applied sciences

MDPI

Article

A Proposal for a Federated Learning Protocol for Mobile and Management Systems

Jakub Michalek †, Vaclav Oujezsky *,†, Martin Holik and Vladislav Skorpil

Department of Telecommunication, Brno University of Technology, Technicka 12, 616 00 Brno, Czech Republic; 186140@vut.cz (J.M.); xholik11@vut.cz (M.H.); skorpil@vut.cz (V.S.)

* Correspondence: oujezsky@vut.cz

† These authors contributed equally to this work.

Featured Application: The proposed protocol can be applied to systems utilizing federated learning. The design takes into account systems that necessitate both redundant and secure communication.

Abstract: In this research paper, we introduce a federated learning communication protocol tailored for emergency management applications. Our primary objective is to tackle the communication challenges that arise in such critical scenarios. In order to overcome the limitations associated with centralized server architectures, we present an innovative communication protocol. This protocol empowers the framework to effectively cooperate with multiple centralized servers, fostering efficient knowledge sharing and model training while ensuring the utmost data privacy and security. By harnessing this protocol, our framework elevates the performance and resilience of vital infrastructure systems operating on the Android platform, thereby facilitating real-time operational scenarios. This research makes a substantial contribution to the field of emergency management applications, as we offer a comprehensive solution that optimizes communication and enables seamless collaboration with numerous centralized servers.

Keywords: Android; communication protocol; federated learning; framework; machine learning; mobile

Citation: Michalek, J.; Oujezsky, V.; Holik, M.; Skorpil, V. A Proposal for a Federated Learning Protocol for Mobile and Management Systems. *Appl. Sci.* **2024**, *14*, 101. https://doi.org/10.3390/app14010101

Academic Editor: Luis Javier Garcia Villalba

Received: 12 November 2023
Revised: 14 December 2023
Accepted: 20 December 2023
Published: 21 December 2023

1. Introduction

Emergency management applications [1] play a critical role in maintaining the efficient operation of various industries. For example, transportation emergency management systems play a pivotal role in significantly augmenting traffic safety and accruing substantial experiential insights in the domain of transportation emergency management. Notably, these systems facilitate expeditious information processing and environmental interaction in the context of traffic accidents. Furthermore, they enable the expeditious estimation of the temporal parameters associated with rescue operations and the evacuation of individuals in the aftermath of such incidents [2]. Thus, it is imperative to develop robust and secure frameworks and communication protocols to enhance their performance and intelligence. As the demand for crisis management grows, especially in terms of coordinated efforts among emergency response teams, there is a pressing need for reliable and rapid information access to address crisis situations effectively. Information systems (IS), particularly web applications based on client–server principles, offer one approach to managing this information. Database systems are essential for organizing logically related data for future use. Modern systems handle complex structured and unstructured data, encompassing IS programs, data analysis, and process management. Local and cloud systems, such as Google Firebase Messaging [3] and MongoDB Realm [4], are employed for notifications, remote configuration, and application management. Safeguarding the integrity of these data is a paramount concern for programmers and institutions alike.

In recent years, federated learning [5,6] has emerged as a promising technique to harness decentralized data sources while preserving data privacy. Research into using

federated learning in emergency management applications is undergoing progressive expansion. For example, a privacy-preserving federated transfer learning approach for disaster classification (FedTL) has been proposed, which uses federated learning for disaster analysis from an image from social networks [7], or an application using the FedResilience algorithm that improves the resilience of resource-constrained critical infrastructures [8]. Even with these developments, the exploration of federated learning within emergency management applications on the Android platform still needs to be explored. While federated learning has shown promise in decentralized data utilization, its application and exploration in the Android ecosystem for emergency management remain limited.

This paper introduces our proposed KI federated learning framework specifically proposed for emergency management applications and mainly our proposed communication protocol. The abbreviation KI is based on the Czech abbreviation for critical infrastructure. The framework and the communication protocol are designed to tackle the unique challenges presented by these systems, which include limited computational resources, diverse data sources, and rigorous security requirements.

The framework incorporates novel communication protocols and machine learning techniques to facilitate collaborative model training while upholding data privacy. It empowers distributed machine learning model training across multiple Android devices, capitalizing on their collective intelligence without compromising the confidentiality of sensitive data.

Furthermore, the framework features adaptive learning algorithms that can dynamically adjust model updates to accommodate the changing characteristics of the infrastructure system. This adaptive capability ensures that the trained models remain accurate and relevant in real-time operational scenarios. The scientific contribution is as follows:

- Introduction of the KI Federated Learning Framework: Development of a novel framework, specifically designed for emergency management applications used with Android systems.
- Innovative Communication Protocol: Proposal of a communication protocol compatible with Apache Kafka and Firebase, contributing to structured communication in federated learning systems.
- Adaptive Learning Algorithms: Integration of adaptive learning algorithms to dynamically adjust model updates, ensuring relevance and accuracy in real-time operational scenarios.
- Expansion of Federated Learning in Android: Exploration and expansion of federated learning applications within the Android platform, contributing to the evolving field of Android-based emergency management systems.
- Simulation and Evaluation: Comprehensive simulation within MATLAB Simulink to assess the efficacy of the proposed communication protocol and framework, ensuring practical feasibility.

In summary, this research makes a significant contribution to the advancement of federated learning in the context of critical infrastructure systems. It provides a communication protocol and practical and secure framework tailored to the Android platform, opening up new possibilities for enhancing the performance and security of these systems. Ultimately, this will increase reliability and resilience in critical operational scenarios.

The paper is structured as follows. Section 2 provides a comprehensive overview of the current state of existing frameworks and tools in the realm of federated learning. In Section 3, we introduce a novel Android federated learning framework tailored to critical infrastructure systems. We delve into the crucial necessity for robust communication between end devices and central servers and the adoption of multiple independent central servers. The principles of our proposed communication protocol used in our framework are elucidated in Section 4, and the results and protocol simulation are presented in Section 5. This is followed by a detailed discussion of the outcomes and our proposed solutions in Section 6. Finally, Section 7 offers a concluding perspective on the entire endeavor.

2. The State of the Art

As part of our research [6], we present a comprehensive overview and comparative analysis of existing frameworks, along with an introduction to the fundamental principles of federated learning. Among the most prominent open-source federated learning frameworks are TensorFlow Federated (TFF) [9] and Federated AI Technology Enabler (FATE) [10]. TFF is tailored for decentralized data processing and boasts support for various algorithms such as FedAvg and FedSGD. It provides both a Federated Learning Application Programmable Interface (API) for high-level interfaces and a Federated Core (FC) API for distributed computing. TFF has demonstrated its efficacy in practical applications, such as mobile keyboard prediction [9]. FATE, on the other hand, places a strong emphasis on secure computing protocols based on homomorphic encryption and multiparty computation. It accommodates a broad spectrum of machine learning algorithms and traditional methods. Notable projects leveraging federated learning include VisionAir [11] for air quality estimation, an object recognition project by Jose Carbacho [12], and a privacy-preserving system for Android malware detection. The Flower project, built on TFF, facilitates federated learning across different servers and devices [13].

For addressing the challenge of heterogeneous data distribution, the Federated Learning with Dynamic Regularization (FedDyn) framework [14] has been developed. It tackles the balance between minimizing the error on individual devices and minimizing the global error by employing dynamic regularization to achieve convergence between local and global errors. Hybrid Federated Dual Coordinate Ascent (HyFDCA) [15] is dedicated to handling the intricacies of hybrid federated learning, where each client manages a specific subset of data samples or features. It utilizes a dual coordinate ascent approach to tackle these challenges. Lastly, the KafkaFed [16] framework introduces an algorithm for federated learning that harnesses Apache Kafka as its communication medium. KafkaFed leverages the robust messaging capabilities of Kafka to facilitate seamless communication and coordination among the participants in the federated learning process.

These diverse frameworks offer a range of approaches and techniques to address the various aspects and challenges of federated learning.

Emergency management applications encompass a wide array of essential systems and services crucial for a society and nation's proper functioning. Android federated learning can have significant applications and implications within the realm of emergency management applications. The integration of federated learning on Android devices within these applications offers several advantages, including improved privacy, localized data processing, and real-time decision-making capabilities. However, it is imperative to exercise meticulous consideration concerning security, resource constraints, data quality, and compliance to ensure the successful and responsible implementation of federated learning in the context of emergency management.

Android's role in emergency management applications is an emerging field that has not received substantial attention thus far. Nonetheless, the Android operating system is gradually expanding beyond mobile devices and into the Internet of Things (IoT) domain. Consequently, it becomes increasingly crucial to explore the implications and potential applications of Android within emergency management systems.

Furthermore, aside from the traditional deployment options, novel Android deployment solutions are emerging, such as Waydroid [17] and LineageOS [18]. These platforms enable the operation of a full-fledged Linux distribution with an Android compatibility layer on Android devices. This breakthrough enhances flexibility and customization, allowing users to leverage Android applications alongside Linux-based functionalities. These evolving deployment options open up exciting new possibilities for Android-based systems in a variety of contexts. Beyond the Android-based systems mentioned above, the current possibilities for implementing machine learning using Google TFlite cover a range of other devices. This includes, for example, STM32 AI [19], where custom models can be implemented on a range of microprocessors. However, this is beyond our research, and we are primarily focused on Android systems.

As a part of our project, we conducted an anonymous survey aimed at exploring the potential requisites and applications of integrating federated learning with Android systems in the context of critical infrastructures. The survey outcomes unveiled several pivotal requirements, with a pronounced focus on the imperative need for secure and dependable data transmission. Notably, the quality of service and ensuring the unwavering delivery of data emerged as critical facets within this domain.

Furthermore, the survey underscored the significance of services relating to data management and machine learning. These services assume a central role in effectively harnessing data resources and deploying machine learning algorithms within the framework of federated learning.

In sum, the survey illuminated the paramount importance of addressing security concerns, establishing a dependable data transmission infrastructure, and providing robust services for data management and machine learning. These elements are essential for realizing the full potential of federated learning when integrated with Android systems for emergency management applications. The subsequent sections elaborate on the proposed framework, taking into account the insights derived from our survey and building upon existing frameworks.

3. The Proposed Framework Components

A fundamental prerequisite in our design is the establishment of a reliable communication channel between the terminal devices and the central server. This channel is crucial for transmitting loss calculations and model weights. Our design accounts for the versatility of this system to be applied across various technologies and includes provisions for backup communication methods. Furthermore, we must address the challenge of preventing the central server from becoming overwhelmed by multiple client requests. To meet this requirement, we accommodate the deployment of multiple independent central servers that synchronize during the learning process. The conceptual framework for connecting these central servers and end clients is depicted in Figure 1.

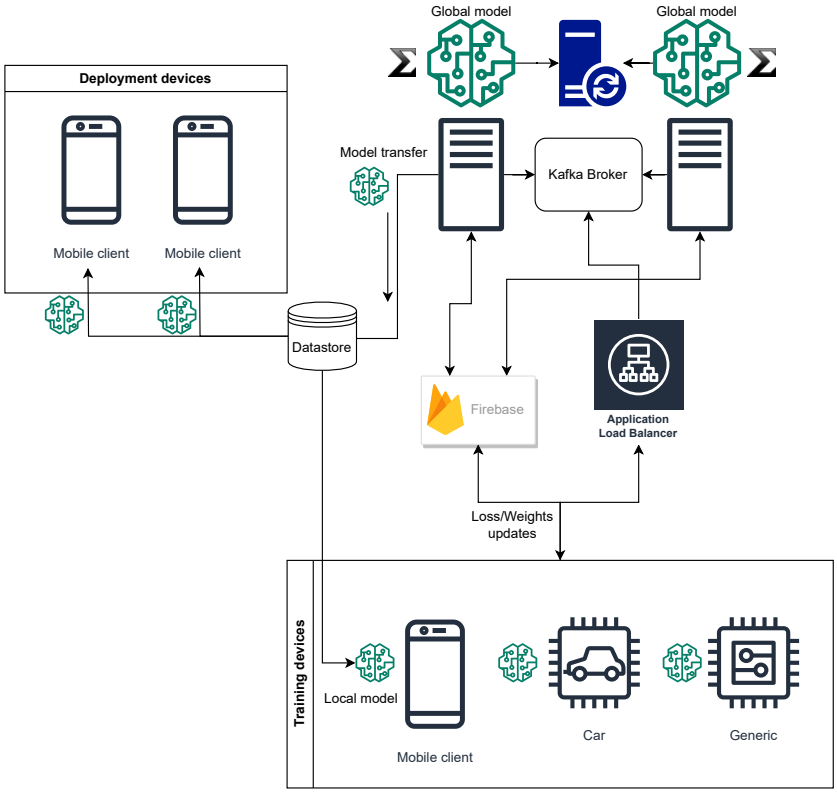

Figure 1. The components of the conceptual framework.

In this design framework, multiple central servers are employed simultaneously, and communication is facilitated using a combination of Apache Kafka [20] and a real-time database. A real-time database is a data repository that actively gathers, processes, and enhances data immediately upon their creation, ensuring that the database remains up-to-date in real time. Kafka serves as an auxiliary communication source rather than the primary one. This choice is motivated by the need to mitigate potential data loss in the event of communication failures, necessitating the implementation of a mechanism for data retransmission. It is worth noting that our design departs from the common practice of using the protobuffer method for data transfer, instead opting for text-based communication as per the specified requirements.

For the real-time database component, we have chosen the Firebase Realtime Database [21] for the development. Additionally, on the end clients, we utilize the default TensorFlow Lite [22] framework provided by Google, while the servers employ TensorFlow as their primary framework. These technology choices align with our intention to leverage the capabilities and compatibility offered by Google technologies to ensure the efficient and effective operation of the system.

4. The Proposed Communication Protocol

The communication protocol proposed in this design is designed to be compatible with both Apache Kafka and the Firebase Realtime Database. In this protocol, all information, including details about model weights, is transmitted in the JavaScript Object Notation (JSON) format. This approach ensures that the data exchanged among the various components of the system can be easily parsed, interpreted, and processed by both Apache Kafka and the real-time database, enabling seamless communication and data synchronization across the system. The learning model within this system encompasses several distinct states, including:

- Recovering from a Checkpoint: Involves restoring the model to a previously saved state.
- Training: Entails the iterative process of optimizing the model's parameters using training data.
- Resaving the Model to a Checkpoint File: Preserves the updated state for future reference.
- Exporting the Weights of the Trained Model to JSON Format: Facilitates their transfer and storage in a human-readable format.
- Importing the Weights from the JSON Format Delivered by the Server: Allows the model to incorporate the updated weights received from the server into its current state.

These defined states enable the model to undergo checkpoint recovery, training, and the exchange of model weights in a structured and coherent manner throughout the learning process.

Furthermore, an additional feature currently in development involves two approaches for calculating the average of the weights. Firstly, the average of the weights is calculated directly on the clients. Secondly, the model itself utilizes TensorFlow functions to calculate the average of the weights.

Figure 2 provides an illustration of the design and implementation of the communication protocol, which is followed by the client and server-side logic, as depicted in Figures 3 and 4. The protocol commences when the client initializes and retrieves the configuration information from the Firebase Realtime Database. This configuration includes details like the current model version. Subsequently, the client queries the database to select the server intended for training by sending a server selection message. All queries, including this one, are sent to the Firebase Realtime Database and/or Kafka.

Furthermore, the communication protocol incorporates a "learning exchange message" designed to convey critical information related to model training. This message comprises several essential components, including a designated client number, priority level, sequence number, and a type entry specifying the message's nature. The type entry distinguishes

between messages containing computed model loss and those intended to transmit updated weights for the global model.

By incorporating these elements into the learning exchange message, the protocol establishes a well-structured format for transmitting vital information relevant to model training. The client number helps identify the sender, while the priority and sequence numbers aid in organizing and prioritizing messages. The type entry allows for the precise interpretation and handling of different data types, ensuring clarity and consistency in the communication process. This standardization promotes effective coordination and synchronization in the learning exchange process between clients and servers.

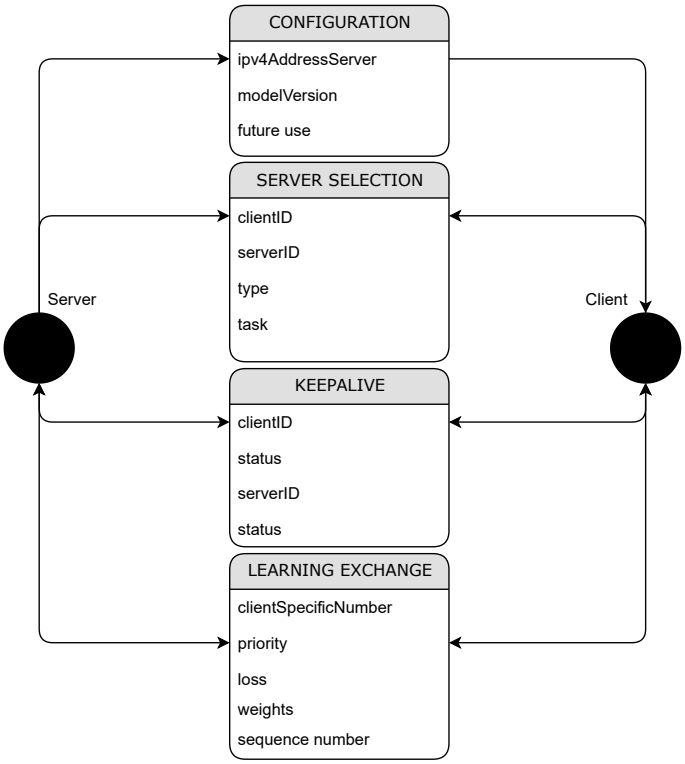

Figure 2. Protocol implementation.

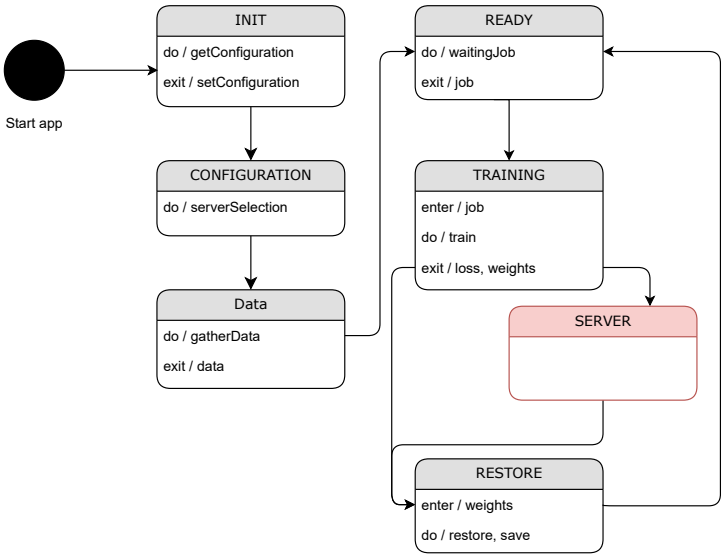

Figure 3. Client site logic.

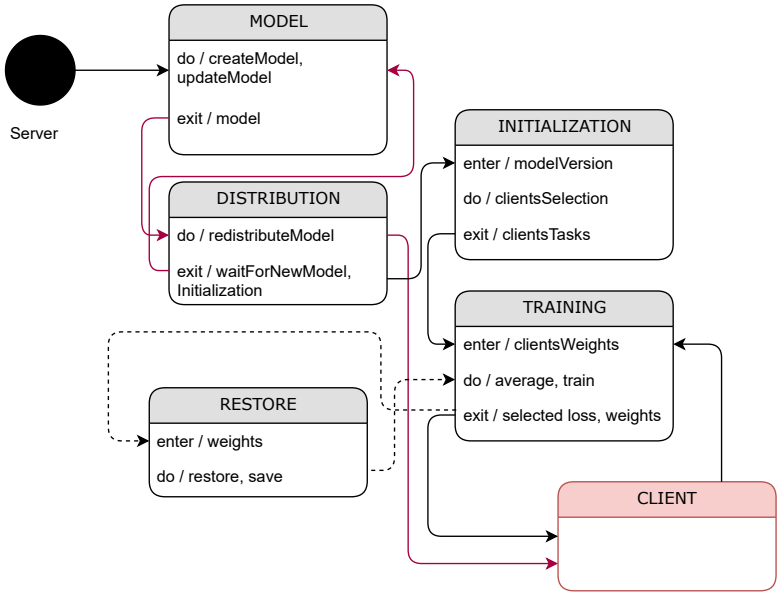

Figure 4. Server site logic.

5. Communication Protocol Simulation

To assess the efficacy of the communication protocol design, a simulation was implemented within the MATLAB R2022a Simulink software [23]. The simulation resources are available in ref. [24]. The simulation comprised three fundamental components: a server, a database, and clients. The operational characteristics of these components are delineated through state transitions, collectively constituting a comprehensive state graph for each. Inter-component communication is facilitated through the transmission of messages and events, grounded in the reciprocal exchange of states.

The constituent elements of the simulation were constructed based on the logic proposed in the communication protocol, augmented by additional logic essential for testing all conceivable scenarios that may arise during their interaction. The simulation itself was formulated to enable the emulation of communication involving any quantity of clients with a single server through a real-time database. The structural details of the individual foundational components are explicated in the subsequent sections.

The state graph corresponding to the client is illustrated in Figure 5. Upon commencement of the simulation, the client initially transitions to the INIT state, wherein it requests configuration information from the database and awaits a response. Subsequent to the configuration retrieval, the client progresses to the CONFIGURATION state, wherein it selects the server for communication. Following this, the client enters the READY state, signaling its preparedness to receive a job for processing and awaits such a job from the server. Upon receipt of a task, the client enters the GATHER_DATA state, wherein client data are collated for learning. Another state, TRAINING_MODEL, involves training the model acquired during configuration, resulting in the creation of a new local model specific to the client. Post each training round, local weights are transmitted to the server, and the client anticipates the reception of new global weights. This iterative process persists until the completion of all rounds specified in the job assignment. Subsequently, the training concludes, and the client, in the RESTORE state, obtains the newly trained global model before reverting to the READY state. This cyclic process repeats periodically, contributing to the refinement of the global model. The OFF state was incorporated into the status graph, which the client enters if it is deactivated in the simulation, thereby facilitating the simulation of random login and logout events of client devices.

The state graph corresponding to the server is depicted in Figure 6. Upon activation, the server transitions to the MODEL state, where a default data model is generated for subsequent enhancement by clients. It then proceeds to the SERVER_INTERFACE state, awaiting job requests from individual clients. Upon receipt of job requests from all reg-

istered clients, jobs are dispatched to clients in the INITIALIZATION state. The server subsequently awaits local weights from clients and, upon reception, transitions to the TRAINING_GLOBAL_MODEL state, wherein these weights are amalgamated into global weights. This process iterates until the conclusion of all rounds specified in the job. Following the overall training, a novel global model is formulated and disseminated to clients. The server reverts to the SERVER_INTERFACE state, awaiting further client task applications.

The state graph for the database, as delineated in Figure 7, is formulated to consist of message and event forwarding between the server and the database. It preserves information pertaining to the status and quantity of clients. Additionally, each component encompasses a state graph for KEEPALIVE messages, which is integral for sustaining the connection between the client and the server via the database. Within the simulation, keepalive messages are periodically dispatched from both clients and the server to the database, which, in turn, monitors potential outages of clients or the server itself, reacting accordingly to these alterations.

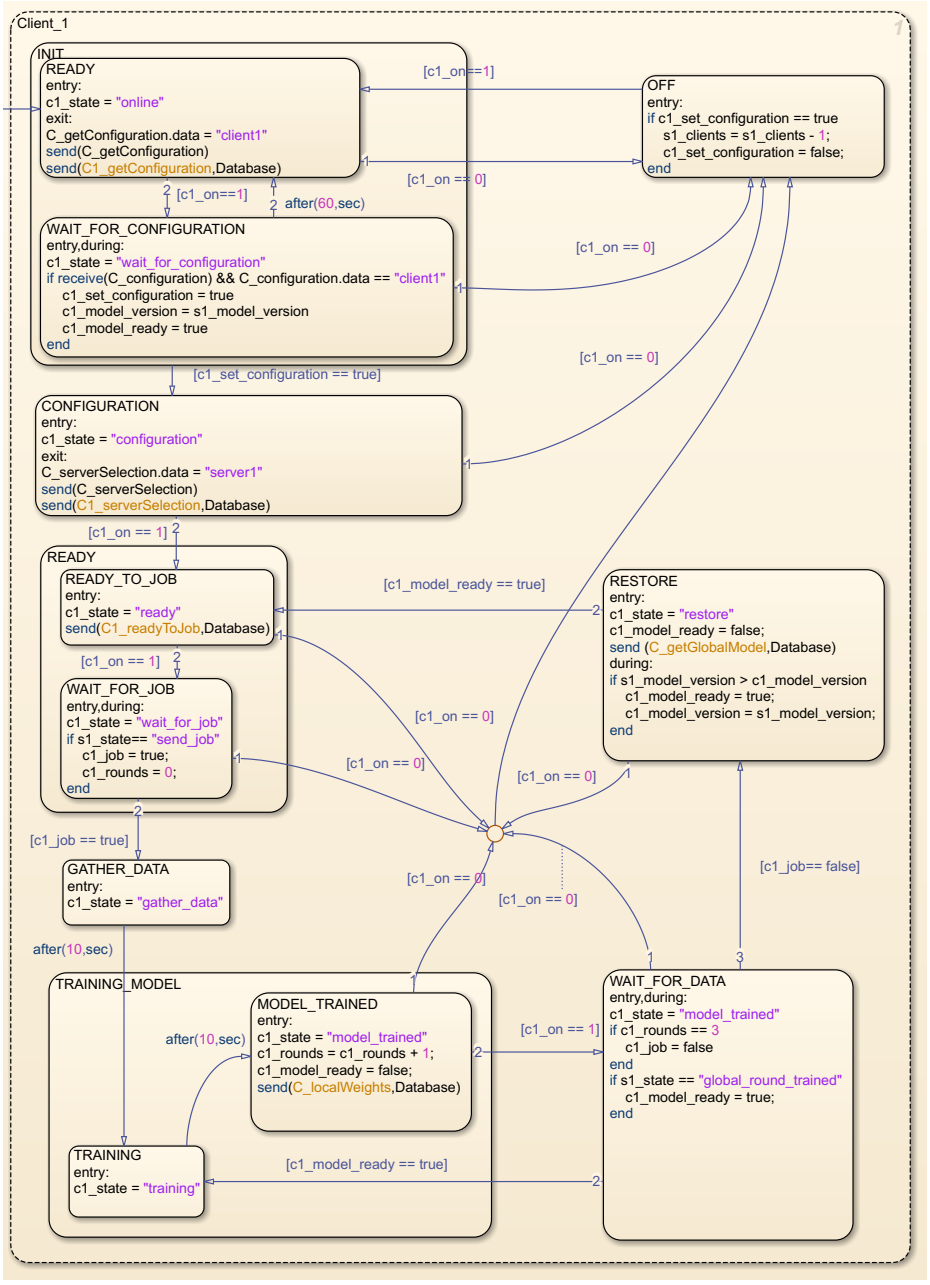

Figure 5. Client state graph.

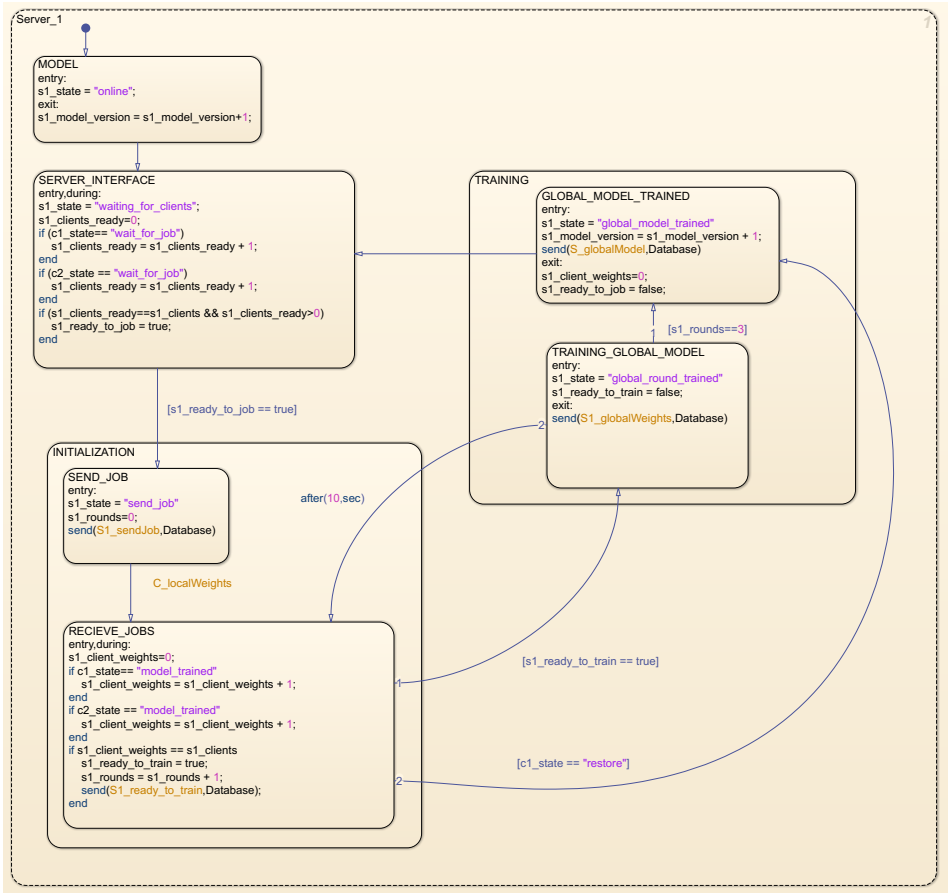

Figure 6. Server state graph.

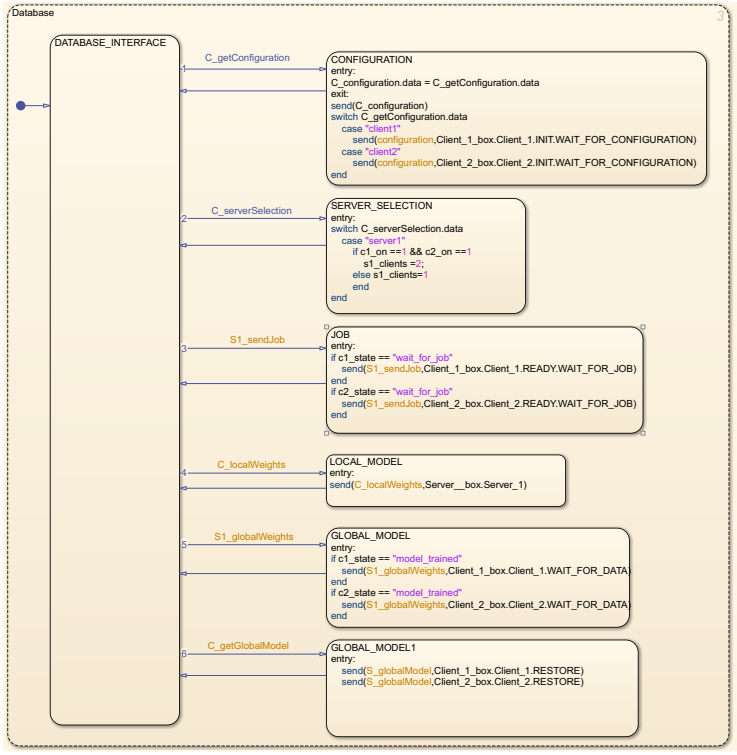

Figure 7. Database state graph.

5.1. The Simulation Results

To evaluate the designed simulation, coverage analysis was employed, specifically Modified Condition/Decision Coverage (MCDC). This analysis method assesses the extent to which the simulation testing encompasses the design, thereby gauging the effectiveness and comprehensiveness of the testing. Furthermore, MCDC aids in identifying any testing gaps, missing requirements, or unintended features by leveraging coverage data acquired during requirements-based testing. This analysis assumes particular significance in the development of systems characterized by stringent security and reliability prerequisites.

For the analysis, a model featuring a server, a database, and two clients was selected. Throughout the analysis, the clients were subjected to random on-and-off operations. The resultant summary of the analysis, encompassing over 250,000 state transitions for each component of the simulation, is presented in Figure 8. Notably, the analysis indicates the stability of the simulation, as it did not yield any error states during the examination.

Summary

Model Hierarchy/Complexity	Run 17							
		Decision		Condition		MCDC		Execution
1. FederatedSystem	158	86%		79%		50%		100%
2. . . . Federated system	157	86%		79%		50%		NA
3. . . . . SF: Federated system	156	86%		79%		50%		NA
4. . . . . SF: Client_1_box	48	81%		88%		50%		NA
5. . . . . SF: Client_1	45	79%		88%		50%		NA
6. . . . . SF: INIT	9	75%		88%		50%		NA
7. . . . . SF: READY	4	75%		NA		NA		NA
8. . . . . SF: TRAINING_MODEL	2	100%		NA		NA		NA
9. . . . . SF: Client_1_keepalive	3	100%		NA		NA		NA
10. . . . . SF: Client_2_box	48	80%		63%		25%		NA
11. . . . . SF: Client_2	45	78%		63%		25%		NA
12. . . . . SF: INIT	9	58%		63%		25%		NA
13. . . . . SF: READY	4	88%		NA		NA		NA
14. . . . . SF: TRAINING_MODEL	2	100%		NA		NA		NA
15. . . . . SF: Client_2_keepalive	3	100%		NA		NA		NA
16. . . . . SF: Database	21	91%		50%		0%		NA
17. . . . . SF: Database_keepalive	10	100%		NA		NA		NA
18. . . . . SF: Server__box	29	94%		100%		100%		NA
19. . . . . SF: Server_1	26	95%		100%		100%		NA
20. . . . . SF: INITIALIZATION	8	100%		NA		NA		NA
21. . . . . SF: TRAINING	2	100%		NA		NA		NA
22. . . . . SF: Server_1_keepalive	3	83%		NA		NA		NA

Figure 8. The summary of model coverage analysis.

Decision condition coverage is a crucial metric for tracking whether all decision conditions and branches within the model have been adequately tested and whether a range of input conditions and their combinations have been effectively addressed. This coverage metric exceeds 80 percent for the majority of components. Additionally, condition coverage analysis produced favorable results for testable conditions, thereby indicating a high degree of reliability within the model. In aggregate, the analysis underscores the well-conceived design and successful implementation of the communication protocol in the SIMULINK-based simulation. Consequently, it signifies the viability of deploying this protocol in practical operational scenarios.

5.2. Implementation

In the case of the implementation of the proposed protocol, we have so far implemented it on Android devices and performed initialization tests with the Firebase database [21]. For the implementation on Android devices, we used the TensorFlow

Lite [22] version (org.tensorflow:tensorflow-lite) for the machine learning implementation [25] and created our own functions for exporting and restoring model weights. Creating functional exports and imports of weights was the biggest challenge. The TensorFlow Lite has (at the time of writing) some limitations, discussed in Section 6. We also implemented a custom state machine that matches the design discussed above. To create Firebase functions on Android devices, we use Firebase libraries and bill of materials (BOM) com.google.firebase:firebase. The minimized version of JSON is shown in Figure 9. This format is used in the Firebase database as well as being sent to a specific topic in Kafka. All Android programming code is in Kotlin.

```
{"configuration":["",{"modelAddress":"","modelVersion":""}],"learningExchange":{"
    clientID":{".priority":"","client":"","loss":"","type":""},"clientSpecificNumber":
    {"client":"","loss":"","priority":"","sequenceNumber":"","type":"","weights":""}},
    "serverSelection":["specificKey",{"clientID":"","serverID":"","task":"","type":""}
    ],"statusKeepAlive":{"clients":{"clientID":"","status":""},"serverID":{"status":""
    }}}
```

Figure 9. The format of JSON messages.

The server part is developed in the Quart [26] environment. The reason is that it is necessary to use asynchronous functions for communication with the Firebase database and Kafka manager. The mentioned Firebase software development kit (SDK) provides tools for automatic updates when the database content changes.

6. Discussion

The incorporation of Apache Kafka and the Firebase Realtime Database ensures a versatile and reliable communication channel, addressing potential challenges such as data loss. However, it is essential to acknowledge and discuss certain limitations and considerations. The proposed framework assumes a certain level of compatibility and integration with Google technologies, such as TensorFlow and Firebase. While this enables efficient operation within the Android ecosystem, it may pose challenges for organizations or systems using alternative technologies.

Furthermore, the simulation results demonstrate the effectiveness of the communication protocol within a controlled environment. However, real-world implementation may introduce additional complexities and unforeseen challenges that require continuous evaluation and adaptation.

TensorFlow Lite on Android devices does come with several limitations and constraints:

- Constraints on Compatibility: TensorFlow Lite may not be fully compatible with all TensorFlow models and operations, especially more complex ones. Modifications or retraining might be necessary to adapt models to work with TensorFlow Lite.
- Limitations on Model Size: TensorFlow Lite imposes restrictions on model size to accommodate the memory and storage limitations of mobile devices. This constraint can affect the types and sizes of models that can be deployed on Android.
- Challenges with Custom Operations: TensorFlow Lite has limited support for custom operations, which can be challenging when working with models that rely on such operations.
- Dynamic Models: Models with variable inputs or architectures may require additional considerations to ensure compatibility with TensorFlow Lite.

Despite these limitations, TensorFlow Lite offers significant benefits, including improved performance, reduced model size, and offline inference capabilities. This makes it a valuable choice for running machine learning models on Android devices.

Additionally, during the implementation, a limitation was encountered with the Firebase Realtime Database. This limitation is specifically related to naming restrictions, which enforce specific conventions to ensure compatibility. To work effectively with the Firebase Realtime Database, it is crucial to adhere to the prescribed naming conventions and structure names accordingly, especially for weights and biases. By following these

naming restrictions, the limitations associated with the Firebase Realtime Database can be effectively managed, ensuring smooth and consistent operation within the implemented communication protocol.

The proposed KI Federated Learning Framework distinguishes itself from KafkaFed [16] through its versatile approach to communication channels. While KafkaFed primarily relies on Apache Kafka, the proposed framework integrates both Apache Kafka and the Firebase Realtime Database. This dual integration not only ensures robustness but also provides a backup communication method, mitigating potential data loss concerns. Additionally, the choice of Firebase Realtime Database facilitates real-time updates and synchronization, fostering efficient communication and coordination among distributed components.

In terms of adaptability to the Android ecosystem, the KI Federated Learning Framework stands out as it is specifically tailored for emergency management applications on Android devices. Leveraging TensorFlow Lite and Firebase, it seamlessly integrates within the Android environment, contributing to enhanced privacy and real-time decision-making capabilities. The structured communication protocol proposed by the KI Federated Learning Framework, utilizing the JSON format, adds a layer of clarity and interpretability.

The integration with Firebase Realtime Database in the KI Federated Learning Framework, while not explicitly emphasized in TFF [9], plays a crucial role. Firebase provides tools for automatic updates, enhancing the efficiency of communication between clients and the server. Moreover, the proposed framework goes a step further by incorporating adaptive learning algorithms. This feature dynamically adjusts model updates based on the changing characteristics of the infrastructure system, ensuring relevance and accuracy in real-time operational scenarios.

7. Conclusions

In conclusion, this article presents a novel and comprehensive Android federated learning framework and communication protocol designed specifically for emergency management applications. This framework addresses the unique challenges posed by such systems, including limited computational resources, diverse data sources, and rigorous security requirements. By facilitating collaborative model training while preserving data privacy and incorporating adaptive learning algorithms, this framework promises to significantly advance the field of federated learning within critical infrastructure systems.

The communication protocol, which combines Apache Kafka and a real-time database, ensures seamless and efficient data exchange, further enhancing the framework's capabilities. TensorFlow Lite on end clients and TensorFlow on servers provide an effective platform for machine learning and model training, aligning with the goals of this project.

The practical implications of the KI federated learning framework are manifold, particularly in the realm of emergency management. By dynamically adapting to real-time conditions, the framework enhances collaboration among response teams, ensuring more efficient emergency management systems. The real-time coordination facilitated by the communication protocol, compatible with Apache Kafka and Firebase, proves invaluable during crises, providing reliable collaboration channels. Moreover, the focus on Android deployment solutions broadens the framework's applicability, offering flexibility across various devices and extending its reach within emergency management scenarios. Importantly, the framework addresses privacy concerns through decentralized data processing, guaranteeing confidentiality in sensitive emergency management situations.

On the theoretical front, the research significantly contributes to the advancement of federated learning. It introduces novel communication protocols and machine learning techniques, fostering the evolution of the federated learning landscape. The proposed communication protocol establishes a theoretical foundation for effective communication within federated learning systems. Its clear structure and defined states provide a framework for structured communication, ensuring clarity and coherence in information exchange.

Looking toward implementation and commercialization, the adaptable nature of the framework extends its utility beyond emergency management. Industries with critical

infrastructure systems stand to benefit from its cross-industry applications. The protocol's compatibility with widely used technologies facilitates seamless integration into existing information systems, offering a practical solution for organizations looking to enhance their data processing capabilities. Furthermore, the framework holds potential for commercialization, especially for companies specializing in data analytics, machine learning, and emergency management solutions.

The research also opens avenues for specialized data security solutions, particularly relevant for cybersecurity companies. The emphasis on privacy-preserving federated learning suggests opportunities to develop tailored security solutions to safeguard sensitive information. Additionally, as federated learning gains importance, organizations seeking guidance in its implementation may create a demand for consultation and training services. This growing significance underscores the diverse commercialization possibilities stemming from the research. In summary, the research not only contributes to federated learning theory but also provides practical solutions with broad applications, presenting promising opportunities for commercialization across various sectors.

This research offers a valuable contribution to the field, with potential applications and implications for emergency management on Android devices. It opens up new possibilities for enhancing the performance and security of these systems, ultimately leading to increased reliability and resilience in critical operational scenarios.

Future work includes the implementation of features like checkpoint recovery, training, and structured weight exchange. With ongoing development and research, this framework holds the promise of improving emergency management applications on the Android platform, making them more secure, efficient, and reliable.

Author Contributions: Conceptualization, V.O.; methodology, V.O. and J.M.; software, J.M. and M.H.; validation, V.S.; investigation, V.O.; resources, V.S.; writing—original draft preparation, V.O. and J.M.; writing—review and editing, V.O.; visualization, V.S.; supervision, V.O.; project administration, V.O.; funding acquisition, V.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the grant of the Ministry of the Interior of the Czech Republic, Open challenges in security research, VK01030152, Android federated learning framework for emergency management applications.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in the article.

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

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programmable Interface
BOM	Bill of Materials
FedDyn	Federated Learning with Dynamic Regularization
FC	Federated Core
FATE	Federated AI Technology Enable
HyFDCA	Hybrid Federated Dual Coordinate Ascent
IoT	Internet of Things
IS	Information Systems
JSON	JavaScript Object Notation
MCDC	Modified Condition/Decision Coverage
SDK	Software Development Kit
TFF	TensorFlow Federated

References

1. Jin, W.; Fang, Y.; Liu, Y.; Yang, J.; Yu, J. Research on Emergency Management Information Business and Information System Framework. In Proceedings of the IEEE 13th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, China, 14–16 July 2023; pp. 53–59. [CrossRef]
2. Chen, N.; Liu, W.; Bai, R.; Chen, A. Application of computational intelligence technologies in emergency management: A literature review. *Artif. Intell. Rev.* **2019**, *52*, 2131–2168. [CrossRef]
3. Firebase Cloud Messaging. Available online: https://firebase.google.com/docs/cloud-messaging/ (accessed on 12 November 2023).
4. MongoDB. Available online: https://www.mongodb.com/ (accessed on 10 November 2023).
5. Li, L.; Fan, Y.; Tse, M.; Lin, K.Y. A review of applications in federated learning. *Comput. Ind. Eng.* **2020**, *149*, 106854. [CrossRef]
6. Michalek, J.; Skorpil, V.; Oujezsky, V. Federated Learning on Android-Highlights from Recent Developments. In Proceedings of the 14th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Valencia, Spain, 11–13 October 2022; pp. 27–30. [CrossRef]
7. Zhang, Z.; He, N.; Li, D.; Gao, H.; Gao, T.; Zhou, C. Federated transfer learning for disaster classification in social computing networks. *J. Saf. Sci. Resil.* **2022**, *3*, 15–23. [CrossRef]
8. Imteaj, A.; Khan, I.; Khazaei, J.; Amini, M.H. FedResilience: A Federated Learning Application to Improve Resilience of Resource-Constrained Critical Infrastructures. *Electronics* **2021**, *10*, 1917. [CrossRef]
9. TensorFlow Federated: Machine Learning on Decentralized Data. Available online: https://www.tensorflow.org/federated (accessed on 18 October 2023).
10. An Industrial Grade Federated Learning Framework. Available online: https://fate.fedai.org/ (accessed on 19 October 2023).
11. Sharma, D.; Diddee, H.H.; Jindal, S.; Grover, S. VisionAir: Using Federated Learning to Improve Model Performance on Android. Available online: https://medium.com/visionair/visionair-using-federated-learning-to-improve-model-performance-on-android-11c6c8014cf4 (accessed on 20 October 2023).
12. Corbacho, J. Federated Learning. Available online: https://proandroiddev.com/federated-learning-e79e054c33ef (accessed on 20 September 2023).
13. Mathur, A.; Beutel, D.J.; de Gusmão, P.P.B.; Fernandez-Marques, J.; Topal, T.; Qiu, X.; Parcollet, T.; Gao, Y.; Lane, N.D. On-device Federated Learning with Flower. *arXiv* **2021**, arXiv:2104.03042.
14. Acar, D.A.E.; Zhao, Y.; Navarro, R.M.; Mattina, M.; Whatmough, P.N.; Saligrama, V. Federated Learning Based on Dynamic Regularization. *arXiv* **2021**, arXiv:2111.04263.
15. Overman, T.; Blum, G.; Klabjan, D. A Primal-Dual Algorithm for Hybrid Federated Learning. *arXiv* **2022**, arXiv:2210.08106.
16. Bano, S.; Tonellotto, N.; Cassarà, P.; Gotta, A. KafkaFed: Two-Tier Federated Learning Communication Architecture for Internet of Vehicles. In Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), Pisa, Italy, 21–25 March 2022; pp. 515–520. [CrossRef]
17. Waydroid. Available online: https://waydro.id/ (accessed on 17 October 2023).
18. LineageOS Android Distribution. Available online: https://lineageos.org/ (accessed on 16 October 2023).
19. STMicroelectronics. Find the Right Solution to Integrate AI into Your Application. Available online: https://stm32ai.st.com/products/ (accessed on 12 November 2023).
20. Apache Software Foundation. Kafka. Available online: https://kafka.apache.org (accessed on 16 October 2023).
21. Moroney, L., The Firebase Realtime Database. In *The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform*; Apress: Berkeley, CA, USA, 2017; pp. 51–71. [CrossRef]
22. TensorFlow Lite. Available online: https://www.tensorflow.org/lite (accessed on 12 November 2023).
23. Documentation, S. Simulation and Model-Based Design. Available online: https://www.mathworks.com/products/simulink.html (accessed on 11 November 2023).
24. Michalek, J.; Vaclav, O. Ouje/KiProtocolSimulation: Publication (Publication). Zenodo. Available online: https://zenodo.org/records/10101429 (accessed on 10 November 2023).
25. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software. *arXiv* **2016**, arXiv:1603.04467.
26. Jones, P. Quart. Available online: https://quart.palletsprojects.com/en/latest/ (accessed on 12 November 2023).