Quasi-Mapping and Satisfying IoT Availability with a Penalty-Based Algorithm

Abstract: The Internet of things and medical things (IoT) and (IoMT) technologies have been deployed to simplify humanity’s life, which the complexity of communications between their layers was increased by rising joining the applications to IoT and IoMT-based infrastructures. The issue is challenging for decision-making and the quality of service where some researchers addressed the reward-based methods to tackle the problems by employing reinforcement learning (RL) algorithms and deep neural networks (DNNs). Nevertheless, satisfying its availability remains a challenge for the quality of service due to the lack of imposing a penalty to the defective devices after detecting faults. This paper proposes a quasi-mapping method to transfer the roles of sensors and services onto a neural network’s nodes to satisfy IoT-based applications’ availability using a penalty-backwarding approach into the NN’s weights and prunes weak neurons and synaptic weights (SWs). We reward the sensors and fog services, and the connection weights between them when are covered the defective nodes’ output. Additionally, this work provides a decision-making approach to dedicate the suitable service to the requester using employing a threshold value in the NN’s output layer according to the application. By providing an intelligent algorithm, the study decides to provide a service based on its availability and updating initial information, including faulty devices and new joined components. The observations and results prove decision-making accuracy for different IoT-based applications by approximately 95.8–97% without imposing the cost. The study reduces energy consumption and delay by approximately 64.71% and 47.4% compared without using neural networks besides creating service availability. This idea affects deploying IoT infrastructures to decision-making about providing appropriate services in critical situations because of removing defective devices and joining new components by imposing penalties and rewards by the designer, respectively.

Keywords: quasi-mapping; Internet of things (IoT); neural network (NN); availability; penalty; decision-making; pruning

1. Introduction

By increasing smart devices and growing up human requirements to access the network and share their records with the service providers, IoT and IoMT technologies had deployed to respond to the requirements. Due to the dynamic feature of IoT, the service providers faced challenges caused by increasing smart devices, which leads to the loss
of quality as the highlighted property in the Internet of Things and Medical Things [1]. According to the rising number of joining applications to IoT infrastructures, the service providers (SPs) faced many different or similar requests to support them by dedicating an appropriate service. Decision-making has a highlighted role to assign a service provider to the requesters with high quality, in which servers can tackle similar requests. In facing the condition, some benchmarks have to determine for deciding between SPs. Many researchers addressed the Internet of Things and Medical Things challenges, including protecting private records, satisfying security, authentication process, and supplies management [2–8]. By defining reliability and availability concepts, some case studies first analyzed the relationship between them and the quality of service for protecting privacy and security [9,10]. Additionally, the researchers defined the trust concept to validate an IoT-application for proving services to a requester, which were guaranteed to successfully and quality [11,12]. The trust defines a flexible concept with supporting different problems that can lead to starvation for some SPs when lacking cover for it. To provide a service, SPs can decide based on various features, including minimal cost, high performance, availability, reliability, and trust [13,14]. According to the importance of protecting private information and records, the researchers proposed decision-making approaches by assuming satisfying IoT applications’ security. The machine learning (ML) algorithms have employed in decision-making, security-predicting, and service-recommending in IoT and IoMT-based applications that were included DNNs, convolution neural networks (CNNs), recurrent neural networks (RNNs), and reinforcement learning (RL) approaches [14–17]. The studies almost utilized the reward-forwarding methods and reinforcement learning to decide about providing a service to the requester after training a DNN with the agents’ background information [18,19]. The reward-based approaches can deteriorate or reduce the confidence and quality of service, according to the lack of the related penalties to the different sensors and server nodes in the Internet of Things. The researchers predicted the malicious attacks and provided a service with forest and decision trees besides employing unsupervised learning algorithms such as RL, which the approaches are classified to the supervised learning [20–23].

According to the review of the neural network’s features, the researchers employed NNs and deep learning accelerators in edge and fog layers against the challenges caused by deploying IoT and IoMT with many nodes that have a relationship to each other [24–28].

The upper mentioned studies demonstrate IoT problems and effective approaches to tackling them that are included:

- Deploying the Internet of Things based on increasing IoT infrastructure applications
- Rising the different joined applications to IoT and its sensors and service provider’s nodes
- Increasing the complexity of the relationship between IoT’s nodes in different layers and their computational operations
- Decision-making to provide a service in facing a huge number of different requests or similar
- The neural networks’ feature and their efficiency in different IoT-applications

After investigating the main problems of deploying IoT and the neural network’s features, we propose a quasi-mapping and availability-aware cloud service algorithm (QAC) to decide about dedicating service to requesters based on a penalty-backwarding approach. The study utilizes an NN’s structure to describe the relationship between IoT nodes in different layers and maps them onto the neural network’s neurons. QAC fines the related weights to the nodes, which are provided a low-quality service or detected as the invisible nodes by monitoring output neuron’ feedback. After diagnosing the weak weights and neurons, the algorithm prunes them and assessments the efficiency of the IoT-application. The pruning weak weights and neurons have a highlighted role to improve the performance and cost of a neural network and also are affected decision-making about an SP due to the loss of the faulty nodes [29]. QAC supports the joined sensors, server nodes, and similar applications to the analysis process for assigning a service provider to a requester such as a relationship between the patients with the emergency conditions.
and nearest available hospitals and medical centers. Before applying other characteristics and features of CAS and the proposed idea, we present the objectives of providing the quasi-mapping the IoT-based applications nodes onto the partial neural networks neurons to clarify its efficiency as follows:

- Decision-making based on assessing the service’s availability by comparing its value with a threshold.
- Online training the neural networks based on updating initial information (Detecting defective devices and new joining components or applications).
- Imposing the penalty into the related weights to the detected faulty nodes.
- Imposing the reward into the related weights to the joined new components to the application or replacing defective nodes with them.
- Pruning the weak weights and neurons and evaluating the service availability after removing them from the IoT-based application.
- Pruning role in improving energy efficiency and latency besides supporting service availability.

By dedicating an availability-aware cloud service instead of a trust-aware approach, we define a flexible condition in facing the critical situation when is requested service with minimal features. A trust-aware service provider creates some restitutions against the critical situations, according to complete losing the identified not-trust servers. The proposed algorithm dedicates an SP to the requester using online training the neural networks with the updated information, including the backwarding the penalties, pruning weak neurons and weights, and the joined new components. We also describe the quasi-mapping IoT nodes on NN’s neurons, penalty-backwarding, availability, and neurons’ operations by the equations. Figure 1 shows following the steps of QAC and the idea for decision-making about assigning a service provider to the requester.

**Figure 1.** The work’s steps.

The existing concepts demonstrate that the study’s main contribution and its novelty include:

- Quasi-mapping the IoT-based application nodes onto the neural networks’ neurons.
- Providing an intelligent availability-aware algorithm to decision-making about dedicating service providers to the requester.
- Presenting the equations to describe quasi-mapping, availability, neuron operation, and penalty-backwarding.
• Providing a penalty-backwarding approach to fines the related weight to the faulty or invisible nodes for pruning them after identifying as the weak weights and neurons

We describe different Internet of Things applications and map their nodes on the neural networks to analyze QAC efficiency in decision-making with high accuracy. After defining an application, we collect a dataset based on the described sensors and their valid values to import them into the simulated nodes and estimate energy consumption and communication delay. This work utilizes the different rates of fault injection and the nodes’ availability assessment to determine a penalty for the identified failure and invisible nodes. By estimating decision-making accuracy and comparing delay and energy before and after performing QAC, the study proves the proposed algorithm and method efficiency to dedicate a service provider to a requester. In comparing other related works, we decide about providing a service to a requester based on analyzing its availability after fining and rewarding the related weights to the identified faulty nodes. The feature has a highlighted role in decision-making when facing critical situations (such as facing accidents and many emergency requests) and timely dedicating the SPs to requesters compared to the other related works with prioritizing trust and security. By online training partial neural networks with the updated initial information, the service provider availability is estimated due to the pruned weak weights and neurons and joined new components or applications. Therefore, we provide a service to a requester based on evaluating all nodes' availability of the application by imposing penalties and rewards using the backwarding method. The issue leads to the prune weak output neurons (as the cloud services) or re-joined the pruned service to IoT infrastructure after rewarding the related weights to the replaced faulty nodes by the new components, which have an impressive effect to improve providing service with recovering the SPs. The study creates an opportunity to re-join the stopped applications by event-based assessing their availability via alerting about connecting new devices instead of the defective and pruned neurons (IoT nodes) using the forwarding method. The service providers’ availability depends on the applications (as an example the healthcare-based applications have higher sensitivity than other applications), which the work defines a threshold value based on their susceptibility. The characteristics create flexibility and scalability for IoT infrastructures developers to provide a suitable service to a requester, which is highlighted in facing critical situations.

We address the purposes to implement the idea in the next sections that are organized as follows: Section 2 reviews the Internet of Things challenges and the presented method tackle them and the related studies to decision-making; We explain the method of collecting a dataset, the quasi-mapping, and an availability-aware cloud service algorithm in Section 3; The observations and simulation results are assessed in Sections 4 and 5 concludes the mentioned concepts of the idea.

2. Related Work

The related work addresses reviewing the different case studies, which have focused on the IoT and IoMT challenges and have proposed the methods to tackle them. This section assesses their efficiency in improving the Internet of Things performance in different aspects, including security, decision-making, trust, availability, and dependability. Additionally, we review the studies, which have employed neural networks and ML algorithms to follow their purposes and have demonstrated the effective role of NNs in facing complicated structures and communications.

Security was defined as the main challenge of the Internet of Things by sharing private information and records to analyze them and provide the related services to the requesters with high quality. Lv et al. [30] reviewed the security problems in the smart city as an IoT-infrastructure, and proposed a new backoff algorithm, and employed a long-range approach to wireless communications for protecting private information and satisfying security with minimal cost. Tawalbeh et al. [5] proposed a new structure for classifying the Internet of Things levels to support its security in edge and cloud layers by defining lower nodes via Amazon Web Service. Some case studies defined a model based
on ML algorithms to predict the attacks and break up the authentication protocols that were endangered the information security in IoT [7,31]. The works utilized the different machine learning algorithms to learn and face them, including artificial NN, DNN, and RNN. By employing the neural network and RL, the studies could better protect the privacy and satisfy the security with learning different passive and active attacks, malware, and malicious information in authentication protocols, and predict them in facing new cases. Ahmed et al. [32] presented a comprehensive systematic lecture review to analyze the highlighted role of machine learning algorithms to apply the security information in the Internet of Things. This work classified the security into three main categories of network, physical, and application attacks, which review demonstrated an impressive effect of multi-layer perceptron to predict and tackle them.

Some researchers addressed trust as a challenge of the Internet of Things because of the increasing complexity of communication between nodes in different levels and computational operations. Trust was almost defined to decision about providing a service and support the security in IoT using detecting and predicting the attacks, which the concept can lead to removing an application by identifying losing its dependability [33]. Altaf et al. [34] followed the issue and provided a trust-based method to detect the malicious components and communications between them in a smart city as an IoT-infrastructure. The study mitigated the service-oriented attacks by defining the trust concept and proving a trust-based service to the requesters after identifying the vulnerabilities in communication between nodes of a smart city. A trust-based method had a highlighted role in service-providing electronic-health applications, which can help IoT-based infrastructure developers to present a confident structure and protect the patient’s privacy records [35]. To provide a service in risky situations, the trust-based methods affect in improving IoT efficiency because of decision-making about the issue after analyzing and predicting the malicious communications and attacks.

Decision-making is challenging for IoT and IoMT because of the increasing number of the joined applications to the infrastructures, which some case studies followed the problem to present the solutions against it. The effective factors in decision-making were addressed as a purpose to provide a method to decide about assigning service to a requester, including timely provide a service, risky situation, cautiously provide a service, and scalability of providing services with increasing SPs and requesters [36]. Pappas et al. [36] focused on a special IoT-application to decision-making about providing a service based on the characteristics of service requesters and providers, which their method assigned service to a requester by defining the effective parameters. Some researchers tried to propose a decision-making method that was supported security besides dedicating service to a requester [37]. Sharma et al. [37] addressed the challenge of decision-making in heterogeneous IoT components and provided the security model to the dependable data transferring and deciding about dedicating a service. The study utilized the learning-based approaches and neural networks for intelligent detecting the anomaly models and malicious devices. The machine learning algorithms and Fuzzy solutions and rules had an impressive effect in following the decision-making, forecasting, detecting, and predicting purposes, which some case studies employed their features to provide a comprehensive analytical Fuzzy-based model to decide about dedicating a service for a special application such as economy’s decisions in an IoT-infrastructure [38,39]. Linear Diophantine and Algebraic models simplified the computations of Fuzzy-based systems, which can affect improving decision-making about the uncertain conditions in IoT and IoMT applications [40,41]. Riaz et al. [42] utilized neutrosophic logics (truth, falsehood, and indeterminacy) as a Fuzzy-based approach to map the features of the patients with multiple personality disorder disease for predicting the treatment duration. Additionally, the researchers proposed a bipolar fuzzy-based model to map the initial characteristics of the disease to correct diagnosis and predict a suitable treatment method for the patients [43].

Resource allocation is challenging for proving the services to the requesters in different IoT applications when facing an SP with many requests. Therefore, some case studies
aimed to provide the approaches to tackle the problem by presenting different algorithms such as whale optimization that was reduced total runtime and cost besides allocating a service to the requester [44]. Li et al. [45] proposed balancing flow in edge computing and an intelligent algorithm to rapid adaptive offloading to reduce the complexity of the computing and data processing at cloud level and fairness allocate the resources besides employing DNNs to fast deciding. The defective nodes led to the increasing energy consumption in the e-health systems as the popular applications of IoT, which the researchers tackled the problem and fairness allocated the health resources by detecting their routes and bypassing them [46–48].

We review the mentioned related works by presenting Table 1 and comparing their weaknesses and advantages [49].

Table 1. Reviewing the existing works and comparing their characteristics [49].

| Case Study     | Method                                      | Advantage                                           | Weakness                                           |
|----------------|---------------------------------------------|-----------------------------------------------------|----------------------------------------------------|
| Lv et al. [30] | -Security protocols                         | -Minimum cost                                       | -Only secure-based service provider               |
|                | -Backoff algorithm                          | -Satisfying IoT security                            | -Lack of decision-making                          |
|                | -Long-range approach                        | -Focusing smart city                                |                                                    |
| Tawalbeh et al. [5] | -Security protocols                        | -Satisfying IoT security                            | -Only secure-based service provider               |
|                | -New structure for IoT levels               | -Focusing on edge computing                         | -Lack of decision-making                          |
|                |                                             | -Reducing the complexity of data processing at the cloud level |                                                    |
| Zhang et al. [7] | -Detecting cyber-attacks                    | -Timely detecting attacks                           | -Only secure-based service provider               |
|                | -Machine learning algorithm                 | -Predicting attacks                                 | -Lack of decision-making                          |
|                | -Authentication protocols                   | -Satisfying IoT security                            |                                                    |
| Aversano et al. [31] | -Detecting cyber-attacks                    | -Timely detecting attacks                           | -Only secure-based service provider               |
|                | -Machine learning algorithm                 | -Predicting attacks                                 | -Lack of decision-making                          |
|                | -Authentication protocols                   | -Satisfying IoT security                            |                                                    |
| Ahmed et al. [32] | -Security protocols                         | -Detecting attacks                                  | -Only secure-based service provider               |
|                | -Machine learning algorithm                 | -Predicting attacks                                 | -Lack of decision-making                          |
|                | -Multi-layer perceptron                     | -Satisfying IoT security                            |                                                    |
| Verma et al. [33] | -Trust-based method                         | -Detecting attacks                                  | -Only trust-based service provider                |
|                | -Dependability                              | -Predicting attacks                                 | -Low flexibility for decision-making              |
|                | -Decision-making                            | -Satisfying IoT trust                               |                                                    |
| Altaf et al. [34] | -Trust-based method                         | -Detecting attacks                                  | -Only trust-based service provider                |
|                | -Detecting malicious components             | -Predicting attacks                                 | -Low flexibility for decision-making              |
|                | -Service-oriented attacks                   | -Satisfying IoT trust in smart city                 |                                                    |
| Arfi et al. [35] | -Trust-based method                         | -Detecting attacks                                  | -Only trust-based service provider                |
|                | -Detecting malicious components             | -Predicting attacks                                 | -Low flexibility for decision-making              |
|                | -Electronic-health applications             | -Satisfying IoT security                            |                                                    |
| Pappas et al. [36] | -Decision-making                           | -Decision-making based on the applications’ characteristics | Increasing complexity of decision-making |
|                | -Defining the applications’ characteristics |                                                     | Low flexibility for decision-making               |
| Sharma et al. [37] | -Detecting anomaly models                   | -Intelligent-detecting                             | -Increasing complexity of decision-making         |
|                | -Detecting malicious components             | -Intelligent-predicting                            | -Low flexibility for decision-making              |
|                | -Neural network                             | -Decision-making in heterogeneous IoT components    |                                                    |
|                | -Learning-based approach                    |                                                     |                                                    |
| Cui et al. [38] | -Pythagorean fuzzy                          | -Identifying IoT adoption barriers                  | -Lack of decision-making                          |
By reviewing the different approaches to face the Internet of Things challenges, the researchers almost followed the security and the related concepts that have helped support the issue, including authentication protocols, trust-based systems, availability, and dependability. The case studies proved the challenge of decision-making in IoT by rising to join new applications to the infrastructure and increasing the complexity of communications between its nodes in different levels. Some researchers proposed the trust-based methods to decision-making that were led to decide about dedicating a secure service by evaluating its confidently. The feature can create some limitations for assigning service to a

| Case Study     | Method                                      | Advantage                                                      | Weakness                                           |
|----------------|---------------------------------------------|----------------------------------------------------------------|---------------------------------------------------|
| Kumar et al. [39] | -Diagnosing infectious diseases             | -Diagnosing infectious diseases                                 | -Approximation model                               |
|                | -Numerical models                           | -Predicting infectious diseases                                 | -Hidden the fault role                             |
|                | -Fuzzy-based approach                       |                                                                |                                                   |
| Wang et al. [40] | -Similarity measurement                     | -Finding similarities with high accuracy                       | -Approximation model                               |
|                | -Fuzzy-based approach                       |                                                                | -Hidden the fault role                             |
| Khan et al. [41] | -Fuzzy-based approach                       | -Can be affected on decision-making about the uncertain conditions in IoT and IoMT | -Approximation model                               |
|                | -Linear Diophantine                         |                                                                | -Hidden the fault role                             |
| Ayub et al. [42] | -Fuzzy-based approach                       | -Can be affected on decision-making about the uncertain conditions in IoT and IoMT | -Approximation model                               |
|                | -Linear Algebra                             |                                                                | -Hidden the fault role                             |
| Riaz et al. [43] | -Fuzzy-based approach                       | -Detecting multiple personality disorder disease               | -Approximation model                               |
|                | -Neutrosophic logics                        | -Predicting multiple personality disorder disease             | -Hidden the fault role                             |
|                | -Medical application                        |                                                                |                                                   |
|                | -Bipolar Fuzzy-based approach               | -Detecting multiple personality disorder disease               | -Approximation model                               |
|                | -Soft mapping                               | -Predicting a suitable treatment method                        | -Hidden the fault role                             |
| Sangaiah et al. [45] | -Resource allocation                       | -Allocating resources in IoT                                   | -Lack of decision-making                          |
|                | -Whale optimization algorithm               | -Reducing total runtime and cost                               | -Low flexibility for allocating resources          |
| Li et al. [46] | -Resource allocation                        | -Balancing flow in the edge computing                          | -Lack of decision-making                          |
|                | -Edge computing                             | -Rapid adaptive offloading                                     |                                                   |
|                | -Intelligent algorithm                      | -Reducing the complexity of data processing at the cloud level |                                                   |
| Luo et al. [47] | -Detecting the route of defective devices   | -Detecting the route of defective devices                       | -Lack of decision-making                          |
|                | -Bypassing the defective nodes              | -Reducing energy consumption                                   | -Increasing complexity of allocating resources    |
|                | -E-health systems                           | -Fairness resource allocation in IoT                           | -Low flexibility                                  |
| Gupta et al. [48] | -Privacy protocols                          | -Supporting the reliability of the communications between wireless-based applications in a decentralized manner | -Only secure-based service provider                |
|                | -Security protocols                         |                                                                | -Lack of decision-making                          |
| This study     | -Providing quasi-mapping algorithm          | -Timely detecting fault                                        | -The complexity of the algorithm steps            |
|                | -Forwarding chain                            | -Reducing total runtime and energy                             |                                                   |
|                | -Backwarding chain                          | -High flexibility and scalability                              |                                                   |
|                | -Partial neural network                     | -Timely decision-making in IoT                                 |                                                   |
|                | -Pruning                                    |                                                                |                                                   |
requester in critical situations where timely services are highlighted more than providing a secure service, as facing emergencies conditions and dedicating health and treatment service with minimal facilities. By identifying losing the trust of an SP, it is removed from the SPs’ list to decision-making and provide it in the critical situations that can lead to starvation in some service providers. Since we propose an intelligent available-aware algorithm for decision-making, which detects the faulty nodes and imposes the penalties and rewards to the related weights to the identified defective device by the forwarding and backwarding chains, respectively. Our method decides about dedicating service to a requester based on its availability value besides supporting providing service in critical situations with minimal facilities.

3. Data Collection and Quasi-Mapping

To face the challenge of decision-making in IoT, we propose a method to decide about assigning a service provider to a requester that consists of quasi-mapping the Internet of Things nodes onto a NN’s neurons. This section presents it and our approach to define an application and collect a dataset, according to the described sensors’ valid information.

3.1. Defining IoT-Based Applications and Collecting a Dataset

To analyze the efficiency of the proposed algorithm, the study first defines different real applications for mapping their nodes onto a neural network’s neurons to dedicate an SP to a requester, as shown in Figure 2a–c. We describe three different applications that are inspired by the real relationship between a set of components in connecting the Internet of Things. Two applications follow medical purposes and the other aims to support road amenities, which are defined to analyze the idea efficiency for the various conditions. As an example, we describe an intensive care unit (ICU) facilities of the hospitals as an IoT-based application that their nodes are included the oxygen condensation, humidity, temperature, and ventilation performance sensors, and some fog and cloud services such as wireless access point (WAP), as shown in Figure 2a. The sensors, and fog and cloud layers nodes are mapped onto the neurons of a NN’s input, hidden, output layers, respectively.

We investigate the ICUs and recovery rooms in the different hospitals with various facilities to analyze the sensors’ output information in standard and critical situations. This work collects a dataset based on monitoring the described sensors’ valid information and generates random values, according to the received information from monitoring the hospitals’ ICUs and recovery rooms [50,51]. We simulate the applications and inject the fault with different rates to estimate the service availability and penalty for faulty nodes by the CupCarbon simulator tool [52]. As shown in Figure 3, the collected dataset consists of the sensors’ output information, fault injection rate, and availability, which is shared on Github (https://github.com/yasamanhosseini/Quasi-mapping-project (accessed on 20 October 2021)) [53].
We investigate the ICUs and recovery rooms in the different hospitals with various facilities to analyze the sensors' output information in standard and critical situations. This work collects a dataset based on monitoring the described sensors' valid information and generates random values, according to the received information from monitoring the hospitals' ICUs and recovery rooms [50,51]. We simulate the applications and inject the fault with different rates to estimate the service availability and penalty for faulty nodes by the CupCarbon simulator tool [52]. As shown in Figure 3, the collected dataset consists of the sensors' output information, fault injection rate, and availability, which is shared on Github (https://github.com/yasamanhosseini/Quasi-mapping-project (accessed on 20 October 2021)) [53].

**Figure 2.** The defined different IoT-based applications and mapping their nodes onto the NNs’ neurons; (a) Hospitals’ emergency unit facilities (Application 1); (b) Hospitals’ ICUs and recovery rooms facilities (Application 2); (c) Roadside parking facilities (Application 3).

**Figure 3.** The collected dataset, including monitoring the sensors’ valid information, fault injection rate, random generating the sensors’ outputs, and nodes availability.
3.2. Equations to Describe Quasi-Mapping and Availability

QAC decides to assign a service provider to a requester by evaluating some highlighted features of IoT-application nodes, including availability, penalty, and rewards. The subsection describes the features, quasi-mapping, and neuron operations by presenting the equations for collecting initial information to perform the algorithm. Table 2 describes the existing parameters, which are employed in the equations.

Table 2. The existing parameters in equations.

| Parameter | Description | Condition |
|-----------|-------------|-----------|
| $L$       | The set of the input layer’s nodes | —         |
| $H$       | The set of the hidden layer’s neurons | —         |
| $O$       | The set of the output layer’s neurons | —         |
| $S$       | The set of the sensor level’s nodes | —         |
| $G$       | The set of the fog level’s services | —         |
| $C$       | The set of the cloud level’s services | —         |
| $V_h$     | Value of hidden layer’s node | $0 \leq V_{hn} \leq 1$ |
| $V_o$     | Value of output layer’s node | $0 \leq V_{on} \leq 1$ |
| $i$       | The number of matrix’s rows for input layer or sensor level | $0 \leq i \leq i_{max}$ |
| $j$       | The number of matrix’s columns for input layer or sensor level | $0 \leq j \leq j_{max}$ |
| $m$       | The number of matrix’s rows for hidden layer or fog level | $0 \leq m \leq m_{max}$ |
| $n$       | The number of matrix’s columns for hidden layer or fog level | $0 \leq n \leq n_{max}$ |
| $p$       | The number of matrix’s rows for output layer or cloud level | $0 \leq p \leq p_{max}$ |
| $q$       | The number of matrix’s columns for output layer or cloud level | $0 \leq q \leq q_{max}$ |
| $W$       | Partial neural network’s weights | —         |
| $d$       | The number of hidden layer’s weights | $0 \leq d \leq d_{max}$ |
| $u$       | The number of output layer’s weights | $0 \leq u \leq u_{max}$ |
| $X$       | The initial valid value of the trained partial NN’s weights | —         |
| $f$       | The number of the detected faults | $0 \leq f \leq f_{max}$ |
| $f_{max}$ | The maximum number of fault | $f_{max} \geq 0$ |
| $i_{max}$ | The maximum number of matrix’s rows for input layer or sensor level | $i_{max} \geq 0$ |
| $j_{max}$ | The maximum number of matrix’s columns for input layer or sensor level | $j_{max} \geq 0$ |
| $m_{max}$ | The maximum number of matrix’s rows for hidden layer or fog level | $m_{max} \geq 0$ |
| $n_{max}$ | The maximum number of matrix’s columns for hidden layer or fog level | $n_{max} \geq 0$ |
| $p_{max}$ | The maximum number of matrix’s rows for output layer or cloud level | $p_{max} \geq 0$ |
| $d_{max}$ | The maximum number of hidden layer’s weights | $d_{max} \geq 0$ |
| $u_{max}$ | The maximum number of output layer’s weights | $u_{max} \geq 0$ |
| $t$       | Available time | $t \geq 0$ |
| $t_{max}$ | Maximum available time | $t_{max} \geq 0$ |
| $S_{expected\ correct}$ | The number of expected correct services | $S_{expected\ correct} \geq 0$ |
| $S_{correct}$ | The number of correct services | $S_{correct} \geq 0$ |
| $S_{total}$ | Total number of services | $S_{total} \geq 0$ |
| Penalty_{if} | Penalty’s impact factor on service availability | Penalty_{if} \geq 0 |

Before presenting the equations, the mentioned parameters of Table 2 are defined as follows:
Definition 1. Includes quasi-mapping \( l_{i,j}, h_{m,n}, \) and \( o_{p,q} \) as the elements of \( |L|_{i\times j}, |h|_{m\times n}, \) and \( |O|_{p\times q} \) matrices by transferring the roles of IoT-based application’s nodes into them where \( |S|_{i\times j}, |H|_{m\times n}, \) and \( |C|_{p\times q} \) matrices demonstrate the sensor, fog, and cloud levels of IoT.

Definition 2. Describes NeuronOperation (Multiply-accumulate operation), in which \( V_{i,j} \) and \( V_{o,p} \) are defined the located neurons’ values in hidden \( (h_{m,n}) \) and output \( (o_{p,q}) \) layers of the neural network.

Definition 3. Explains estimating the penalty value for detecting the fault with \( i \) number where is indexed with \( f \) and \( S \) describes as the service by three different situations, including the expected correct service \( (S_{\text{expected correct}}) \) and correct service \( (S_{\text{correct}}) \) at \( t \) time.

Definition 4. Applies to impose the estimated penalty value \( (\text{Penalty}_{f}) \) for the hidden layers with \( d \) weight numbers, which are related to the hidden layer weights \( (W_{i}) \) and indexed with \( f \).

Definition 5. Indicates imposing penalty \( (\text{Penalty}_{f}) \) to the related weights of output \( (W_{o}) \) layers for the indexed fault with \( f \) where \( d \) is weight number.

Definition 6. Applies estimating availability of an SP based on identifying the number of correct services \( (S_{\text{correct}}) \) compared to the total number of services \( (S_{\text{total}}) \) at \( t \) time where \( 0 \leq t \leq t_{\text{max}} \) is met before imposing the penalty value to the related weights to the detected faulty nodes.

Definition 7. Relates assessing the availability value \( (\text{Availability}_{f}) \) to the nodes due to the number of expected correct service \( (S_{\text{expected correct}}) \) compared to the total number of services \( (S_{\text{total}}) \) at \( t \) time after detecting \( i \) fault and imposing penalty \( (\text{Penalty}_{f}) \) to the connected weight to the defective or faulty nodes where \( 0 \leq t \leq t_{\text{max}} \) is met.

We describe the sensor, fog, and cloud layers of IoT as the different matrices and also the input, hidden, and output layers of the neural network, which Equation (1) demonstrates quasi-mapping the Internet of Things nodes onto NN’s neurons.

\[
\begin{align*}
L & = \begin{bmatrix}
  l_{0,0} & \cdots & l_{0,j} \\
  \vdots & \ddots & \vdots \\
  l_{i,0} & \cdots & l_{i,j}
\end{bmatrix} \\
H & = \begin{bmatrix}
  h_{0,0} & \cdots & h_{0,n} \\
  \vdots & \ddots & \vdots \\
  h_{m,0} & \cdots & h_{m,n}
\end{bmatrix} \\
O & = \begin{bmatrix}
  o_{0,0} & \cdots & o_{0,q} \\
  \vdots & \ddots & \vdots \\
  o_{p,0} & \cdots & o_{p,q}
\end{bmatrix} \\
S & = \begin{bmatrix}
  s_{0,0} & \cdots & s_{0,j} \\
  \vdots & \ddots & \vdots \\
  s_{i,0} & \cdots & s_{i,j}
\end{bmatrix} \\
G & = \begin{bmatrix}
  g_{0,0} & \cdots & g_{0,n} \\
  \vdots & \ddots & \vdots \\
  g_{m,0} & \cdots & g_{m,n}
\end{bmatrix} \\
C & = \begin{bmatrix}
  c_{0,0} & \cdots & c_{0,q} \\
  \vdots & \ddots & \vdots \\
  c_{p,0} & \cdots & c_{p,q}
\end{bmatrix}
\end{align*}
\]

QuasiMapping = \[
\begin{cases}
  l_{i,j} = s_{i,j} \quad & (j = 0) \land (0 \leq i \leq i_{\text{max}}) \\
  h_{m,n} = g_{m,n} \quad & (n = 0) \land (0 \leq m \leq m_{\text{max}}) \\
  o_{p,q} = c_{p,q} \quad & (q = 0) \land (0 \leq p \leq p_{\text{max}})
\end{cases}
\]

Additionally, Figure 4a,b shows the mapping operations that are defined IoT and a neural network based on three-dimensional structures. The red-color pointers illustrate mapping the Internet of Things levels’ nodes to a partial neural network’s neurons.
The multiply-accumulate (MAC) operations are performed in the neurons of hidden and output layers of the partial NN. Equation (2) describes the operations for hidden and output layers where 
\( (n=0) \land (0 \leq m \leq m_{\text{max}}) \) and 
\( (q=0) \land (0 \leq p \leq p_{\text{max}}) \) are met, respectively.

\[
\begin{align*}
d_{\text{max}} &= i_{\text{max}} \times m_{\text{max}} \times u_{\text{max}} = m_{\text{max}} \times p_{\text{max}} \\
\text{NeuronOperation} &= \begin{cases} 
V_{hm,n} = \sum_{i=0}^{i_{\text{max}}} \sum_{d=0}^{d_{\text{max}}} (l_{i,j} \times W_d) , & (n=0) \land (0 \leq m \leq m_{\text{max}}) \\
V_{op,q} = \sum_{m=0}^{m_{\text{max}}} \sum_{u=0}^{u_{\text{max}}} (l_{i,j} \times W_u) , & (q=0) \land (0 \leq p \leq p_{\text{max}})
\end{cases}
\end{align*}
\]

After injecting fault with random rates onto the simulated IoT nodes, we define the penalty value for the identified invisible or failed components and impose it into partial NN’s weights, which Equation (3) demonstrates penalty estimation.

\[
\begin{align*}
\text{Penalty}_{if} = \frac{\sum_{d=0}^{d_{\text{max}}} \left( S_{\text{expected correct}} - S_{\text{correct}} \right)}{S_{\text{total}}}
\end{align*}
\]

This work fines the related weights to the mapped nodes, which are detected as the invisible or faulty components at t time after simulating and injecting fault with random rates on them (Using CupCarbon tool), as shown in Equations (4) and (5) for hidden and output layers.

\[
\begin{align*}
W_d &= X_d - \sum_{f=0}^{f_{\text{max}}} \text{Penalty}_{if} , & 0 \leq d \leq d_{\text{max}} \\
W_u &= X_u - \sum_{f=0}^{f_{\text{max}}} \text{Penalty}_{fu} , & 0 \leq u \leq u_{\text{max}}
\end{align*}
\]
We describe availability based on Equation (6) before imposing the penalty into the related weights to the invisible nodes, in which Equation (7) illustrates the estimated value for the availability of the fined node by the determined penalty.

\[
\text{Availability} = \frac{\sum_{t=0}^{t_{\text{max}}} S_{\text{correct}}}{S_{\text{total}}} \quad (6)
\]

\[
\text{Availability}_{\text{Penalty}} = \left( \frac{\sum_{t=0}^{t_{\text{max}}} S_{\text{expected correct}}}{S_{\text{total}}} \right) - \text{Availability} \quad (7)
\]

By describing the effective parameters in decision-making via the presented equations, we provide the initial information for QAC to decide about assigning an SP to a requester with the estimated penalty and availability values.

3.3. Quasi-Mapping and an Availability-Aware Cloud Service Algorithm

This work presents a three-phase algorithm to decide about assigning a service provider to a requester based on analyzing the SPs’ availability, which the subsection addresses the issue. The main three phases of QAC consist of quasi-mapping the nodes of the IoT-application onto a partial neural network’s neurons, decision-making, and identifying the joined components or similar applications to the mentioned structure. To clearly explain the algorithm’s operations, we focus on application 2 (Figure 4b) and present two methods for decision-making to demonstrate the algorithm’s independence to define a threshold value.

QAC phases are classified into sub-phases to organize their operations per phase and individually describe them, which Algorithm 1 demonstrate the sub-phases of the algorithm’s phase 1. QAC phase 1-1 performs quasi-mapping the Internet of Things nodes in the levels of the sensor, fog, and cloud onto the neurons of the partial NN’s input, hidden, and output layers, parallel, as shown in Algorithm 1.

---

Algorithm 1. Quasi-mapping and availability-aware Cloud service’s (QAC)1-1; 1-2; 1-3; 1-4

Parameter declaration:
S: Sensor = \{Oxygen condensation, Humidity, Temperature, Ventilation performance, The number of beds\}
G: Fog service = \{Data collection device, Wireless access point, Hospital’s ICU monitoring\}
C: Cloud service = \{Ambulance station\}
i: Sensor index = \{0,1, \ldots ,3,4\}
j: Fog index = \{0,1,2\}
p: Cloud index = \{0\}
t: Available time whereas t \subseteq \text{Natural number and } t \geq 0
Fault: The number of detected faults in the period of t time whereas Fault \subseteq \text{Integer and } Fault \geq 0
Masked: The number of masked faults in the period of t time whereas Masked \subseteq \text{Integer and } Masked \geq 0

Inputs:
1- SV: IoT sensor’s value (Oxygen condensation, Humidity, Temperature, Ventilation performance, The number of recovery room’s beds) whereas SV \subseteq \text{Natural number}
2- HiddenLayerWeights: Hidden layer’s weights whereas HiddenLayerWeights \subseteq \text{Natural number}
3- OutputLayerWeights: Hidden layer’s weights whereas OutputLayerWeights \subseteq \text{Natural number}

Output: CloudService: The output’s value for assigning service to requester in cloud level whereas 0 \leq \text{CloudService} \leq 1

1: Start
2: Procedure QTS (SV, HiddenLayerWeights, OutputLayerWeights)
3: Start phase1-1 // Quasi-mapping IoT’s nodes onto partial NN’s nodes
4: do in parallel
5: for each S in Sensor do // Quasi-mapping IoT sensors onto input layer’s nodes of PNN
6: for i in Sensor index do
7: (Input layer’s node) \_i \leftarrow S_i
8: end for;
Algorithm 1. Cont.

9: \textbf{end for;}
10: \textbf{for each G in Fog service do} // Quasi-mapping IoT fog services onto hidden layer’s nodes of PNN
11: \textbf{for j in Fog index do}
12: \hspace{1em} (Hidden layer’s node)\_j \leftarrow G\_j
13: \textbf{end for;}
14: \textbf{end for;}
15: \hspace{1em} (Output layer’s node)\_p \leftarrow C\_p // Quasi-mapping IoT fog services onto output layer’s nodes of PNN
16: \textbf{end parallel;}
17: \textbf{end phase 1-1}
18: \textbf{Start phase 1-2} // Multiply-Accumulate (MAC) Operations in hidden and output layers of partial NN
19: \textbf{do in parallel}
20: \textbf{for each i in Sensor index do}
21: \hspace{1em} \textbf{for j in Fog index do}
22: \hspace{2em} \textbf{for p in Cloud index do}
23: \hspace{3em} (Hidden layer’s MAC value)\_j \leftarrow \sum_{i=0}^{i=\text{maximum}} SV\_i \times \text{HiddenLayerWeights}_h \hspace{1em} h = i \times j
24: \hspace{3em} \text{Normalizing} \hspace{1em} (Hidden layer’s Mac value)\_j \hspace{1em} 0 \leq (Hidden layer’s Mac value)\_j \leq 1 \hspace{1em} j = \text{maximum}
25: \hspace{3em} (Output layer’s MAC value)\_p \leftarrow \sum_{i=0}^{i=\text{maximum}} SV\_i \times \text{OutputLayerWeights}_o \hspace{1em} o = j \times p
26: \hspace{3em} \text{Normalizing} \hspace{1em} (Hidden layer’s Mac value)\_j \hspace{1em} 0 \leq (Hidden layer’s Mac value)\_p \leq 1
27: \hspace{2em} \textbf{end for;}
28: \hspace{1em} \textbf{end for;}
29: \hspace{1em} \textbf{end parallel;}
30: \textbf{end phase 1-2}
31: \textbf{Start phase 1-3} // Fault detection based on forwarding and fail stop mechanism
32: \textbf{While each t in Available time do}
33: \hspace{1em} \textbf{for SV miss IoT’s sensor value do} // Fault detection on PNN layers’ node using forwarding chain
34: \hspace{2em} \textbf{for i in Sensor index do}
35: \hspace{3em} \textbf{for j in Fog index do}
36: \hspace{4em} \textbf{for p in Cloud index do}
37: \hspace{5em} \text{Fault}_t\_i = \text{Fault}_{(t-1)}\_i + 1 // Determining defective sensor at t time and its connected weights to the hidden layer
38: \hspace{5em} \text{Fault}_t\_j = \text{Fault}_{(t-1)}\_j + 1 // Determining defective fog service at t time and its connected weights to the output layer
39: \hspace{5em} \text{Fault}_t\_p = \text{Fault}_{(t-1)}\_p + 1 // Determining defective cloud service at t time
40: \hspace{4em} \textbf{end for;}
41: \hspace{3em} \textbf{end for;}
42: \hspace{2em} \textbf{end for;}
43: \hspace{1em} \textbf{end while;}
44: \hspace{1em} \textbf{end phase 1-3}
45: \textbf{Start phase 1-4} // Availability creation based on backwarding chain
46: \textbf{While each t in Available time do}
### Algorithm 1. Cont.

| Line | Code                                                                 | Description                                                                 |
|------|----------------------------------------------------------------------|-----------------------------------------------------------------------------|
| 49:  | for p in Cloud index do // Investigating quality of service at t time in cloud level (PNN’s output layer) based on comparing output’s neuron value with the threshold’s value to determine penalty’s value |                                   |
| 50:  | if (Output layer’s MAC value) \( t_p < \) (Threshold’s value) \( p \) do |                                   |
| 51:  | Penalty\( _t \) = (Threshold’s value) \( p \) − (Output layer’s MAC value) \( t_p \) |                                   |
| 52:  | do in parallel                                                     |                                   |
| 53:  | (OutputLayerWeights) \( p \) ← (OutputLayerWeights) \( p \) − \( \sum \limits_{p=0}^{max \_t} \sum \limits_{t=0}^{Fault \_t} \) Penalty\( _{Fault \_t} \) | // Availability creation on output layer’s weights |
| 54:  | for j in Fog index do                                             |                                   |
| 55:  | (HiddenLayerWeights) \( j \) ← (HiddenLayerWeights) \( j \) − \( \sum \limits_{j=0}^{max \_t} \sum \limits_{t=0}^{Fault \_t} \) Penalty\( _{Fault \_t} \) | // Availability creation on hidden layer’s weights |

Phase 1-4 is dedicated to fault-masking by joining new components to the application’s nodes instead of the failed or faulty components. In this phase, we reward the neurons and related weights to the detected faulty nodes in facing a joined new component or receiving the correct service from them, as shown in Algorithm 3.

Algorithm 3 shows phase 3-2, which has included updating the initial information for the proposed algorithm by joining a new application to the structure or new components to run QAC based on the updated information.

In phase 1-2, the proposed algorithm addresses the multiply-accumulate operations into the neurons of hidden and output layers and normalizes the estimated MAC’s values, which Algorithm 1 demonstrates. We identify the faults in sensor, fog, and cloud levels’ nodes at t time and register them to analyze their availability in QAC’s phase 1-3, as shown in Algorithm 1.

After detecting faults in different levels of IoT, the proposed algorithm evaluates the output neuron’s value and utilizes the backward chain to apply the estimated penalty’s value the related weights in hidden and output layers when (Output layer’s MAC value) \( t_p < \) (Threshold’s value) \( p \) is met, which Algorithm 1 illustrates the operation in phase 1-4.

Phase 2 of QAC is classified into two sub-phases to decision-making about the services providers based on two methods, as shown in Algorithms 2 and 3. In phase 2-1, we decide about dedicating the service by estimating an average value and performing MAC operation into the output neuron, which has pruned the output with (OutputWeight) \( \simeq 0 \) condition.

Algorithm 2 demonstrates decision-making based on method 2, which defines a threshold value and compares it to the output neuron’s value and pruning the output by applying (OutputNeuron’s value) \( \simeq 0 \) condition.
Algorithm 2. Quasi-mapping and availability-aware Cloud service\’s(QAC)2-1; 2-2

1: Start phase 2-1 // Decision-making about dedicating cloud service based on method 1 (Weight pruning)
2: OutputWeight = (Output layer\’s MAC value) − (Maximum value of output) + (Minimum value of output).
3: do in parallel
4: OutputNeuron\’s value ← (Output layer\’s MAC value) × (OutputWeight)
5: if (OutputWeight) ≃ 0 do
6: Pruning OutputWeight and OutputNeuron
7: end if;
8: end parallel;
9: end phase 2-1
10: Start phase 2-2 // Decision-making about dedicating cloud service based on method 2 (Neuron pruning)
11: OutputNeuron\’s value = ((+1) × (Output layer\’s Mac value)) + ((−1) × (Threshold\’s value))
12: if (OutputNeuron\’s value) ≃ 0 do
13: Pruning OutputNeuron
14: end if;
15: end phase 2-2

Algorithm 3. Quasi-mapping and availability-aware Cloud service\’s(QAC)3-1; 3-2

1: Start phase3-1 // joining a new component instead of defective component
2: While each t in Available time do
3: Start phase3-1 // joining a new component instead of defective component
4: While each t in Available time do
5: for SV in IoT\’s sensor value do // Removing penalty from the connected weights to recovered nodes
6: for i in Sensor index do
7: for j in Fog index do
8: for p in Cloud index do
9: Reward\_t = Penalty\_t − Penalty\_t−1 // t≠0 and Reward\_t > 0
10: Fault\_t = 0 // Determining new joined sensor at t time and removing penalty from its connected weights to the hidden layer
11: Masked\_t = Masked\_t−1 + 1
12: Fault\_t = 0 // Determining new joined fog service or received data from the new sensor at t time and removing penalty from its connected weights to the output layer
13: Masked\_t = Masked\_t−1 + 1
14: (HiddenLayerWeights) ← (HiddenLayerWeights) + ∑ j = 0 t = 0 Reward\_Masked\_t
15: Fault\_p = 0 // Determining new joined cloud service at t time in case of receiving correct information from fog level (hidden layer\’s nodes)
16: Masked\_p = Masked\_p−1 + 1
17: (OutputLayerWeights) ← (OutputLayerWeights) + ∑ p = 0 t = 0 Reward\_Masked\_p
Algorithm 3. Cont.

| Line | Code |
|------|------|
| 18:  | end for; |
| 19:  | end for; |
| 20:  | end for; |
| 21:  | end for; |
| 22:  | end while; |
| 23:  | end phase 3-1 |

After presenting the algorithm and its operations, this work explains the steps of imposing penalties and reward to the related weights to the detected faulty nodes and the joined new components, respectively, as shown in Figures 5–9.

Figure 5. Two different methods to decision-making about dedicating service to a requester with high quality and forwarding chain to fault detection: (a) Method 1 based on pruning weak Weight (output weight) and output neuron in the output layer; (b) Method 2 is based on pruning weak output neuron in the output layer.

Figure 6. Backwarding chain to trust creation: (a) Imposing penalty to output and hidden layer’s weight and trust creation considering the method 1; (b) Imposing penalty to output and hidden layer’s weight and trust creation considering the method 2.
Figure 7. The efficiency of a joined sensor to input layer’s nodes instead of the defective node for different methods to decision-making about dedicating service to a requester with high quality and forwarding chain to fault detection: (a) Method 1 based on pruning weak weight (output weight) and output neuron in output layer; (b) Method 2 is based on pruning weak output neuron in the output layer.

Figure 8. Backwarding chain to trust creation: (a) Imposing penalty and reward to output and hidden layer’s weight and trust creation considering method 1; (b) Imposing penalty and reward to output and hidden layer’s weight and trust creation considering method 2.
The efficiency of a similar joined application with an additional option for different methods to decision-making about dedicating service to a requester with high quality and forwarding chain to fault detection: (a) Method 1 based on pruning weak weight (output weight) and output neuron in output layer; (b) Method 2 based on pruning weak output neuron in the output layer.

Figure 5a, b address the fault detection process based on method 1 and method 2, in which the weak output’s neuron and weight are pruned by them using providing the average and threshold values, respectively. The faults are detected by the forwarding chain and penalties are imposed on the weights using the backwarding chain.

We impose the estimated penalty value to the related weight to the defective or faulty nodes by employing the backwarding chain, as shown in Figure 6a, b.

The study defines two hypotheses to assess the efficiency of QAC and the idea, including facing the joined new components and applications to the mentioned structure. Figure 7a, b demonstrate diagnosing the joined new sensor instead of the defective sensor with similar characteristics to the application by the forwarding chain.

After detecting the joined new sensor and masking the faulty component, we reward the related weights to the masked node by the backwarding chain, as shown in Figure 8a, b.
By joining a new application with a different feature, QAC decides about dedicating an SP to a requester based on the updated initial information that is related to the application’s features and the output neurons’ availability condition. Figure 9a,b illustrate the decision-making process by joining a new application to the structure. We demonstrate the process by a flowchart as follows:

According to the reviewed concepts of the work, QAC was provided as an intelligent algorithm that has detected the faulty components and joined new nodes and applications by the forwarding chain. The algorithm utilized the backwarding chain to impose penalties and reward to the related weights to the detected faulty and new components, which feature created a flexible structure for decision-making in critical situations. Due to the features of the neural networks, this work proposed the quasi-mapping method to transfer the role of IoT-applications’ nodes into the NNs’ neurons to decide about assigning a service provider to a requester. We provided a service based on analyzing its availability and pruned the weak output neuron and weights by imposing the penalty. The feature also created the opportunity to reconnect the removed SPs with low availability in facing joining new components instead of defective components and masking their fault.

4. Simulation Results

According to the mentioned purposes and contributions of the idea, we evaluate the dedicated SPs’ availability and decision-making accuracy to validate QAC efficiency in improving the Internet of Things performance. This section addresses the issue and proves the highlighted role of decision-making based on the availability-aware algorithm, which reports communication delay and energy consumption caused by transferring data between IoT nodes.

To analyze the quasi-mapping and decision-making performance, we first map an application’s nodes onto a neural network’s neurons and train it based gradient-distance method. The study describes the neural networks based on python co-design and utilizes the TensorFlow libraries to define the functions by the Jupyter notebook [54,55]. After training the NN by the initial information, we simulate the application nodes and inject the faults with random rates using the CupCarbon simulator tool, which is described based on Java co-design in MAC and Ubuntu LTS 16.04 [52]. By injecting the faults into the nodes and updating the initial information about their situation, the IoT-application components are re-mapped onto the neural network’s neurons and are imposed the estimated penalties.
into the related weights to the faulty devices and are retrained the NN by the updated information. We repeat the operations to analyze service availability by changing the fault injection rate. Additionally, this work defines the randomly joined new components instead of defective nodes and a new application with different features to the mentioned structure that simulates them to report delay and energy consumption by the CupCarbon tool. Due to the joined new components and applications, we reward the related weights to them and decide about providing a service to a requester by defining new value for the weights and retraining the neural network using the updated information.

To assess QAC’s efficiency for decision-making, this work first analyzes the relationship between fault rates and output neurons’ availability because of presenting three different applications and SPs. Figure 10a–f proves the indirect relationship between them for SP1 and SP2 in the presented three applications, which correlation coefficient between fault rates and availability’s values is dependent on the applications. The simulation results indicate the availability and fault rate by approximately 0.37 and 0.85 for application 1 and SP2 in time slot 5, which proves the indirect relationship between the mentioned parameters. As shown in Figure 10a–f, the rising time slot affects improving or deteriorating the availability of an SP of the application because of facing faulty nodes and joining new components instead of the defective devices. According to the presented Equation and algorithm, we estimate the availability of a service provider based on the different parameters (including the expected number of correct services, the total number of services, and the number of correct services) after imposing the penalties and rewards, which the weak weights and neurons are pruned by detecting the defective devices. By increasing the fault rate, the simulation results illustrate a less reducing availability for application 3 than other applications because of its less sensitivity to the occurring faults. We report them in different time slots, which is included the maximum available time ($t_1 \leq t \leq t_{\text{max}}$).

Application 3 aims to support the road welfare services such as the distance between vehicles and road resorts or filling stations, which have less sensitivity compared to the medical or healthcare application against the faults rates. We define a threshold value for analyzing the service availability in the output layer of the neural network to prune the weak weights and neurons of the layer and remove the service provider with minimum availability, which its value is determined based on the applications. Therefore, the application sensitivity to the fault rate affects assessing the SPs’ availability because of estimating the penalty and reward value based on the feedback of the output neuron’s value.

![The relationship between fault rate and availability for SP1 of application 1](image1)

![The relationship between fault rate and availability for SP2 of application 1](image2)

Figure 10. Cont.
According to the highlighted role of decision-making accuracy, we evaluate the parameter for the three provided applications after performing the algorithm and imposing the penalties and reward to the related weights to the detected faulty nodes and joined new components. As shown in Figure 11a,b, decision-making accuracy and standard deviations are approximately 97% and 0.03 for application 2, which is also estimated for the other applications by comparing the expected and assigned SP numbers. CAS provides a service to a requester based on analyzing its availability, in which the estimated value is defined as the expected SP by the Equations (6) and (7). The study decides about dedicating a service provider to a requester with high accuracy by providing an availability-aware algorithm and employing the forwarding and backwarding chains to detect the fault and imposing the penalty and pruning the weak weights and neurons. We estimate the accuracy of decision-making and standard deviation by repeating QACs operations when facing a new request and condition at the available time.
Figure 11. Comparing the expected and estimated availability values in application 2: (a) Comparing the expected and estimated availability values without losing a correct decision; (b) Comparing the expected and estimated availability values with losing a correct decision (Red colored pointer demonstrates the incorrect decision-making).

Also, this work estimates the total energy and delay by comparing the mentioned parameters before and after pruning the weak weights and neurons, in which the output neurons are available as the service providers at the cloud level. To evaluate the mentioned parameters, we simulate the application nodes and describe their operation based on Java and Python co-designs as the scripts by the CupCarbon tool. The connections between the application’s nodes are defined due to the relationship between them after mapping onto the NN’s neurons. We utilize the router nodes, in which several nodes have to connect to one device (A located node on the fog or cloud level such as a wireless access point node). The CupCarbon determines a specific area for connecting a set of nodes without using the router. Additionally, the simulator reports the energy consumption, total energy consumption, battery level, and total battery level for each simulated node (in different levels), which we assess the total energy consumption, including all of the reported mentioned parameters. As the setting of the simulator to report delay, we determine the unit of estimating total runtime using CupCarbon, which estimates the parameter based on the communication between nodes after describing their operations and reading input files as a dataset. By assessing the energy consumption and communication delay, we prove increasing the Internet of Things performance with minimal cost before facing invisible an SP. Figure 12a–e illustrates improving total delay and energy by approximately 47.4% and
64.71% after pruning the weak weights and neurons with losing minimal availability in approximately 1–3%. As shown in Figure 12c,d, we estimate the total delay and energy for the other related works (Without CAS) and compare them with the proposed idea when running the algorithm before pruning them (Before pruning with CAS). The simulation results indicate reducing total energy and delay by approximately 47.3% and 19.9% for CAS compared to the other related works. In comparing the results between other related works and CAS, we simulate application 2 without mapping their nodes (Fore other works) onto the neural network and injecting faults onto the similar nodes with the utilized same rates for the proposed idea by the CupCarbon tool. The proposed algorithm has more efficiency (Improving total delay and energy) than other works because of mapping the applications nodes onto the NN neurons and simulating them based on the connections between the neurons in different layers of a partial neural network. By analyzing the reported results in Figure 12a,d, we prove the highlighted role of pruning weak neurons and the related weights to the detected faulty nodes in improving the performance of the IoT-based applications for the proposed idea compared (After detecting and pruning) to other works (Without CAS).

The simulation results show that the maximum contributions of total energy are related to the routers and switches, and middle nodes with the maximum number of connections, as shown in Figure 13. By occurring faults on the components and imposing the penalty to the related weights to them, we prune the components where their values are close to zero and improve the total delay and energy.

![Total delay for application 2](image)

![Total energy for application 2](image)

![Comparing total energy for application 2](image)

![Comparing total delay for application 2](image)

**Figure 12. Cont.**
Figure 12. Comparing availability, the delay and energy before and after detecting the faults in application 2: (a) Comparing the total delay between three phases of before pruning, after detecting fault and pruning, and after detecting fault without pruning; (b) Comparing the total energy between three phases of before pruning, after detecting fault and pruning, and after detecting fault without pruning; (c) Comparing total energy between QAC and other related works (Without CAS); (d) Comparing total delay between QAC and other related works (Without CAS); (e) Comparing the total energy between three phases of before pruning, after detecting fault and pruning for the different available SPs of application 2 (Red colored indexes demonstrate the SP numbers).

By comparing the total delay and energy between three phases, the simulation results prove the highlighted role of pruning the weak weights and neurons in improving IoT-application performance. We simulate the pruned neural network after imposing the penalties into the related weights to the detected faulty nodes to assess the Internet of Things-based application efficiency that the removed nodes led to improve the communication delay and energy consumption.

Figure 13. Energy consumption contribution of the different. Simulated components of IoT for application 2.
As shown in existing concepts and reported results, we employed the hierarchical approach to assess the idea and QAC efficiency in decision-making by quasi-mapping the Internet of Things levels’ nodes onto the partial neural networks’ neurons and simulated the IoT applications by the CupCarbon. After injecting faults with random rates, QAC re-mapped the application nodes onto the NN’s neurons to impose the estimated penalty value into the related weights to the detected faulty components and analyzed the availability of output neurons (Cloud services). Additionally, we rewarded the related weights to the joined new components and evaluate the service’s availability by joining new applications or new nodes instead of the defective devices. Since simulation results first analyzed the relationship between fault rates and availability values for different service providers of the applications to demonstrate that their correlation coefficient depended on the application and its sensitivity to occur the faults. The results proved that the proposed algorithm decided about dedicating service to a requester based on assessing its availability with high accuracy by comparing the expected SPs and assigned service providers. Additionally, we reported the total delay and energy caused by communications between the IoT-application nodes, which illustrated the highlighted role of pruning the related weak weights and neurons to the detected faulty nodes. To verify the reported results, we shared all logs of simulating the applications on Github (https://github.com/yasamanhosseini/Quasi-mapping-project (accessed on 20 October 2021)) [53].

By reviewing the proposed idea and reported experimental and simulation results, the work can support the different IoT-based applications to decision-making with high accuracy and availability.

5. Conclusions
The Internet of Things follows gathering the records and creating an infrastructure for communications between different devices and remote control them to service the requesters with high quality. By rising the joined applications to IoT infrastructure, the framework developers faced some problems, including protecting private information, service availability and dependability, and security. Some case studies addressed the problems to provide the different methods and protocols to satisfy the purposes and improve the efficiency of the Internet of Things. Decision-making is also challenging in IoT by increasing the number of service providers with different characteristics and features to dedicate them to the requesters. We followed the issue to propose a decision-making method, which was utilized the neural network’s features to decide about providing an SP to a requester with high quality. The study presented an availability-aware algorithm that injects the fault to the IoT-based applications’ nodes with different rates for decision-making and dedicating a service provider based on its availability. We mapped the IoT levels’ nodes onto the partial neural network’s neurons to online training the NNs with the updated information in facing faulty components and identifying the joined new nodes or application to the structure. By employing the forwarding chain to detect the faults or the joined new components and the backwording chain to impose the penalties and reward to the weights, the proposed algorithm pruned the weak neuron and weights and evaluate the service providers’ efficiency to dedicate them to the requester. The feature created an opportunity to dedicate a service provider in critical condition with minimal facilities when facing an emergency request and assigned SPs with high availability to other requesters.

Author Contributions: A.M.R.: Formal analysis, Methodology, Writing—original draft, Writing—review & editing. R.A.N.: Writing—review & editing, Software, Formal analysis. S.A.: Writing—review & editing, Methodology S.Y.H.M.: review & editing. M.H.: Conceptualization, Funding acquisition, Project administration, Supervision. All authors have read and agreed to the published version of the manuscript.

Funding: The authors acknowledge that this research was supported by Xiamen University Malaysia Research Fund (Grant No: XMUMRF/2019-C3/IECE/0006).

Institutional Review Board Statement: Not applicable.
Informed Consent Statement: Not applicable.

Acknowledgments: The authors acknowledge Kamran Siddique and Amir Haider for their contributions in review, editing, proofreading, and funding acquisition.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fu, X.; Yu, F.R.; Wang, J.; Qi, Q.; Liao, J. Dynamic service function chain embedding for NFV-enabled IoT: A deep reinforcement learning approach. IEEE Trans. Wirel. Commun. 2019, 17, 507–519. [CrossRef]

2. Amanullah, M.A.; Habeeb, R.A.; Nasaruddin, F.H.; Gani, A.; Ahmed, E.; Nairn, A.S.; Akim, N.M.; Imran, M. Deep learning and big data technologies for IoT security. Comput. Commun. 2020, 151, 495–517. [CrossRef]

3. Alladi, T.; Chamola, V.; Sikdar, B.; Choo, K.K. Consumer IoT: Security vulnerability case studies and solutions. IEEE Consum. Electron. Mag. 2020, 9, 17–25. [CrossRef]

4. Kumar, R.; Tripathi, R. Towards design and implementation of security and privacy framework for internet of medical things (iomt) by leveraging blockchain and iips technology. J. Supercomput. 2021, 77, 7916–7955. [CrossRef]

5. Tawalbeh, L.A.; Muheidat, F.; Tawalbeh, M.; Quwaider, M. IoT Privacy and security: Challenges and solutions. Appl. Sci. 2020, 10, 4102. [CrossRef]

6. Khalid, U.; Asim, M.; Baker, T.; Hung, P.C.; Tariq, M.A.; Rafferty, L. A decentralized lightweight blockchain-based authentication mechanism for IoT systems. Clust. Comput. 2020, 10, 1–21. [CrossRef]

7. Zhang, J.; Shen, C.; Su, H.; Arafain, M.T.; Qu, G. Voltage over-scaling-based lightweight authentication for IoT security. IEEE Trans. Comput. 2021, 1, 1. [CrossRef]

8. Legenvre, H.; Henke, M.; Ruile, H. Making sense of the impact of the internet of things on Purchasing and Supply Management: A tension perspective. J. Purch. Supply Manag. 2020, 26, 100596. [CrossRef]

9. Tchernykh, A.; Babenko, M.; Chervyakov, N.; Miranda-L ópez, V.; Avetisyan, A.; Drozdov, A.Y.; Rivera-Rodriguez, R.; Radchenko, G.; Du, Z. Scalable data storage design for nonstationary IoT environment with adaptive security and reliability. IEEE Internet Things J. 2020, 7, 10171–10188. [CrossRef]

10. Nguyen, T.A.; Min, D.; Choi, E. A hierarchical modeling and analysis framework for availability and security quantification of IoT infrastructures. Electronics 2020, 9, 155. [CrossRef]

11. Abou-Nassar, E.M.; Iliyasu, A.M.; El-Kafrawy, P.M.; Song, O.Y.; Bashir, A.K.; Abd El-Latif, A.A. DiTrust chain: Towards blockchain-based trust models for sustainable healthcare IoT systems. IEEE Access 2020, 8, 111223–111238. [CrossRef]

12. Djedjig, N.; Tandjaoui, D.; Medjek, F.; Romdhani, I. Trust-aware and cooperative routing protocol for IoT security. J. Inf. Secur. Appl. 2020, 52, 102467. [CrossRef]

13. Gope, P.; Gheraibia, Y.; Kabir, S.; Sikdar, B. A secure IoT-based modern healthcare system with fault-tolerant decision making process. IEEE J. Biomed. Health Inform. 2020, 25, 862–873. [CrossRef]

14. Mekala, M.S.; Viswanathan, P. (t, n): Sensor Stipulation with THAM index for smart agriculture decision-making IoT system. Wirel. Pers. Commun. 2020, 111, 1909–1940. [CrossRef]

15. Almiani, M.; AbuGhazleh, A.; Al-Rahayfeh, A.; Atiwei, S.; Razaque, A. Deep recurrent neural network for IoT intrusion detection system. Simul. Model. Pract. Theory 2020, 101, 102031. [CrossRef]

16. Rahmani, A.M.; Mirmahaleh, S.Y.; Hosseinzadeh, M. An intelligent algorithm to recommend percent vegetation cover (ARVC) for PM 2.5 reduction. Air Qual. Atmos. Health 2020, 13, 859–870. [CrossRef]

17. Khan, H.; Shah, PM.; Shah, M.A.; ul Islam, S.; Rodrigues, J.J. Cascading handcrafted features and Convolutional Neural Network for IoT-enabled brain tumor segmentation. Comput. Commun. 2020, 153, 196–207. [CrossRef]

18. Chowdhury, M. Reward-based service provisioning scheme for UAV-MEC assisted IoT infrastructures. Int. J. Ad Hoc Ubiquitous Comput. 2021, 37, 113–124. [CrossRef]

19. Zhang, J.; Bhuian,M.Z.; Xu, Y.; Singh, A.K.; Hsu, D.F.; Luo, E. Trustworthy target tracking with collaborative deep reinforcement learning in Edge/AI-Aided IoT. IEEE Trans. Ind. Inform. 2021, 18, 1301–1309. [CrossRef]

20. Yadav, V.K.; Yadav, R.K.; Verma, S.; Venkatesan, S. CP2EH: A comprehensive privacy-preserving e-health scheme over cloud. J. Supercomput. 2021, 28, 1–31. [CrossRef]

21. Srinidhi, N.N.; Sagar, C.S.; Shreyas, J.; SM, D.K. An improved PRoPHET-Random forest based optimized multi-copy routing for opportunistic IoT networks. Internet Things 2020, 11, 100203. [CrossRef]

22. Manikandan, R.; Patan, R.; Gandomi, A.H.; Sivanesan, P.; Kalyanaraman, H. Hash polynomial two factor decision tree using IoT for smart health care scheduling. Expert Syst. Appl. 2020, 141, 112924. [CrossRef]

23. Xiong, X.; Zheng, K.; Lei, L.; Hou, L. Resource allocation based on deep reinforcement learning in IoT edge computing. IEEE J. Sel. Areas Commun. 2020, 38, 1133–1146. [CrossRef]

24. Singh, S.K.; Jeong, Y.S.; Park, J.H. A deep learning-based IoT-oriented infrastructure for secure smart city. Sustain. Cities Soc. 2020, 60, 102252. [CrossRef]

25. Lir, T. Deep Learning for IoT. In Proceedings of the 2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC), Austin, TX, USA, 6 November 2020; pp. 1–4.
26. Mirmahaleh, S.Y.; Reshadi, M.; Shabani, H.; Guo, X.; Bagherzadeh, N. Flow mapping and data distribution on mesh-based deep learning accelerator. In Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip, New York, NY, USA, 17–18 October 2019; pp. 1–8.

27. Mirmahaleh, S.Y.; Reshadi, M.; Bagherzadeh, N. Flow mapping on mesh-based deep learning accelerator. J. Parallel Distrib. Comput. 2020, 144, 80–97. [CrossRef]

28. Mirmahaleh, S.Y.; Reshadi, M.; Bagherzadeh, N.; Khademzadeh, A. Data scheduling and placement in deep learning accelerator. Clust. Comput. 2021, 10, 1–9. [CrossRef]

29. Mirmahaleh, S.Y.; Rahman, A.M. DNN pruning and mapping on NoC-Based communication infrastructure. Microelectron. J. 2019, 94, 104655. [CrossRef]

30. Lv, Z.; Qiao, L.; Kumar Singh, A.; Wang, Q. AI-empowered IoT security for smart cities. ACM Trans. Internet Technol. 2021, 21, 1–21. [CrossRef]

31. Aversano, L.; Bernardi, M.L.; Cimitile, M.; Pecori, R. A systematic review on Deep Learning approaches for IoT security. Comput. Sci. Rev. 2021, 40, 100389. [CrossRef]

32. Ahmad, R.; Alsmadi, I. Machine learning approaches to IoT security: A systematic literature review. Internet Things 2021, 29, 100365. [CrossRef]

33. Verma, R.; Chandra, S. A systematic survey on fog steered IoT: Architecture, prevalent threats and trust models. Int. J. Wirel. Inf. Netw. 2021, 28, 116–133. [CrossRef]

34. Altuf, A.; Abbas, H.; Iqbal, F.; Khan, M.M.; Rauf, A.; Kanwal, T. Mitigating service-oriented attacks using context-based trust for smart cities in IoT networks. J. Syst. Archit. 2021, 115, 102028. [CrossRef]

35. Arfi, W.B.; Nasr, I.B.; Kondrateva, G.; Hikkerova, L. The role of trust in intention to use the IoT in eHealth: Application of the modified UTAUT in a consumer context. Technol. Forecast. Soc. Chang. 2021, 167, 120688. [CrossRef]

36. Pappas, N.; Caputo, A.; Pellegrini, M.M.; Marzi, G.; Michopoulos, E. The complexity of decision-making processes and IoT adoption in accommodation SMEs. J. Bus. Res. 2021, 131, 573–583. [CrossRef]

37. Sharma, P.K.; Dennison, M.; Raglin, A. IoT solutions with multi-sensor fusion and signal-image encoding for secure data transfer and decision making. arXiv 2021, arXiv:2106.01497.

38. Cui, Y.; Liu, W.; Rani, P.; Alrasheedi, M. Internet of Things (IoT) adoption barriers for the circular economy using Pythagorean fuzzy SWARA-CoCoSo decision-making approach in the manufacturing sector. Technol. Forecast. Soc. Chang. 2021, 171, 120951. [CrossRef]

39. Kumar, S.; Ahmadian, A.; Kumar, R.; Kumar, D.; Singh, J.; Baleanu, D.; Salimi, M. An efficient numerical method for fractional SIR epidemic model of infectious disease by using Bernstein wavelets. Mathematics 2020, 8, 558. [CrossRef]

40. Wang, P.; Wang, J.; Wei, G.; Wei, C. Similarity measures of q-rung orthopair fuzzy sets based on cosine function and their applications. Mathematics 2019, 7, 340. [CrossRef]

41. Khan, N.; Yaqoob, N.; Shams, M.; Gaba, Y.U.; Riaz, M. Solution of Linear and Quadratic Equations Based on Triangular Linear Diophantine Fuzzy Numbers. J. Funct. Spaces 2021, 2021, 8475863. [CrossRef]

42. Ayub, S.; Shabir, M.; Riaz, M.; Aslam, M.; Chinram, R. Linear Diophantine Fuzzy Relations and Their Algebraic Properties with Decision Making. Symmetry 2021, 13, 945. [CrossRef]

43. Riaz, M.; Hashmi, M.R. M-polar neutrosophic soft mapping with application to multiple personality disorder and its associated mental disorders. Artif. Intell. Rev. 2021, 54, 2717–2763. [CrossRef]

44. Riaz, M.; Tehrim, S.T. Bipolar fuzzy soft mappings with application to bipolar disorders. Int. J. Biomath. 2019, 12, 1950080. [CrossRef]

45. Sangiah, A.K.; Hosseinabadi, A.A.; Shareh, M.B.; Bozorgi Rad, S.Y.; Zolfagharian, A.; Chilamkurthy, N. IoT resource allocation and optimization based on heuristic algorithm. Sensors 2020, 20, 539. [CrossRef]

46. Li, X.; Qin, Y.; Zhou, H.; Cheng, Y.; Zhang, Z.; Ai, Z. Intelligent Rapid adaptive offloading algorithm for computational services in dynamic internet of things system. Sensors 2019, 19, 3423. [CrossRef]

47. Luo, Y.; Li, W.; Qiu, S. Anomaly detection based latency-aware energy consumption optimization for IoT data-flow services. Sensors 2020, 20, 122. [CrossRef]

48. Gupta, R.; Shukla, A.; Mehta, P.; Bhattacharya, P.; Tanwar, S.; Tyagi, S.; Kumar, N. Vahak: A blockchain-based outdoor delivery scheme using uav for healthcare 4.0 services. In Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 6–9 July 2020; pp. 255–260.

49. Mehta, P.; Gupta, R.; Tanwar, S. Blockchain envisioned UAV networks: Challenges, solutions, and comparisons. Comput. Commun. 2020, 151, 518–538. [CrossRef]

50. Tonello, S.; Borghetti, M.; Sardini, E.; Serpelloni, M. Improved portable measuring device for real-time humidity and temperature monitoring in intensive care unit. IEEE Instrum. Meas. Mag. 2020, 23, 79–86. [CrossRef]

51. Tanaka, H.; Yokose, M.; Takaki, S.; Mihara, T.; Saigusa, Y.; Goto, T. Evaluation of respiratory rate monitoring using a microwave Doppler sensor mounted on the ceiling of an intensive care unit: A prospective observational study. J. Clin. Monit. Comput. 2021, 30, 1–9. [CrossRef] [PubMed]

52. Simulator. Available online: http://cupcarbon.com/ (accessed on 20 October 2021).

53. Dataset. Available online: https://github.com/ysamhanosseini/Quasi-mapping-project (accessed on 20 October 2021).
54. Platform. Available online: https://jupyter.org/ (accessed on 20 October 2021).
55. Library. Available online: https://www.tensorflow.org/ (accessed on 20 October 2021).