FedDebug: Systematic Debugging for Federated Learning Applications

Waris Gill[1], Ali Anwar[2], Muhmmad Ali Gulzar[1]

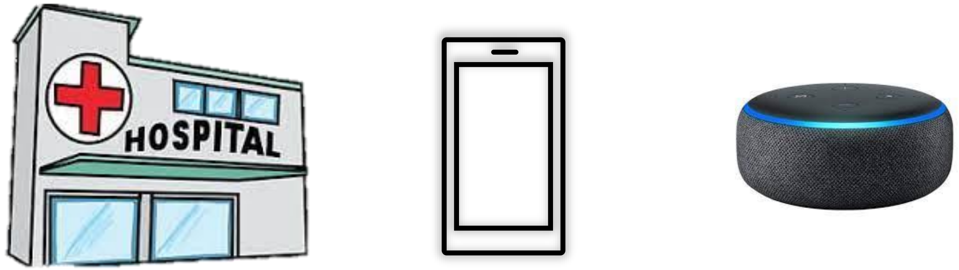
The 45th IEEE/ACM International Conference on Software Engineering

[1] VIRGINIA TECH

[2] UNIVERSITY OF MINNESOTA

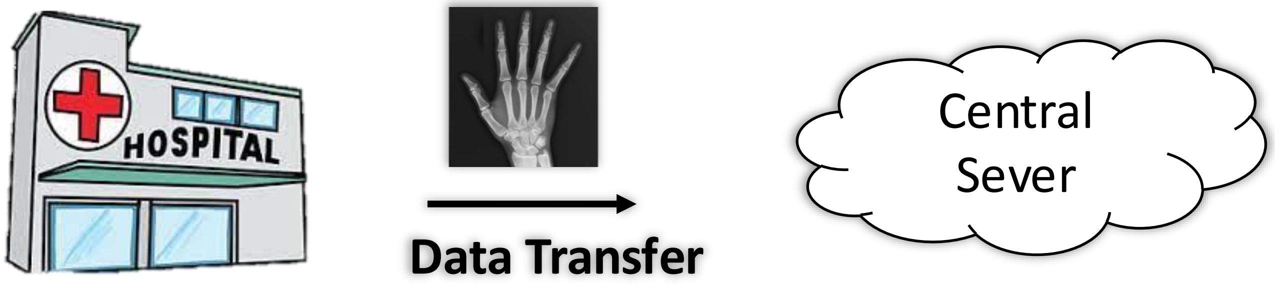
Functional Reusable Available

Why Federated Learning (FL)?



Hospital, phones and smart devices generate wealth of data.

ML training require transfer of data to the central server.



Data Transfer

Central Sever

Simply **sending raw data** to train an ML model is **not feasible**:

- ❑ Data is sensitive
- ❑ Privacy laws enforced by the governments

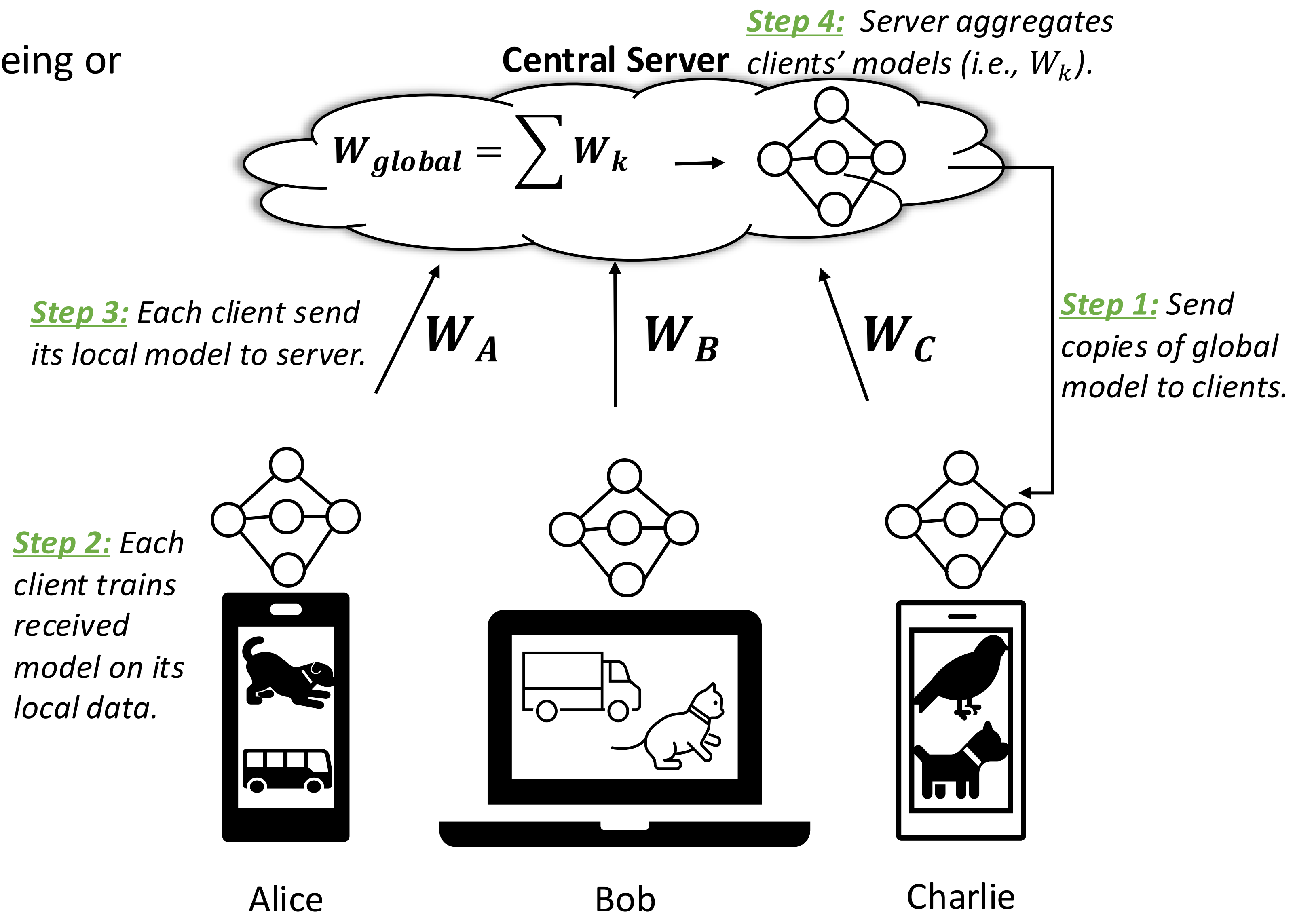
HIPAA
Health Insurance Portability
and Accountability Act

GDPR

What is Federated Learning (FL)?

FL trains an AI model without anyone seeing or touching private data.

- ❑ Step 1-4 is a single FL training **round.**
- ❑ Training continues for hundreds of rounds.



Step 4: *Server aggregates clients' models (i.e., W_k).*

Step 1: *Send copies of global model to clients.*

Step 2: *Each client trains received model on its local data.*

Step 3: *Each client send its local model to server.*

Real World Examples

Siri

Alexa

Google's Gboard

Takeaway: *FL trains high quality AI model without accessing clients' private data.*

Debugging Problem in FL

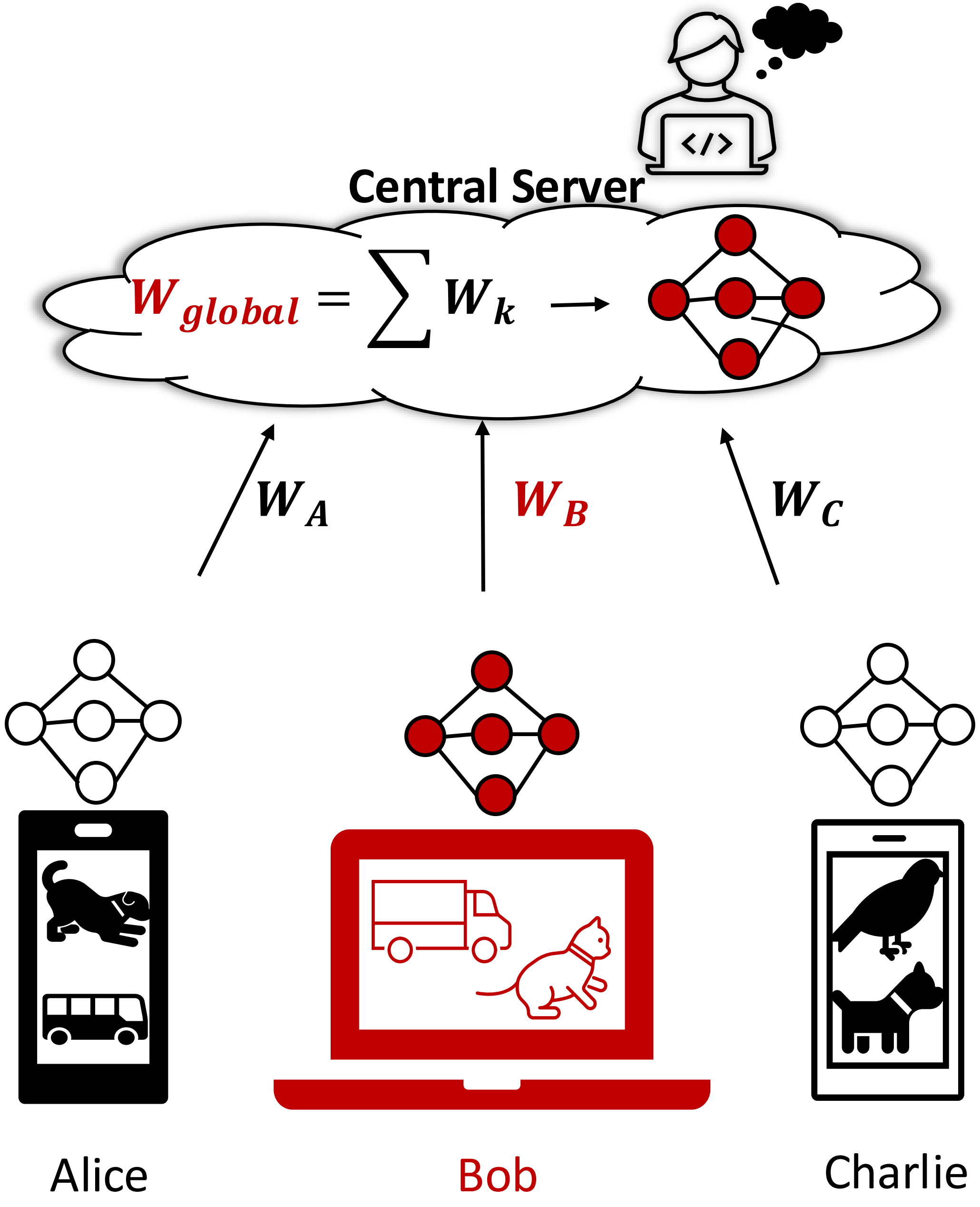
- ❑ Suppose that **Bob's model** becomes **faulty** during its local training.

  Faulty Client

  - ❑ Natural (faulty sensor/camera)
  - ❑ Malicious (Backdoor Attack)

- ❑ During aggregation, Bob's model ($\boldsymbol{W_B}$) also makes the global model (W_{global}) faulty.

How can an *FL developer* at the central server, automatically find Bob?

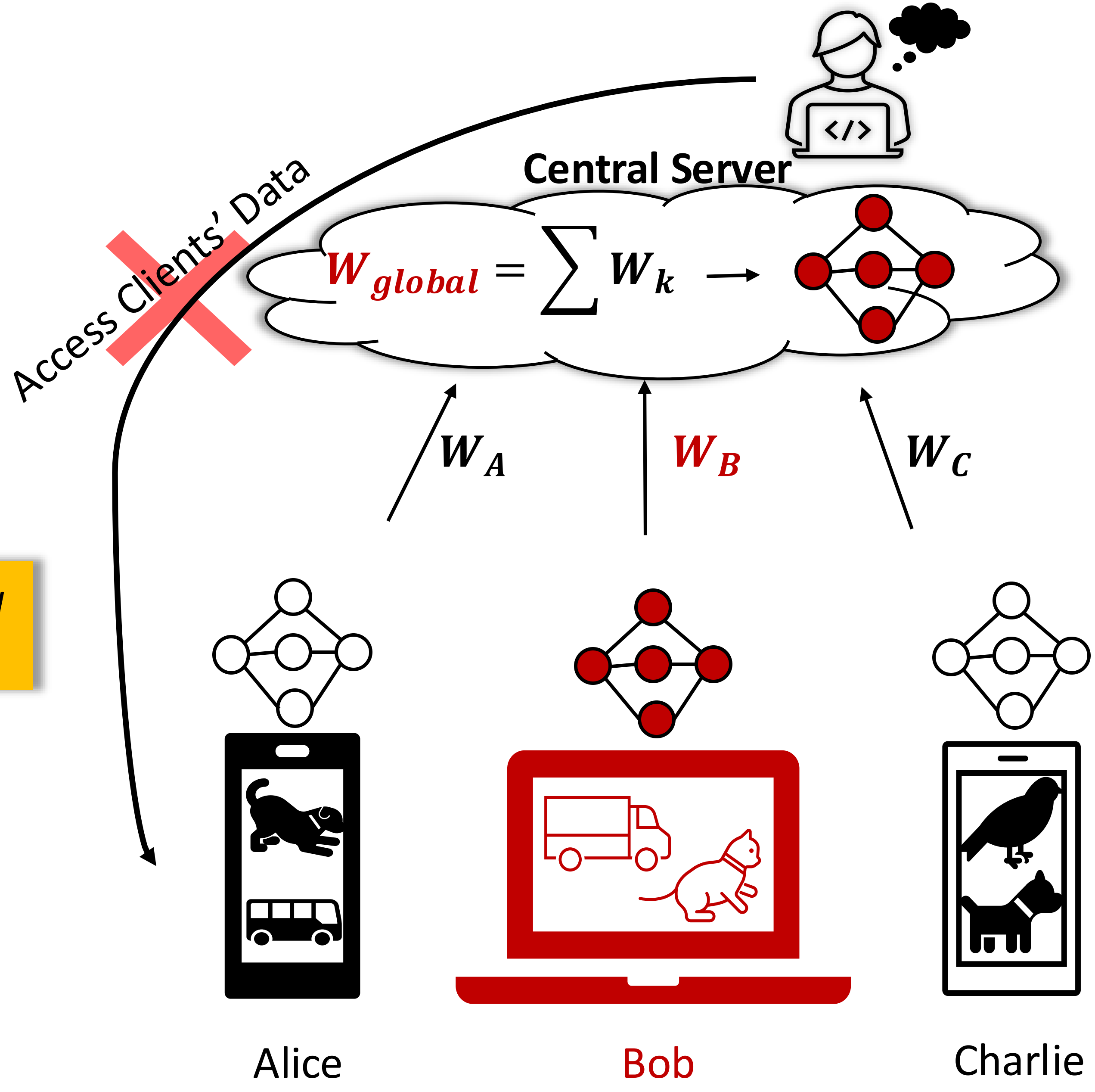


Trivial Solution

Developer accesses the clients' data to evaluate each model to find the faulty client.

However, FL forbids to access clients' data.

How do we find Bob without accessing clients' data or collecting new dataset at the aggregator?



Our Contribution: FedDebug

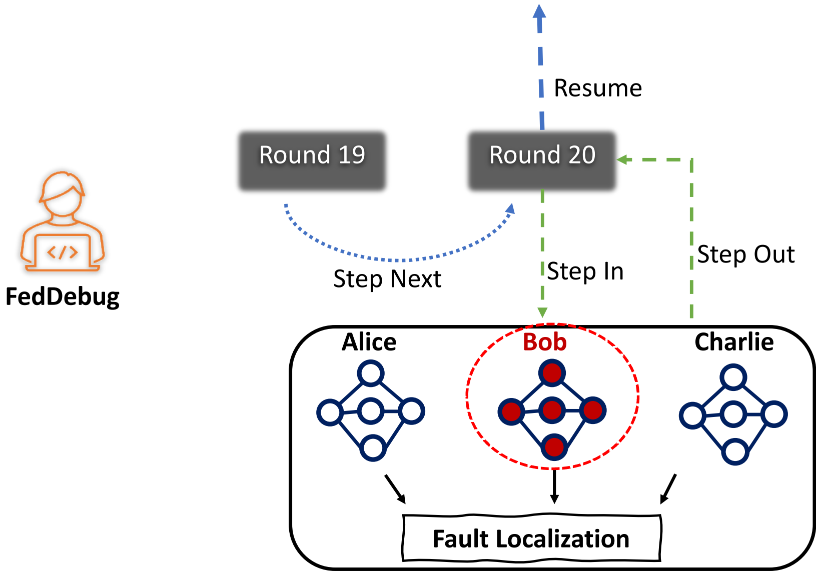
Interactive Debugging

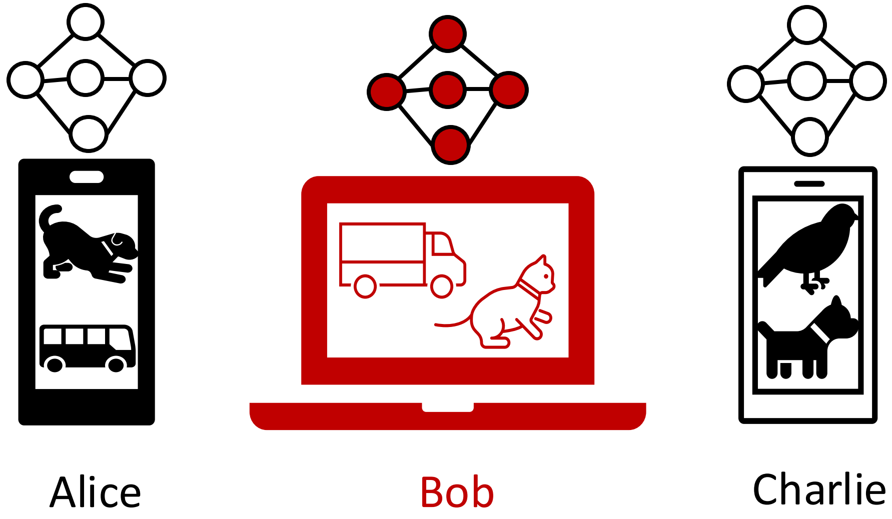
+

Fault Localization

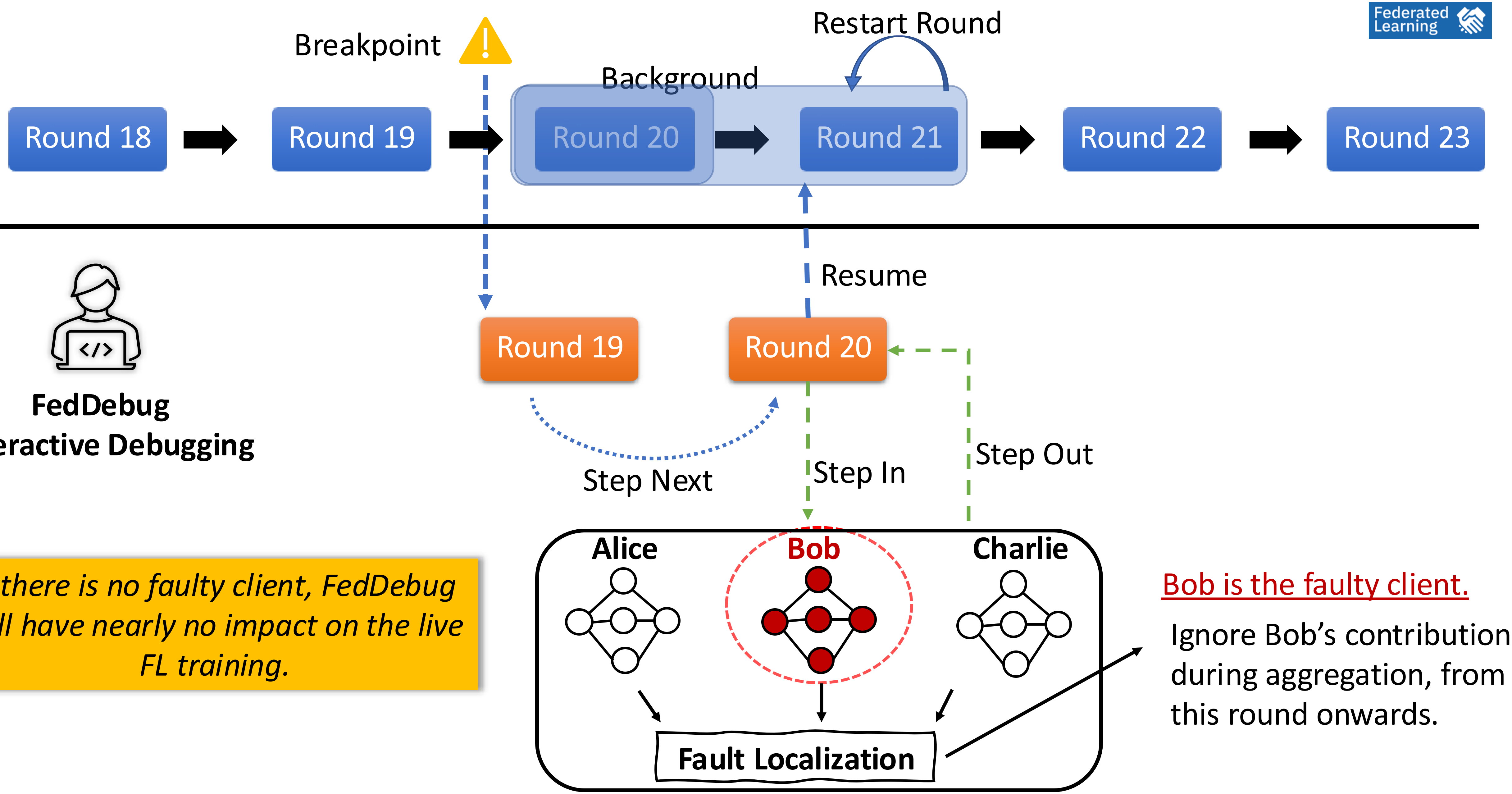
FedDebug's lightweight Interactive debugging assist a developer to inspect any FL training round.

FedDebug's fault localization technique finds the faulty client (Bob) during interactive debugging.





Interactive Debugging- with a Faulty Client



Breakpoint

Background

Restart Round

Round 18 → Round 19 → Round 20 → Round 21 → Round 22 → Round 23

Resume

Round 19

Round 20

Step Next

Step In

Step Out

FedDebug
Interactive Debugging

Alice

Bob

Charlie

Fault Localization

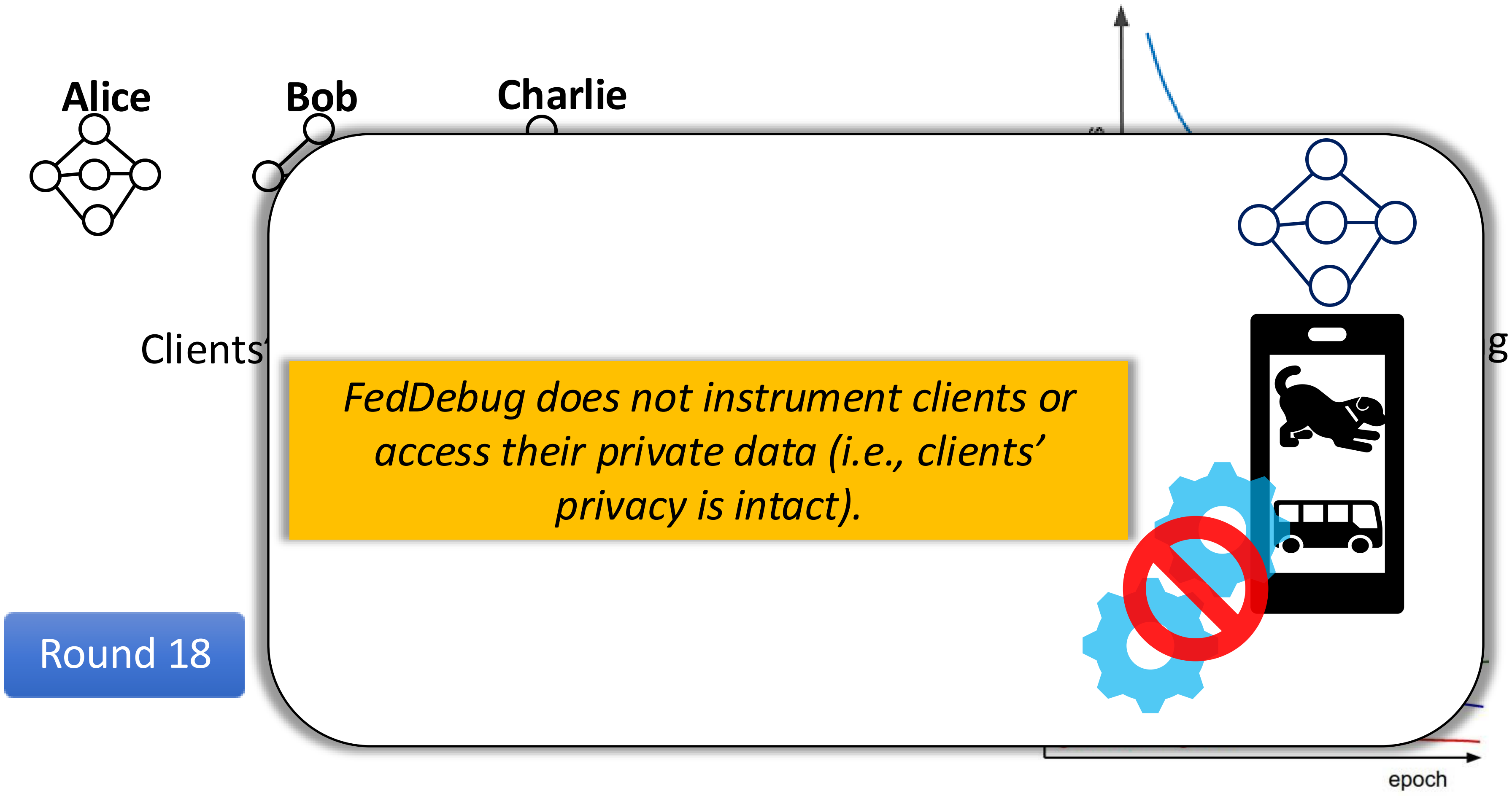
If there is no faulty client, FedDebug will have nearly no impact on the live FL training.

Bob is the faulty client.

Ignore Bob’s contribution, during aggregation, from this round onwards.

What information is collected in FedDebug?

FedDebug collects:

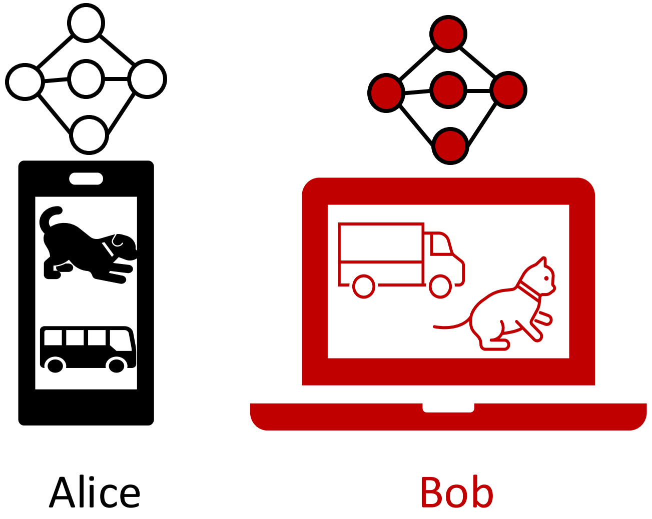


FedDebug does not instrument clients or access their private data (i.e., clients' privacy is intact).

Round ID

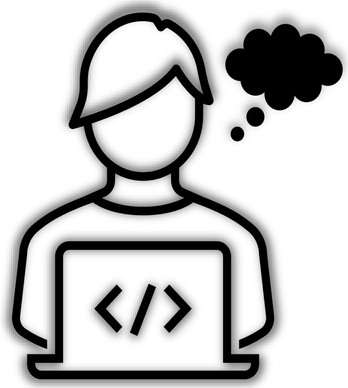
Hyperparameters (e.g., learning rate, epochs)

Localizing Faulty Clients in FL



Now, let's discuss how FedDebug localizes Bob at the central server.

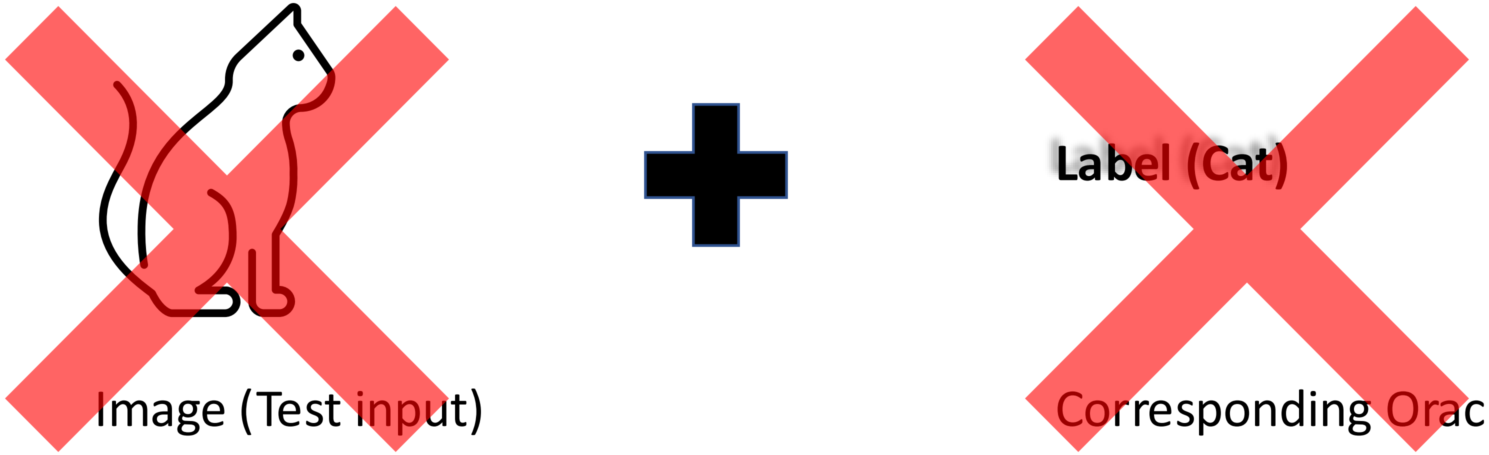
How to automatically find a faulty Client in FL?



To find a fault we require two things :

- □Test Input
- □Test Oracle

Example: To test a neural network we require



Label (Cat)

Image (Test input)

Corresponding Oracle

In FL, Developer can't access the clients' data, which limits existing ML testing solutions.

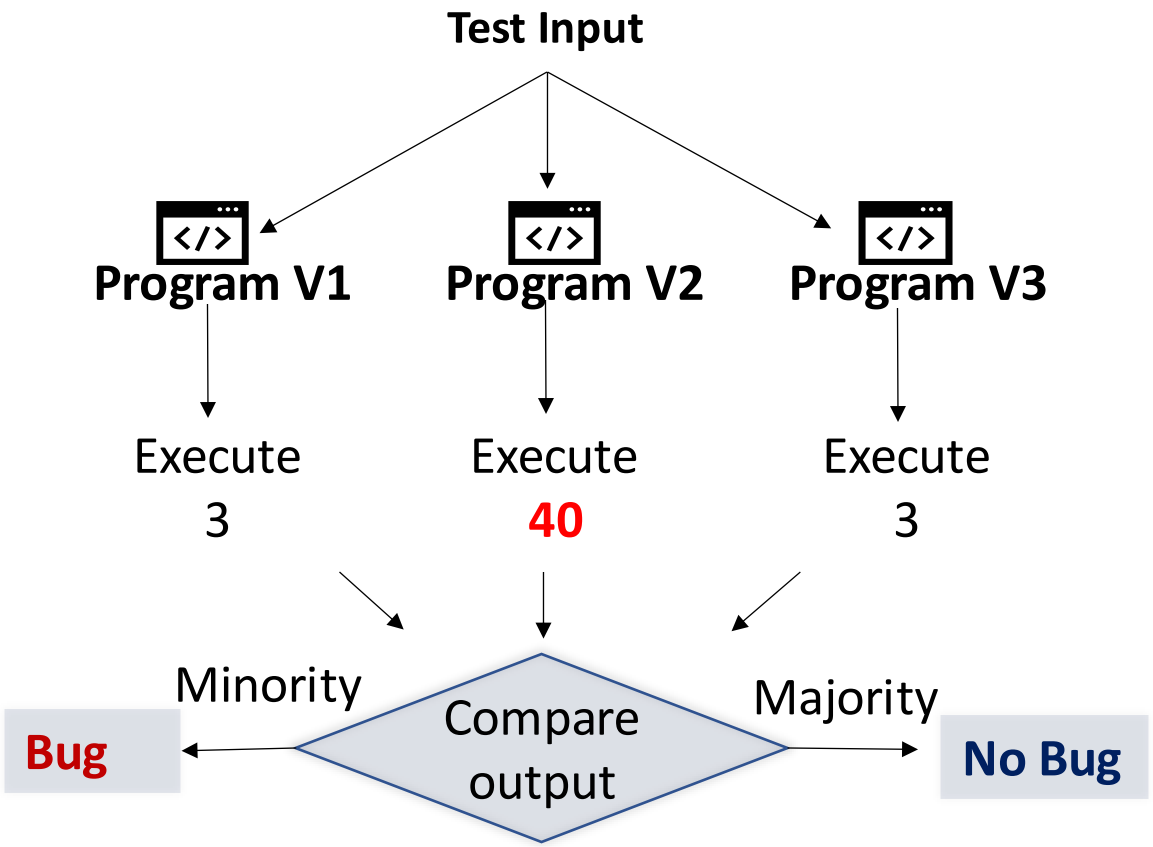
*One possible way to fix this issue is with **Differential Execution**.*

Background: Differential Execution

It executes two or more **comparable programs** on the **same test input** and compare the resulting outputs to identify a **bug**.

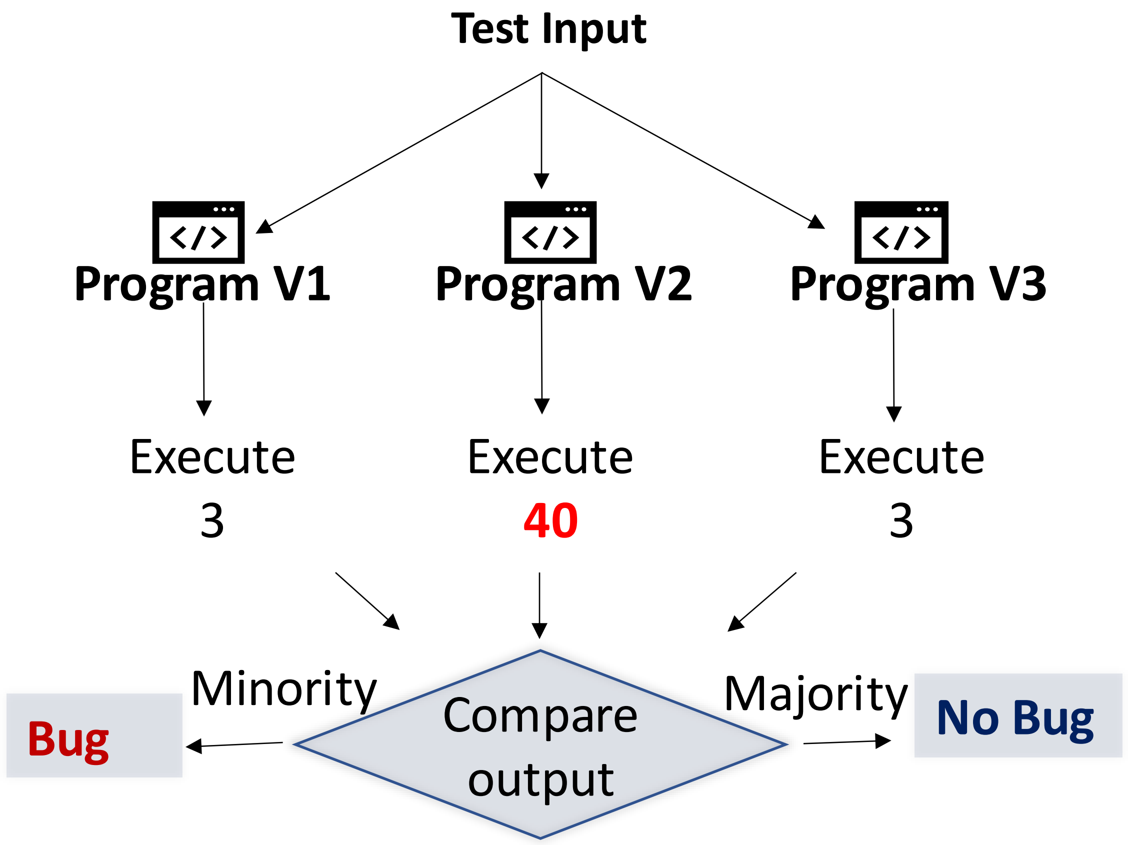
Comparison can be done at **different levels**:

- Output comparison
- Byte code execution comparison
- Crashing Comparison



Differential Execution in Federated Learning

Programs Differential Execution



Test Input

Program V1

Program V2

Program V3

Execute 3

Execute **40**

Execute 3

Compare output

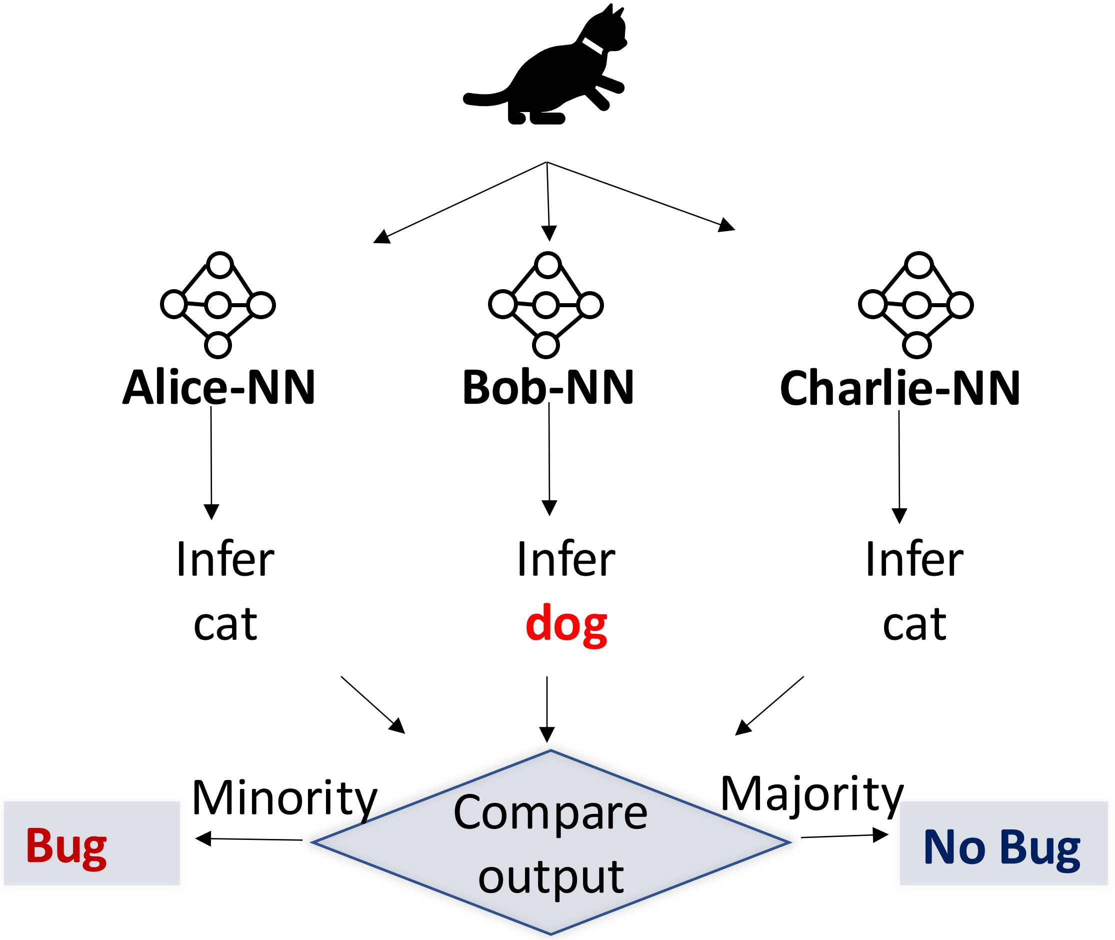
Minority

Majority

Bug

No Bug

FL Clients' Models Differential Execution



Alice-NN

Bob-NN

Charlie-NN

Infer cat

Infer **dog**

Infer cat

Compare output

Minority

Majority

Bug

No Bug

Problem: *The FL developer cannot access clients' data. How can we solve this issue?*

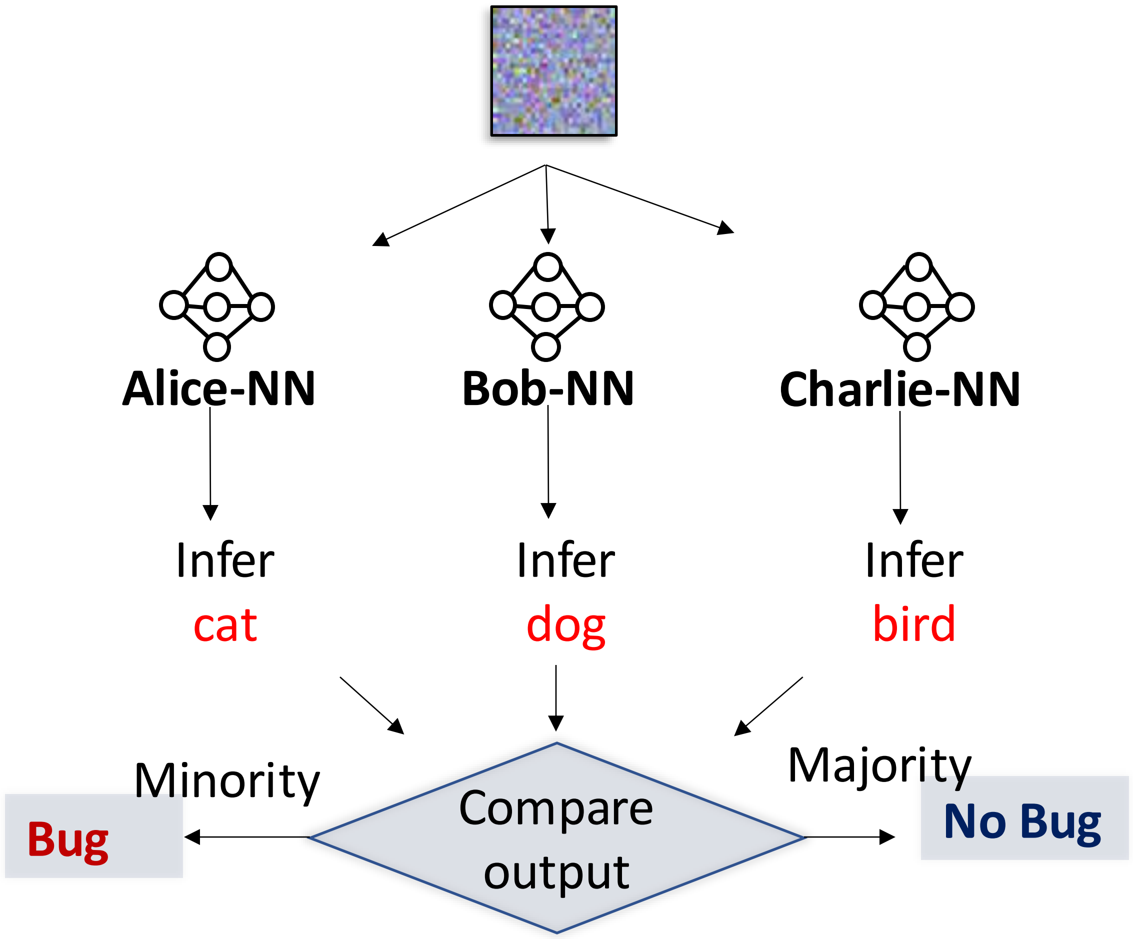
Possible Solution: *Generate random inputs at central server.*

Differential Execution in FL: Random Input

- Its impossible to assign a real-label to a **random input**. Each client may produce **different outputs**.

- How can we solve it?
  Similar to **byte code execution comparison,** compare the **internal behaviors** of clients' models.

Clients' Models Differential Execution on Random Input



Alice-NN

Bob-NN

Charlie-NN

Infer cat

Infer dog

Infer bird

Compare output

Minority

Bug

Majority

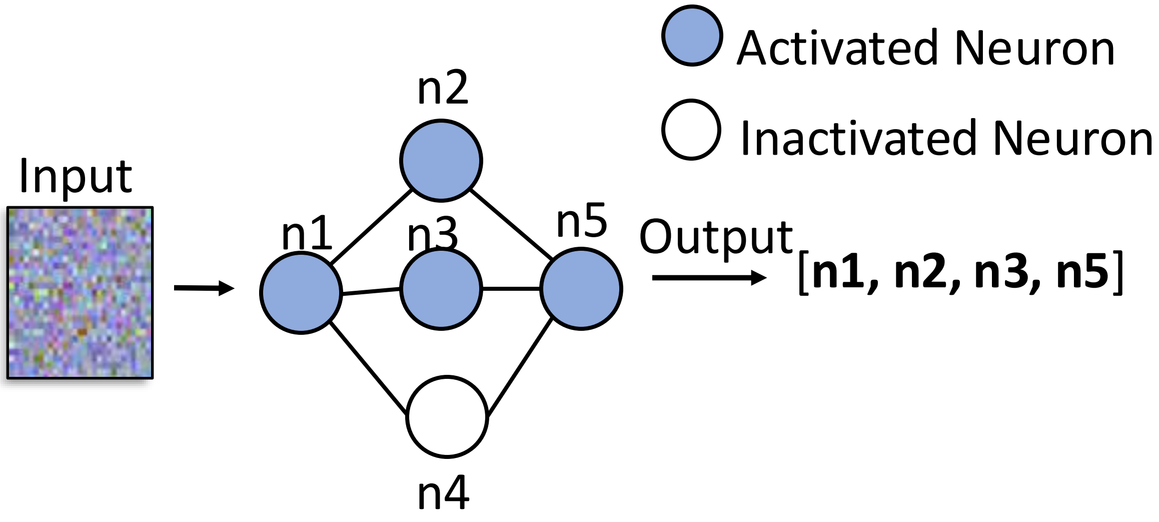
No Bug

Faulty client will have different internal behavior w.r.t others.

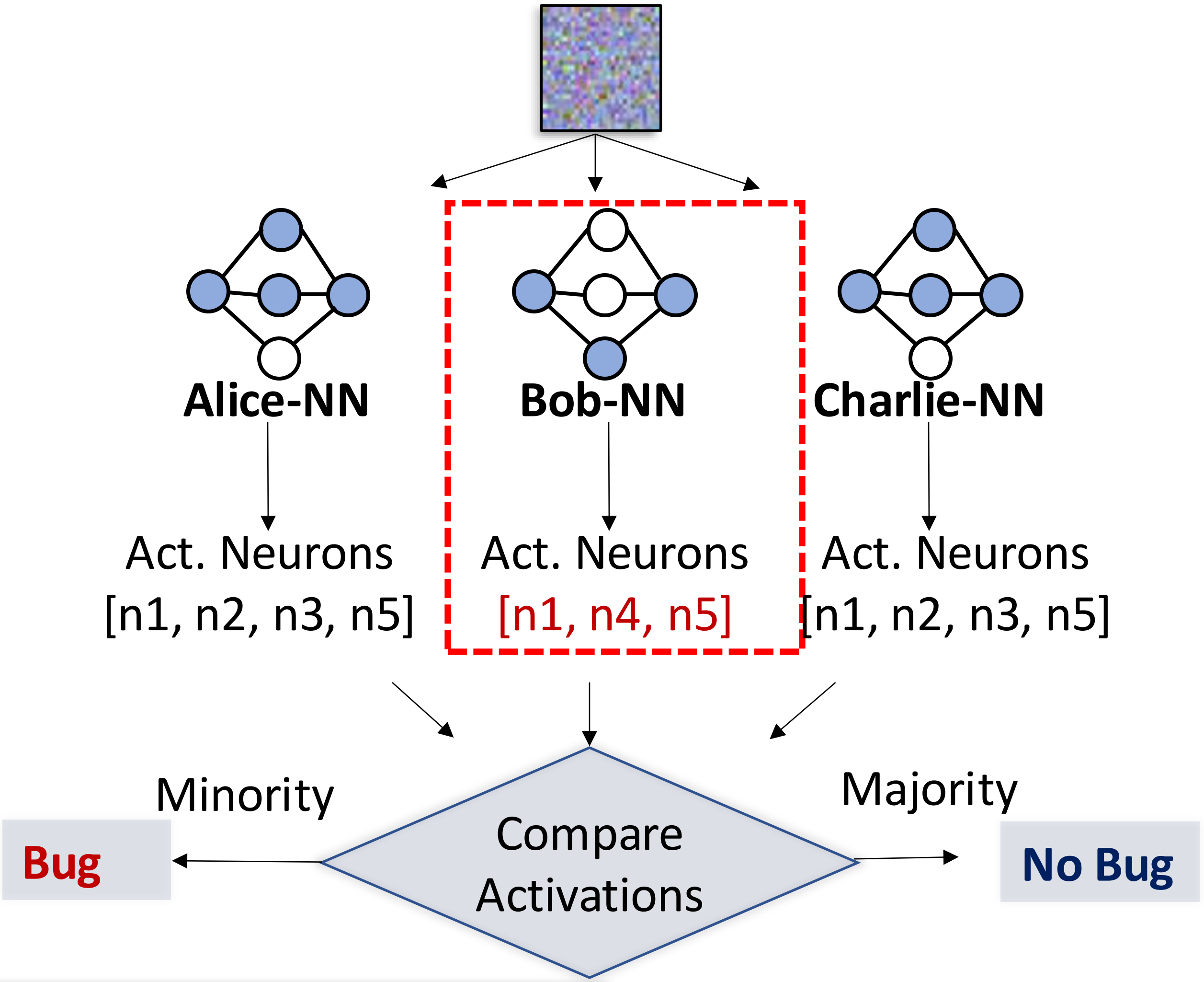
Differential Execution in FL: Capturing Client Behavior

How do we capture internal behavior of a neural network?

Capture the activated neurons on a given input.



Differential Execution with <u>Neuron Activations</u>



Act. Neurons [n1, n2, n3, n5]

Act. Neurons [n1, n4, n5]

Act. Neurons [n1, n2, n3, n5]

Minority

Compare Activations

Majority

Bug

No Bug

Bob is a faulty client as its activations are different w.r.t to other clients.

FedDebug Implementation

- FedDebug is supported in IBMFL framework.
- Fault localization is completely independent of IBMFL framework.

IBM®

Federated Learning

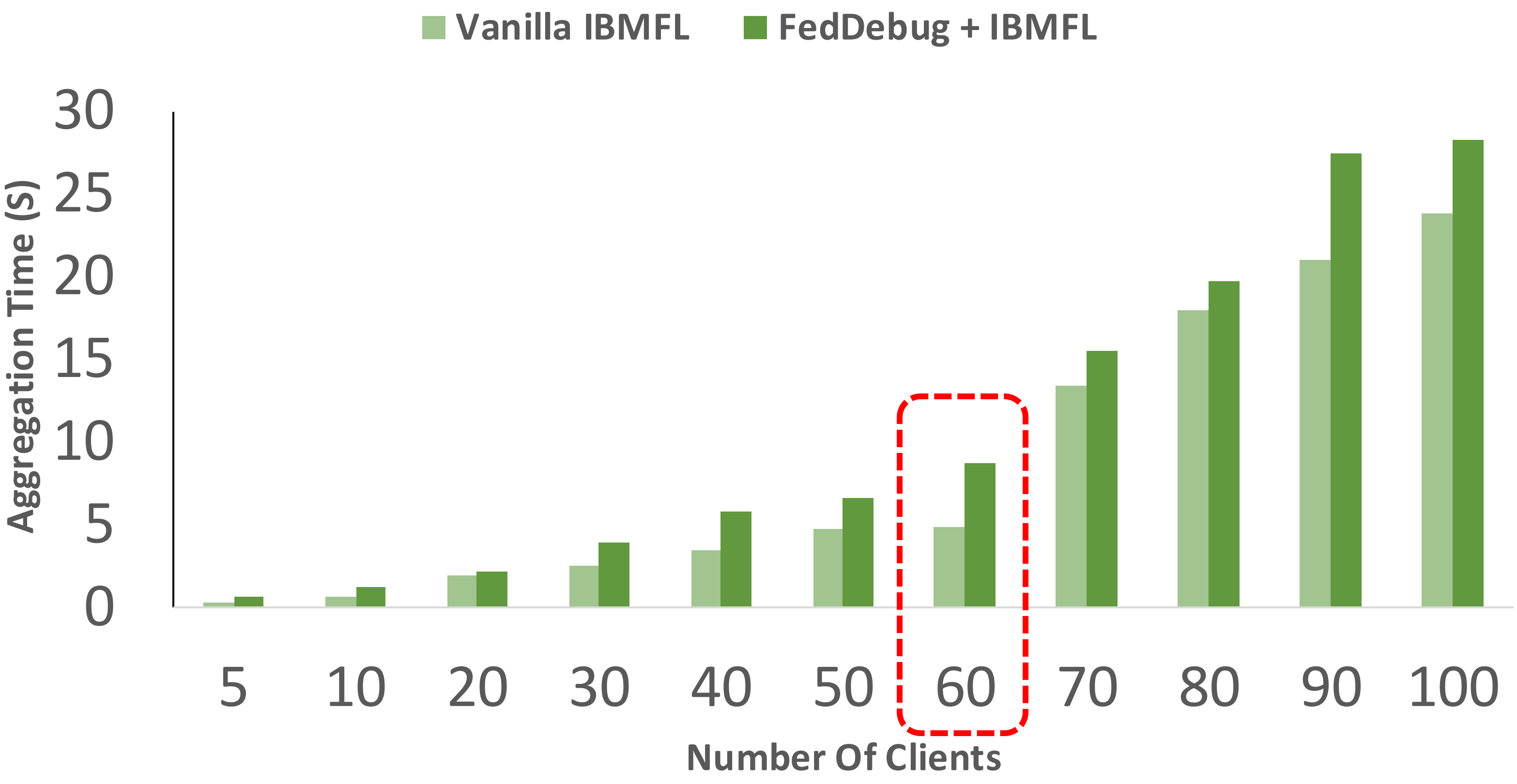
Evaluation Goals

❑Performance

❑Fault Localization Accuracy

❑Localizing Multiple Faulty Clients

Performance: Aggregation Overhead



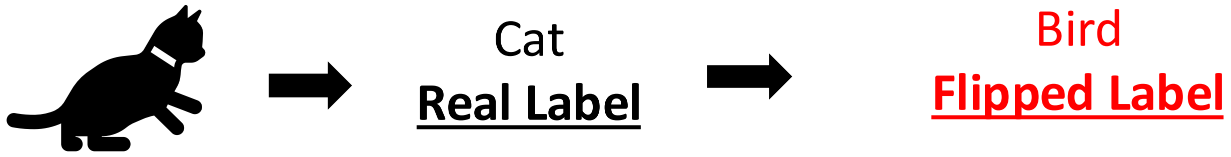
Example: In **60 Clients setting**:

- IBMFL aggregation time is 4.8 seconds.
- FedDebug+IBMFL aggregation time is 8.7 seconds.

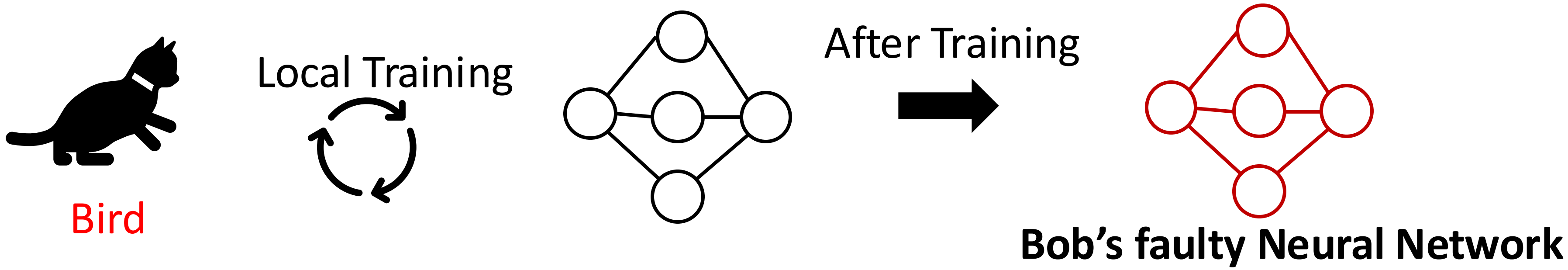
FedDebug adds about 48% to the aggregation time, but **it's negligible at just 1.2%** compared to round training time.

How to make a client (Bob) faulty in FL?

Flipped the labels of the client’s training data.



When Bob locally trained its neural network on flipped images, it becomes a faulty client.



GitHub

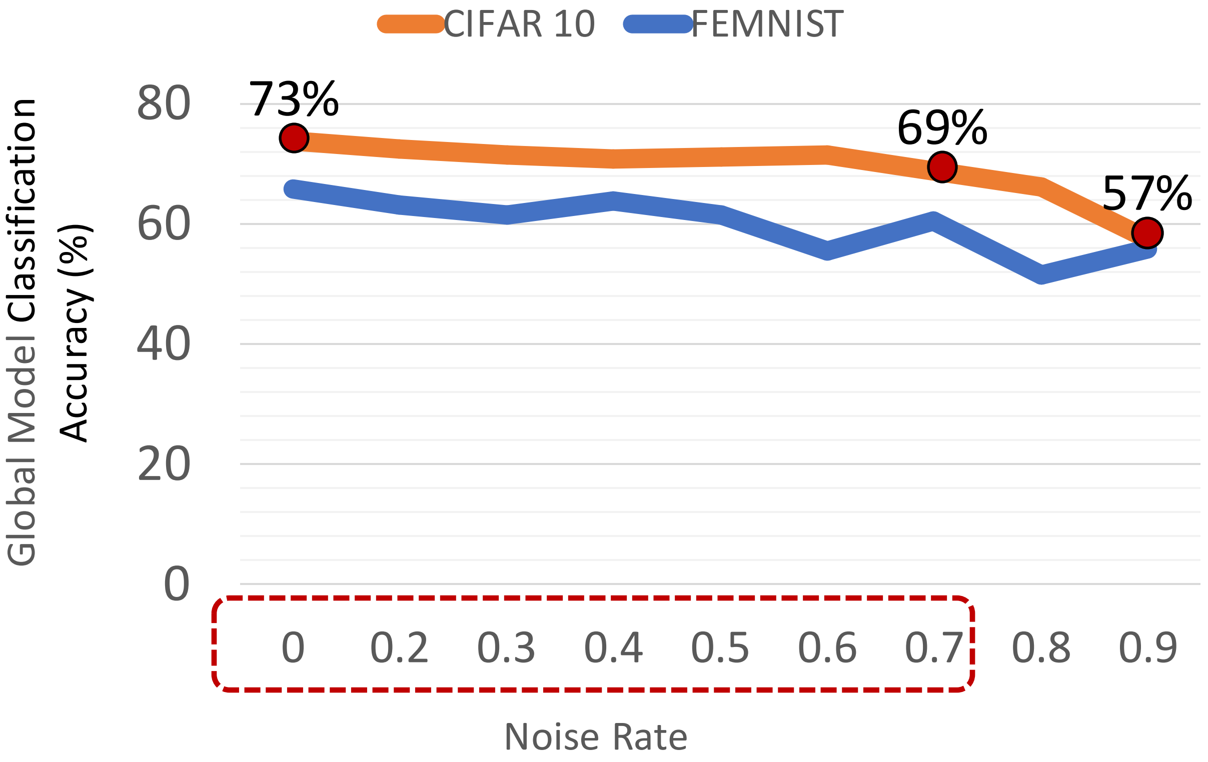
Strength of a faulty client is determined by the **noise rate.**

$$\textbf{noise rate} = \frac{\#\ of\ Flipped\ Labels}{Total\ Labels}$$

*We constructed 68 unique FL configurations by varying datasets, clients, architectures, number of faulty client as a **benchmark** for future research.*

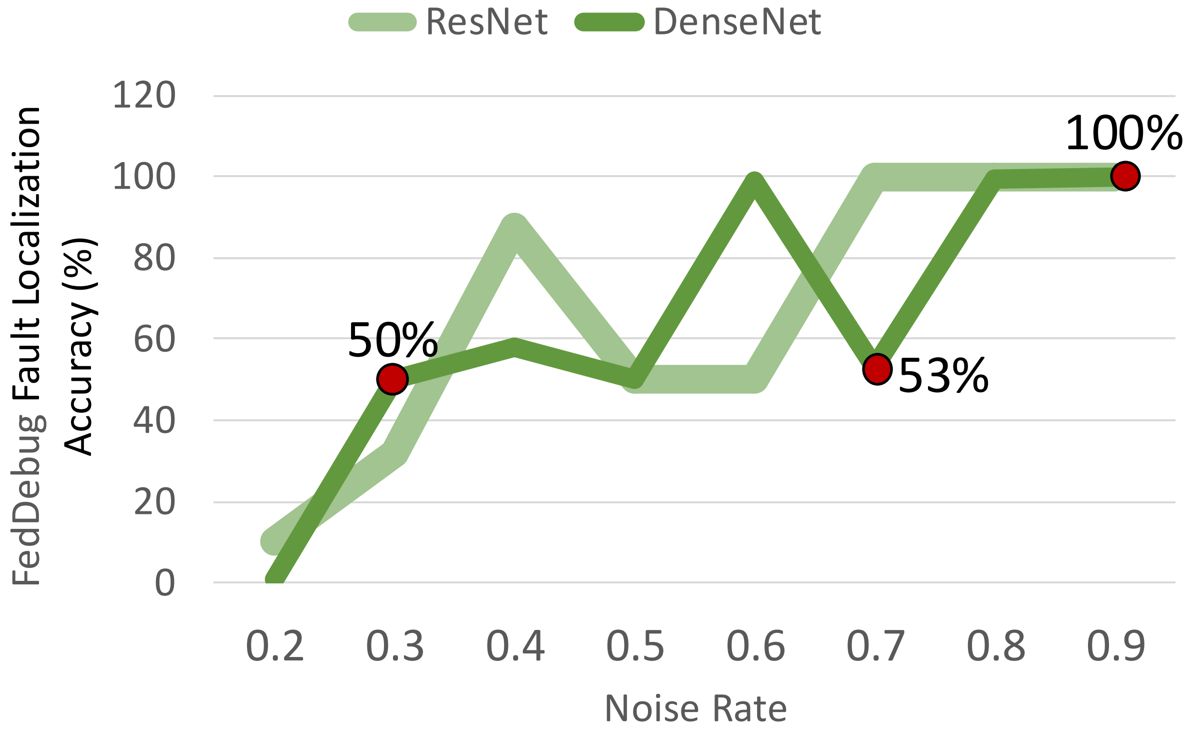
Fault Localization Accuracy

What is a representative noise rate for simulating a faculty client.



Low noise rates up to to 0.7, barely affect the global model performance.

FedDebug's resilience against different degrees of faults



FedDebug effectively localizes faulty clients even with low noise rates.

Localizing Multiple Faulty Clients

- **DenseNet neurons** learns better features compared to ResNet.
- **Dense concatenation** among its layers is the reason behind this advantage.
- Thus, FedDebug performs well when the clients contain DenseNet.

# of Faulty Clients	Total Clients	Architecture	Localization Accuracy (CIFAR)	Localization Accuracy (FEMNIST)
5	30	ResNet	100	98
7	30	ResNet	100	97.1
5	30	DenseNet	100	100
7	30	DenseNet	100	100
5	50	ResNet	54	60
7	50	ResNet	57.1	62.9
5	50	DenseNet	100	100
7	50	DenseNet	100	95.7

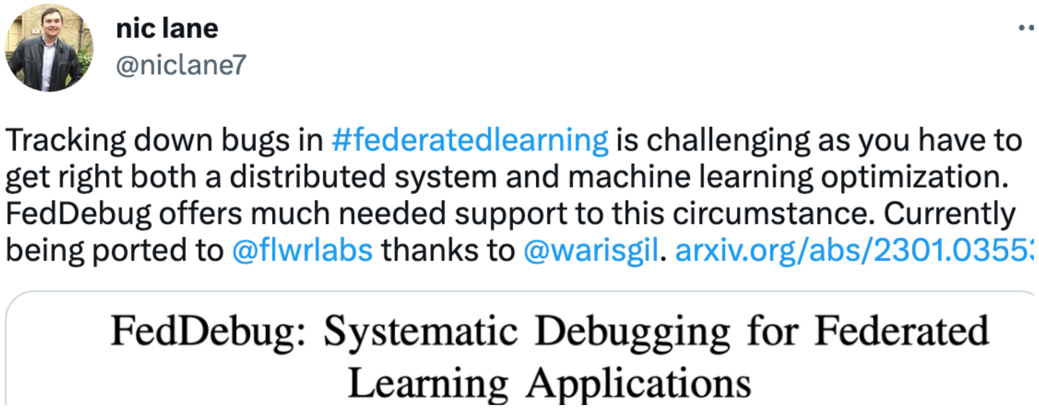
FedDebug identifies multiple faulty clients with an average accuracy of 90%.

Conclusion

FedDebug is the **first open-source debugging and testing framework** for FL applications.

Currently available in **IBM FL Framework**.

Porting to **Flower FL Framework** is in progress.



nic lane
@niclane7

Tracking down bugs in #federatedlearning is challenging as you have to get right both a distributed system and machine learning optimization. FedDebug offers much needed support to this circumstance. Currently being ported to @flwrlabs thanks to @warisgil. arxiv.org/abs/2301.0355

FedDebug: Systematic Debugging for Federated Learning Applications

Complete artifact is available at https://github.com/SEED-VT/FedDebug

Thank you ☺

Functional Reusable Available