engineering proceedings

MDPI

Proceeding Paper

Testing Bluetooth Signals for Device Identification Using Neural Network †

Martin Pap, Bálint Dugár, Balázs Lukács * and Lilik Ferenc *

Digital Development Center, Széchenyi István University, 9026 Győr, Hungary; martin.pap@ddc.sze.hu (M.P.); balint.dugar@ddc.sze.hu (B.D.)

* Correspondence: balazs.lukacs@ddc.sze.hu (B.L.); lilikf@sze.hu (L.F.)

† Presented at the Sustainable Mobility and Transportation Symposium 2026, Győr, Hungary, 8 October 2026.

Abstract

In an era where IT security has paramount importance, the identification of devices holds a significance comparable to that of individuals. With the pervasive adoption of wireless technology, understanding the informational content within wireless data, particularly when intercepted by third-party devices, becomes crucial. This study aims to measure the effectiveness of a system that can capture Bluetooth signals from a device and, using these signals, create an image that can be used to identify the device with a neural network approach. Bluetooth packets were captured in an isolated environment, filtered, and transformed into images via post-processing Python code. A Convolutional Neural Network (CNN) based on a modified LeNet architecture was trained using 20,000 data samples per device. Testing with 36 different hyperparameter combinations showed accuracy above 95% for known device classes and approximately 40% confidence when presented with images from unknown devices. The results confirm that devices can be identified with acceptable accuracy based on Bluetooth signal fingerprints.

Keywords: Bluetooth; radio frequency fingerprinting; device identification; convolutional neural network; software-defined radio; machine learning; wireless security

1. Introduction

In an era where IT security has paramount importance, the identification of devices holds a significance comparable to that of individuals [1]. With the pervasive adoption of wireless technology, understanding the informational content within wireless data, particularly when intercepted by third-party devices, becomes crucial. Our study aims to measure the effectiveness of a system which can capture Bluetooth signals from a device and using this signal create an image which can be used to identify the device with a neural network approach. Our solution is built up on similar works where great accuracy was achieved using 5G and RF signals [2–4].

The components responsible for emitting radio frequency signals may vary in type and manufacturer from device to device, but they will only operate within certain tolerance limits. This means that there may be some degree of variation between two devices, which could potentially make the device traceable [5]. The characteristic of RF fingerprint-based identification is such that it cannot be avoided: when a device emits a signal, it becomes trackable [6]. However, the extent and value of these characteristics can vary in several ways, including temperature and the level of RF noise surrounding the target device.

Check for updates

Academic Editors: Gábor Dogossy, Szabolcs Kocsis-Szürke, Dániel Csikor, Dániel Serfőző, Miklós Kuczmann, András Lajos Nagy, László Lendvai and Ambrus Zelei

Published: 13 October 2026

Copyright: © 2026 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.

After the data capture in an isolated environment, the Bluetooth packets were filtered and transformed into images via a post-processing Python (v3.10) code, resulting in images for analysis. The images representing the Bluetooth fingerprint of the device were used to train a Convolutional Neural Network (CNN). Based on the papers [2,3] we used a slightly modified LeNet [7] architecture. For training we used 20,000 data samples per device. For the best results we tried multiple hyperparameter combinations which resulted in 36 different models; for each we saved the best (based on validation accuracy) and the last model.

From these 36 models with results from a test dataset we selected the one which showed the best results in accuracy across classes and also which showed the best scores for distinguishing devices that do not belong to any classes. This approach reached accuracy above 95% with classes introduced during training and accuracy around 40% when shown images not belonging to any of the classes. This approach requires data from each class/device we want to separate, which makes it harder to adopt in a real scenario. The models proved that devices can be identified with acceptable accuracy based on Bluetooth fingerprint image.

2. Bluetooth Data Collection

2.1. Measurement Procedure

During the measurement, it was necessary to measure the selected devices in an environment where the Bluetooth signal sent by the subject under examination could be captured in its purest form possible. Another important aspect is that only the data sent by the tested device should be included in the recorded data. The shielded measuring box made by Rohde & Schwarz provides a practical solution for this purpose, offering a sufficiently isolated environment during the process. Its built-in antenna designed for the Bluetooth frequency range is also suitable for conducting the measurements. The setup used for data capturing can be seen in Figure 1.

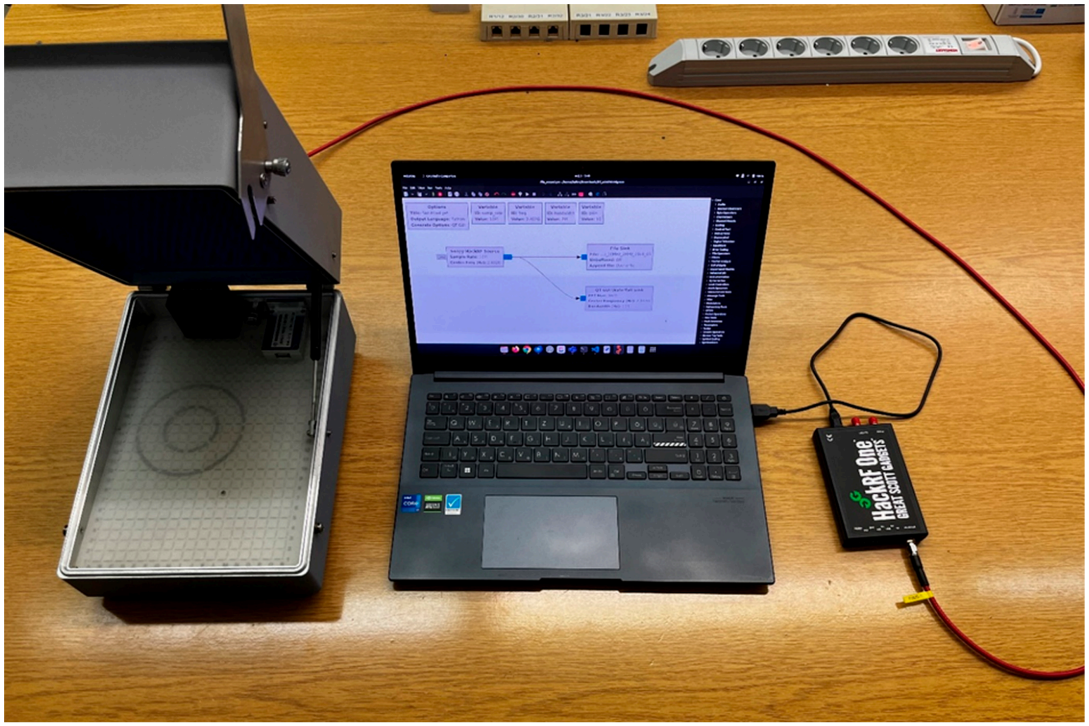


Figure 1. The measurement setup used for recording Bluetooth packets.

https://doi.org/10.3390/engproc2026157022

The device used to record radio frequency signals was the HackRF One (Great Scott Gadgets, Lakewood, CO, USA), which is a Software Defined Radio (SDR). By connecting them to a computer they can be freely configurable and set up for any use case (if the specific SDR is able to communicate on the required frequency band). The software side of the system is often an application called GNURadio (v3.10.9.0), which is a GUI tool that can generate the specific code for the needs of the user. Using graphical blocks connected, the user can set up a graph that tailors the capabilities of the device to the requirements of the project. For capturing Bluetooth packets the configured graph is quite simple (Figure 2).

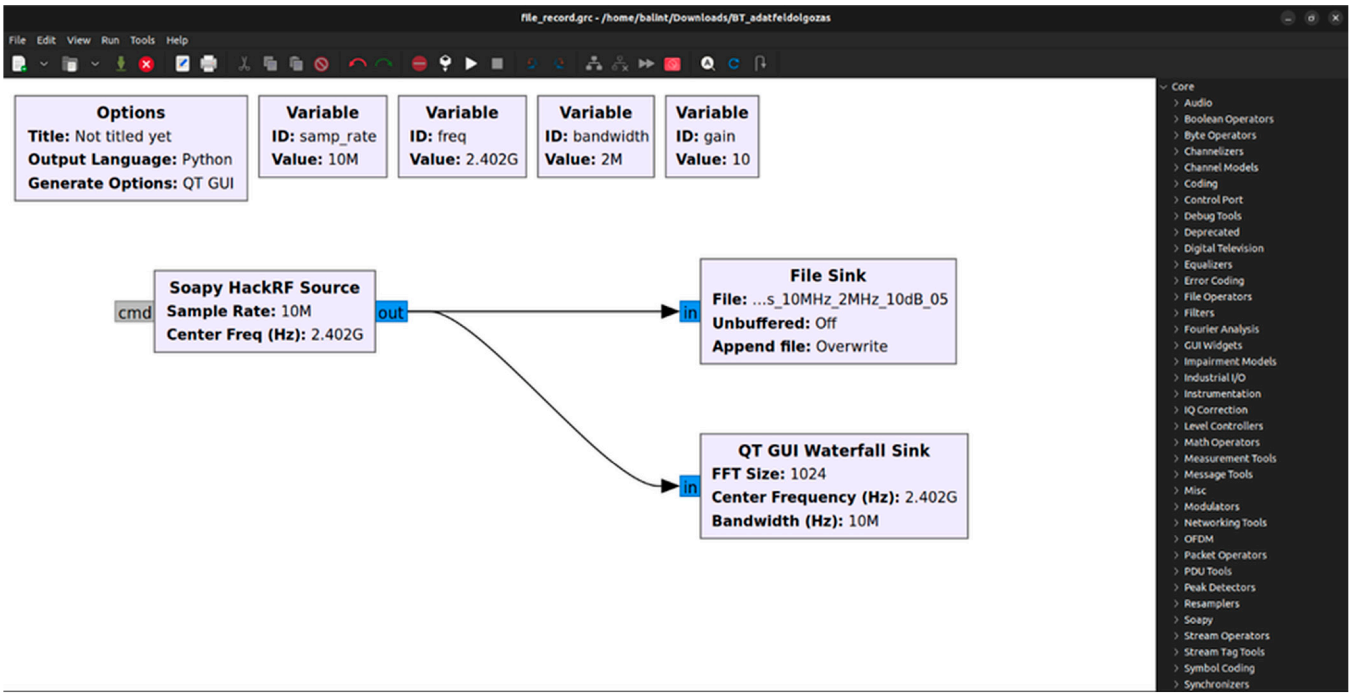


Figure 2. The graph used for the measurement with GNU Radio.

The "QT GUI Waterfall Sink" is not a necessary block; it only demonstrates the incoming radio frequency signals for the user using a waterfall diagram. The main block is the Soapy HackRF Source, which configures the SDR to the required parameters for the measurement: the center frequency was set to 2402 MHz, which is the center frequency for one of the three advertisement channels (channel 37) [8]. During the measurements, the smartphones were set to look for other devices, which results in a large number of Bluetooth packets on the advertisement channel. The sample rate was set to 10 MHz, which is 10× oversampling compared to the approximately 1 Megabaud transfer speed sent by the Bluetooth device, resulting in more data for generating the figures for AI recognition. The bandwidth was set to 2 MHz, so any other signal on other channels was filtered out.

2.2. Rapidly Growing Raw Data Size

To ensure sufficient data capture, it is important to consider that the recorded data quantity increases very quickly. During the measurement, packets sent by each device were recorded in multiple parts, with each file containing one minute of data, resulting in a size of 5 GB per file. For the AI measurement, 20,000 packets per device were required, meaning that around 120 recordings per device were needed, resulting in 600 GB of data per device.

Although this size can be reduced later, as the sampling rate can be halved from the previously used 10 MHz, instantly halving the data accumulation per device. For the AI device recognition, the measurement and analysis of two devices were performed for the training: an iPhone 13 mini and a Samsung Galaxy S21 Ultra. Different methods were

required to access the desired measurement data for the two devices, because iOS and Android handle Bluetooth connections slightly differently. In each case, airplane mode was active to ensure that only Bluetooth signals were transmitted.

The processing time is also influenced by the large data packets, as processing and converting this amount of data is a significant task even for a computer: a single file containing raw data consists of approximately 1.3 billion complex numbers (20,000,000 × 60). The final image-generating Python code processed, converted, and generated the images in just under 6 min, therefore the processing time for one device's data is approximately 120 × 6 = 720 min, or 12 h.

2.3. Bluetooth Signal Types Based on Recorded Data

Based on the representation of recorded data, two types of packet types can be identified: packets with less occurrence exhibit relatively higher signal strength, typically positioned farther apart from each other, whereas the remaining packets, with relatively lower signal strength, occur more densely and are recorded in larger quantities. The latter occurred during Bluetooth scanning. Overall, to ensure that both types of packets are adequately represented in the graphs forming the basis of machine learning, the number of recorded packets with higher signal strength must reach 10,000. The representation of recorded Bluetooth packets is shown in Figure 3.

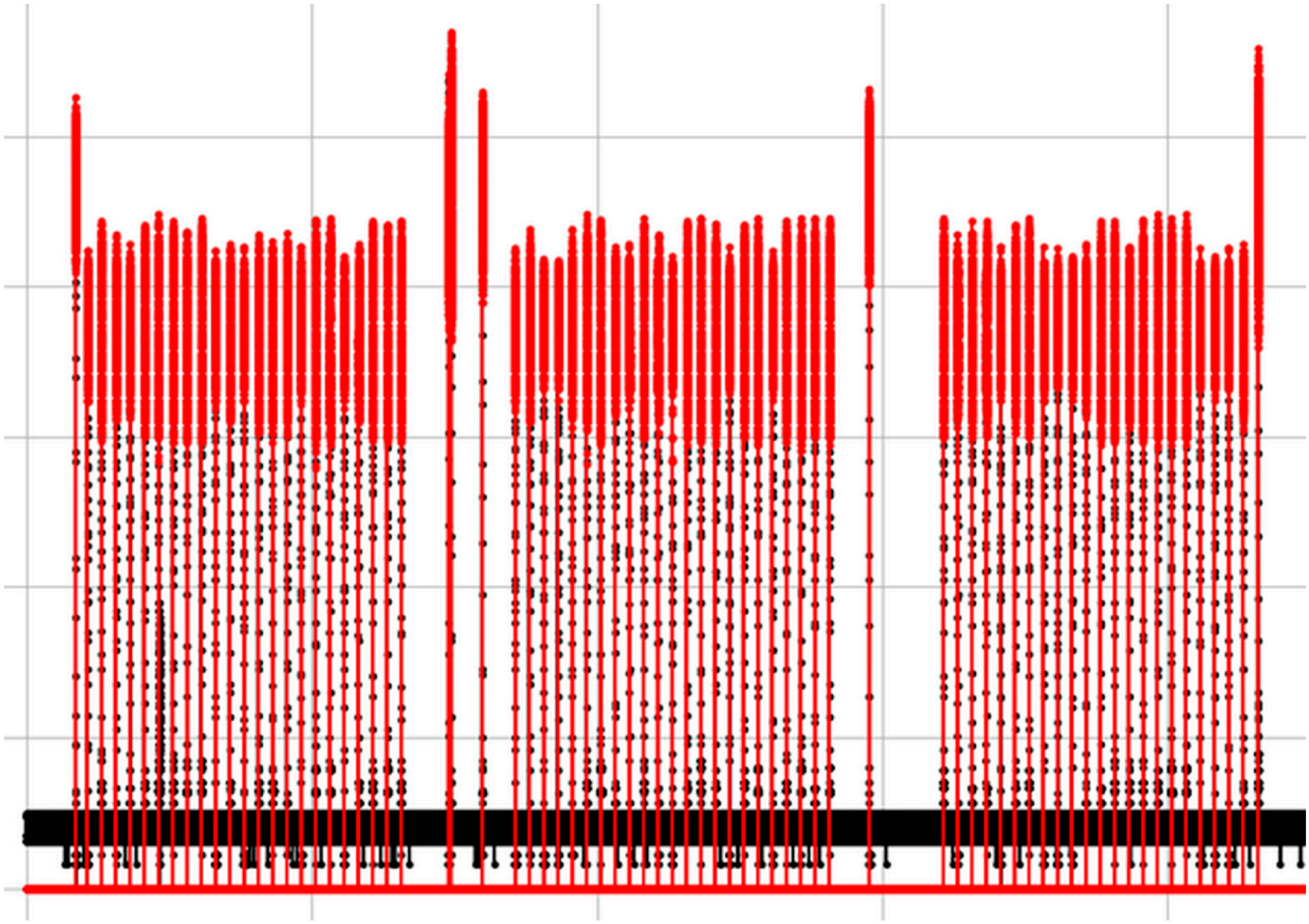


Figure 3. Representation of recorded Bluetooth packets.

https://doi.org/10.3390/engproc2026157022

3. Discrimination Methods Procedure and Results

3.1. Coordinate Representation

One method of discriminating Bluetooth signals involves representing characteristics using a coordinate system, which encapsulates the unique features of a particular receiver. During the procedure, the following two characteristics were analysed: firstly, the extent of deviation from the carrier frequency that the device is originally intended to transmit on, which in this case is 2402 MHz. Secondly, the frequency hopping magnitude applied during Bluetooth data transmission. The former is abbreviated as CFO (Carrier Frequency Offset) [9], while the latter is referred to as FSK deviation (FSK refers to the modulation applied, where alternation between two frequencies conveys binary data).

The measured Bluetooth signals were represented based on the aforementioned two characteristics, primarily for three devices: a Samsung Galaxy S21 Ultra, a ZTE Blade A72, and an iPhone 13 mini. For the representation, 20 recordings of 1-min intervals were analysed per device. After the above process, the results can be seen in Figure 4 (in this case there are only 10 recordings present to keep the illustration more manageable).

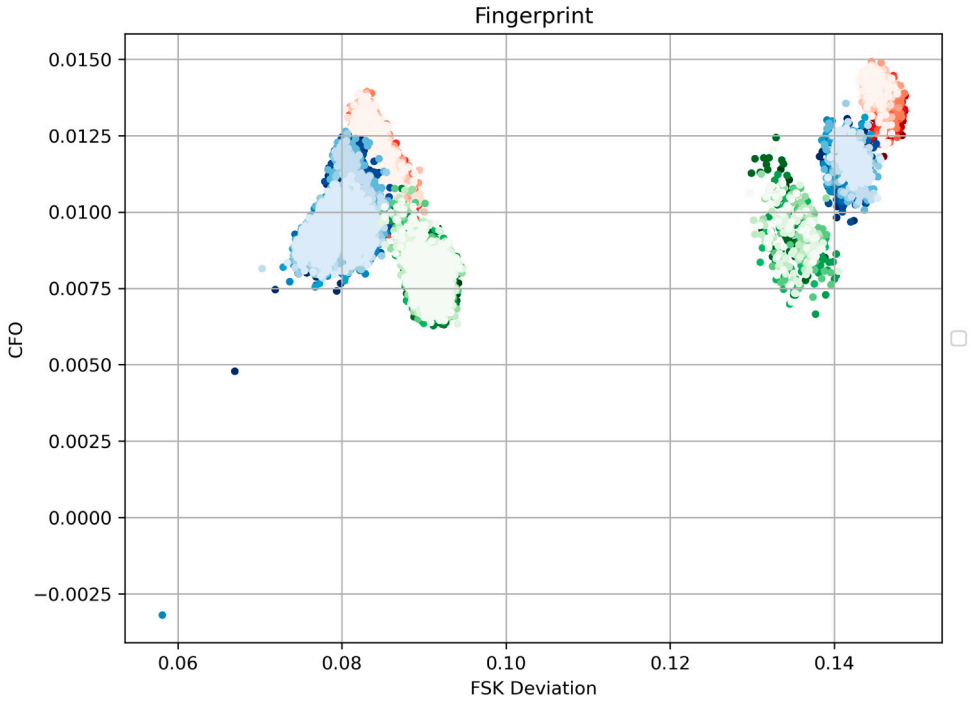


Figure 4. Representation of the data from the three phones in the coordinate system.

In the image, two clusters of signals can be observed, indicating the presence of two different types of packets sent by the devices. In both cases, it is evident that the results corresponding to the same device point to a specific area; however, there are overlapping areas between the devices as well. Figure 5 provides a closer view of the degree of overlap between the results.

As a fourth device, recordings were also made from an iPhone 14, the results of which are represented in purple in Figure 6. Since recordings had already been made with an iPhone 13 mini, theoretically, the results of the iPhone 14 should be situated close to this nominated area.

https://doi.org/10.3390/engproc2026157022

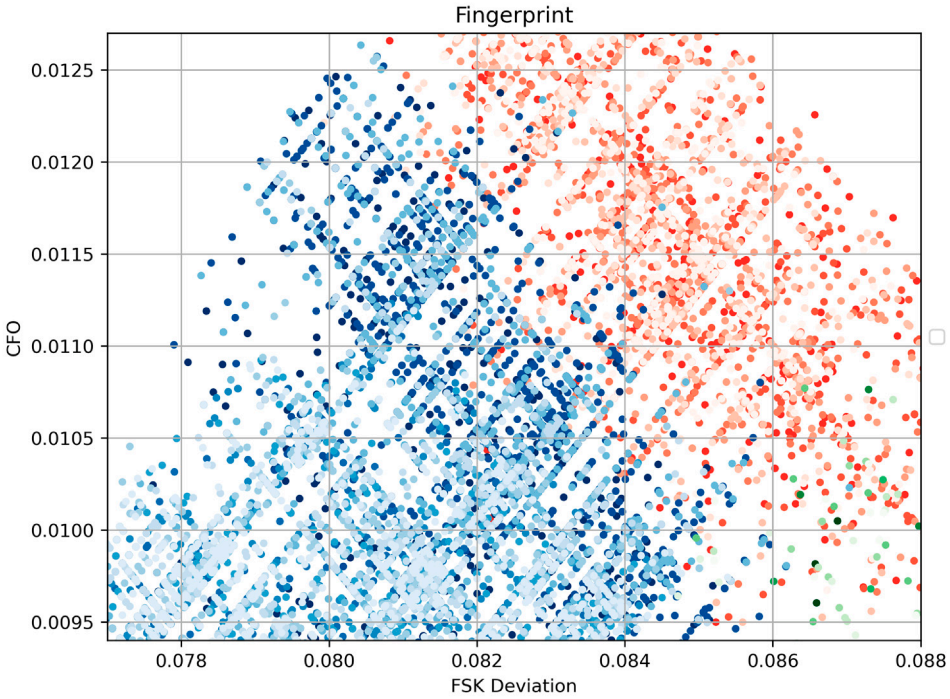


Figure 5. Diagram illustrating the degree of overlap between the results.

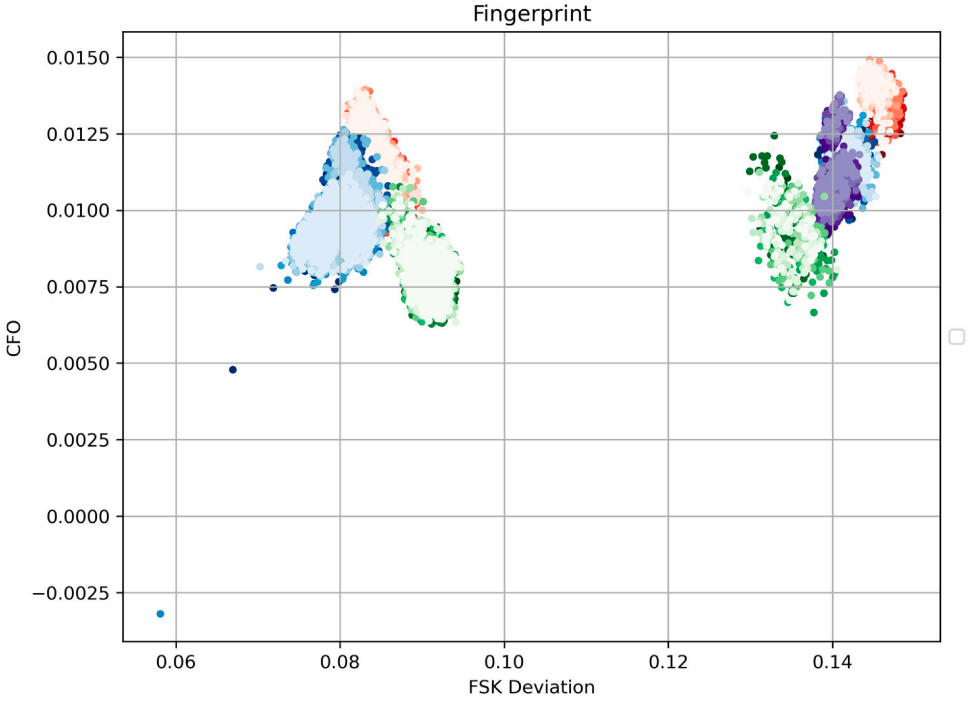


Figure 6. Representation of the results of the iPhone 14 with the previously depicted three devices.

Comparing Figures 7 and 8, the results are closely situated to the device of the same manufacturer; however, stronger overlap is observed between the points corresponding to the two iPhones.

https://doi.org/10.3390/engproc2026157022

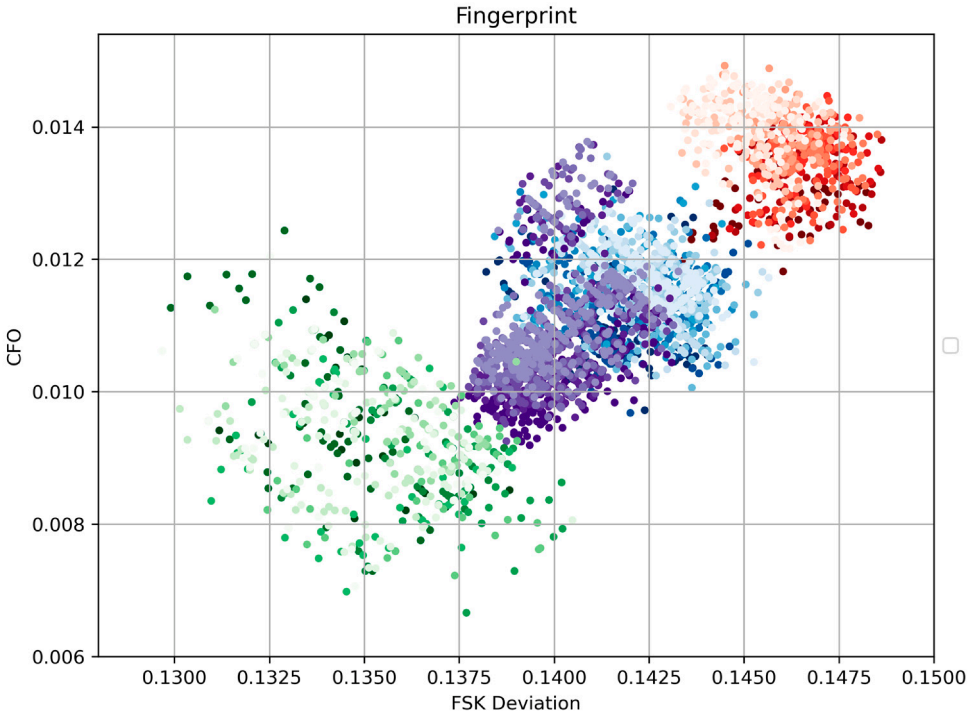


Figure 7. Illustration of the results of the iPhone 14 on an approximate image.

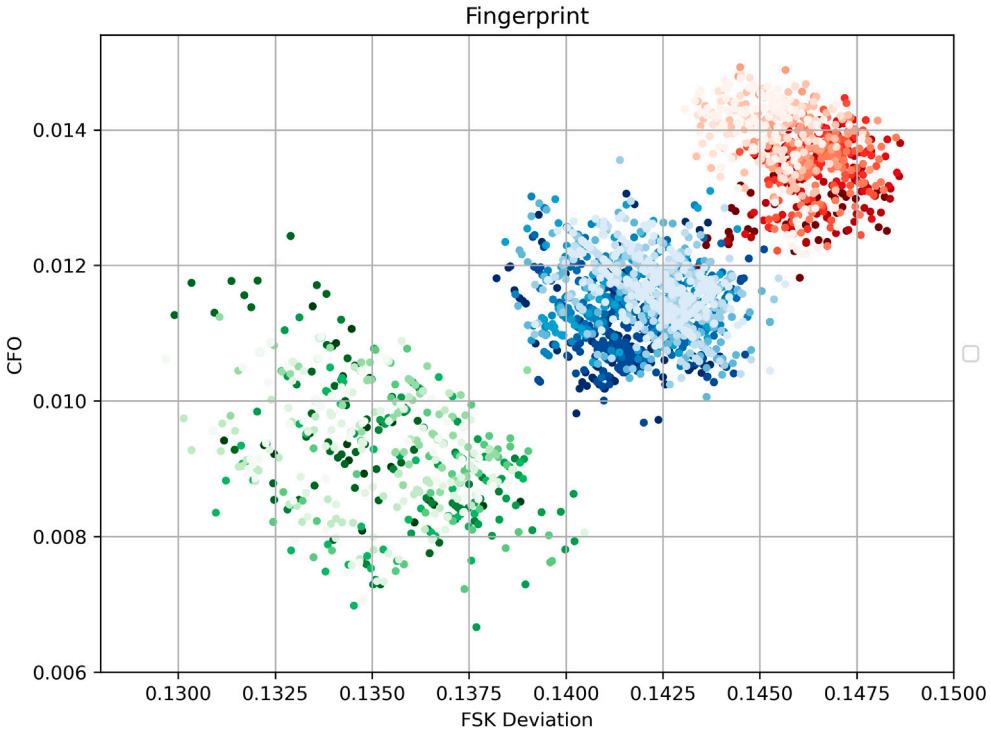


Figure 8. The same approximate image segment without the representation of the iPhone 14.

From this initial test, it follows that the method will be able to approximately determine which phone sent the examined packet. Most likely, the phone type and/or manufacturer can be named using the procedure alone. However, the above measurements were conducted under the most favorable conditions in terms of signal acquisition. To determine practical accuracy, several aspects of the results need to be examined:

(1) Impact of Noise on Results. This is expected to be the most significant hindrance to the method. In real-life scenarios, additional devices and signals operate within

the Bluetooth frequency band, greatly complicating the readability of the Bluetooth packets. In extreme cases, this could result in a larger dispersion of points on the graph, making it harder to specify devices.

(2) Analysis of Multiple Devices for Database Construction. To precisely determine the accuracy of the procedure, as many devices as possible need to be recorded in the database. However, recording and processing each device is a very time-consuming process.

(3) The Effect of Temperature on the Results. The operation of the components responsible for RF transmitting in a device can vary based on its temperature. The oscillator that works in such a module can ever so slightly change its frequency, affecting the characteristics of the transmitted signal and therefore its RF fingerprint as well.

3.2. Generating Images for AI-Based Packet Recognition

Generating images for AI-based device recognition constituted a significant portion of the development tasks. Bluetooth signals can be classified into two groups based on their nature (when performing scanning for other available Bluetooth devices or advertising themselves). From a machine learning perspective, this meant that roughly the same number of images needed to be generated for both types. After proper processing, images similar to Figures 9 and 10 can be obtained. The first image was generated based on the data of a Bluetooth packet from the Samsung Galaxy S21 Ultra, while the second image was based on the Bluetooth packet of the iPhone 13 mini.

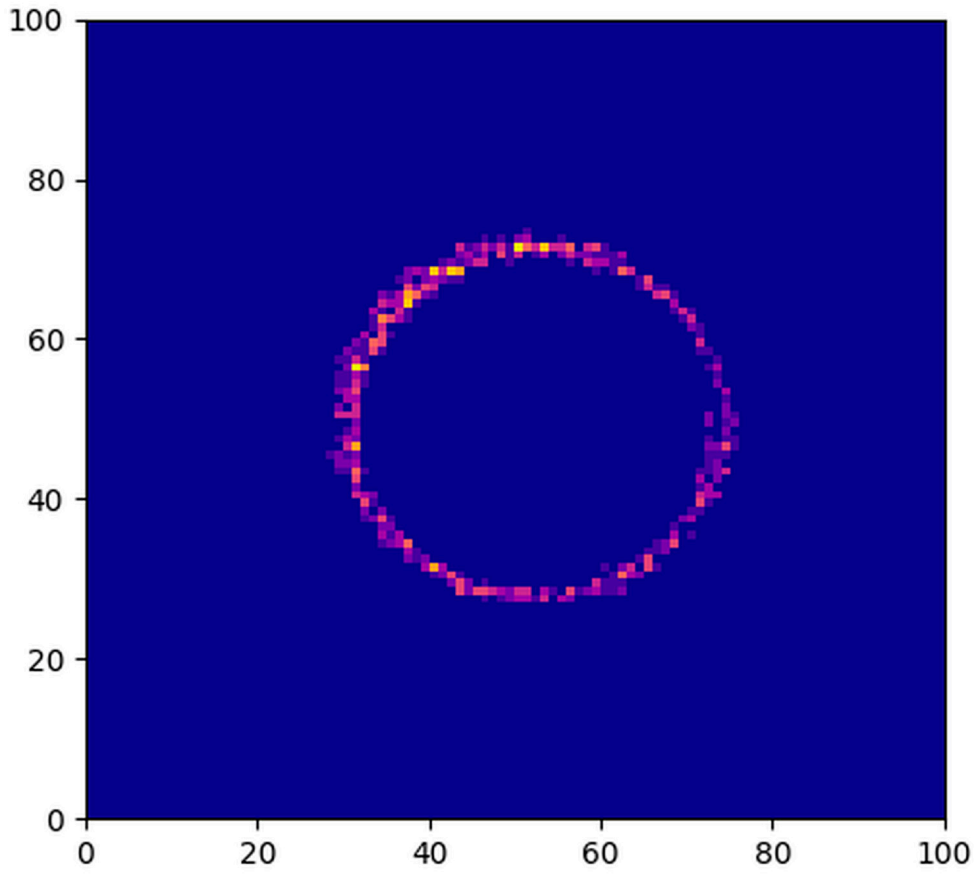


Figure 9. Representation of one Bluetooth packet from the examined Samsung Galaxy S21 Ultra.

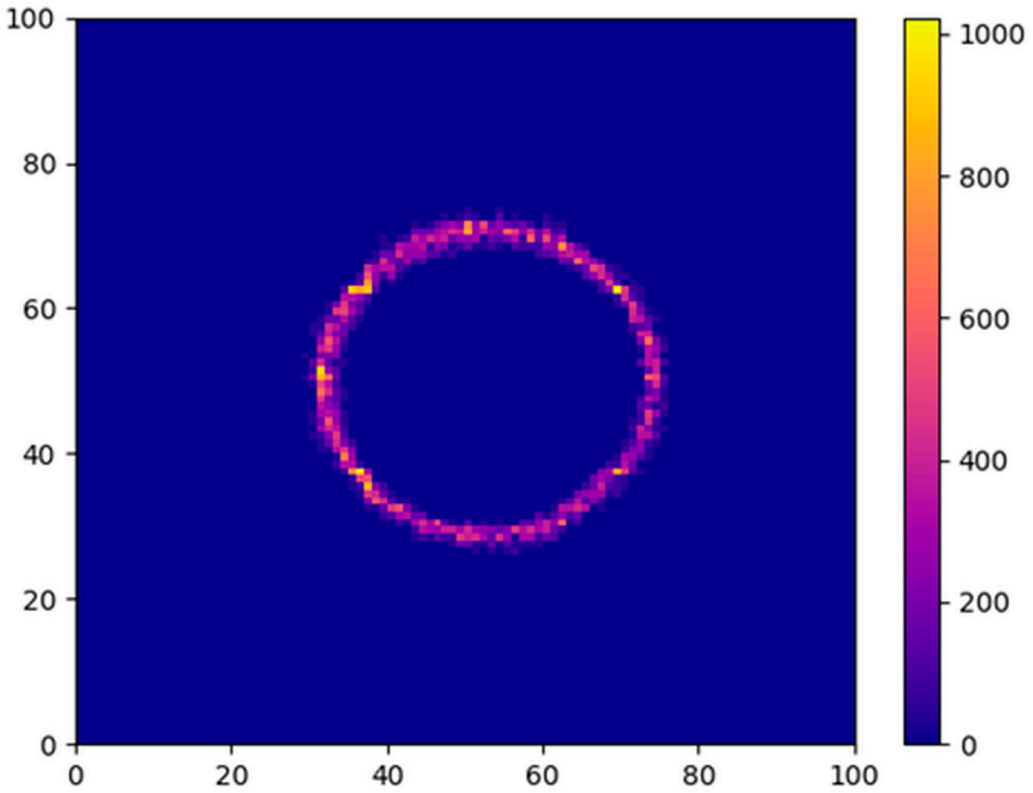


Figure 10. Representation of one Bluetooth packet from the examined iPhone 13 mini.

Regarding the size of the image, the publication upon which the method was based used 64 × 64-pixel images for generation. However, in our own implementation, it was found that working with images of at least 369 × 369 pixels is necessary to ensure that the results are not altered by the image generation process. Although this step was necessary, it also increased the required training time.

3.3. The Challenges of the Figure Procedure

In terms of representation, a challenge was to appropriately select and process the signal by the software. Several methods were tested, but during this initial development period, a method that could perfectly segregate and extract the packets was not yet developed. Another difficulty in the process was that the devices also alter their transmission signal strength, so a finely tuned packet search parameterization could not be directly applied to other devices, and even among samples recorded later from the same device, adjustments were necessary.

Nonetheless, the Python code used changed its efficiency depending on the recorded segments: in most cases, it was able to generate appropriate graphs either 100% or 90–80% of the time, while in other cases, it showed results of 50% or worse. Incorrect processing primarily meant improper packet grouping, where the majority of cases failed to classify the devices into the two specified signal shape types.

The image generation procedure also posed further challenges. In several instances, unexpected side effects were observed in the generated image, including the streaking of graphs (Figure 11) and improper packet cropping (Figure 12). The reason for the streaking and the necessarily larger figure size is arguably the matplotlib library used for generating pictures. It is recommended to find other alternatives to check for better results.

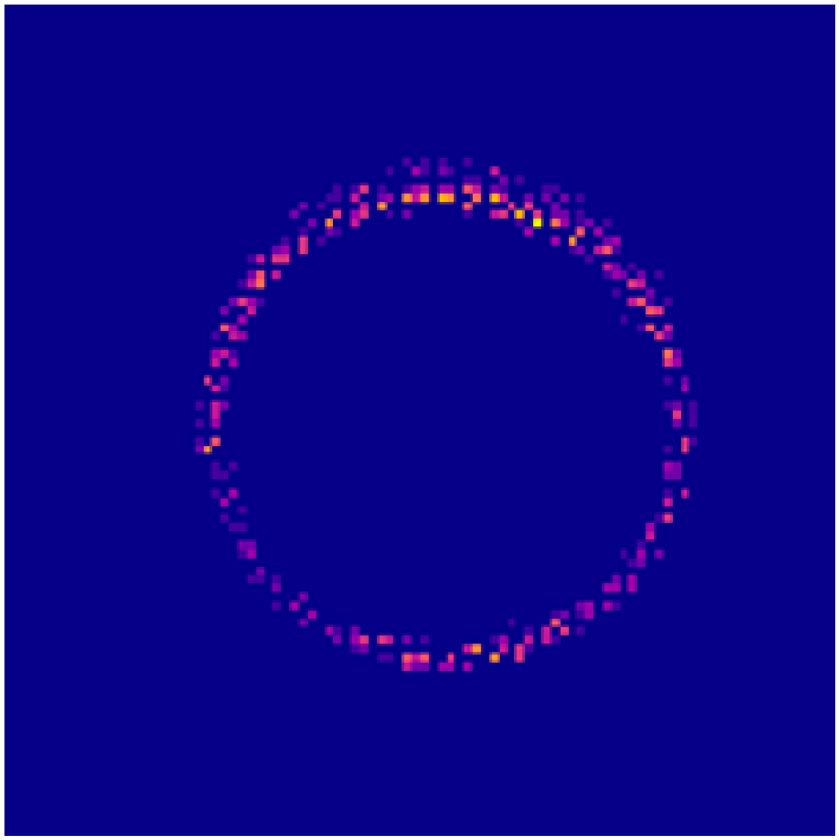


Figure 11. An example illustrating the streaking observed in the resulting graph. The graph exhibits streaks resembling horizontal and vertical hashing patterns.

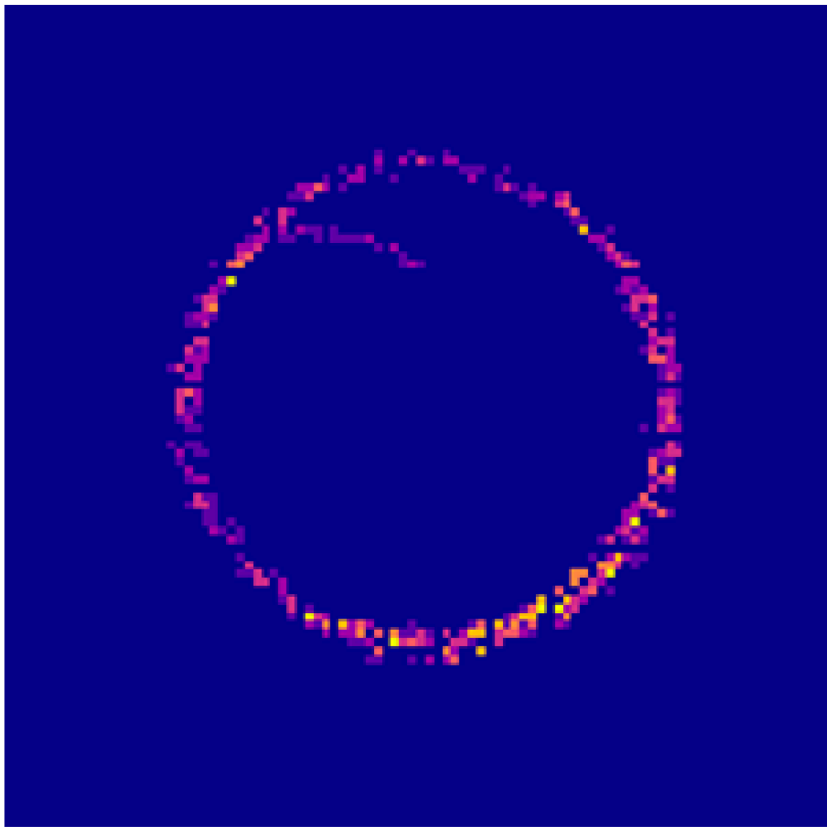


Figure 12. An illustrative example of improper packet cropping for a single packet.

https://doi.org/10.3390/engproc2026157022

4. Bluetooth Fingerprint Classification

Since we created an image-based fingerprint from the Bluetooth signals, we used an artificial neural network architecture that proved to be efficient for this type of task. The original LeNet-5 is a 7-layer architecture not counting the input (Figure 13). Our approach was similar but slightly different. We used the LeNet-5 and based on some tests we added dropouts as a regularization method. Also, we used only 2 convolution blocks. The biggest difference we applied is a temperature scaling softmax [10] layer as an output.

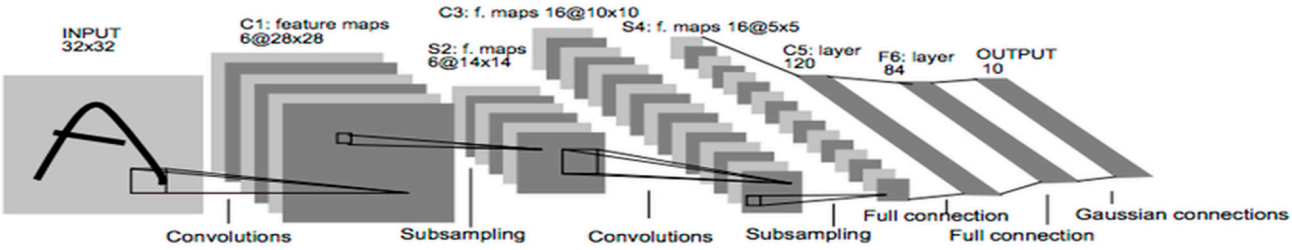


Figure 13. Architecture of LeNet-5, a convolutional neural network [7].

We wanted to achieve a model that gives high accuracy for one class if the input is from a known class and gives low accuracy across all classes if the input is from an unknown device. The softmax function with temperature scaling adjusts the logits (inputs) using a temperature parameter T before applying the softmax operation:

$$\text{SOFTMAX}(z_i) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$$

where z_i is the i-th element of the input logits vector and T is the temperature parameter, which adjusts the sharpness of the softmax distribution.

By increasing the temperature parameter, the output probabilities become more diffuse, meaning that the model becomes less confident in its predictions. Conversely, decreasing the temperature parameter sharpens the output distribution, making the model more confident in its predictions. Our model uses two convolution blocks. Each has a Conv2D layer, then a batch normalization layer, then a MaxPooling2D layer. After the convolutional blocks we use a flatten layer to convert our data from a 2D matrix to a 1D vector. Then we have the fully connected block with 3 Dense layers and a dropout. We use MaxPooling instead [11] of average pooling to get the most dominant features. The inclusion of a dropout layer [12] within the model facilitates enhanced generalization by stochastically deactivating a subset of neurons during training epochs.

4.1. Training and Testing

We trained each model for 30 epochs. Based on our previous tests with the model and the data, the model reached an acceptable accuracy within 30 epochs. We used a single GPU for training with a batch size of 100. 80 percent of the dataset was used to train, and 20 percent was used to validate. During training we monitored the validation accuracy, and we saved the best and the last model.

During training we tried different hyperparameters. To find the optimal solution we changed the temperature, the optimizer, and the learning rate. We tried 4 temperature values, 3 learning rates, and 3 optimizers. Combining these, we had 36 different models and for each we had the last and the best. Using these 72 models we ran tests on an independent dataset which contained images for class 1 and class 2 that were not used during training. Also, we created a third group with images from a device that is different from the 2 devices used during training.

https://doi.org/10.3390/engproc2026157022

We ran the predictions with all the models for all the images. Our goal was to find the model with the best accuracy predicting class 1 and class 2, and a model that provides confidence levels during predictions that can be used for thresholding out class 3, which represents devices unknown to the model. Using the predictions with probability thresholding we can categorize the model output into 3 groups: class 1 and class 2 (which were part of the training), and if the probability is below the threshold, then class 3 (the unknown device).

4.2. Results

Our results showed that with the artificial neural network architecture we are able to decide which Bluetooth fingerprint belongs to which device across 2 devices, and we can detect unknown devices with a confidence level that is higher than pure guess. Based on the model's confidence level as shown in Table 1 we chose two of our best models. During testing we used a 95% threshold for each model to identify unknown devices. Any prediction below this confidence level was considered as an uncertain prediction connected to an unknown device.

Table 1. Confidence scores when predicting images for best models.

Model	Class 1 avg. Confidence	Class 2 avg. Confidence	Class 3 avg. Confidence
2.0_0.0001_sgd_best_model.keras	77.22	78.73	40.25
5.0_0.001_adam_best_model.keras	99.84	97.60	23.49

From all the models we chose two. One which had great results when we needed to detect unknown devices, with lower confidence deciding between class 1 and class 2. And another model with high confidence classifying class 1 and class 2 but much lower results with unknown devices. The chosen two models have the following parameters: Model 1: Optimizer: SGD, Learning rate: 0.0001, Temperature: 2.0. Model 2: Optimizer: Adam, Learning rate: 0.001, Temperature: 5.0.

The SGD optimizer-based model is the best in terms of detecting unknown devices while maintaining acceptable confidence in identifying known classes. The ADAM optimizer-based model is the best in terms of identifying known devices, but it has a lower confidence level when presented with an unknown device.

Figure 14 shows the confusion matrix of the SGD optimizer-based model. It is clear that confusion between class 1 and class 2 is none, which means that the model can separate the 2 known classes very well. Unknown devices are confused with class 2. During the data collections we used 1 device as the unknown class; it was an Android device just as class 2. Our assumption is that class 2 and class 3 are confused since they have more similarities than class 1, which is an iPhone device.

Figure 15 shows the confusion matrix of the ADAM optimizer-based model. It is clear to see that it has better values with class 1 and class 2, and similarly to the SGD model, a high confusion rate between class 2 and class 3 with unknown devices.

https://doi.org/10.3390/engproc2026157022

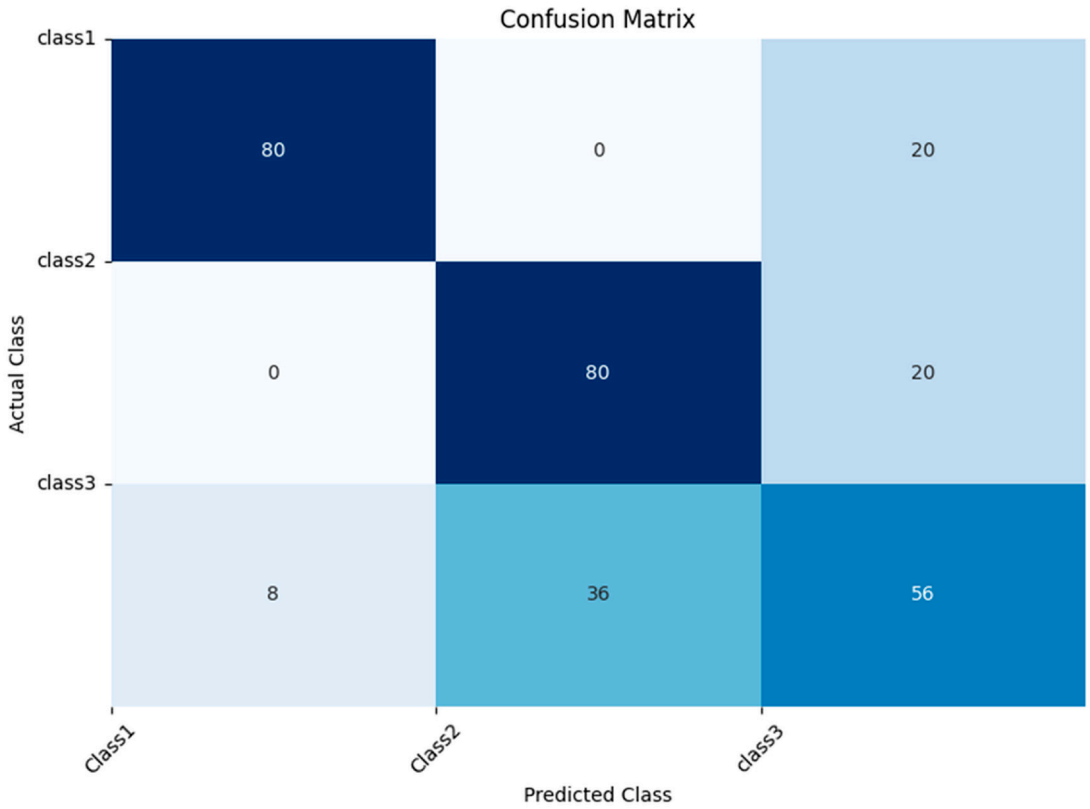


Figure 14. Confusion matrix of 2.0_0.0001_sgd_best model.

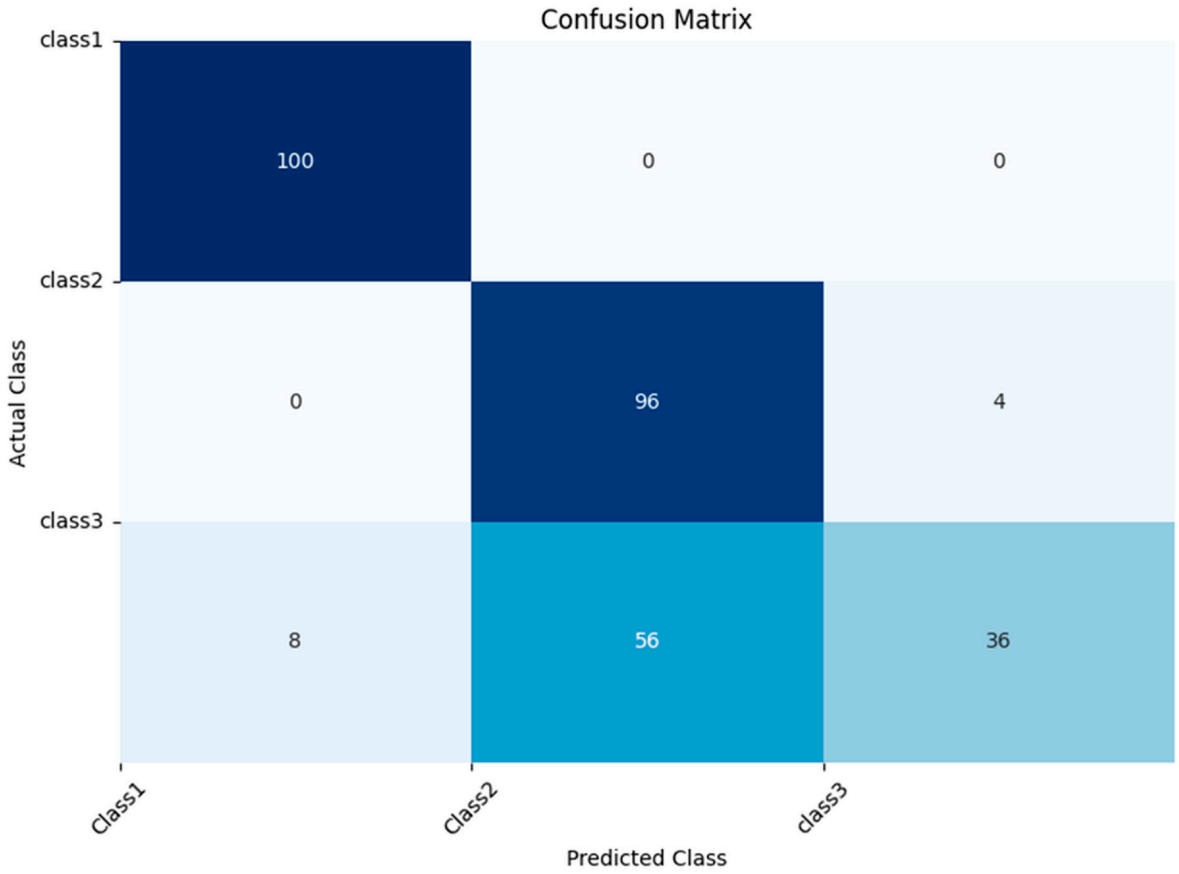


Figure 15. Confusion matrix of 5.0_0.001_adam_best model.

5. Conclusions

In this study, we explored the efficacy of using neural networks to identify devices based on Bluetooth fingerprints, with a particular focus on distinguishing between two classes of devices. Our findings demonstrate promising results, indicating strong accuracy in classifying both Class 1 and Class 2 devices. Moreover, we observed the potential of our model to effectively identify unknown devices under certain conditions, thus highlighting its versatility and applicability in real-world scenarios.

One key area of interest for future research is understanding the impact of increasing the number of classes on both the confidence of predictions and overall accuracy. Additionally, we must consider how introducing noise to the fingerprint measurements may influence the robustness of our model. Investigating noise-resilient techniques and incorporating them into our model architecture could enhance its performance in noisy environments. Furthermore, our study serves as a foundational exploration into the feasibility of using neural networks for device identification based on Bluetooth fingerprints, providing a solid baseline for future research to build upon.

Author Contributions: Conceptualization, B.D., M.P. and B.L.; Methodology, B.D. and M.P.; Software, B.D. and M.P.; Validation, B.D., M.P. and B.L.; Formal Analysis, B.D. and M.P.; Investigation, B.D. and M.P.; Data Curation, B.D. and M.P.; Writing—Original Draft Preparation, B.D. and M.P.; Writing—Review & Editing, B.L., L.F.; Visualization, B.D. and M.P.; Supervision, L.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to privacy restrictions.

Acknowledgments: The authors would like to thank the Digital Development Center at Széchenyi István University for providing the equipment and infrastructure necessary for this research.

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

References

1. Xu, Q.; Zheng, R.; Saad, W.; Han, Z. Device fingerprinting in wireless networks: Challenges and opportunities. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 94–104. [CrossRef]
2. Fu, H.; Dong, H.; Yin, J.; Peng, L. Radio frequency fingerprint identification for 5G mobile devices using DCTF and deep learning. *Entropy* **2023**, *26*, 38. [CrossRef] [PubMed]
3. Peng, L.; Zhang, J.; Liu, M.; Hu, A. Deep learning based RF fingerprint identification using differential constellation trace figure. *IEEE Trans. Veh. Technol.* **2020**, *69*, 1091–1095. [CrossRef]
4. Robyns, P.; Marin, E.; Lamotte, W.; Quax, P.; Singelée, D.; Preneel, B. Physical-layer fingerprinting of LoRa devices using supervised and zero-shot learning. In Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec 2017), Boston, MA, USA, 18–20 July 2017.
5. Hall, J.; Barbeau, M.; Kranakis, E. Detecting rogue devices in Bluetooth networks using radio frequency fingerprinting. In Proceedings of the Communications and Computer Networks (CCN 2006), San Francisco, CA, USA, 30 October–1 November 2006; pp. 108–113.
6. Givehchian, H.; Bhaskar, N.; Herrera, E.R.; Rodrigo, H.; Dameff, C.; Bharadia, D.; Schulman, A. Evaluating physical-layer BLE location tracking attacks on mobile devices. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22–26 May 2022.
7. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
8. Bluetooth Special Interest Group. Core Specification—Bluetooth Technology Website. Available online: https://www.bluetooth.com/specifications/specs/ (accessed on 30 May 2024).

9. Sun, W.; Paek, J.; Choi, S. CV-Track. In *HotWireless '17: Proceedings of the 4th ACM Workshop on Hot Topics in Wireless*; Association for Computing Machinery: New York, NY, USA, 2017.
10. Guo, C.; Pleiss, G.; Sun, Y.; Weinberger, K.Q. On calibration of modern neural networks. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 1321–1330.
11. Boureau, Y.-L.; Ponce, J.; LeCun, Y. A theoretical analysis of feature pooling in visual recognition. In Proceedings of the 27th International Conference on Machine Learning (ICML 2010), Haifa, Israel, 21–24 June 2010; pp. 111–118.
12. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.

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.

https://doi.org/10.3390/engproc2026157022