*electronics*

MDPI

*Article*

# Evaluation of Thermal Stress on Heterogeneous IoT-Based Federated Learning †

**Yi Gu *, Liang Zhao, Tianze Liu and Shaoen Wu**

Department of Information Technology, Kennesaw State University, 680 Arntson Drive, Marietta, GA 30060, USA; lzhao10@kennesaw.edu (L.Z.); tliu11@students.kennesaw.edu (T.L.); swu10@kennesaw.edu (S.W.)

* Correspondence: ygu5@students.kennesaw.edu

† This paper is an extended version of our paper published in Proceedings of the 2024 ACM Southeast Conference (ACM SE '24), Marietta, GA, USA, 18–20 April 2024. https://doi.org/10.1145/3603287.3651222.

**Abstract:** Federated learning is a novel paradigm allowing the training of a global machine-learning model on distributed devices. It shares model parameters instead of private raw data during the entire model training process. While federated learning enables machine learning processes to take place collaboratively on Internet of Things (IoT) devices, compared to data centers, IoT devices with limited resource budgets typically have less security protection and are more vulnerable to potential thermal stress. Current research on the evaluation of federated learning is mainly based on the simulation of multi-clients/processes on a single machine/device. However, there is a gap in understanding the performance of federated learning under thermal stress in real-world distributed low-power heterogeneous IoT devices. Our previous work was among the first to evaluate the performance of federated learning under thermal stress on real-world IoT-based distributed systems. In this paper, we extended our work to a larger scale of heterogeneous real-world IoT-based distributed systems to further evaluate the performance of federated learning under thermal stress. To the best of our knowledge, the presented work is among the first to evaluate the performance of federated learning under thermal stress on real-world heterogeneous IoT-based systems. We conducted comprehensive experiments using the MNIST dataset and various performance metrics, including training time, CPU and GPU utilization rate, temperature, and power consumption. We varied the proportion of clients under thermal stress in each group of experiments and systematically quantified the effectiveness and real-world impact of thermal stress on the low-end heterogeneous IoT-based federated learning system. We added 67% more training epochs and 50% more clients compared with our previous work. The experimental results demonstrate that thermal stress is still effective on IoT-based federated learning systems as the entire global model and device performance degrade when even a small ratio of IoT devices are being impacted. Experimental results have also shown that the more influenced client under thermal stress within the federated learning system (FLS) tends to have a more major impact on the performance of FLS under thermal stress.

**Keywords:** thermal stress; federated learning systems; internet of things

check for updates

**Citation:** Gu, Y.; Zhao, L.; Liu, T.; Wu, S. Evaluation of Thermal Stress on Heterogeneous IoT-Based Federated Learning. *Electronics* **2024**, *13*, 3140. https://doi.org/10.3390/electronics13163140

Academic Editor: Fernando De la Prieta Pintado

Received: 28 June 2024
Revised: 29 July 2024
Accepted: 6 August 2024
Published: 8 August 2024

**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

Conventionally, training a machine learning model involves transferring raw data to a central location for computation and analysis. However, this centralized approach can be impractical due to factors such as large datasets, user privacy concerns, or regulatory restrictions on data collection. To address these issues, Google introduced federated learning [1] in April 2017. Initially tested on Gboard for Android phones, federated learning enables collaborative machine learning on distributed entities like Internet of Things (IoT) devices. In federated learning, models are trained locally on clients, and only the gradients or model parameters are sent to a central server for aggregation. After years of real-world practice, federated learning is proven to allow model training in distributed

and heterogeneous devices. Heterogeneous devices, as their name suggests, are a set of devices with more than one kind of processor or computing core. In real-world IoT systems, heterogeneous devices are very common, as IoT devices working in the same system tend to have various designs and products. In addition, sharing model parameters instead of raw data has made federated learning a promising solution when protecting user privacy during machine learning and has become more important recently. However, as federated learning is typically performed in distributed systems with multiple potentially untrusted devices, it is easier for the federated learning system (FLS) to be impacted and influenced [2]. In particular, when clients in FLS are low-end IoT devices with limited resources, impacts like thermal stress can easily prevent clients from offering their best performance.

Computing devices consume energy and produce heat, which are side effects during operation. If the heat generated becomes excessive and cannot be dissipated timely, it can lead to performance reductions or even physical damage to computing devices. Thermal stress is a sort of impact that has various implementations, such as creating excessive heat and using the heat to cause damage to the performance or even construction of the impacted system. In the context of federated learning, thermal stress may disrupt the normal functioning of the FLS, thus leading to adverse consequences, including decreased CPU/GPU frequencies, longer training times, and increased power consumption, ultimately reducing overall system performance. Our previous work [3] constructed a real-world FLS on isomorphic IoT devices, simulated thermal stress, and conducted preliminary evaluations and analyses.

***Study Objective*** In this paper, we first extended our previous real-world homogeneous FLS [3] on IoT devices to a real-world heterogeneous FLS, still based on the flower framework [4]. Secondly, we simulated thermal stress on a GPU for Jetson Nanos, as in our previous work [3], and also added a simulation of thermal stress on CPU for Raspberry PIs in this work. The simulation tools we used include Jetson Benchmarks [5] and Stress-ng [6]. Thirdly, we evaluated the performance with various measurements and metrics using Jetson-Stats [7] for Jetson Nanos, as in our previous work [3], and we chose a new platform called Glances [8] for measuring Raspberry PIs. Fourthly, we analyzed the influence of thermal stress on our heterogeneous FLS.

***Research Gap*** Currently, few studies have focused on federated learning systems (FLS) deployed on real-world IoT devices, and even fewer have examined the impact of thermal stress on heterogeneous FLS. Our work is among the first to address this gap. Additionally, we improved the homogeneous FLS from our previous work to develop a heterogeneous FLS by incorporating Raspberry PI4 clients alongside Jetson Nano clients. This research aims to explore the effect of thermal stress on real-world heterogeneous IoT-based FLS.

***Hypotheses*** We have two hypotheses within our current study. The first one is that we consider thermal stress may affect heterogeneous IoT-based FLS. The second one is that we considered the ratio difference of our FLS clients under thermal stress, which may have different impacts on the FLS. That is, even if only one client is under thermal stress, the performance of FLS may be affected. As the ratio of clients under thermal stress increases, the influence of thermal stress on FLS may increase accordingly.

***Contributions*** Our prominent contributions in this paper are summarized as follows:

- To the best of our knowledge, the present work is among the first to evaluate the performance of federated learning under thermal stress on real-world heterogeneous IoT-based systems.
- We conducted experiments of various scenarios when federated learning clients are under different thermal stresses with measuring metrics, including CPU utilization rate, GPU utilization rate, temperature, and power consumption. We added 67% more training epochs compared with our previous work [3].
- We varied the proportion of clients under thermal stress in each group of experiments and systematically quantified the effectiveness and real-world impact of thermal stress on the low-end, heterogeneous, IoT-based federated learning system. We added 50% more clients compared with our previous work [3].

## 2. Related Work

***Federated Learning***. Federated learning was first introduced by Google [1] in April 2017, being tested and trained on Gboard for Android phones. IoT devices such as mobile phones and tablets often store data, including sensitive personal information, in their memory. Collecting these data centrally for traditional machine learning is not only resource-consuming but also poses significant privacy risks. Unlike traditional machine learning methods, federated learning trains the global model collaboratively on IoT devices in which data are located, transferring only trained model parameters and gradients instead of data to the server. These parameters and gradients are then processed on the server using specific algorithms. McMahan et al. [9] introduced the federated average algorithm (aka FedAVG), which later became a foundational method for aggregating updates received at the server. Google proposed the TensorFlow federated (TFF) framework [10], and Bonawitz et al. introduced FedScale [11], both supporting single-node simulation. Ryffel et al. proposed PySyft [12], and Beutel et al. introduced flower [4]. The frameworks support additional functions like multi-node execution and heterogeneous computation.

***Thermal Stress***. Thermal stress can weaken performance or even damage the structure of affected systems, leading to possible unfavorable consequences such as communication conversion, excessive energy consumption, and heat-led hardware damage. Previous research has revealed various negative impacts of thermal stress from different perspectives. Masti et al. [13] performed RSA decryption on specific CPU cores on edge devices, as well as using thermal side channels for communication, showing thermal stress and its possibility of influencing the work of edge devices and communication between edge devices. Tian et al. [14] also found a similar phenomenon among users who rent the same FPGA over a period of time; certain thermal channels can be manipulated to converted channels and lead to manipulation in communication. Kong et al. [15] found that certain malicious commands can lead to fine-grained and specified over-temperature spots, thus causing certain physical damage in the instruction cache. Gao et al. [16] used workloads that can cause excessive thermal changes to raise the temperature in data centers to a high situation and conducted measures on thermal and proposed effective thermal stress vectors, all to reveal the vulnerability of data centers and areas likely tp face thermal stress. In follow-up work, Gao et al. conducted additional testing on thermal stress under various scenarios and with thermal-related metrics measured [17]. Duchatellier et al. [18] studied the effect of thermal stress on edge devices and the vulnerability of cloud-edge systems. Jaspinder et al. [19] studied RSPP and related thermal side-channel attacks and their influence. Our previous work [3] constructed a real-world FLS on isomorphic IoT devices, simulated thermal stress on FLS clients, and conducted preliminary evaluations and analyses on the measurement of metrics like CPU or GPU utilization rate to figure out the impact of thermal stress on FLS.

***Summary*** Table 1 is an author contribution table that illustrates the current literature on related research topics. We can conclude from Table 1 that, in the current literature, only a limited amount of work focuses on FLS deployed on real-world IoT devices, and even less work has been done to evaluate the influence of thermal stress on real-world IoT-based FLS. This study addresses the research gap in understanding how thermal stress affects the performance and reliability of heterogeneous IoT-based FLS, which is essential for the practical deployment of FLS in various real-world applications. Our work is among the first attempts to carry out thermal stress on heterogeneous IoT-based FLS. Compared with our previous work [3], we added 50% more clients within our FLS, added 67% more training epochs within training rounds, extended our thermal stress varieties by adding thermal stress on CPUs, and chose new tools for measuring the metrics within FLS. We also improved our FLS from a homogeneous FLS within our previous work [3] to a heterogeneous FLS within this study, adding three Raspberry PI4 clients to our previous Jetson Nano clients. We consider both FLS and thermal stress, aiming to understand the effect and impact of thermal stress on real-world heterogeneous IoT-based FLS. This study not only enhances the understanding of the effects of thermal stress on FLS but

also provides valuable insights and methodologies for future research and development in this field.

**Table 1.** The authors contribution table.

| Authors | Content | FLS Type | Node(s) | Thermal Influence |
|---|---|---|---|---|
| Google (2017) [1] | World's first FLS on Gboard | Real World | Multi | N/A |
| McMahan et al. (2017) [9] | FLS aggregation algorithm | N/A | N/A | N/A |
| Google (2020) [10] | Tensorflow FLS | Simulated | Single | N/A |
| Bonawitz et al. (2019) [11] | FedScale FLS | Simulated | Single | N/A |
| Ryffel et al. (2018) [12] | PySyft | Simulated | Multi | N/A |
| Beutel et al. (2020) [4] | Flower | Real World | Multi | N/A |
| Masti et al. (2015) [13] | RSA decryption on specific CPU cores | N/A | N/A | Thermal side channels |
| Tian et al. (2019) [14] | FPGA TDM attack | N/A | N/A | Thermal channels manipulation |
| Kong et al. (2010) [15] | Malicious commands | N/A | N/A | Certain overheat spot |
| Gao et al. (2017) [16] | Excessive workloads | N/A | N/A | Thermal Stress |
| Gao et al. (2018) [17] | Excessive workloads under multiple scenarios | N/A | N/A | Thermal stress |
| Jaspinder et al. (2024) [19] | RSPP and thermal side-channel attacks | N/A | N/A | Thermal side-channel attacks |
| Gu et al. (2024) [3] | Effect of thermal stress on IoT-based, homogeneous, real-world FLS | Real World | Multi homogeneous | Thermal Stress |
| This paper | Effect of thermal stress on IoT-based, heterogeneous, real-world FLS | Real World | Multi Heterogeneous | Thermal Stress |

The "N/A"s within the table means the authors didn't mention related content within their paper.

## 3. Methodology

### *3.1. System Description*

In an FLS, clients train a global model collaboratively under the coordination of a central server. Each client trains its local data for a local model, and the central server carries out a weighted aggregation of local models to formulate a global model. An iteration of federated learning is as follows. (1) All clients download the global model $W_{t-1}$ from the central server. (2) Client $k$ trains its local data to obtain a local model $W_{t,k}$ (local model for client k in the t-th round of communication). (3) All clients upload updated local model parameters and gradients to the central server. (4) After receiving all data, the central server carries out weighted aggregation using algorithms like FedAVG or FedBN to obtain the global model named $W_t$ (the global model in the t-th round of communication). After multiple rounds of iterations, a final model $W$ will be produced, which is close to the results of centralized machine learning under the same model as the global model using the same dataset. Our FLS also works under the same principle mentioned above.

To set up our FLS as a heterogeneous real-world IoT-based FLS, we decided to set our FLS clients with heterogeneous IoT devices. After the evaluation of device capability, we decided to choose three Jetson Nanos and three Raspberry PI4s as our clients, as they have different CPU cores and differences in GPU equipment, thus meeting the definition of heterogeneity.

### *3.2. Stress Implementation*

Our thermal stress approach aims to generate excessive heat within the FLS, leveraging this heat or its side effects to impact system performance or cause damage. In an FLS, clients are typically deployed on devices with fewer resources than the central server, making them more susceptible to thermal stress. Heat is primarily generated on IoT devices during computation, with more resource-intensive programs producing more excessive heat over the same duration on the same device. Therefore, to create excessive heat, we selected highly resource-demanding programs, which were loaded onto IoT devices acting as clients and run concurrently with the FLS, simulating thermal stress on our FLS. This simulated thermal stress affects the first three steps of a federated learning iteration. Figure 1 illustrates the general framework of our FLS and the simulated thermal stress on the system. Using these methods, we can simulate and evaluate thermal stress on the FLS.

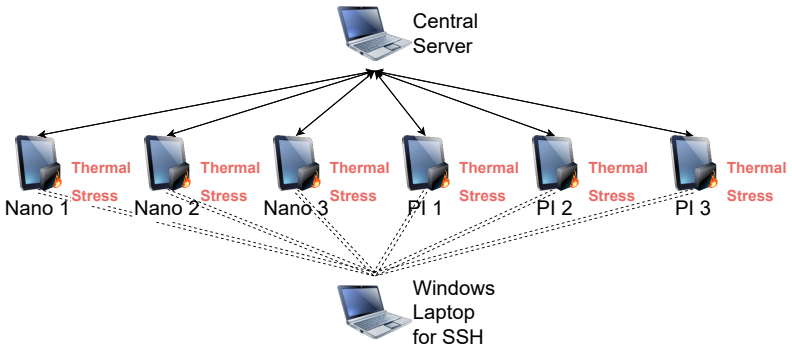
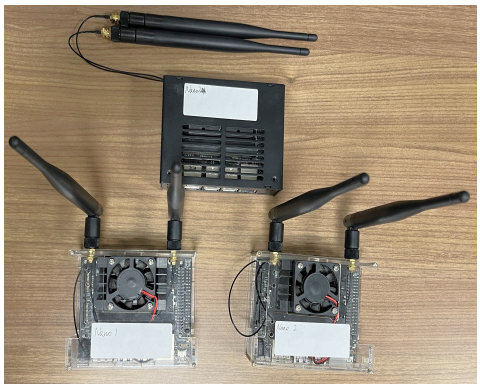



**Figure 1.** Schematic diagram of federated learning system under thermal stress.

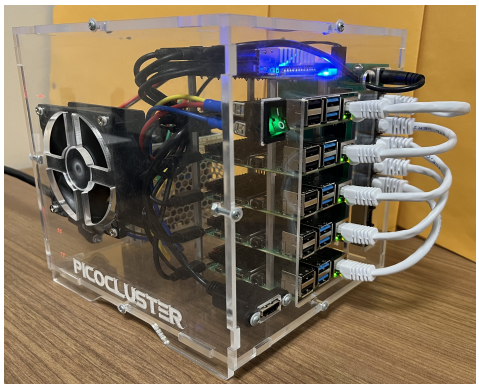
## 4. Experimental Evaluation

### *4.1. Evaluation Methodology*

To create a real-world multi-node heterogeneous FLS and see how it performs under thermal stress, we chose to use the flower framework. In this paper, we chose three Jetson Nanos and three Raspberry PI4s as clients and a Lambda laptop as the central server, using gRPC Protocol to conduct client–server communication. We also chose another Windows laptop SSH for our six clients for the convenience of our control. Figure 2 shows the real-world picture of our FLS clients. In the left part of Figure 2, each of the three Jetson Nanos in the picture reflects one of the clients. In the right part of Figure 2, the PI4 cluster contains five PI4s in total, and we chose to use three of the PI4s within the cluster for our evaluation.



(a) Our Jetson Nano clients within the federated learning system




(b) Our Raspberry PI clients within the federated learning system

**Figure 2.** Our federated learning system experiment setup.

As we can see from the figure, we named the three Jetson Nanos as Nano1, Nano2, and Nano3. As for the Raspberry PI4s, we named them PI1, PI2, and PI3, respectively. We named the Lambda laptop as Lambda. The configurations of the seven devices are shown in Table 2. Note that Nano1 and Nano2 share the same configuration; Nano3 has its own configuration. PI1, PI2, and PI3 share the same configuration. Also, considering that the Windows laptop serves only SSH, its configuration has no effect on our framework and thus will not be mentioned.

**Table 2.** The configuration of devices running a federated learning system.

| | **Lambda** | **Nano1-2** | **Nano3** | **PI1-2-3** |
|---|---|---|---|---|
| CPU | Intel i7-12800H | ARM A57 | ARM A57 | ARM A72 |
| CPU Core Num | 14 | 4 | 4 | 4 |
| GPU | RTX 3080 Max-Q | Maxwell | Maxwell | N/A |
| GPU Core Num | 6144 | 128 | 128 | N/A |
| FAN | Yes | Yes | No | Yes |
| MEM | 32 GB | 4 GB | 4 GB | 8 GB |
| Disk | 1 TB | 128 GB | 128 GB | 128 GB |

The "N/A"s within the table means the device don't have certain parameters.

In this paper, we chose to use the "embedded-devices" example on flower's GitHub official website [20] as our FLS basis. We then carried out further customization on the provided version of the aggregation algorithm FedAVG to make sure our six clients could be functional in the evaluation simultaneously. We also modified the provided server and client scripts to make sure arguments can be specified while running the scripts using terminal mode. For the server, the only thing to do is get server files ready and have certain environments installed properly, as required in the requirement description file. For a Jetson Nano FLS client, the first thing to do is to install JetPack 4.6.1 on the 128 GB microSD card. Note that the storage of the micro SD card should not be below 64 GB or certain issues of not having enough storage left will occur in later stages. Set the device following system instructions step by step. Create a new user group in the complimentary docker with NVIDIA Jetson and add a new user to it. After that, get the FLS client's files ready and run them in the terminal to create a docker image for FLS clients. Later, FLS clients will be running in the docker images. Also, considering that the latest version of CUDA supported on our Jetson Nanos is outdated, we chose to use the CPU to run clients. For a Raspberry PI4 client, the first thing to do is install the Ubuntu Server 64-bit version operation system. We chose the Ubuntu Server 22.04.03 LTS. We then got all the required client-side files and environment requirements ready using the requirement description file provided. Clients on Raspberry PI4s will run locally on the devices using a CPU, considering they do not have GPUs within the devices.

For this paper, we trained clients using the basic CNN model under the Pytorch 1.13.1 [21] framework. We also chose to use MNIST [22] as our dataset. MNIST, with the full name of Mixed National Institute of Standards and Technology, is a dataset composed of handwritten digits. With the purpose of classifying handwritten digits, MNIST is one of the most widely used datasets in machine learning. The sheer size of the dataset helps the predictive analytics of deep learning. This dataset contains 60,000 images within the training set and 10,000 images within the testing set, formatted as $28 \times 28$ pixel monochrome images. In our training, the 60,000 images within the training set are evenly split into six partitions, each assigned to a different client. For each client, the partition of the training set is then divided, with 90% for training and 10% for validation. The training rounds are assigned on the server side and are set to 5, the training epoch in each round is set to 2. After each round, an aggregation to the server is made, and testing is carried out after aggregation. Also, we set it on the server side to make sure all clients connected to the server are sampled.

### *4.2. Thermal Stress Simulation*

Our FLS clients consist of three Raspberry PIs and three Jetson Nanos, all of which are IoT devices. Of all the components in an IoT device, the CPU and GPU are the primary parts responsible for running high-computing, resource-demanding programs, making them the main sources of excessive heat. To simulate thermal stress, we can deploy programs that heavily consume CPU or GPU resources while the FLS is operational. During our preliminary experiments, we observed some significant adverse effects of thermal stress. Since Raspberry PIs only have CPUs, we could only deploy high-CPU-consuming programs while the FLS is running. Initially, we used a typical CPU stress test provided by Stress-ng [6] on Raspberry PIs, which resulted in a 120-fold increase in FLS running time for the same task. For Jetson Nanos, running high-CPU-consuming programs while the FLS was active consistently led to CPU overload and system breakdowns. This could be because both the FLS clients and the high-CPU-consuming programs were running on the CPUs, leaving insufficient computing resources for the system to function properly.

To achieve a more systematic and quantified evaluation, we aimed to apply an appropriate amount of thermal stress on our FLS clients to ensure that thermal stress affects the FLS while still allowing it to function. Based on preliminary experiments, we chose Jetson Benchmarks [5] for high-GPU-consuming programs on Jetson Nanos and Stress-ng [6] for high-CPU-consuming programs on Raspberry PIs. Note that these high-computation, resource-consuming programs were only be deployed on our FLS clients.

Jetson Benchmarks, provided by Nvidia, include Inception V4, ResNet-50, OpenPose, VGG-19, YOLO-V3, Resolution, and Unet. These benchmarks utilize GPU + 2DLA and are designed to test the extreme performance of Jetson devices, making them ideal for our simulation. To run these benchmarks, we first needed to prepare the environment requirements, then downloaded the models and a CSV file containing all parameters, and finally, run the benchmark scripts via the terminal. Running all benchmarks on Jetson Nano typically takes no less than two hours.

Stress-ng [6] is a versatile and portable stress test suite compatible with various computer systems. It was initially designed to push devices to their limits to uncover hardware issues like thermal over-runs and operating system bugs under heavy utilization. Stress-ng can stress-test a system by targeting different physical subsystems and various operating system kernel interfaces. It offers over 80 CPU-specific stress tests that exercise floating-point operations, integer calculations, bit manipulation, and control flow. To align with the high-GPU-consuming programs on Jetson Nanos and ensure the FLS runs within a reasonable timeframe, we decided to stress only one of the four CPU cores on Raspberry PIs for a period of two hours to simulate thermal stress.

### *4.3. Measurements of Metrics*

Several metrics of FLS clients were considered, and specific tools were chosen to monitor these metrics. The purpose of these measurements was to evaluate the performance of FLS clients both during normal operation and under thermal stress in order to understand the impact of thermal stress.

#### 4.3.1. Training Time and Accuracy

Our FLS trains machine learning models on specific datasets. The accuracy of the trained model is an important factor in determining the performance of FLS. On the other hand, considering that our FLS carries out tasks with the same load in each group of experiments, system running time is also a very important factor in assessing the performance of FLS. As a complementary function, FLS clients automatically log the length of training time and evaluation time for each training round. The FLS server automatically logs the training accuracy of each round. By documenting and analyzing these logs, we can compare the training times and accuracies of clients under normal conditions and under thermal stress, allowing us to observe the differences.

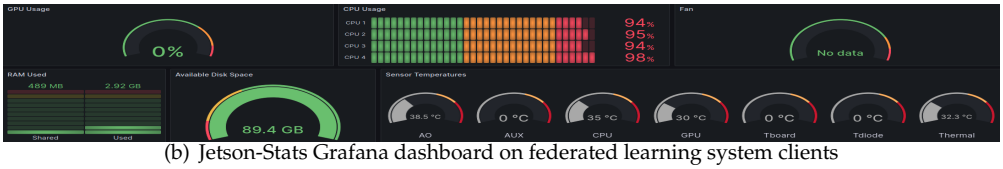
#### 4.3.2. CPU and GPU Utilization Rate and Total Energy Consumption

To determine the effect of the thermal stress, the utilization rate of FLS clients can be documented and analyzed. As our FLS runs on a CPU and thermal stress runs on a CPU or GPU, the utilization rate of the CPU and GPU can be key values to see how thermal stress will affect our clients. Also, considering thermal stress is highly demanding regarding computer resources, and thus, might cost extra energy, total energy consumption (TOT) can be another key metric to see the influence of thermal stress. For Jetson Nanos, all three can be documented by jtop-logger provided by Jetson-Stats [7], which is a Python file and automatically logs the condition of our FLS client's system into a CSV file on a one-second per log basis. For Raspberry PIs, however, it has no GPU, and there are no methods to derive its total energy consumption. Thus, we can only determine the CPU utilization for evaluation. For monitoring, we chose glances [8], an open-source system cross-platform monitoring tool that allows for the real-time monitoring of various aspects of the system, such as CPU, memory, disk, network usage, etc.

We also have another timely displayed Jetson-Stats-Grafana dashboard [23] to show the real-time utilization rate of the CPU, GPU, TOT, and other metrics. The dashboard first collects Jtop-gathered data using a certain API provided and uses certain scripts to regulate data into certain forms that can be accepted and then transfer those data to the host running Prometheus [24]. Then, it has another host running Grafana [25], which is the platform for our dashboard. By importing certain dashboard distribution files and pulling data from Prometheus, Grafana can offer a timely display of utilization rates in certain metrics. We also modified the Jetson-Stats-Grafana dashboard for it to fit our Jetson Nano clients. As for Raspberry PIs, Glances comes with an easy-to-read dashboard. Figure 3 shows our Jetson-Stats-Grafana dashboard on Jetson Nano1 without any other program running, and Figure 4 is the dashboard of our FLS clients while training the local model.




(a) Jetson-Stats Grafana dashboard on Jetson Nano1




(b) Jetson-Stats Grafana dashboard on federated learning system clients

**Figure 3.** Jetson-Stats Grafana dashboards.

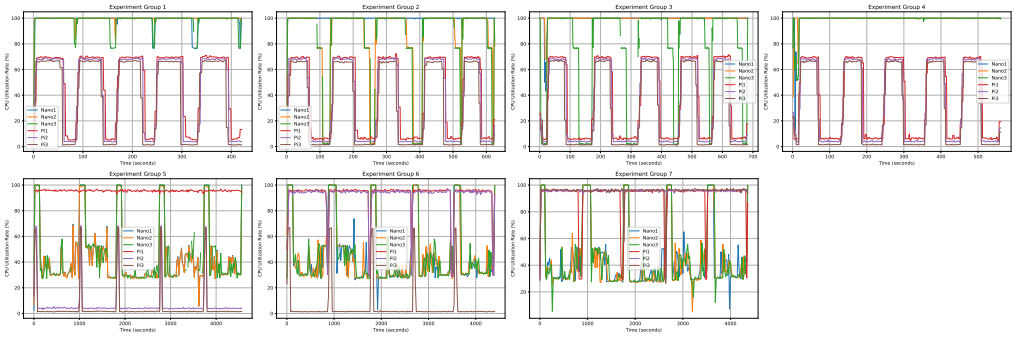



**Figure 4.** CPU utilization rate.

#### 4.3.3. Temperature

While inducing thermal stress, temperature plays a vital part in all metrics. By evaluating the temperature of FLS clients under or not under thermal stress, we can easily get to know how FLS clients are influenced under thermal stress from this perspective. On Jetson Nanos, we chose to use Jtop-stats to log temperature-related data and used the Jetson-Stats-Grafana dashboard for real-time display. On Raspberry PIs, we chose to use Glances for the same function.

### *4.4. Experiment Process and Outcomes*

In this paper, we set seven groups of experiments to systemically qualify how thermal stress influences the FLS. Each group of experiments is conducted on a real-world FLS running on a server and six clients. The server and the six clients are exactly as we listed in Table 2. We also offer the maximum power support for our clients, which are the 5V4A powerlines for Jeston Nanos and 5.1V3A USB-C powerlines for Raspberry PIs. For each group of experiments, the settings related to FLS are all the same. The FLS trains a CNN network from Pytorch [21] on the MNIST [22] dataset. Training contains five rounds and two epochs within each round. After each round of training, an evaluation will be processed. The MNIST [22] dataset that we chose is randomly distributed into six even shares; each of the clients has a specific share of data assigned to itself locally.

Within the seven groups of experiments, the number of clients under thermal stress varies from 0 to 6, meaning the ratio of clients under thermal stress varies from 0% to 100%. A detailed description of which clients are under thermal stress within each group of experiments is as follows. For the first group, all clients run federated learning without thermal stress. For the second group, one client (Nano1) runs federated learning under thermal stress, while the other three clients run federated learning without thermal stress. For the third group, two clients (Nano1 and Nano2) run federated learning under thermal stress, while the other two clients run federated learning without thermal stress. For the fourth group, three clients (Nano1, Nano2, and Nano3) run federated learning under thermal stress while the other client runs federated learning without thermal stress. For the fifth group, four clients (Nano1, Nano2, Nano3, and PI1) run federated learning under thermal stress, while the other client runs federated learning without thermal stress. For the sixth group, five clients (Nano1, Nano2, Nano3, PI1, and PI2) run federated learning under thermal stress, while the other client runs federated learning without thermal stress. For the seventh group, all clients run federated learning under thermal stress, while the other clients run federated learning without thermal stress.

In addition to the FLS and thermal stress simulation, we also set up data-logging tools and system-monitoring dashboards and had them running while we conducted our experiments. For Jetson Nanos, we run jtop-logger as the data logging tool, and the Jetson-Stats-Grafana dashboard was used as the system-monitoring dashboard. For Raspberry PIs, we run Glances as both the data-logging tool and the system-monitoring dashboard. Both data-logging tools and dashboards are run as long as the FLS is running to collect real-time metrics from clients and to monitor real-time client performances. Another thing to mention is that after each experiment, we downloaded the automatically generated FLS log from the server and six clients, which contains system running time, training accuracy, and other related metrics measurements.

All groups of experiments were carried out successfully. However, we failed to run the Grafana dashboard on any client under thermal stress, with any attempt ending in system congestion and crash conditions. For utilization rates and temperate, we listed out all our results from Figures 4–7. For training time and accuracy, we have general time and accuracy data listed in Table 3 and detailed training and evaluation time within round 1 listed in Table 4.

**Table 3.** Running time and accuracy for experiments.

| | Running Time | Acc RND1 | Acc RND2 | Acc RND3 | Acc RND4 | Acc RND5 |
|---|---|---|---|---|---|---|
| EXP1 | 0:07:01 | 0.9630 | 0.9835 | 0.9857 | 0.9882 | 0.9887 |
| EXP2 | 0:10:25 | 0.9757 | 0.9852 | 0.9878 | 0.9912 | 0.9908 |
| EXP3 | 0:10:49 | 0.9658 | 0.9813 | 0.9853 | 0.9887 | 0.9887 |
| EXP4 | 0:11:21 | 0.9585 | 0.9810 | 0.9840 | 0.9870 | 0.9872 |
| EXP5 | 1:16:03 | 0.9755 | 0.9863 | 0.9883 | 0.9900 | 0.9905 |
| EXP6 | 1:18:21 | 0.9703 | 0.9823 | 0.9851 | 0.9886 | 0.9881 |
| EXP7 | 1:19:80 | 0.9658 | 0.9727 | 0.9858 | 0.9875 | 0.9887 |

**Table 4.** Training and evaluating time for Round 1 clients.

| | E1 TRN | E2 TRN | R1 EVA |
|---|---|---|---|
| EXP1Nano1 | 00:35 | 00:34 | 00:01 |
| EXP1Nano2 | 00:33 | 00:33 | 00:02 |
| EXP1Nano3 | 00:35 | 00:36 | 00:02 |
| EXP1PI1 | 00:27 | 00:26 | 00:01 |
| EXP1PI2 | 00:26 | 00:27 | 00:01 |
| EXP1PI3 | 00:26 | 00:26 | 00:01 |
| EXP2Nano1 | 00:56 | 00:45 | 00:02 |
| EXP2Nano2 | 00:37 | 00:37 | 00:02 |
| EXP2Nano3 | 00:38 | 00:36 | 00:01 |
| EXP2PI1 | 00:27 | 00:27 | 00:01 |
| EXP2PI2 | 00:27 | 00:26 | 00:01 |
| EXP2PI3 | 00:25 | 00:26 | 00:01 |
| EXP3Nano1 | 00:50 | 00:53 | 00:02 |
| EXP3Nano2 | 00:47 | 00:47 | 00:02 |
| EXP3Nano3 | 00:35 | 00:33 | 00:02 |
| EXP3PI1 | 00:25 | 00:27 | 00:01 |
| EXP3PI2 | 00:26 | 00:25 | 00:01 |
| EXP3PI3 | 00:27 | 00:25 | 00:01 |
| EXP4Nano1 | 00:58 | 00:51 | 00:02 |
| EXP4Nano2 | 00:50 | 00:51 | 00:02 |
| EXP4Nano3 | 00:54 | 00:54 | 00:02 |
| EXP4PI1 | 00:28 | 00:27 | 00:01 |
| EXP4PI2 | 00:27 | 00:27 | 00:01 |
| EXP4PI3 | 00:25 | 00:26 | 00:01 |
| EXP5Nano1 | 00:46 | 00:52 | 00:03 |
| EXP5Nano2 | 00:55 | 00:50 | 00:04 |
| EXP5Nano3 | 01:00 | 00:59 | 00:03 |
| EXP5PI1 | 06:38 | 06:35 | 00:12 |
| EXP5PI2 | 00:26 | 00:27 | 00:02 |
| EXP5PI3 | 00:28 | 00:26 | 00:01 |
| EXP6Nano1 | 00:52 | 00:46 | 00:02 |
| EXP6Nano2 | 00:43 | 00:45 | 00:02 |
| EXP6Nano3 | 00:46 | 00:46 | 00:02 |
| EXP6PI1 | 06:34 | 06:27 | 00:12 |
| EXP6PI2 | 06:38 | 06:36 | 00:11 |
| EXP6PI3 | 00:25 | 00:27 | 00:02 |
| EXP7Nano1 | 00:52 | 00:49 | 00:03 |
| EXP7Nano2 | 01:00 | 01:00 | 00:02 |
| EXP7Nano3 | 00:45 | 00:44 | 00:02 |
| EXP7PI1 | 06:29 | 06:48 | 00:12 |
| EXP7PI2 | 06:31 | 06:49 | 00:11 |
| EXP7PI3 | 06:36 | 06:25 | 00:10 |

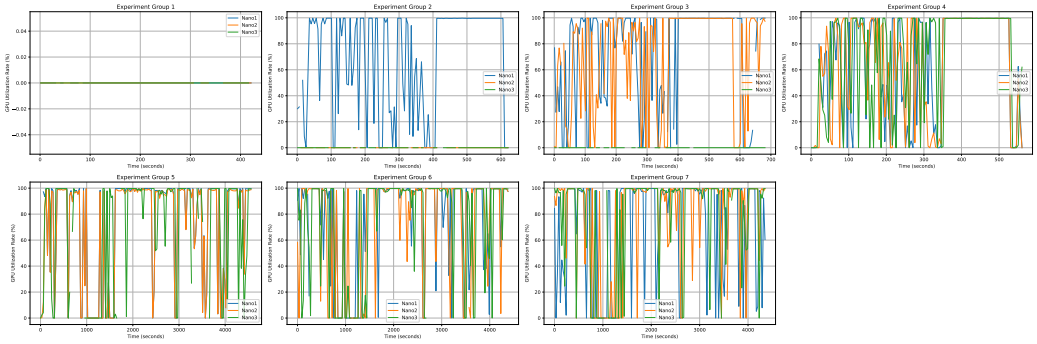



**Figure 5.** GPU utilization rate.

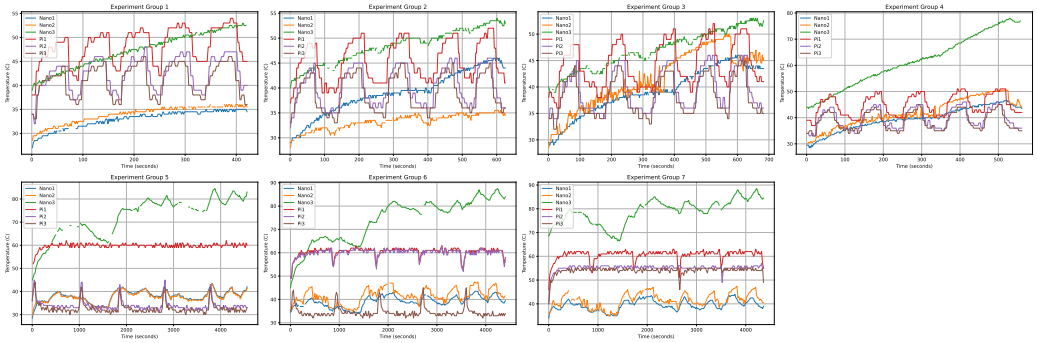



**Figure 6.** Temperature utilization rate.

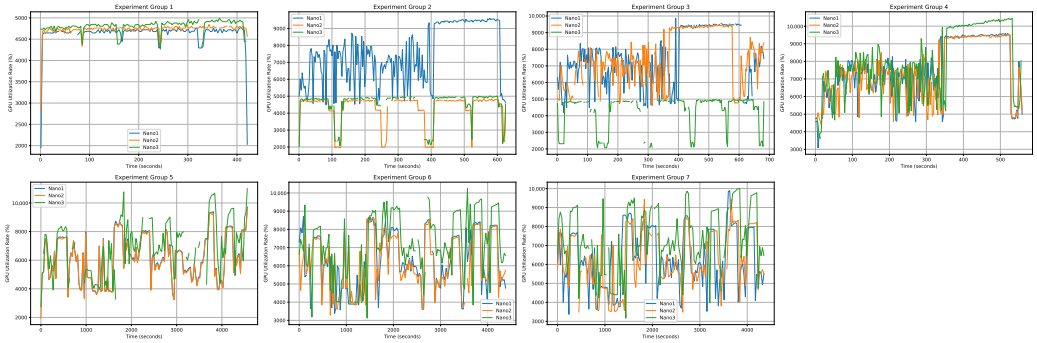



**Figure 7.** Power consumption.

## 5. Discussion and Analysis

### *5.1. Impact on CPU and GPU Utilization Rate*

Figure 4 shows that when federated learning system (FLS) clients run their training rounds without thermal stress, their CPU utilization rates remain at a certain maximum value: nearly 100% for Jetson Nano clients and around 70% for Raspberry PI clients. However, under thermal stress, the CPU utilization rates exhibit different patterns. For Jetson Nano clients under thermal stress, the utilization rates remain at around 100%, showing little change compared to non-stressed conditions. In contrast, for Raspberry PI clients under thermal stress, the utilization rates tend to rise to around 100%, indicating a 30% increase. This difference may be because we are applying direct thermal stress to the CPUs on Raspberry PIs, whereas on Jetson Nanos, the stress is applied to the GPUs.

If clients are not under thermal stress, they typically have a much lower CPU utilization rate while running the evaluation round. If clients are under thermal stress, then even during their evaluation round, they tend to have a nearly 100% utilization rate. Note that clients without thermal stress must wait for clients under thermal stress to finish their training. During this waiting time, clients without thermal stress will also have a lower CPU utilization rate. Clients that are under thermal stress but are less influenced by it, like

the Nano clients in Experiments 5 to 7, also have to wait for more affected clients to finish their training before moving on to the next step. During such periods, they also tend to have a lower CPU utilization rate, around 40%.

As shown in Figure 5, Jetson Nano clients without thermal stress typically exhibit very low GPU utilization rates, with only occasional short periods of utilization rate peaks. However, clients under thermal stress tend to experience fluctuations in their GPU utilization rates, but for the most part, the utilization rate remains high.

### *5.2. Temperature and Power Consumption*

When it comes to temperature, the patterns for clients on different devices vary. For Jetson Nano clients, as shown in Figure 6, Jetson Nano clients without thermal stress, such as Nano 1 and 2 in Experiment 1, maintain a stable temperature of around 30 to 35 Celsius. However, other normal FLS clients like Nano 3 exhibit temperature fluctuations between 40 and 55 Celsius. In Experiment 4, while Nano 1 and 2 are under thermal stress, their temperatures range between 40 and 50 Celsius, showing less stability compared to when they are not under stress. Nano 3 can reach up to 80 Celsius under thermal stress. The differences are possible because Nano 1 and 2 are equipped with fans while Nano 3 is not.

For Raspberry PI clients, as depicted in Figure 6, normal clients without thermal stress tend to maintain temperatures between 30 and 40 Celsius. However, when Raspberry PI clients are under thermal stress, their temperatures can rise to around 60 Celsius, nearly doubling, even when equipped with fans. Additionally, similar to the CPU utilization rate, when some clients in an FLS are under thermal stress, other clients without stress tended to experience temperature drops while waiting for the stressed clients to finish their training rounds and during evaluations. Clients less affected by thermal stress also tended to experience temperature drops while waiting for the more affected clients to complete their training rounds.

For power consumption, as shown in Figure 7, normal Jetson Nano clients running federated learning system training rounds typically consume between 4500 mW and 5000 mW while working. Power consumption can drop to 2000 mW when these clients are running evaluations or waiting for other under-stress clients to finish their training. However, clients under thermal stress tend to consume more than 5000 mW, fluctuating between 5000 and 9000 mW and sometimes reaching 10,000 mW, or even 12,000 mW. From Experiments 4 to 7, we can see that under the same thermal stress, Nano 3 consumes more power than Nano 1 or 2, possibly because Nano 1 and 2 have fans while Nano 3 does not. When both PIs and Nanos are under thermal stress simultaneously, Jetson Nanos, being less affected by thermal stress than PIs, also tend to experience power consumption drops while waiting for the PIs to finish their training rounds.

### *5.3. Impact on Training Time and Accuracy*

From Tables 3 and 4, we observe that when only Jetson Nano clients are under thermal stress, the running time for each round of the under-stress Nano clients, as well as the system's overall running time, nearly doubles. However, when both Jetson Nano and Raspberry Pi clients are under thermal stress, the running time for each round of the Raspberry Pi clients and the system's overall running time increases by almost 12 times. This indicates that applying thermal stress to the computational components utilized by the FLS has a significantly greater impact on the FLS performance than applying thermal stress to computational components not directly used by the FLS.

From Table 3, we can also conclude that the higher the ratio of clients under thermal stress, the longer the system's overall training time. However, regarding training accuracy, we did not observe any significant changes, whether the FLS was under thermal stress or not. A possible explanation for this could be that we are running a CNN on simple tasks, such as MNIST handwriting figure identification, and using a very large training set. These conditions make it easy for accuracy to remain high and resistant to the effects of thermal stress. Additionally, during data processing, we noticed that the system status

logs for clients under thermal stress occasionally had brief periods of missing data. Upon examining the raw data logs, we found that this is likely because thermal stress consumed too many resources during those brief periods, leaving insufficient resources for our data loggers to function properly.

### *5.4. Analysis and Insights*

From the experimental results, we found that thermal stress causes nodes in FLS to experience significantly higher CPU and GPU utilization rates, often reaching 100%. It also causes the temperature of nodes in the FLS to rise considerably, with Jetson Nanos reaching around 80% and Raspberry PIs around 96% on average, and temperatures can climb as high as 80 degrees Celsius. Additionally, thermal stress and excessive heat can lead to a substantial increase in the Jetson Nano clients' power consumption, ranging from 60% to as high as 140%. Nodes without fans tend to exhibit higher temperatures and power consumption compared to those with fans under the same thermal stress conditions.

When any node is under thermal stress, all other clients must wait for that specific client to finish its training before moving to the next round, which increases the overall training time. When clients are affected by thermal stress to varying degrees, the less-affected clients also have to wait for the more affected ones to finish, further increasing the training time. If thermal stress is not directly applied to the component running the FLS, the running time may only double. However, if thermal stress is directly applied to the component running the FLS, it can significantly impact the running time. Even minimal stress on one of the CPU cores can cause the running time to increase by 12 times, while typical stress can cause the running time to increase up to 120 times. However, if the type of thermal stress remains unchanged, whether it is not directly applied, directly applied, or a combination of both, the increase in time is not significant if only the ratio of stressed clients rises when clients are already under stress. Therefore, we can conclude that the training time of an FLS under thermal stress is determined by the "slowest" client within the FLS, e.g., the client most affected by the thermal stress.

In our previous work [3], we deployed MobileNet-v2/3 on the CIFAR-10 dataset [26] and found that the performance of FLS could decrease by as much as 21% under thermal stress in terms of model accuracy. However, in this evaluation, such a performance drop was not observed. A possible explanation might be related to the complexity of the task that the FLS is running. An FLS running a simpler task, such as training a CNN on the MNIST dataset for handwriting digit identification with a sufficient training set, tends to be more robust and less likely to be influenced by thermal stress. In contrast, when the task becomes more complex, the FLS might exhibit less robustness and be more easily influenced in terms of performance.

Generally speaking, we found that thermal stress can seriously affect our FLS, leading to higher CPU and GPU utilization, increased temperature, greater power consumption, extended training times, and even preventing the FLS from performing normally, thereby impacting system robustness. Based on the experimental results, we suggest integrating a data-driven anomaly detection system [27] into federated learning systems. This system should focus on abnormal utilization rates, temperature, and power consumption to detect thermal stress.

## 6. Conclusions, Limitations and Future Work

### *6.1. Conclusions*

In this paper, we are among the first to evaluate the performance of federated learning on real-world heterogeneous IoT-based systems under thermal stress. We used high-CPU-consuming and high-GPU-consuming programs to simulate thermal stress and varied ratios of clients under stress in the system to see the influence of thermal stress. Extensive experiment results have shown that thermal stress can cause low-end, IoT-based federated learning systems to have extended training time, extreme computing resources utilization, as much as 86% higher in temperature, more heat conducted, an average of 167% more

power consumption made, and even threats to system robustness. It is also shown from experimental results that the most influenced client under thermal stress within the FLS tends to have the most major impact on the performance of FLS under thermal stress. Future research may utilize the findings within this study as patterns to identify if an FLS is under thermal stress and develop FLS thermal stress detection and defense systems accordingly. This study may also be treated as a basis to explore further related issues of thermal stress on FLS.

### *6.2. Limitations*

Despite the current outcomes of our current research, certain limitations of our research still exist. To begin with, our current research was only conducted under a limited scale of IoT devices, and further studies can be transplanted to larger-scale FLSs for more enhanced results. Secondly, we only conducted our research based on equally distributed data, while the influence of thermal stress on FLS based on non-iiD data, which is more common within daily practice, is yet to be discussed. Last but not least, topics like whether different machine learning models have an influence on the performance of FLS under thermal stress are yet to be illustrated by further experiments.

### *6.3. Future Work*

Future work includes evaluations of larger networks and their performance under non-IID data distributions among IoT devices. We will also try to find out the influence of the machine learning model used in the FLS on the performance of IoT-based FLS under thermal stress in our future work. As wireless federated learning has developed prosperously in recent years, research related to the area has also been vast. Hou et al. [28] focused on efficient federated learning for the metaverse via multiple methods, and Amiri et al. [29] focused on federated learning over wireless fading channels. Our future work will also include an exploration of the impact of thermal stress on wireless federated learning systems.

**Author Contributions:** Conceptualization, Y.G. and L.Z.; methodology, Y.G. and L.Z.; software, Y.G. and T.L.; validation, Y.G.; formal analysis, Y.G.; investigation, Y.G.; resources, Y.G., L.Z. and S.W.; data curation, Y.G. and T.L.; writing—original draft preparation, Y.G. and T.L.; writing—review and editing, Y.G., L.Z. and S.W.; visualization, Y.G.; supervision, L.Z. and S.W.; project administration, L.Z. and S.W.; funding acquisition, L.Z. and S.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** All data used for this research are open-source data which we cited. The data as outcomes of our work are all detailed and listed within figures and tables in the context. As entities should not be multiplied unnecessarily, we chose not to reveal our data further.

**Conflicts of Interest:** The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

## References

1. McMahan, B.; Ramage, D. Federated Learning: Collaborative Machine Learning without Centralized Training Data. Available online: https://research.google/blog/federated-learning-collaborative-machine-learning-without-centralized-training-data/ (accessed on 24 June 2024).
2. Gouissem, A.; Abualsaud, K.; Yaacoub, E.; Khattab, T.; Guizani, M. Federated Learning Stability Under Byzantine Attacks. In Proceedings of the 2022 IEEE Wireless Communications and Networking Conference (WCNC), Austin, TX, USA, 10–13 April 2022; pp. 572–577. [CrossRef]
3. Gu, Y.; Zhao, L.; Deng, B.; Wu, S. Evaluation of Thermal Stress on IoT-based Federated Learning. In Proceedings of the 2024 ACM Southeast Conference, ACM SE '24, Marietta, GA, USA, 18–20 April 2024; pp. 291–296. [CrossRef]
4. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K.H.; Parcollet, T.; de Gusmão, P.P.B.; et al. Flower: A Friendly Federated Learning Research Framework. *arXiv* **2020**, arXiv:2007.14390.
5. Bhide, A.; Kulkarni, A. NVIDIA-AI-IOT Jetson_Benchmarks. Available online: https://github.com/NVIDIA-AI-IOT/jetson_benchmarks (accessed on 24 June 2024).

6. King, I.C. ColinIanKing-stress-ng. Available online: https://github.com/ColinIanKing/stress-ng (accessed on 24 June 2024).
7. Bonghi, R. Jetson-Stats. Available online: https://github.com/rbonghi/jetson_stats (accessed on 24 June 2024).
8. Hennion, N. Nicolargo-Glances. Available online: https://github.com/nicolargo/glances (accessed on 24 June 2024).
9. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the Artificial Intelligence and Statistics, PMLR, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
10. Google. Tensorflow Federated: Machine Learning on Decentralized Data. Available online: https://www.tensorflow.org/federated (accessed on 24 June 2024).
11. Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečný, J.; Mazzocchi, S.; McMahan, H.B.; et al. Towards Federated Learning at Scale: System Design. *arXiv* **2019**, arXiv:1902.01046.
12. Ryffel, T.; Trask, A.; Dahl, M.; Wagner, B.; Mancuso, J.; Rueckert, D.; Passerat-Palmbach, J. A Generic Framework for Privacy Preserving Deep Learning. *arXiv* **2018**, arXiv:1811.04017.
13. Masti, R.J.; Rai, D.; Ranganathan, A.; Müller, C.; Thiele, L.; Capkun, S. Thermal Covert Channels on Multi-core Platforms. In Proceedings of the 24th USENIX Security Symposium (USENIX Security 15), Washington, DC, USA, 12–14 August 2015; pp. 865–880.
14. Tian, S.; Szefer, J. Temporal Thermal Covert Channels in Cloud FPGAs. In Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Seaside, CA, USA, 24–26 February 2019; pp. 298–303.
15. Kong, J.; John, J.K.; Chung, E.Y.; Chung, S.W.; Hu, J. On the Thermal Attack in Instruction Caches. *IEEE Trans. Dependable Secur. Comput.* **2010**, *7*, 217–223. [CrossRef]
16. Gao, X.; Xu, Z.; Wang, H.; Li, L.; Wang, X. Why "Some" Like It Hot Too: Thermal Attack on Data Centers. In Proceedings of the 2017 ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems, Urbana-Champaign, IL, USA, 5–9 June 2017; pp. 23–24.
17. Gao, X.; Xu, Z.; Wang, H.; Li, L.; Wang, X. Reduced Cooling Redundancy: A New Security Vulnerability in a Hot Data Center. In Proceedings of the NDSS, San Diego, CA, USA, 18–21 February 2018.
18. Duchatellier, J.; Holmes, T.; Suo, K.; Shi, Y. An Empirical Study of Thermal Attacks on Edge Platforms. In Proceedings of the 2021 ACM Southeast Conference, New York, NY, USA, 15–17 April 2021; pp. 175–179.
19. Kaur, J.; Das, S. RSPP: Restricted Static Pseudo-Partitioning for Mitigation of Cross-Core Covert Channel Attacks. *ACM Trans. Des. Autom. Electron. Syst.* **2024**, *29*, 1–22. [CrossRef]
20. Afermarq; Tanertopal. Federated Learning on Embedded Devices with Flower. *arXiv* **2023**, arXiv:2007.14390.
21. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2019.
22. LeCun, Y.; Cortes, C. MNIST Handwritten Digit Database. Available online: https://yann.lecun.com/exdb/mnist/ (accessed on 24 June 2024).
23. Svcavallar. Jetson-Stats-Grafana-Dashboard. Available online: https://github.com/svcavallar/jetson-stats-grafana-dashboard (accessed on 24 June 2024).
24. Authors, P. Prometheus Monitoring Systems and Time Series Database. Available online: https://prometheus.io/ (accessed on 24 June 2024).
25. Labs, G. Grafana: The Open Observability Platform. Available online: https://grafana.com/ (accessed on 24 June 2024).
26. Krizhevsky, A.; Nair, V.; Hinton, G. CIFAR-10 (Canadian Institute for Advanced Research). Available online: https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 24 June 2024).
27. Saridou, B.; Bendiab, G.; Shiaeles, S.N.; Papadopoulos, B.K. Thermal Management in Large Data Centres: Security Threats and Mitigation. In Proceedings of the Security in Computing and Communications: 8th International Symposium, SSCC 2020, Chennai, India, 14–17 October 2020; Revised Selected Papers 8; Springer: Chennai, India, 2021; pp. 165–179.
28. Hou, X.; Wang, J.; Jiang, C.; Meng, Z.; Chen, J.; Ren, Y. Efficient federated learning for metaverse via dynamic user selection, gradient quantization and resource allocation. *IEEE J. Sel. Areas Commun.* **2023**, *42*, 850–866. [CrossRef]
29. Amiri, M.M.; Gündüz, D. Federated learning over wireless fading channels. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 3546–3557. [CrossRef]


**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.