T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

A FEDERATED LEARNING SYSTEM ON RASPBERRY PI 4 FOR METAL SURFACE INSPECTION

OZAN DORUK YAVUZ

SUPERVISOR
DOÇ.DR. YAKUP GENÇ

GEBZE
2025

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

A FEDERATED LEARNING SYSTEM ON RASPBERRY PI 4 FOR METAL SURFACE INSPECTION

OZAN DORUK YAVUZ

SUPERVISOR
DOÇ.DR. YAKUP GENÇ

2025
GEBZE

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on .../.../..... by the following jury.

JURY

Member
(Supervisor) : Doç.Dr. Yakup Genç

Member : Prof.Dr. Yusuf Sinan Akgül

ABSTRACT

The project aimed to show and test the benefits of federated learning using Raspberry Pi and different devices. For this purpose, defects on metal surfaces (such as scratches, holes, major damages, etc.) were detected and identified using a model trained with YOLOv8 object detection and the federated learning method.

The benefits of federated learning include ensuring and proving the privacy of image data by keeping the photos stored locally, without sending them to any external location. The model was trained just using the weights obtained locally by the devices.

Another objective was to achieve parallelism through the collaborative work of machines contributing to our shared model. This setup enabled multiple local devices to work together without overusing the resources of a single device.

The project also aimed to achieve higher efficiency with a model trained on multiple devices compared to models running on single devices.

Using the Raspberry Pi's camera module, the goal was to gather data and contribute to the model. The model, which can also act as the main model for collecting data and detecting defects on metal surfaces, was designed to use the camera for defect detection.

Performance metrics of devices operating with different data, on different local devices, and with different numbers of epochs were collected and analyzed through comparisons.

At the end of the detection process, TP, TN, FN, and FP values were used to calculate the accuracy of the detections. These calculations were automated through scripts that could be executed after the detection process. The accuracy of these results was compared across different scenarios.

Keywords: Federated Learning, Raspberry Pi, Defect Detection on Metal Surfaces, YOLOv8, Data Privacy, Performance Analysis, Image Detection.

ÖZET

Projede federe öğrenmenin faydalarını Raspberry Pi ve farklı cihazlar kullanarak göstermek ve test etmek hedeflendi. Bunun için metal yüzeyler üzerindeki bozulmalar, (çizik, delik, büyük bozulma ve benzeri olarak.) YOLOv8 obje tespiti kullanılarak eğitilen modelde, federe öğrenim yöntemiyle algılandı ve tespit edildi.

Federe öğrenmenin faydaları olarak, yerel depolamadan başka bir yere gönderilmeyen fotoğraflar ve sadece makinelerin yerelinde elde ettiği ağırlıklarla eğitilen model sayesinde, bu fotoğraf verilerinin gizliliğinin sağlanması ve kanıtlanması amaçlandı.

Birden fazla yerel cihazda çalışarak ortak modele katkı sağlayan makinelerin beraber çalışabilme ve tek bir makinenin kaynaklarını harcamama özelliği sayesinde paralelliğin elde edilmesi de amaçlandı.

Birden fazla cihazla eğitilen modelde, tek olarak çalışan cihazlarına kıyasla daha yüksek bir verimlilik elde etme amaçlandı.

Raspberry Pi cihazının kamera modülü sayesinde verileri elde edip modele katkı sağlamak hedeflendi. Verilerin toplanıldığı ve metal üzerindeki kusurların tespit edildiği ana model olarakta çalışabilen modelde, kusur tespitini bu kamera sayesinde yapabilmesi hedeflendi.

Farklı yerel cihazlarda, farklı verilerle çalışan ve farklı epoch sayıları ile test edilen cihazların performans metrikleri çıkarıldı ve karşılaştırılarak analiz edildi.

Tespit sürecinin de sonunda TP, TN, FN ve FP verileri kullanılarak verimlilik tespitini sağlayan tespitler çıkarıldı. Bu kodlar tespit sürecinden sonra çalıştırılabilecek şekilde, script olarak otomatikleştirildi. Bu sonuçların verimliliği farklı senaryolar üzerinden karşılaştırıldı.

Anahtar Kelimeler: Federe Öğrenim, Raspberry Pi, Metal Yüzeylerde Kusur Algılama, YOLOv8, Veri Gizliliği, Performans Analizi, Görüntü Algılama.

ACKNOWLEDGEMENT

I am grateful for the invaluable support, resources and guidance provided by my supervisor, Doç.Dr.Yakup Genç. I also extend my thanks to Prof.Dr.Yusuf Sinan Akgül, who provided additional fixes and guidance as part of the jury.

Ozan Doruk YAVUZ

CONTENTS

1. INTRODUCTION

Defect detection on metal surfaces, such as scratches, holes, and other issues, is important for ensuring quality and reliability in many industries. This project focuses on using federated learning with the YOLOv8 model to make the detection process efficient and secure.

The goal is to use Raspberry Pi devices with their camera modules to detect defects locally without sharing image data to external servers. Federated learning helps train a shared model using data from multiple devices, ensuring data privacy and reducing the need for transferring large amounts of information.

This project also aims to make the detection process faster and more efficient by using multiple devices working together. The results will show how federated learning and YOLOv8 can be used to improve defect detection while maintaining privacy and making better use of limited hardware resources which can be end up with better accuracy metrics.

2. THEORETICAL FOUNDATION

2.1. Federated Learning

Federated learning is a distributed machine learning approach that allows multiple devices to train a shared model without transferring their raw data to a central server. This ensures data privacy while reducing communication overhead. Each device trains the model locally and shares only the updated model weights, which are then combined at a central server.

In this project, both Raspberry Pi devices and personal computers are used interchangeably as clients and servers in the federated learning setup. This configuration shows how devices with different hardware capabilities can work in a federated environment. This setup also highlights the adaptability of federated learning in scenarios where devices may take on dual roles, contributing to training as clients and managing updates as servers.

2.2. YOLOv8 Object Detection

YOLOv8 is a lightweight and accurate object detection model. It processes images and identifies objects with high precision and speed, making it suitable for both Raspberry Pi devices and personal computers. YOLOv8 was chosen for this project because of its ability to handle complex tasks like defect detection efficiently across light-weight hardware.

The model is trained to detect different defects on metal surfaces, such as scratches, holes, and deformations. Its architecture allows it to run efficiently on both edge devices and more powerful personal computers, enabling collaborative training in the federated setup.

2.3. Defect Detection on Metal Surfaces

Detecting defects such as scratches, holes, and deformations is critical for maintaining the quality of metal products. In this project, defect detection is performed using YOLOv8, trained collaboratively through federated learning. A dataset of defect images was prepared, annotated, and distributed across devices for local training.

The federated setup ensures that raw data remains on the devices, addressing

privacy concerns while enabling efficient training across a distributed network. The collaborative process results in a global model that benefits from the different data collected by different devices.

2.4. Performance Metrics

The performance of the federated learning system is measured using various metrics, including true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). These metrics are used to calculate accuracy, precision, recall, and F1 score, which provide a detailed view of how well the system performs in detecting defects.

Precision-recall (PR) curves and confusion matrices are also analyzed for individual clients to better understand the model's performance across different devices. These metrics help identify specific strengths and weaknesses in the system, such as its ability to detect subtle defects or avoid false detections.

The results are compared between models trained on different devices and those trained together using federated learning. This analysis shows how the combination of Raspberry Pi devices and personal computers improves accuracy, efficiency, and overall system performance.

3. PROJECT DESIGN

3.1. System Architecture

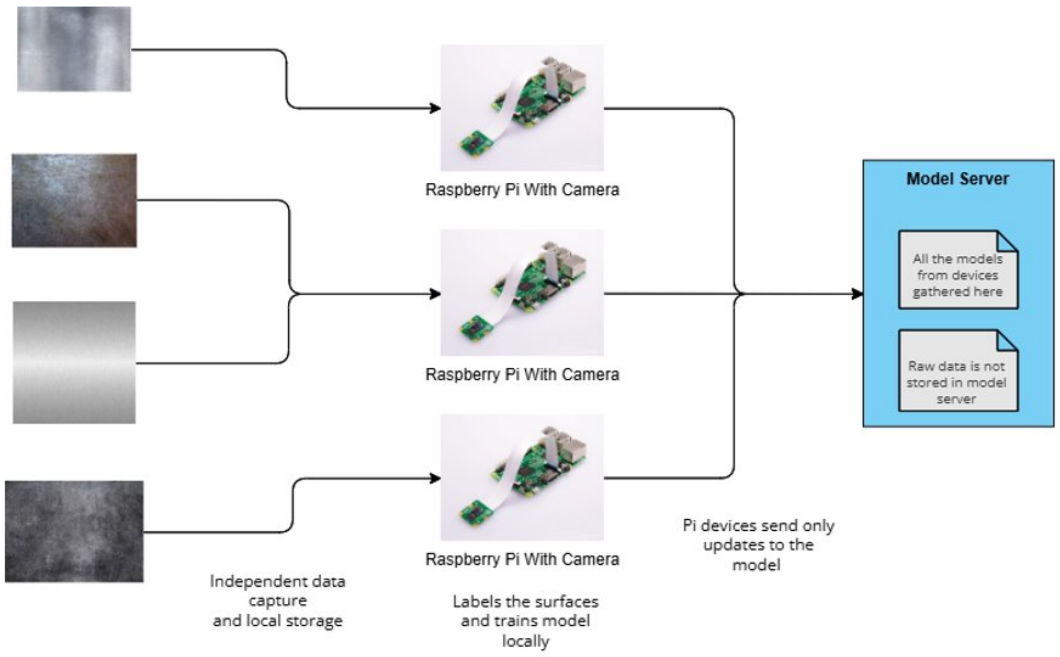

Figure 3.1: System Architecture.

In this project, federated learning is implemented to detect defects on metal surfaces using Raspberry Pi devices and personal computers. The system is composed of two main components: local devices (Raspberry Pi and PCs), a central model server.

The local devices act as both clients and servers interchangeably. Each device collects image data from metal surfaces using a camera (Raspberry Pi) or existing datasets (PCs). The data is processed locally to train a YOLOv8 model. Only the updated model weights are sent to the central model server via a secure SSH connection with privacy and secure communication.

The central model server aggregates model weights received from multiple devices. This aggregated global model is redistributed to all devices through the same secure SSH connection.

Together, this architecture ensures secure collaboration between devices, enabling distributed training without compromising data privacy. The use of SSH strengthens the system's reliability and security while maintaining seamless communication across

devices. In addition, system is not only dependent to the SSH and can be send over different types of connection with only sending a very small data.

3.2. Hardware Setup

Figure 3.2: Photo Of the Raspberry Pi with Camera Module 3.

The hardware for this project includes Raspberry Pi devices and personal computers. The Raspberry Pi devices, built on the ARMv7l architecture, are equipped with Camera Module 3 to capture images of metal surfaces. These devices are compact and efficient, which directly inserted to the model and used with its predefined libraries. The camera modules provide clear and detailed images of defects, such as scratches

and holes, which are used for training the YOLOv8 model.

In addition to Raspberry Pi devices, personal computers are used to provide additional computational power. These machines complement the Raspberry Pi by handling more intensive training tasks and supporting the overall workflow of the system. The combination of Raspberry Pi devices and personal computers creates a flexible and scalable hardware setup for federated learning.

3.3. Software Design

The software design for this project is modular, with specific scripts handling key aspects of the federated learning workflow, including dataset preparation, local training, aggregation, predictions, and evaluation.

3.3.1. Dataset Preparation

The dataset is divided among clients using the ”splitter.py” script. This script splits the main dataset into training, validation, and test subsets and distributes them evenly among the specified number of clients. Each client receives a structured dataset, ensuring that images and their corresponding labels are correctly matched. This step is critical for maintaining consistency in training and ensuring balanced data distribution across devices.

3.3.2. Local Training

The ”client.py” script is responsible for training the YOLOv8 model on each client’s local dataset. Each client initializes a pretrained YOLOv8 model (”yolov8n.pt”) and fine-tunes it using its specific dataset. Parameters such as batch size, number of epochs, and image size can be adjusted to accommodate the computational limitations of the client devices. Once training is complete, the model weights are saved locally, ready to be sent to the server for aggregation.

3.3.3. Model Aggregation

The ”server.py” script aggregates the model weights received from all clients. Using a federated averaging algorithm, the script combines the weights to create a global model. This aggregated model benefits from the diverse datasets across clients without requiring access to their raw data. The global model is saved and redistributed to clients for further training or inference.

3.3.4. Workflow Simulation

The entire federated learning process is orchestrated using the ”simulate.py” script. This script sequentially runs the ”client.py” script on each client to perform local training and then triggers the ”server.py” script to aggregate the weights. By automating the workflow, ”simulate.py” ensures smooth coordination between clients and the server.

3.3.5. Predictions

Predictions are made using the ”predict yolo.py” script. This script applies the aggregated model to test images and saves the detection results. For images where no objects are detected, placeholder files are created to maintain consistency in the output. This step ensures that every test image has a corresponding output, simplifying further analysis.

3.3.6. Label Validation

The ”labelChecker.py” (Checks the invalid labels.) and ”check labels.py” (Checks the empty ground truth and prediction files.) scripts are used to validate the dataset labels. These scripts identify missing or invalid labels and ensure that all label files are correctly formatted. This step is essential for maintaining the quality of the training and validation datasets.

3.3.7. Evaluation

The system’s performance is evaluated using the ”calculate tp fp fn.py” and ”calculate metrics.py” scripts. These scripts calculate metrics such as true positives (TP), false positives (FP), false negatives (FN), precision, recall, F1 score, and accuracy. The metrics provide insights into the effectiveness of the model and help compare results across different configurations, such as varying numbers of clients or training epochs.

3.4. Dataset Preparation

The dataset preparation process is a critical step in this project to ensure proper data distribution among clients for federated learning. This is handled using the ”splitter.py” script, which automates the division of the dataset into client-specific

subsets.

The script first divides the main dataset into three subsets: training, validation, and testing. Each subset is then split evenly among the specified number of clients, ensuring balanced data distribution. Images and their corresponding labels are carefully matched to maintain consistency and avoid errors during training. The data for each client is stored in a structured format, with separate folders for train, valid, and test subsets.

In addition to splitting the dataset, a data cleaning process was applied to improve the quality of the training data. During this step, data that caused over-fitting on class-3 (Class for scratches.) was identified and reduced. Empty or incorrect data entries, such as mislabeled or missing annotations, were also cleaned for each client to receive a valid and well-prepared dataset. This cleaning process reduces noise in the data, leading to better training results and more reliable model performance.

By ensuring balanced distribution and high-quality data, the dataset preparation and cleaning process enables each client to contribute effectively to the federated learning process.

3.5. Training and Aggregation

The training process in this project involved running 5 epochs for each client, which took approximately 2 to 3 hours per client depending on the hardware capabilities. Each client trained the YOLOv8 model locally on its allocated dataset. The training was managed using the ”client.py” script, which allowed for efficient and independent training on each client’s device.

During training, the pretrained YOLOv8 model (”yolov8n.pt”) was fine-tuned using client-specific data. Parameters such as batch size, image size, and the number of epochs were adjusted to suit the computational capacity of each device. After completing the training process, the locally trained weights were saved and prepared for aggregation.

The aggregation process was handled by the ”server.py” script. This script collected the model weights from all clients and combined them using a federated averaging algorithm. The aggregated global model was saved and redistributed to the clients for further training or inference. This iterative process ensured that the global model benefited from the different datasets shared across the clients.

The combination of local training and server-side aggregation allowed the system to achieve collaborative learning while maintaining data privacy. This project successfully implemented a federated learning system for defect detection on metal surfaces, achieving significant progress in model accuracy, communication efficiency, and data

privacy.

To begin with, the accuracy of the federated model improved notably after additional data cleaning and increased training time. For Client 1, the precision-recall (PR) curve showed significant improvement after the removal of over-fitting data for class 3 and the elimination of empty or incorrect labels.

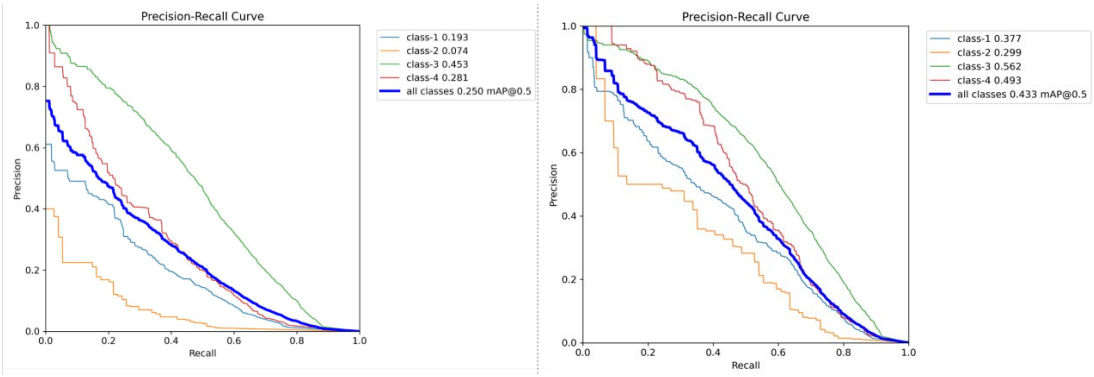

Figure 3.3: PR-Curves of the Client-1 before and after data cleaning.

This enhancement resulted in more consistent predictions, reducing false positives and false negatives. The accuracy of Client 1 increased significantly, validating the impact of the data cleaning process and longer training periods.

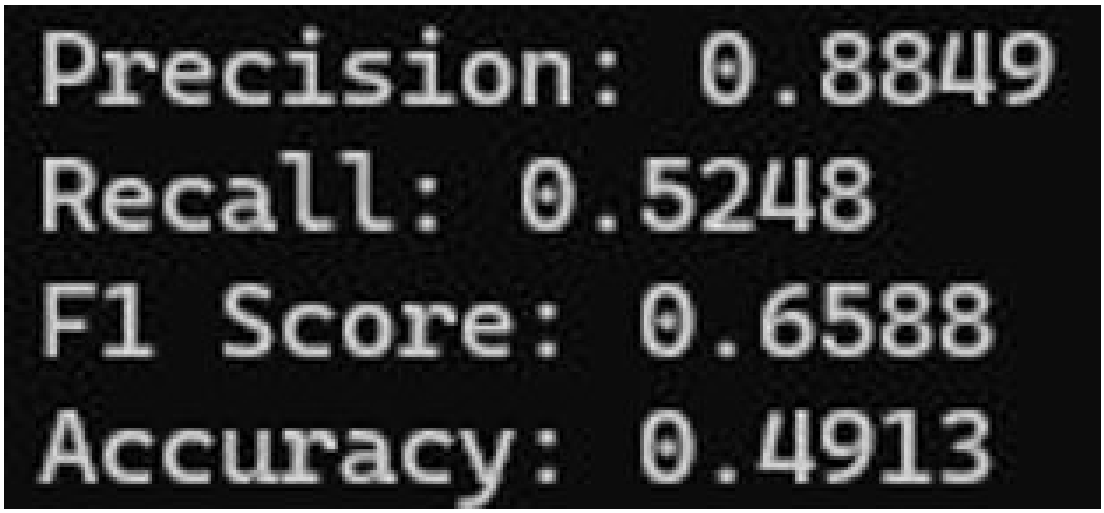

Figure 3.4: Accuracy before the data cleaning and increase in training time (With 3 clients).

The system was evaluated using a three-client configuration, which demonstrated a 6.56% improvement in accuracy compared to the two-client setup. This improvement

reflects the benefits of collaborative training across multiple devices, leveraging diverse datasets for a stronger global model.

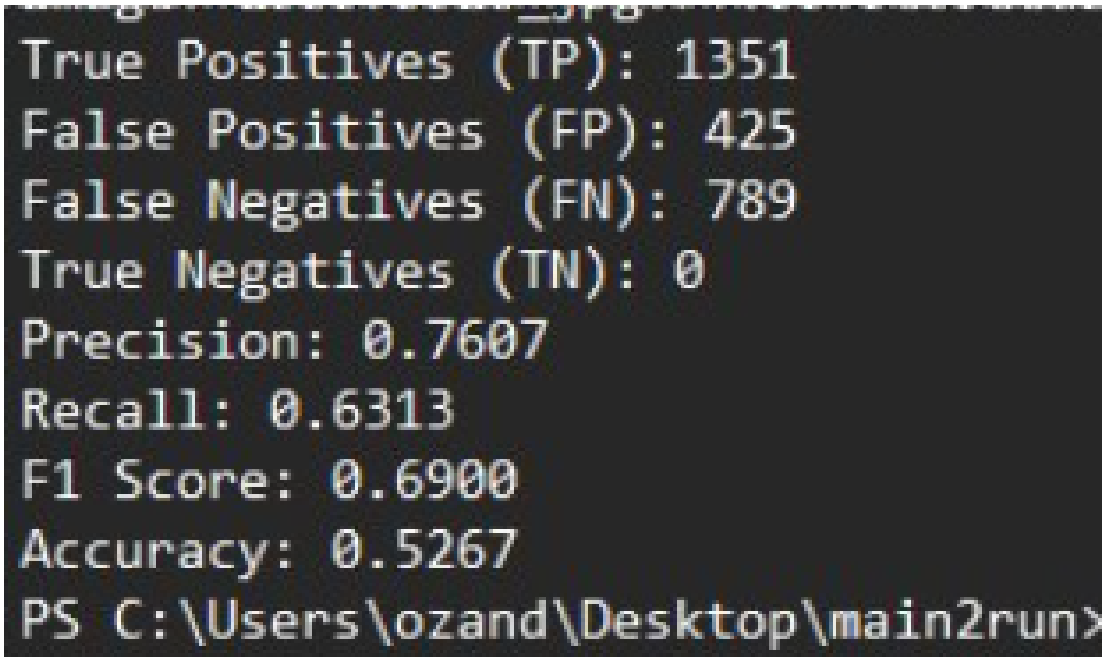

Figure 3.5: Accuracy after the data cleaning and increase in training time (With 2 clients).

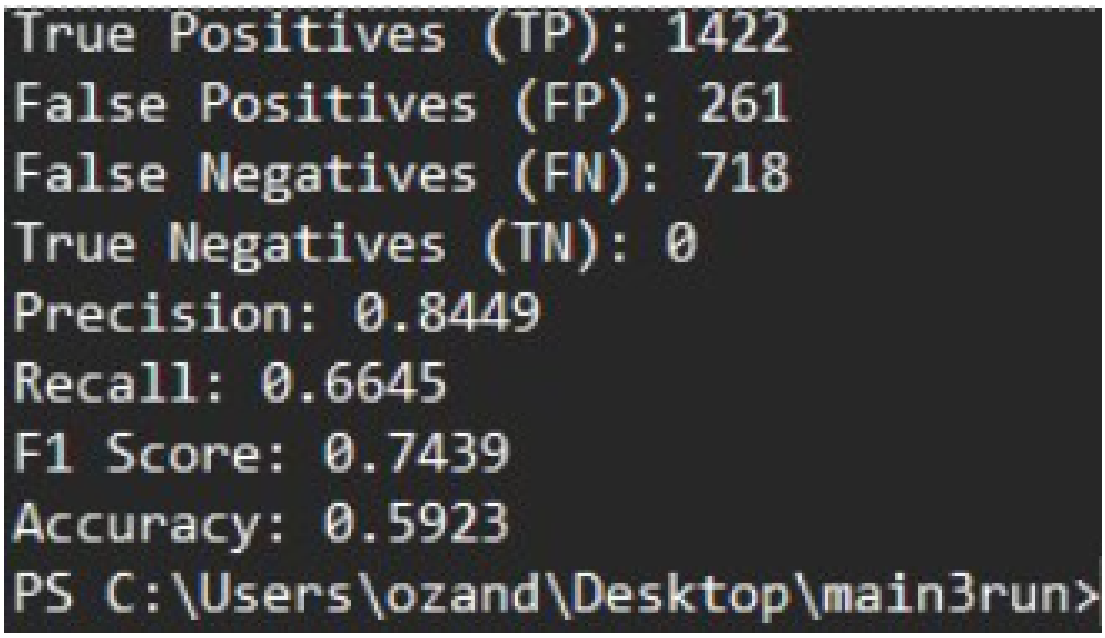

Figure 3.6: Accuracy after the data cleaning and increase in training time (With 3 clients).

The communication speed between clients and the model server was another critical evaluation metric. Using SSH for transferring model weights, the average transfer time was recorded at 6 seconds, well below the acceptable threshold of 35 seconds. This efficient communication ensured smooth operation during federated learning rounds, even with the hardware constraints of Raspberry Pi devices.

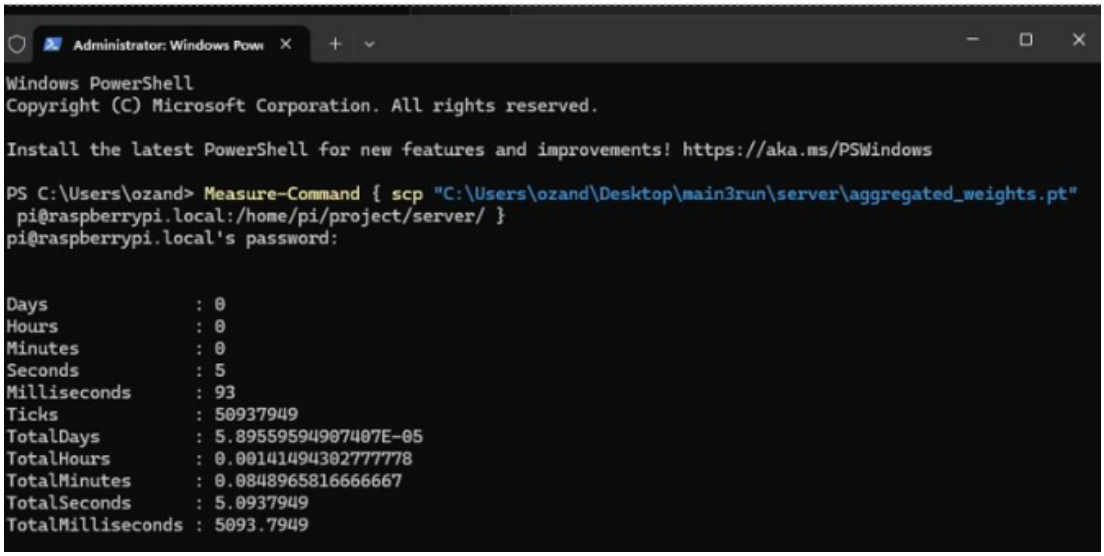

Figure 3.7: Speed of the data transfer of the weights.

The results also emphasize the success of the federated learning approach in maintaining data privacy. By keeping raw data localized on each client and only sharing model updates, the system minimized the risk of sensitive data exposure while still achieving collaborative training. This aspect is particularly valuable for industrial applications where data security is a priority.

Overall, the improvements in PR curves for Client 1, accuracy gains across all clients, and the efficient communication protocol highlight the effectiveness of the system. These results confirm the viability of the federated learning setup for scalable and privacy-preserving defect detection in real-world scenarios.

4. CONCLUSIONS

In conclusion, this project successfully implemented and measured its results of our federated learning system for defect detection on metal surfaces, achieving improvements in model accuracy, communication efficiency, and data privacy. The use of Raspberry Pi devices and personal computers in a collaborative setup showed the usage of scalable and private training across multiple devices.

Future work can focus on optimizing training time, integrating additional defect classes, and exploring hardware-specific enhancements to improve the system's overall performance and adaptability.

BIBLIOGRAPHY

[1] Visual Paradigm Online. “Diagrams.” (2025), [Online]. Available: https://online.visual-paradigm.com/ (visited on 01/14/2025).

[2] TensorFlow. “Tensorflow federated (tff).” (2025), [Online]. Available: https://www.tensorflow.org/federated (visited on 01/14/2025).

[3] OpenCV. “Opencv.” (2025), [Online]. Available: https://opencv.org (visited on 01/14/2025).

[4] Raspberry Pi. “Raspberry pi os.” (2025), [Online]. Available: https://www.raspberrypi.com/software/ (visited on 01/14/2025).

[5] Raspberry Pi. “Raspberry pi 4 real world tests.” (2025), [Online]. Available: https://unixetc.co.uk/2019/07/07/raspberry-pi-4-real-world-tests/ (visited on 01/14/2025).

[6] daniil.khoroshev.pmi@gmail.com. “Severstal-steel-defect-detection dataset.” (2025), [Online]. Available: https://universe.roboflow.com/daniil-khoroshev-pmi-gmail-com/severstal-steel-defect-detection (visited on 01/14/2025).

[7] Ultralytics. “Yolo model.” (2025), [Online]. Available: https://docs.ultralytics.com/ (visited on 01/14/2025).

[8] Wei Huanga, Tianrui Li, Dexian Wang, Shengdong Du, Junbo Zhang. “Fairness and accuracy in federated learning.” (2025), [Online]. Available: https://arxiv.org/pdf/2012.10069 (visited on 01/14/2025).

[1]–[8]