A in (12) as 1. Fig. 16 shows the reference vectors before and after 500 iterations of pre-training. Since we implemented two SOMs in the hardware, two types of reference vectors ($k = 1, 2$) are illustrated in the figure. In Figs. 16-(c) and 16-(d), each reference vector is represented as an RGB color rectangle because the RGB color is also a three-dimensional vector of the eight-bit integer. Figs. 16-(a) and 16-(c) show that the initial reference vectors were zero, and Figs. 16-(b) and 16-(d) show that the SOMs in the LA were self-organized by the pre-training. Note that owing to the differences in the data dimension and distribution, the self-organized maps between the SOMs were significantly different.

Fig. 17 shows the SOM errors between the reference vectors of the winner neurons and data during the pre-training.

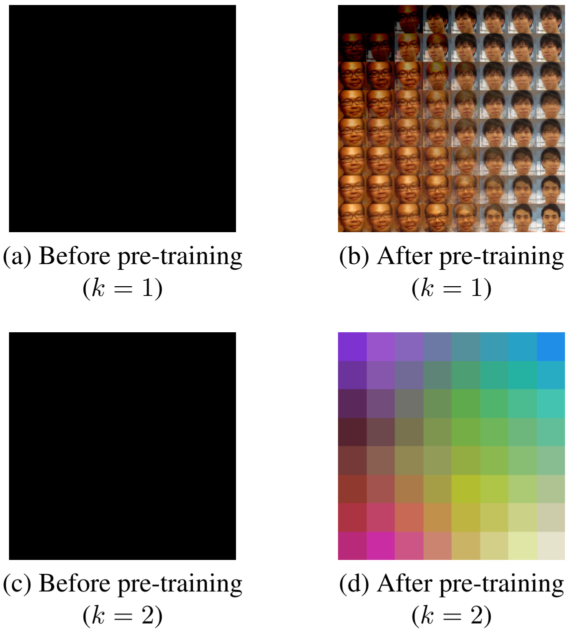

(a) Before pre-training ($k = 1$)
(b) After pre-training ($k = 1$)
(c) Before pre-training ($k = 2$)
(d) After pre-training ($k = 2$)

FIGURE 16. Measured results of FPGA implementation: Reference vectors before and after the pre-training.

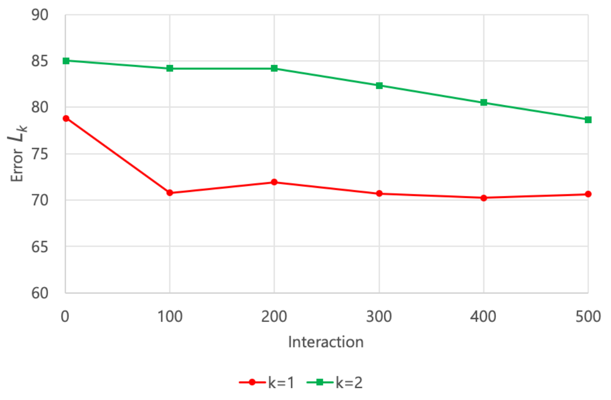


FIGURE 17. Measured results of FPGA implementation: Error of the SOM in LA between the reference vectors of the winner neurons and data during the pre-training.

In the case of the SOM for face, the validation images used in experiment A were used for data to compute the error. In the case of the SOM for place, one-hot vectors that represented places A and B were used for data to compute the error. The SOM error in k-th LA L_k was computed as follows:

$$L_k = \frac{1}{P}\sum_{p=1}^{P}\frac{1}{N}\|\boldsymbol{x}_k(p) - \boldsymbol{w}_{i_c,j_c}\|_1, \tag{14}$$

where $\boldsymbol{x}_k(p)$ is the p-th validation data and P is the total number of validation data. During pre-training, the error decreased and converged. Note that the maximum allowable value of the error is 255 because both input and reference vectors were eight-bit integer values.

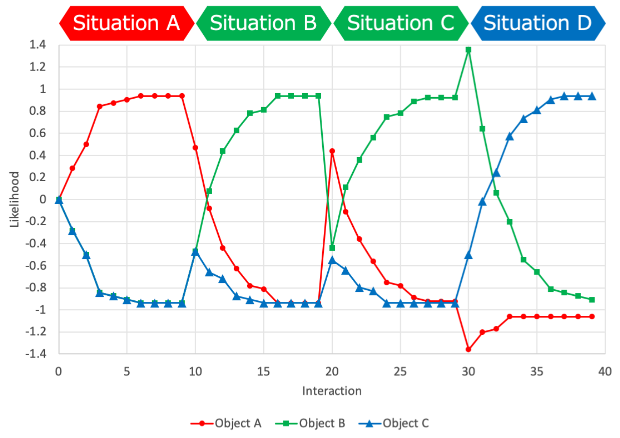


FIGURE 18. Measured results of FPGA implementation: Outputs of the CE during classical conditioning.

After the pre-training, we executed a classical conditioning test on the FPGA, where we simultaneously afforded US and CS. The combinations of US and CS are defined in Table 1. First, the US and CS of situation A were afforded for ten iterations. Then, the US and CS of situation B were also similarly afforded for ten iterations. Similar procedure was followed for situations C and D. We set A in (12) as 4, and β in (9) as 0.1. Fig. 18 shows the outputs of the CE during the classical conditioning test on the FPGA. At the beginning of the test, the output for object A increased, and the other outputs monotonically decreased. Then, the situation was changed at the tenth iteration, and the output for object B increased to adapt to the situation. Similarly, the situation was changed at the 30-th iteration, and the output for object C started increasing. This experimental result showed that even though the hardware was implemented using the simplified hardware-oriented algorithm, the hardware of the model was able to learn the combinations of CS and US (also regarded as the customers' preferences) with a few training iterations.

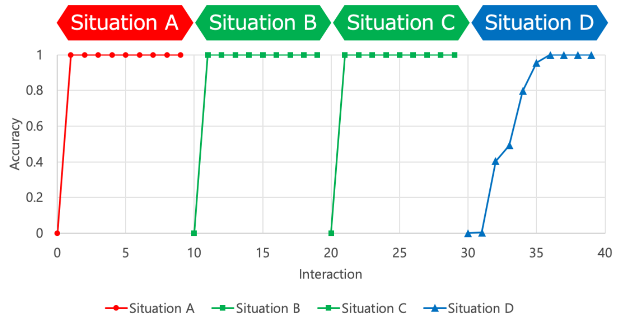


FIGURE 19. Measured results of FPGA implementation: Accuracy rate of the classical conditioning model with respect to the validation images during the human-robot interactions.

Moreover, to verify the generalization performance of the proposed hardware, we used the generalization test corresponding to the test in experiment A. Figure 19 shows the accuracy rate of the model with respect to the validation images during the interactions and indicates that the hardware adapted all situations within ten human robot interactions.

TABLE 4. Execution time of the software.

Implementation @ device	Execution time [s]
C++ @ A53	251.3 m
C++ @ i5	22.0 m
Python + NumPy @ TX2	42.3 m
Python + CuPy @ TX2	18.7 m

TABLE 5. Measured results of FPGA implementation: Execution time of the hardware.

	Execution time [s]
Entire hardware	197.4 μ
Empty hardware	90.7 μ

TABLE 6. Estimated chip powers.

	Power [W]
Entire hardware	5.009
Empty hardware	3.748

D. PERFORMANCE OF THE HARDWARE

We measured the processing speed of the hardware. The XCZU9EG FPGA used in this experiment was a system-on-chip (SoC) FPGA, integrating a processing system (PS) and programmable logic (PL). In this experiment, the proposed hardware was implemented on the PL. Software on the PS passed input data to the PL and received computation results from the PL. We measured the execution times of the training mode from the onset of data input until the end of the receipt of computation results. The operation of the hardware was repeated 500 times. For comparison between the processing speed of the hardware and that of software, we implemented software and executed it on PS in FPGA (Arm Cortex-A53), Intel Core i5 4670K, and a CPU in NVIDIA Jetson TX2 (Nvidia Denver and Arm Cortex-A57). For Arm Cortex-A53 and Intel Core i5, we implemented the classical conditioning model using C++. For the CPU in Jetson TX2, we implemented it using Python and NumPy [40]. Moreover, we replaced the model computations by NumPy with those by CuPy [41] to execute them on a GPU in Jetson TX2. We set a mode of Jetson TX2 as Max-N where all cores in Jetson TX2 were available. Table 4 shows the average execution times of the software, and Table 5 shows the average execution times of the hardware. As shown in these tables, the hardware was faster than the software even in the GPU. The execution time of the hardware included time for data transfer. We measured the execution time of only the logic of the model, by implementing a hardware function that computed nothing but received dummy CS and returned dummy CR. We measured the execution time of the empty hardware, which was measured when the clock frequency for data transfer was operated at 150 MHz.

TABLE 7. Comparison of previous studies on the amygdala models for robot applications.

	EMA [28]	SAFEL [29], [30]	This work
Structure of model	Includes LA and CE (Single SOM and FCNN)	Excludes LA and CE (FCNN)	Includes LA, CE, and perception parts (Multiple SOMs and FCNN)
Hardware implementation	FPGA	Not implemented on hardware	FPGA

We measured the power consumption of the hardware (both entire hardware and empty hardware). Table 6 shows the chip powers estimated by Vivado. As shown in the table, the FPGA, including the proposed hardware, consumed 5.009 W.

V. DISCUSSION

A. MODEL

Table 7 shows the comparison of previous studies on the amygdala models for robot applications. Compared to the EMA [28], the proposed model can be applied to more complex tasks. The EMA fetches single information from the environment for CS; therefore, it cannot deal with combinations of multiple factors in the environment. Furthermore, the EMA cannot be used if factors that are related to US are unknown. However, our model fetches significant information from the environment for CS. Therefore, our model can deal with combinations of factors in the environment, meaning that our model can be used even if we do not know which factors are related to US.

Compared to the amygdala model in SAFEL [29], [30], our proposed model is more useful for robot applications because it includes the functions of LA, CE, and perception parts. By introducing the LA model, our model can accept unknown stimuli. For example, the model beforehand learns situations where a young male customer and an old female customer appear. When a young female customer appears, a neuron located between a neuron for the young male and a neuron for the old female, will react. Supposing that a relation exists between age, gender, and preferences, our model can estimate the preference of the unknown customer.

B. HARDWARE

Using the proposed hardware-oriented algorithm, the hardware of the LA module does not require any multipliers. If we implement the classical conditioning model with the same structure as the model in the experiment, by not using the hardware-oriented algorithm, the LA module requires 6,291,840 multiplication operations: (1) requires 1,572,864 multiplication operations, (2) requires 128 multiplication operations, (3) requires 256 multiplication operations, and (6) requires 4,718,592 multiplication operations. Therefore, the proposed algorithm is effective for reducing hardware resources (Table 3). Furthermore, although the hardware was implemented using the simplified hardware-oriented algorithm, the hardware of the model could self-organize the reference vectors of the LA module (Fig. 16) and learn the combinations of US and CS with a few training iterations (Fig. 18).

Although Xilinx Vitis AI [42] can implement DL framework-based models, such as Keras, TensorFlow, and Caffe, into FPGAs replacing the floating-point variables with fixed-point variables, our proposed hardware-oriented algorithm is more effective in implementing the model into FPGAs because replacing multiply operations and exponential functions with bit-shift operations drastically reduces the hardware resources. Vitis AI cannot achieve such an effective algorithm.

Since we used SOMs with 8×8 neurons in this study, the use of eight-bit fixed-point variables is sufficient to compute SOM algorithm and solve the task. Additionally, even if SOM hardware with more neurons is required in some tasks, we can easily change the bit width of all variables to adapt to other tasks because the hardware is parameterized.

Tables 4 and 5 show that the proposed hardware operated at 150 MHz is 1,273 times, 111.4 times, 214.3 times, and 94.73 times faster than the software executed on Arm Cortex-A53, Intel Core i5 4670K, CPU of Jetson TX2, and GPU of Jetson TX2, respectively. These findings reveal that the computation of the proposed hardware is more effective than that of the software, even on the GPU. Moreover, SDSoC estimated a latency of the training mode on the hardware as 15,395 CLK. Thus, the ideal execution time of the hardware without data transfer is 6.67 ns/CLK $\times$ 15,395 CLK $=$ 102.6 μs. In addition, Table 5 also shows that the execution time of the logic is 197.4 μs - 90.7 μs $=$ 106.7 μs if data transfer time is ignored.

VI. CONCLUSION

In this study, we proposed a novel amygdala-inspired classical conditioning model that learns preferences after a few repeated human-robot interactions. We implemented the model on an FPGA by proposing a hardware-oriented algorithm that reduces multiplication operations more than 6,000,000 times. Using the hardware-oriented algorithm, we designed a dedicated hardware for the model on an XCZU9EG FPGA that is more than 1,200 times faster than software on an Arm Cortex-A53 CPU. The hardware can learn preferences through a few training iterations, even though the simplified algorithm is used for the implementation.

In future studies, we aim to integrate the amygdala-inspired model with other brain-inspired models. The proposed model has been integrated with the perception parts, regarded as a neocortex model. Furthermore, by introducing hippocampus and entorhinal cortex models and integrating them with the model, functions for episodic memory using representation of place and time can be realized.

ACKNOWLEDGMENT

This article is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

REFERENCES

[1] L. Iocchi, D. Holz, J. Ruiz-del-Solar, K. Sugiura, and T. van der Zant, ''RoboCuphome: Analysis and results of evolving competitions for domestic and service robots,'' *Artif. Intell.*, vol. 229, pp. 258–281, Dec. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370215001174

[2] G. E. Hinton, S. Osindero, and Y.-W. Teh, ''A fast learning algorithm for deep belief nets,'' *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, ''Gradient-based learning applied to document recognition,'' *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ''ImageNet classification with deep convolutional neural networks,'' in *Proc. NIPS*, vol. 1, 2012, pp. 1097–1105.

[5] J. Redmon and A. Angelova, ''Real-time grasp detection using convolutional neural networks,'' in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 1316–1322.

[6] S. Hori, Y. Ishida, Y. Kiyama, Y. Tanaka, Y. Kuroda, M. Hisano, Y. Imamura, T. Himaki, Y. Yoshimoto, Y. Aratani, K. Hashimoto, G. Iwamoto, H. Fujita, T. Morie, and H. Tamukoh, ''Hibikino-MusashiHome 2017 team description paper,'' 2017, *arXiv:1711.05457*. [Online]. Available: http://arxiv.org/abs/1711.05457

[7] T. Ono, Y. Tanaka, Y. Ishida, Y. Abe, K. Kamanaru, D. Kamimura, K. Nakamura, Y. Nishimura, S. Tokuno, Y. Mii, M. Yamauchi, Y. Uemura, T. Hashimoto, Y. Nakamura, I. Uchino, D. Kanaoka, T. Hanyu, K. Tsukamoto, T. Morie, and H. Tamukoh, ''Hibikino-MusashiHome 2020 team description paper,'' 2020, *arXiv:2005.14451*. [Online]. Available: http://arxiv.org/abs/2005.14451

[8] Y. Ishida and H. Tamukoh, ''Semi-automatic dataset generation for object detection and recognition and its evaluation on domestic service robots,'' *J. Robot. Mechtron.*, vol. 32, no. 1, pp. 245–253, Feb. 2020.

[9] L. Weiskrantz, ''Behavioral changes associated with ablation of the amygdaloid complex in monkeys.,'' *J. Comparative Physiol. Psychol.*, vol. 49, no. 4, pp. 381–391, 1956.

[10] J. E. LeDoux, *The Emotional Brain: The Mysterious Underpinnings of Emotional Life*. New York, NY, USA: Simon and Schuster, 1996.

[11] J. P. Aggleton, *The Amygdala: A Functional Analysis*. Oxford, U.K.: Oxford Univ. Press, 2000.

[12] E. T. Rolls, *Emotion Explained*. Oxford, U.K.: Oxford Univ. Press, 2005.

[13] E. A. Phelps and J. E. LeDoux, ''Contributions of the amygdala to emotion processing: From animal models to human behavior,'' *Neuron*, vol. 48, no. 2, pp. 175–187, Oct. 2005.

[14] H. Nakahara, H. Yonekawa, T. Fujii, M. Shimoda, and S. Sato, ''GUINNESS: A GUI based binarized deep neural network framework for software programmers,'' *IEICE Trans. Inf. Syst.*, vol. E102.D, no. 5, pp. 1003–1011, 2019.

[15] I. P. Pavlov, *Conditioned Reflexes: An Investigation of the Physiological Activity of the Cerebral Cortex*. Oxford, U.K.: Oxford Univ. Press, 1927.

[16] J. LeDoux, C. Farb, and D. Ruggiero, ''Topographic organization of neurons in the acoustic thalamus that project to the amygdala,'' *J. Neurosci.*, vol. 10, no. 4, pp. 1043–1054, Apr. 1990.

[17] E. Tsvetkov, W. A. Carlezon, F. M. Benes, E. R. Kandel, and V. Y. Bolshakov, ''Fear conditioning occludes LTP-induced presynaptic enhancement of synaptic transmission in the cortical pathway to the lateral amygdala,'' *Neuron*, vol. 34, no. 2, pp. 289–300, Apr. 2002.

[18] J. L. Armony, D. Servan-Schreiber, J. D. Cohen, and J. E. LeDoux, ''An anatomically constrained neural network model of fear conditioning.,'' *Behav. Neurosci.*, vol. 109, no. 2, pp. 246–257, 1995.

[19] J. Morén and C. Balkenius, ''A computational model of emotional learning in the amygdala,'' in *Proc. Animals Animats, 6th Int. Conf. Simulation Adapt. Behaviour*, 2000, pp. 115–124.

[20] E. M. Izhikevich, ''Solving the distal reward problem through linkage of STDP and dopamine signaling,'' *Cerebral Cortex*, vol. 17, no. 10, pp. 2443–2452, Oct. 2007.

[21] Y. Li, K. Nakae, S. Ishii, and H. Naoki, ''Uncertainty-dependent extinction of fear memory in an amygdala-mPFC neural circuit model,'' *PLOS Comput. Biol.*, vol. 12, no. 9, Sep. 2016, Art. no. e1005099.

[22] E. T. Rolls, ''A theory of emotion and consciousness, and its application to understanding the neural basis of emotion,'' in *The Cognitive Neurosciences*, 1995, pp. 1091–1106.

[23] E. M. Izhikevich, ''Simple model of spiking neurons,'' *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.

[24] C. L. Hull, *Principles of Behavior: An Introduction to Behavior Theory*. New York, NY, USA: D. Appleton-Century, 1943.

[25] D. W. Leonard, ''Partial reinforcement effects in classical aversive conditioning in rabbits and human beings.,'' *J. Comparative Physiol. Psychol.*, vol. 88, no. 2, pp. 596–608, 1975.

[26] R. A. Rescorla, ''Partial reinforcement reduces the associative change produced by nonreinforcement,'' *J. Experim. Psychol. Animal Behav. Processes*, vol. 25, no. 4, p. 403–414, 1999.

[27] N. J. Mackintosh, *The Psychology of Animal Learning*. New York, NY, USA: Academic, 1974.

[28] S. Sonoh, K. Horio, S. Aou, and T. Yamakawa, ''An emotional expression model inspired by the amygdala,'' *Int. J. Innov. Comput., Inf. Control*, vol. 5, pp. 1147–1160, May 2009.

[29] C. Rizzi Raymundo and C. G. Johnson, ''An artificial synaptic plasticity mechanism for classical conditioning with neural networks,'' in *Advances in Neural Networks—ISNN*, vol. 8866. Cham, Switzerland: Springer, 2014, pp. 213–221.

[30] C. Rizzi Raymundo, C. G. Johnson, and P. A. Vargas, ''Fear learning for flexible decision making in robocup: A discussion,'' in *Robot World Cup*. Cham, Switzerland: Springer, 2017.

[31] T. Kohonen, ''Self-organized formation of topologically correct feature maps,'' *Biol. Cybern.*, vol. 43, no. 1, pp. 59–69, 1982, doi: 10.1007/BF00337288.

[32] Y. Tanaka and H. Tamukoh, ''Hardware implementation of brain-inspired amygdala model,'' in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[33] T. Shinozaki, ''Biologically inspired feedforward supervised learning for deep self-organizing map networks,'' in *Proc. NIPS Workshop Represent. Learn. Artif. Biol. Neural Netw. (MLINI)*, 2016, pp. 1–7.

[34] *Zynq UltraScale+ MPSoC*. Accessed: Oct. 26, 2020. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html

[35] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase, ''Development of human support robot as the research platform of a domestic mobile manipulator,'' *ROBOMECH J.*, vol. 6, no. 1, Dec. 2019.

[36] G. Shires and H. Wennborg, ''Web speech API specification,'' W3C Community Group, Final Report, 2012.

[37] J. Redmon and A. Farhadi, ''YOLO9000: Better, faster, stronger,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6517–6525.

[38] *SDSoC Development Environment*. Accessed: Oct. 26, 2020. [Online]. Available: https://www.xilinx.com/products/design-tools/software-zone/sdsoc.html

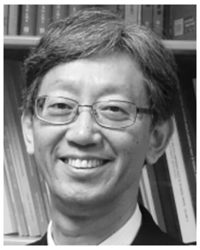
[39] *Vivado High-Level Synthesis*. Accessed: Oct. 26, 2020. [Online]. Available: https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html

[40] T. E. Oliphant, *A Guide to NumPy*, vol. 1. New York, NY, USA: Trelgol Publishing, 2006.

[41] R. Okuta, Y. Unno, D. Nishino, S. Hido, and C. Loomis, ''Cupy: A numpy-compatible library for nvidia gpu calculations,'' in *Proc. ML Syst. Workshop NIPS*, 2017, p. 151.

[42] *Vitis AI*. Accessed: Oct. 26, 2020. [Online]. Available: https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html

YUICHIRO TANAKA (Graduate Student Member, IEEE) received the B.E. and M.E. degrees from the Kyushu Institute of Technology, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree with the Graduate School of Life Science and Systems Engineering. Since 2019, he has also been a Research Fellow of the Japan Society for the Promotion of Science (JSPS). He is also a Student Member of IEICE.



TAKASHI MORIE (Member, IEEE) received the B.S. and M.S. degrees in physics from Osaka University, Osaka, Japan, in 1979 and 1981, respectively, and the Dr.Eng. degree from Hokkaido University, Sapporo, Japan, in 1996. From 1981 to 1997, he was a member of the Research Staff with Nippon Telegraph and Telephone Corporation (NTT). From 1997 to 2002, he was an Associate Professor with the Department of Electrical Engineering, Hiroshima University, Higashihiroshima, Japan. Since 2002, he has been a Professor with the Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, Kitakyushu, Japan. His research interests include VLSI implementation of neural networks and new functional nanodevices.

HAKARU TAMUKOH (Member, IEEE) received the B.Eng. degree from Miyazaki University, Japan, in 2001, and the M.Eng. and Ph.D. degrees from the Kyushu Institute of Technology, Japan, in 2003 and 2006, respectively. He was a Postdoctoral Research Fellow of the 21st Century Center of Excellent Program, Kyushu Institute of Technology, from April 2006 to September 2007. He was an Assistant Professor of the Tokyo University of Agriculture and Technology from October 2007 to January 2013. He is currently an Associate Professor with the Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology. His research interests include hardware/software complex systems, digital hardware design, neural networks, soft-computing, and home service robots. He is also a member of IEICE, SOFT, JNNS, JSAI, and RSJ.