Fontys ICT

**[Labeling noise to harmony]**

**IT AND SOFTWARE**

**Technical Report**

**Project Manager: Li Li**

**Period Ending: [20-02-2024] - [xx]**

Table of Contents

[Research 1](#_Toc164328088)

[1.1 What is a federated learning system? 2](#_Toc164328089)

[How does it work? 2](#_Toc164328090)

[Advantages 2](#_Toc164328091)

[Disadvantages 2](#_Toc164328092)

[Applications 2](#_Toc164328093)

[Future of Federated Learning 2](#_Toc164328094)

[1.2 Implementing a federated learning system 3](#_Toc164328095)

[2.1 What are the effects of noisy labels present in local datasets on the overall federated learning model's performance? 5](#_Toc164328096)

[3.1 How can I improve the accuracy of a federated learning model by mitigating label noise in local datasets? 8](#_Toc164328097)

[Embedding-based Discovery of Noisy Labels 9](#_Toc164328098)

[Model Confidence as a Proxy for Label Noise 10](#_Toc164328099)

[Nearest Neighbor-based Correction (NNC) 10](#_Toc164328100)

[Adaptive Knowledge Distillation (AKD): 10](#_Toc164328101)

[Noise-aware Federated Averaging (NA-FedAvg) 11](#_Toc164328102)

# Research

This project is investigating the following main question:

* How to decrease label noise in a federated learning system, so that the accuracy of the model gets improved.

To understand the context, I first explore:

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

# What is a federated learning system?

Imagine training a powerful AI model across millions of devices without collecting their data. Federated Learning achieves this by:

* **Decentralized Training:** The model resides on individual devices, learning from local data privately.
* **Privacy-Preserving Mechanisms:** Secure aggregation techniques and differential privacy protect raw data.
* **Communication Efficiency:** Only model updates, not original data, are exchanged, minimizing bandwidth usage.

## How does it work?

1. **Model Initialization:** A pre-trained model is distributed to devices.
2. **Local Training:** Each device trains the model on its local data privately.
3. **Model Updates:** Local updates are encrypted and sent to a central server.
4. **Model Improvement:** The server aggregates updates without revealing individual data, improving the global model.
5. **Iteration:** The updated model is sent back to devices, repeating the process for further improvement.

## Advantages

* **Data Privacy:** Individuals retain control over their data, complying with stricter regulations.
* **Security:** Centralized data storage vulnerabilities are eliminated.
* **Data Heterogeneity:** Diverse data from various sources leads to better model performance.
* **Scalability:** Learning happens on individual devices, reducing centralized computation burden.

## Disadvantages

* **Communication Overhead:** Frequent communication between devices and server can be resource-intensive.
* **Heterogeneity Challenges:** Device capabilities and data distributions can impact model accuracy.
* **Privacy Guarantees:** Balancing privacy with model performance requires careful design.

## Applications

* **Mobile and Edge Computing:** Personalized recommendations, on-device language translation.
* **Healthcare:** Collaborative disease prediction, privacy-preserving medical imaging analysis.
* **Finance:** Fraud detection, personalized financial advice.
* **Internet of Things (IoT):** Federated learning for smart devices, anomaly detection.

## Future of Federated Learning

FL research actively addresses challenges like communication efficiency, privacy guarantees, and applicability to complex tasks. Its potential to revolutionize machine learning while respecting privacy is undeniable.

# Implementing a federated learning system

During this project our project group is using Flower. Flower is an unified approach to federated learning, analytics, and evaluation. It makes it way more simple to turn a centralized learning project into one using federated learning.

To change a project using centralized learning to federated learning is quite straightforward.

Since in federated learning you treat clients separately, you have to create a FlowerClient that handles a single client.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

The class takes the trainloader, valloader and net (which is the model).

Each client takes it’s own train and valloader, so the data is differently distributed between clients.

Afbeelding met tekst, Lettertype, schermopname, wit

Automatisch gegenereerde beschrijving

In flower you can choose to create a simulation, where all clients and the server run on one instance, but it is also possible to run all clients and the server in separate instances (terminals). Using simulation is the easiest approach, so that’s what I am using.

To run a simulation you have to pass the amount of clients, so the server knows for how many clients to look out for, a method to create instances of the FlowerClient class, config to define how many rounds the simulation should run for, a strategy to aggregate all the data (defaults to FedAvg) and resources which defines how much CPU/GPU you want to use for the simulation.

Afbeelding met tekst, Lettertype, schermopname, algebra

Automatisch gegenereerde beschrijving

# 2.1 What are the effects of noisy labels present in local datasets on the overall federated learning model's performance?

To find this out I started with using an example project of Cifar-10, which is a dataset with images and labels. First I ran a federated learning simulation with the current dataset. And then I mixed up the labels randomly to see in what way it would affect the performance of the simulation.

While using the clean dataset of Cifar-10, I received up to 0.7 accuracy with centralized learning.

Afbeelding met tekst, lijn, diagram, Perceel

Automatisch gegenereerde beschrijving

While using the Cifar-10N dataset, which practically is the same dataset, but then with scrambled labels. Only achieved an accuracy up to 0.55 with centralized learning.

Afbeelding met tekst, lijn, diagram, schermopname

Automatisch gegenereerde beschrijving

Then I started implementing federated learning for the Cifar-10 Dataset. With the FedAvg strategy, I achieved an accuracy of 0.57 after only 4 epochs.

Afbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijving

Since I am focusing on Speech Data, I then started working with the Speech Commands dataset of google. This dataset has 1000 short speech clips for 18 different words. The only issue with this dataset was that it took a really long time to train and run. So I manually shrinked the dataset down so it only has 100 speech clips for the different words.

First of all starting with clean labels and centralized learning I achieved 0.68 accuracy after only 5 epochs, with only a train loss of 0.7.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Then I scrambled up the labels of the dataset and received this accuracy:

TODO

After finishing with centralized learning, I starting training with federated learning. Once again I started off with clean labels.

# 3.1 How can I improve the accuracy of a federated learning model by mitigating label noise in local datasets?

First of all, it is useful to know what label noise itself is. A simple example is when you have a dataset with images. And there is a cat, but during the creation of the dataset a human accidently named it a dog. Then the label is not correct with the image in the dataset. And having these incorrect labels is called label noise.

To figure out ways to handle label noise, I studied a research paper made by students of TU Eindhoven. <https://paperswithcode.com/paper/federated-learning-with-noisy-labels>

This paper explores a challenge in training machine learning models called label noise. Label noise happens when data gets labeled incorrectly.

In traditional deep learning, researchers have explored various techniques to address label noise, including:

* **Loss correction techniques:** These techniques adjust the importance of training examples to reduce the impact of noisy labels.
* **Regularization techniques:** These techniques make the model less sensitive to specific data points, which can help reduce the influence of noisy labels.
* **Data augmentation techniques:** These techniques artificially create new training data to improve the model's ability to handle variations in the data, including noisy labels.
* **Noise filtering techniques:** These techniques identify and remove noisy labels from the training data before training the model.
* **Early stopping:** This technique stops training the model before it memorizes the noisy labels.

Federated Learning (FL) is a privacy-preserving approach where models are trained on devices without sending all data to a central server. However, FL introduces new challenges for handling label noise because the noise patterns can vary across different devices.

This paper provides a summary of existing techniques for handling label noise in FL. These techniques can be broadly categorized into:

* **Data filtering techniques:** These techniques aim to identify and remove noisy data points on devices.
* **Label correction techniques:** These techniques attempt to correct noisy labels on devices.
* **Weighted aggregation techniques:** These techniques adjust how the model updates from different devices are combined based on the estimated noise levels on those devices.

## Embedding-based Discovery of Noisy Labels

a new method to find noisy labels in FL using embeddings. Embeddings are a way to represent data points as locations in a high-dimensional space. They use a pre-trained model (like a map) to create these embeddings for each data point on a device (like pinpointing a location on the map).

Because the embeddings are created without using the labels, they are not affected by the label noise itself (like using a map that doesn't depend on the names of places). Then, they use a technique called k-Nearest Neighbors (kNN) to find outliers in the embedding space. These outliers are likely to be noisy data points (like a house on a lake in the middle of a desert on the map).

Finally, they use a voting approach to potentially correct the labels of these outliers based on their neighbors. They estimate the amount of label noise on each device by counting how many labels they think are wrong based on the embeddings.

This approach is interesting because it can find noisy labels on devices without sending all the data to a server, which helps protect privacy in FL.

## Model Confidence as a Proxy for Label Noise

a new method to find noisy labels in FL without using a separate pre-trained model. Instead, they use a scoring function to rank the data points on each device based on how confident the model is about its predictions. A low score indicates the model is unsure about the label, which suggests it might be noisy.

Here's how it works:

1. **Scoring Function:** They use a function called "energy score" to compute a score for each data point. This score tells you how confident the model is about its prediction.
2. **Threshold:** They set a threshold based on the scores of all the data points on a device. Data points with scores below the threshold are considered likely to have noisy labels.
3. **Noise Level:** They estimate the amount of label noise on each device by counting how many data points have scores below the threshold.

## Nearest Neighbor-based Correction (NNC)

1. **Find Noisy Labels:** They use embeddings and kNN to find data points with likely noisy labels, just like before.
2. **Correct Noisy Labels:** When they find a data point with a likely noisy label, they use the kNN prediction as the new, corrected label. They update the labels on the device itself.
3. **Train Model:** The device trains the model using the corrected data points.

## Adaptive Knowledge Distillation (AKD):

instead of directly changing the labels, AKD trains the model in a way that avoids memorizing the noisy data points.

1. **Find Noisy Labels:** They estimate the amount of label noise on each device, just like before.
2. **Help Model Avoid Noisy Data:** They use a pre-trained model or the model from the server to create a kind of "hint" for the device's model. This hint helps the device's model focus on learning the correct patterns from the data and avoid memorizing the noisy labels.
3. **Train Model:** The device trains the model using the original data and the "hint" from the server.

## Noise-aware Federated Averaging (NA-FedAvg)

Instead of changing the labels or the model itself, they change how the model updates from different devices are combined at the server.

1. **Estimate Noisy Labels:** Devices estimate how much noisy data they have, just like before.
2. **Combine Updates with Noise in Mind:** When the server combines the updates from all the devices, it gives more weight to the updates from devices with less noisy data and less weight to the updates from devices with more noisy data.

Attached to the paper is a project called FedLN, and with this project they also analyzed speech data. And for all the methods they proposed and tried out, Nearest Neighbor-based Correction (NNC) works the best for speech datasets. That’s why I will first focus on implementing this strategy to mitigate label noise.

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving