Federated Learning Implementation on Arduino

Report submitted in fulfillment of the requirements for the B.Tech. Project of

Fourth Year B.Tech.

by

Niranshi Agarwal and Khushi Jamod

19074009 and 19075039

Under the guidance of

Dr. Hari Prabhat Gupta

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI

Varanasi 221005, India

22 November 2022

Dedicated to

My parents, professors, exploratory project convener, mentor and everyone who helped and motivated me in successful completion of this project.

<u>Declaration</u>

I certify that

1. The work contained in this report is original and has been done by myself and the general supervision of my supervisor.

2. The work has not been submitted for any project.

3. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi
Date:22 November,2022

Niranshi Agarwal and Khushi Jamod
IDD Student and B.Tech. Student Respectively
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Certificate

*This is to certify that the work contained in this report entitled "**Federated Learning Implementation on Arduino**" being submitted by **Niranshi Agarwal and Khushi Jamod (Roll No. 19074009 and 19075039)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Place: IIT (BHU) Varanasi
Date: 22 November,2022

Dr Hari Prabhat Gupta
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Abstract

In this project, we have worked on implementing federated learning on arduino devices.The Arduino Nano 33 BL devices are used. The main motive of federated learning is to address privacy concerns and solve the problem of data silos. We have implemented this approach for word recognition problem. Here, the entire training process is performed on the arduino device, which is a tiny ml device, with limited memory resources and CPU. A small-sized model is used, with the training data recorded from the device itself. Here, we found that more frequent rounds of federated learning is needed, as the training dataset recordings tend to be small in number.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Recently, federated learning (FL) has emerged as a famous distributed machine learning prototype for its approach to addressing privacy concerns and solving the problem of data silos. People are more anxious than ever about who has access to and where their data is shared.

In FL, a joint global model is trained in a decentralized manner under the coherence of a central server. In every communication round, from the central server, each participating device downloads the current model and then, with its data, computes an updated model locally. Then these locally trained models are aggregated by the central server through averaging weights. Finally, a single integrated and updated global model is sent back to the devices. Furthermore, federated learning systems have the architecture to divide a large chunk of computing effort into tiny pieces that is suitable for many lower-capacity devices.

Machine Learning on edge devices is a rapidly growing field at the intersection of embedded systems and machine learning, allowing significant insights, data collection, and algorithmic development that were previously impossible. Here, two major concerns are no aggregation of data on a single server to access due to privacy issues, a new method to combine the learning of the edge devices that doesn't require a large amount of on-device memory, and continuous communication with a central server.

1.2 Related Work

The existing approaches to implement federated learning on edge devices, like Arduino, allow only inferential analysis on the edge devices, whereas the training is performed on other systems. This off-device training prevents modification to the model once deployed. The edge device cannot learn incrementally or directly. These are the different works in this direction:

1. Disabato and Roveri introduced an incremental algorithm based on transfer learning and k-nearest neighbor to support inference and on-device learning of ML and DL solutions on embedded devices and IoT units.

2. Transfer learning in combination with federated learning is explored in the work of Kopparapu et al., who proposed TinyFedTL. The advantage of transfer learning is that only a subset of the total neural network layers is trained.

Chapter 2

Architecture

2.1 Overview

This section describes the development of the implementation.

2.2 Hardware Setup

We must set up the hardware for the user to engage with the program. The application is installed on an Arduino Nano 33 BLE Sense board, which incorporates many necessary parts. The built-in IMU (inertial measuring units) sensors on the board will record the accelerometer data.

2.3 Neural Network Model

2.3.1 ANN Model

Our implementation focuses on on-device training of the neural network model. For that, the model should be small enough to fit into the memory requirements of the Arduino device. Hence it eliminates the possibility of using a complex model in the implementation. Therefore, the model consists of an input layer of size 30 (threedimensional data with ten readings for each sample), a single hidden layer of 4 nodes, and the output layer of three nodes. Finally, the model produces 132 weights and seven biases. The weights and biases are stored in 4B floats, contributing to the final size of the model, which is 556 B. These bytes are stored in the RAM of the board.

2.3.2 CNN Model

We also implemented a CNN model for training on the Arduino Nano BLE sense. The model consists of an input layer of 30 nodes, a 2x2 kernel, a hidden layer of 5 nodes, and an output layer of size 3. Finally, the model produces a total of 117 weights and biases.

2.4 Model training

2.4.1 ANN Model

Initially, the devices are given a model with random weights. The device records 15 samples, five samples for each class for training. AIFES library implements the ANN model on which the device is trained. Sigmoid and softmax activations are used for hidden and output layers with adam optimizer and cross-entropy loss.

2.4.2 CNN Model

A new model is obtained each time we start the application. The device records the accelerometer data for five samples of the three classes. The model uses the accelerometer readings for forward propagation, giving us the final values of the three output nodes. Mean square error is obtained with the output value and the expected one. Finally, the delta is calculated for each neuron to update weights and biases to perform the backpropagation.

2.5 Workflow

A federated learning server, numerous clients, and their interactions make up this design. After a predetermined number of client node training epochs, the federated learning server computes the model averaging using the received local models. The result of the model averaging is a new global model. The client nodes receive the new global model and train their local models using the available data at each node.
The communication between the federated learning server and the client nodes is done through Bluetooth. The client nodes(Peripheral devices) are connected to a central device which is then connected through the serial port to the federated learning server(FL server). The peripheral devices transfer the weights after training through Bluetooth to the central device, which transfers them further to the FL server via serial port. The FL server sends updated weights to the central device through the serial port, which then transfers them to peripheral devices. The federated learning server is implemented in python. The pyserial library is used here for the server and central device communication via serial ports. FedAvg technique is used here for the aggregation of the models. The parameters are averaged on the server after receiving all the weights from the clients.

2.6 Federated Learning

The devices start training as soon as they receive the total samples. The devices turn on their BLE as soon as they complete training. The peripheral devices get connected to the central device and set up communication between the FL server and peripheral devices. The central device gets connected to one peripheral device at a time. The peripheral devices send weights to the central device, and the central device sends them to the FL server via serial port. After receiving all the weights, the FL server aggregates them by averaging them and sends them back to each peripheral device via the central device. After receiving weights, the peripheral devices start another round of training. Testing is done after all the rounds of FL are completed.

The data is exchanged in bytes. For the FL server and central device, bytes are received at the serial input buffer in the central device from the server, and the bytes received by the server are sent through the serial output buffer of the device.

Chapter 3

Experiment

3.1 Data Collection and Training

Five samples of each gesture class are collected in each of the peripheral devices. For each gesture, the accelerometer data is captured through the IMU sensors of the Arduino nano BLE 33 sense. The peripheral device starts training after collecting 15 samples. Devices turn on their BLE after completing the training.

3.2 Model Transfer

Transferring model weights and architecture between the edge device and the global server was challenging. The communication between devices and the global server is done through a central device with Bluetooth and a python server that uses a pyserial library.

3.3 Evaluation

To evaluate the global model, the global model is tested on local devices with test samples individually. The final accuracy is calculated by taking the average of the accuracies obtained from all the devices in a particular iteration.

Chapter 4

Results

4.1 Accuracy

The inference accuracy was 83 percent for the ANN Model and 50 percent for the CNN model.

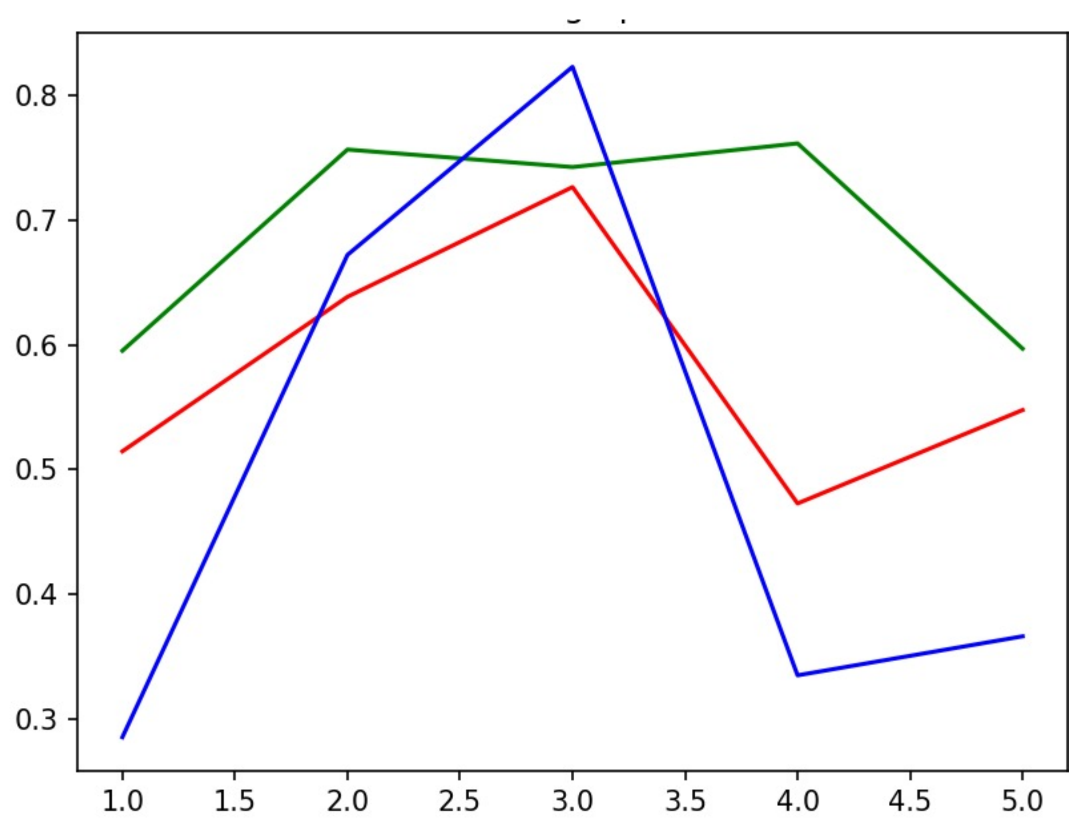

Figure 4.1 Accuracy vs Number of FL rounds for ANN Model Device 1 - Green, Device 2 - Red, Device 3 - Blue

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Here we presented the experimental work undertaken with a gesture recognition application that performs on-device training with federated learning of a neural network model using Arduino Nano 33 BLE Sense boards. The work gives practical insights into how on-device training can be implemented and analyzes the performance issues encountered when combining model training and federated learning on a recent microcontroller board.

5.2 Future Work

Future work will further explore federated learning training to more deeply understand its capacity to train versatile models if there are only limited local training data for the clients. For this purpose, we must explore how to combine transfer learning with pruning and model compression to train comparatively complex models on edge devices like Arduino.

Chapter 6

Bibliography

1) Arcadius Tokognon, C.; Gao, B.; Tian, G.Y.; Yan, Y. Structural Health Monitoring Framework Based on Internet of Things: A Survey. IEEE Internet Things J. 2017, 4, 619–635.

2) Kopparapu, K.; Lin, E. TinyFedTL: Federated Transfer Learning on Tiny Devices. arXiv 2021, arXiv:2110.01107.

3) Ray, P.P. A Review on TinyML: State-of-the-art and Prospects. J. King Saud Univ. Comput. Inf. Sci. 2021.

4) Mathur, A.; Beutel, D.J.; de Gusmão, P.P.B.; Fernandez-Marques, J.; Topal, T.; Qiu, X.; Parcollet, T.; Gao, Y.; Lane, N.D. On-device Federated Learning with Flower. arXiv 2021, arXiv:2104.03042.

5) Ren, H.; Anicic, D.; Runkler, T. TinyOL: TinyML with Online-Learning on Microcontrollers. arXiv 2021, arXiv:2103.08295.

6) Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated Machine Learning: Concept and Applications. ACM Trans. Intell. Syst. Technol. 2019, 10, 1–19.

7) https://arxiv.org/abs/2110.01107