FEDERATED LEARNING
MACHINE LEARNING WITH PRIVACY ON THE EDGE

Mike Lee Williams • @mikepqr • mlw@cloudera.com
Federated Learning
• Data must be collected for a specific purpose.
• That purpose must be one to which the user has consented.
• The data must be necessary to achieve that purpose.
• It must be deleted when it is no longer necessary for any purpose.

— The GDPR, paraphrased
Consumers have a right to reasonable limits on the personal data that companies collect and retain. Companies should collect only as much personal data as they need to accomplish [clearly specified purposes]. Companies should securely dispose of or de-identify personal data once they no longer need it, unless they are under a legal obligation to do otherwise.

— The 2012 White House report *Consumer Data Privacy in a Networked World*
Don’t collect it. Don’t store it. Don’t keep it.

— Maciej Ceglowski, *Haunted by Data*, Strata NY 2015
PRACTICALITIES
THE FEDERATED LEARNING SETTING
Pear can help you

- make an album of your child to share easily with grandma
- write better emails
- write better texts

A N A L Y Z I N G

"I've got Hey! I've got the tickets"

"Use ‘I think’ less"

it just needs access to your data...
FEDERATED AVERAGING

TURBOFAN TYCOON

turbofan.fastforwardlabs.com
CHALLENGES
Power consumption
Dropped connections

Stragglers
Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning by Hitaj et al. (2017)
APPLICABILITY
WHEN TO CONSIDER FEDERATED LEARNING

• You care about **privacy** and/or
• You care about **bandwidth** or other system resources

• You have established a performance **benchmark and**
• Your model would be improved by access to **more training data**

• You are doing deep learning
• (Although if you are, check out **PySyft** and **TF-Federated**)
USE CASES

• Predictive maintenance/industrial IOT
• Smartphones
• Healthcare (wearables, drug discovery, prognostics, etc.)
• Browsers
• Retail video analytics
• Enterprise/corporate IT (chat, issue trackers, email, etc.)
Malay Haldar / Machine Learning Engineer @Airbnb
Sravya Tirukkovalur / Senior Machine Learning Engineer @Adobe
Jibin Liu / Software Engineer @Ebay
Alex Beutel / Senior Research Scientist @Google
Eric Tramel / Federated Learning R&D Lead @OWKIN
Mantas Matelis / Software Engineer @Cardiogram
Avesh Singh / Software Engineer & Technical Lead @Cardiogram

Deep Learning in Practice

HOST

Mike Lee Williams
Research Engineer @Cloudera Fast Forward Labs
From Cloudera

- An introduction to Federated Learning (Cloudera VISION blog, business audience)
- Federated learning: distributed machine learning with data locality and privacy (CFFL blog, more technical)
- Turbofan Tycoon (working prototype, see FFL blog post for some details)

Other blog posts

- Collaborative Machine Learning without Centralized Training Data (Google research blog)
- Federated Learning for Firefox (Firefox on florian.github.io)
- Federated Learning for wake word detection (snips.ai on medium.com)

Papers

- Communication-Efficient Learning of Deep Networks from Decentralized Data by McMahan et al. (Google, 2016)
- Practical Secure Aggregation for Privacy-Preserving Machine Learning by Bonawitz et al. (Google, 2017)
- Federated Multi-Task Learning by Smith et al. (2017)
- A generic framework for privacy preserving deep learning by Ryffel et al. (PySyft, 2018)
- Federated Learning for Mobile Keyboard Prediction by Hard et al. (Google, 2018)
- Towards Federated Learning at Scale: System Design by Bonawitz et al. (Google, 2019)
```python
def round(self):
    """
    Do a round of federated learning:
    - instruct each node to train and return its model
    - replace the server model with the weighted average of the node models
    - replace the node models with the new server model
    """
    updates = [node.train() for node in self.nodes]
    self.fedavg([u for u, node in zip(updates, self.nodes)])
    self.push_model(node for node in self.nodes)
```
def fedavg(self, updates):
    
    Replace the server model with the weighted average of the node models.
    
    `updates` is a list of dictionaries, one for each node, each of which has `state_dict` (the weights on that node) and `n_samples` (the amount of training data on that node).
    
    ```python
    N = sum(u["n_samples"] for u in updates)
    for key, value in self.model.state_dict().items():
        weight_sum = (u["state_dict"][key] * u["n_samples"] for u in updates)
        value[:] = sum(weight_sum) / N
    ```