Federated Deep Learning for Healthcare Data

Erik Babu

23 June 2020

Supervisor: Prof. Daniel Rueckert

Motivation

Motivation

- Study was conducted on the accuracy of cancer diagnoses in 2015
- 16 observers had to decide whether images of breast tissue were cancerous
- Observers achieved 99% classification accuracy (majority voting)

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0141357

Motivation

- The identity of the observers?

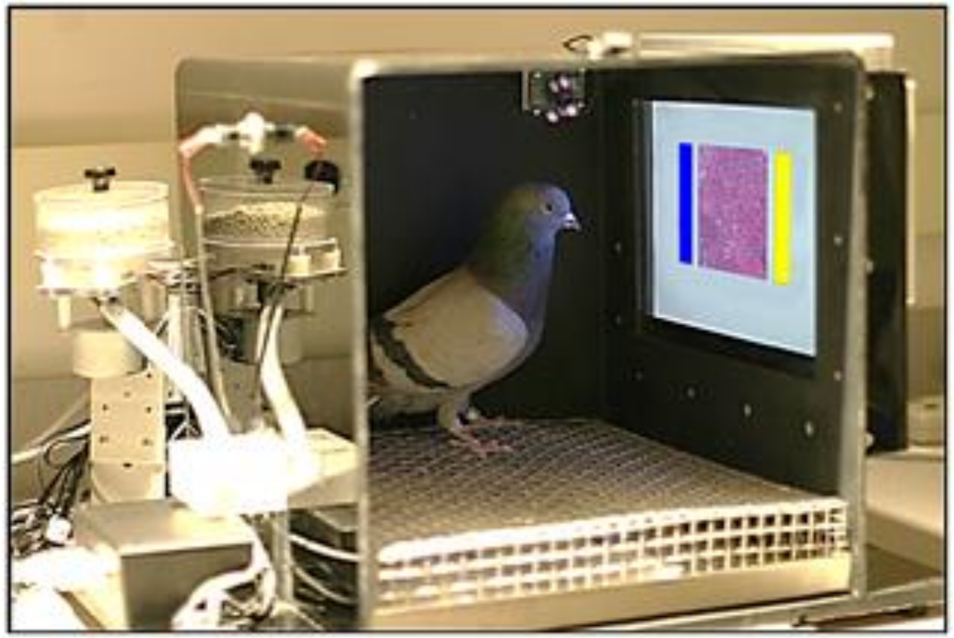
https://www.insidermonkey.com/blog/11-highest-paying-states-for-radiologists-474636/

Motivation

› The identity of the observers?

Motivation

› The identity of the observers?



Identifying patterns in medical data is not a uniquely human skill!

https://www.roblox.com/library/126796917/pigeon-head

Motivation

Identifying patterns in medical data is not a uniquely human skill!

› DL algorithms excel at uncovering patterns from data

› They have the potential to address medical diagnosis problems

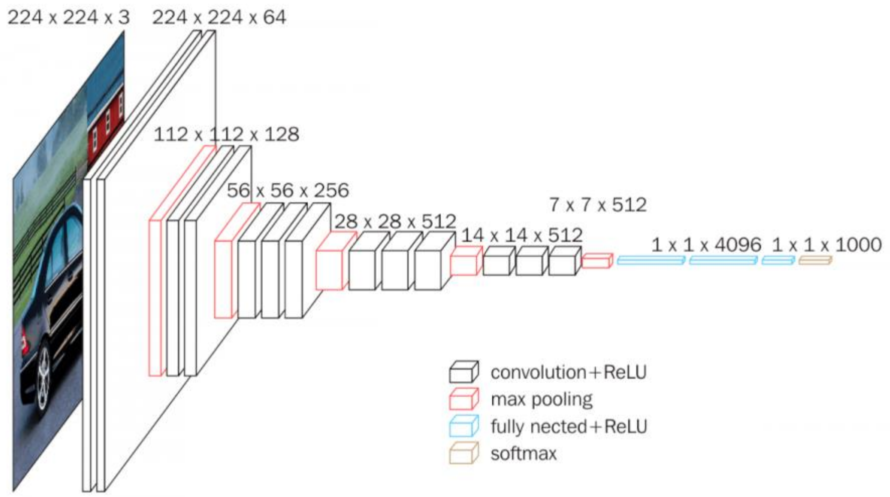
- But they require **large training sets** to achieve **high performance**

https://medcitynews.com/2020/01/googles-ai-beats-humans-at-detecting-breast-cancer-sometimes/

Motivation

SOTA DL classification models are data-hungry

- E.g. VGG16
- More than 138M trainable parameters
- All parameters tuned for model to converge to a solution
- Insufficient training data leads to poor approximation

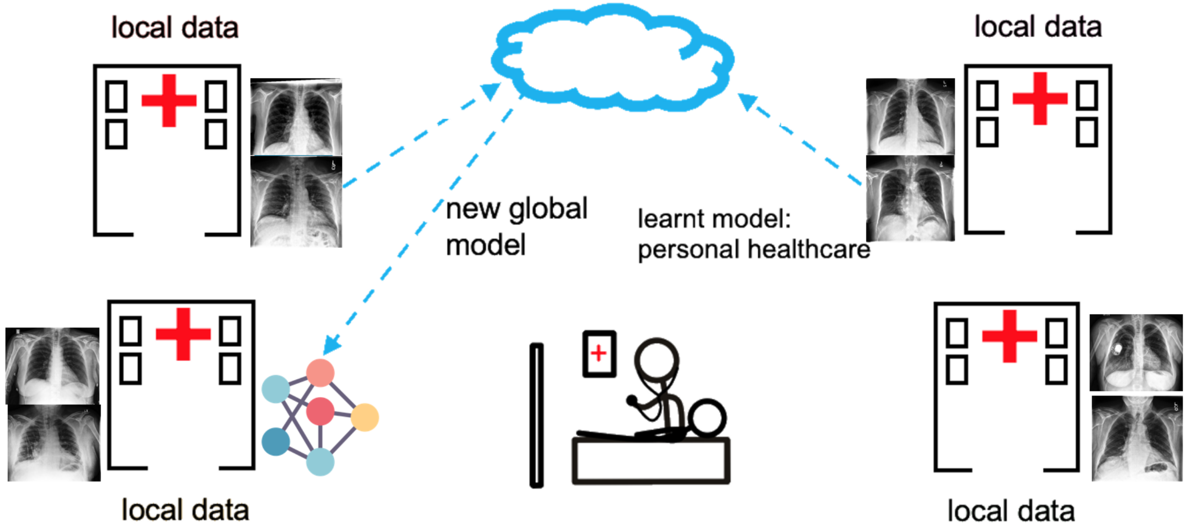


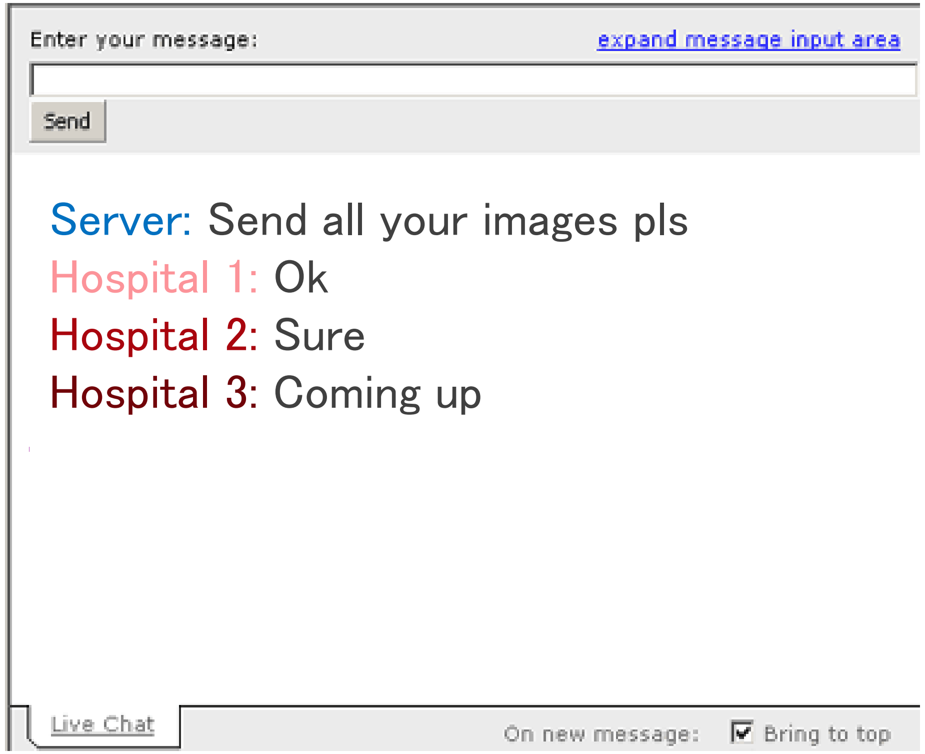
VGG16

https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96
http://www.image-net.org/

Possible Solution

Aggregate the raw data centrally

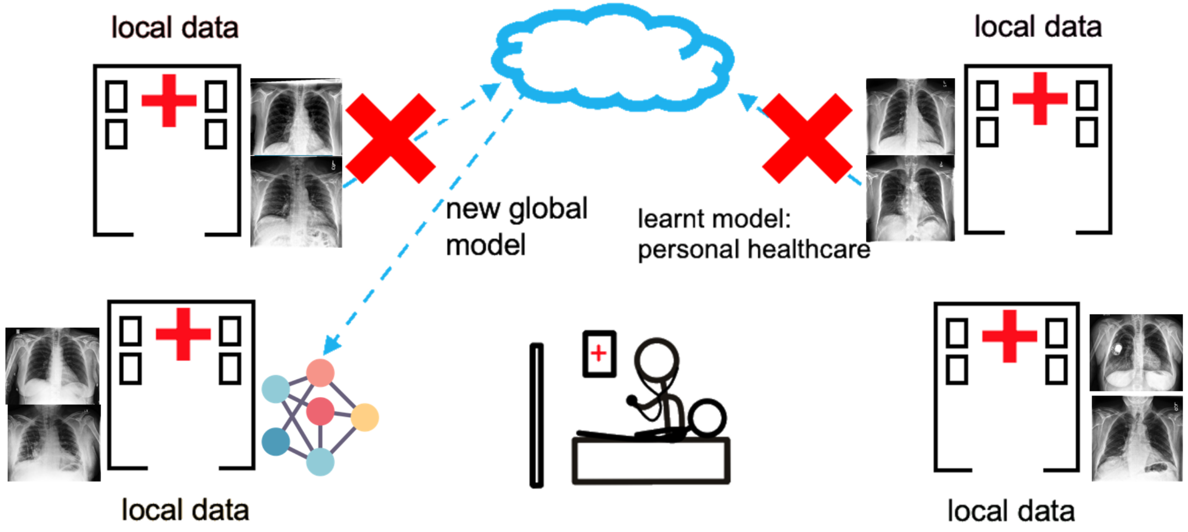




https://recollective.com/help/Advantage/tools/chat/chat_permanent.htm

Impossible Solution

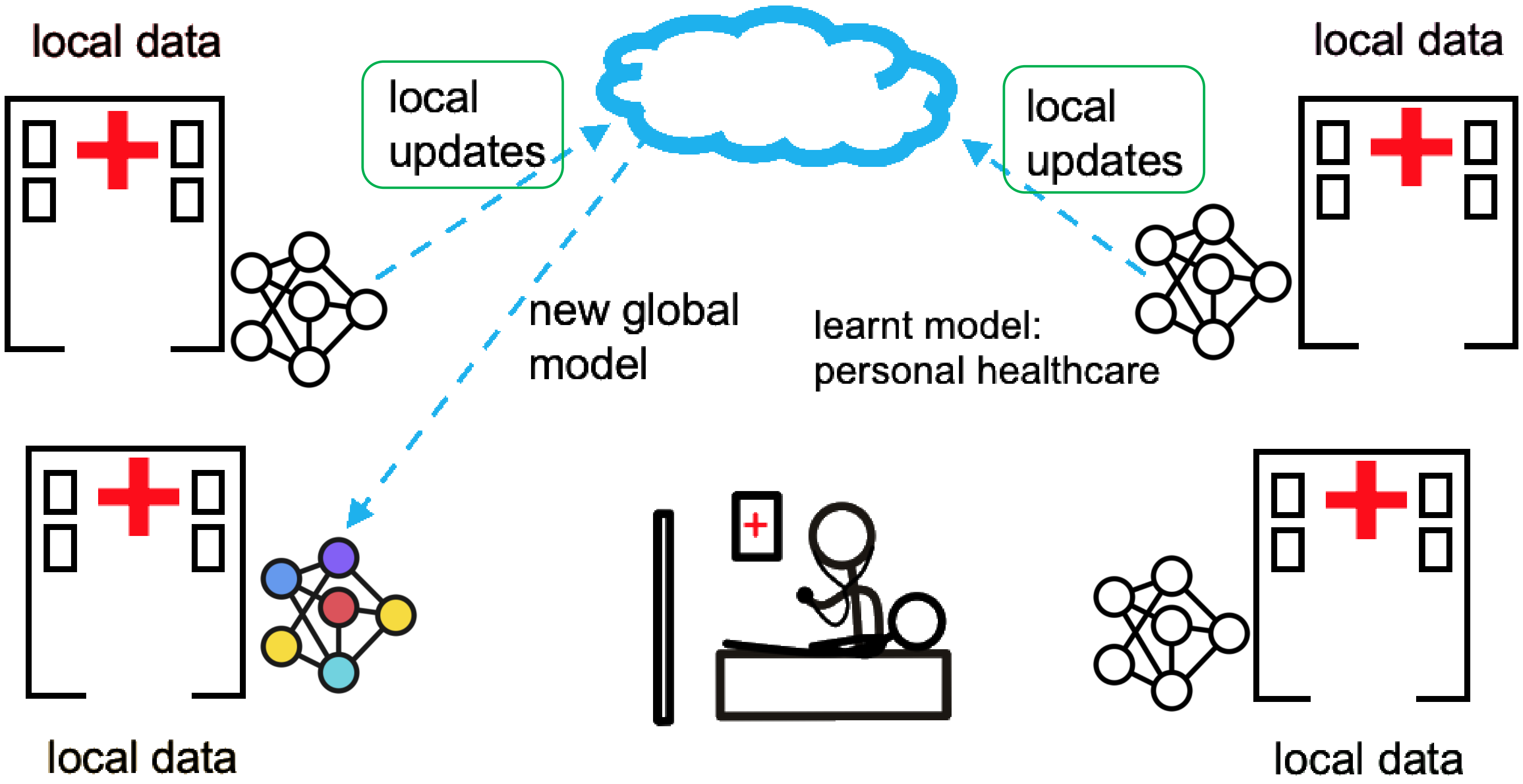
Aggregate the raw data centrally



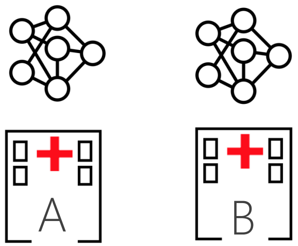
https://gdpr-info.eu/
https://www.cdc.gov/phlp/publications/topic/hipaa.html

Our Solution

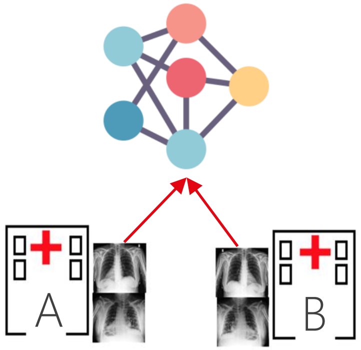
Federated Learning (FL)



Terminology



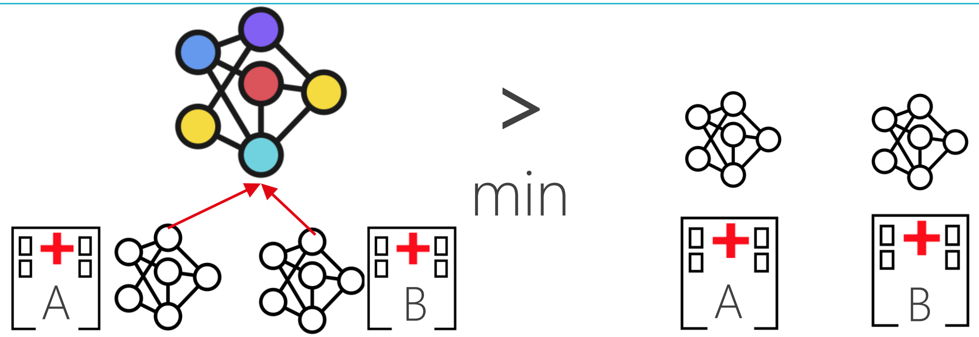
- Institutional/Baseline models
- Expected **lower bound**



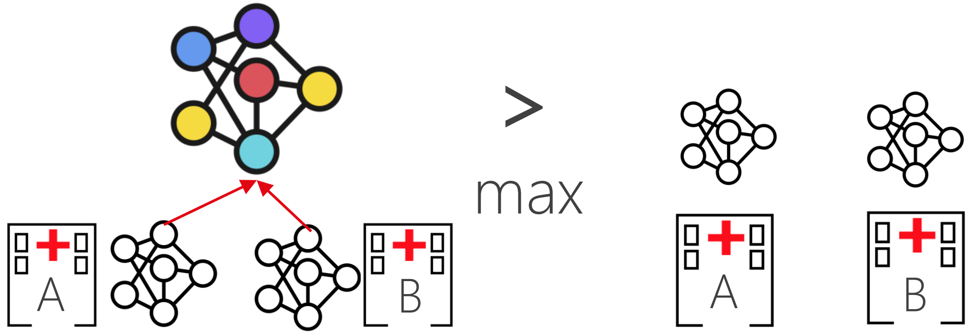
- Centralised/Benchmark model
- Expected **upper bound**

Investigation

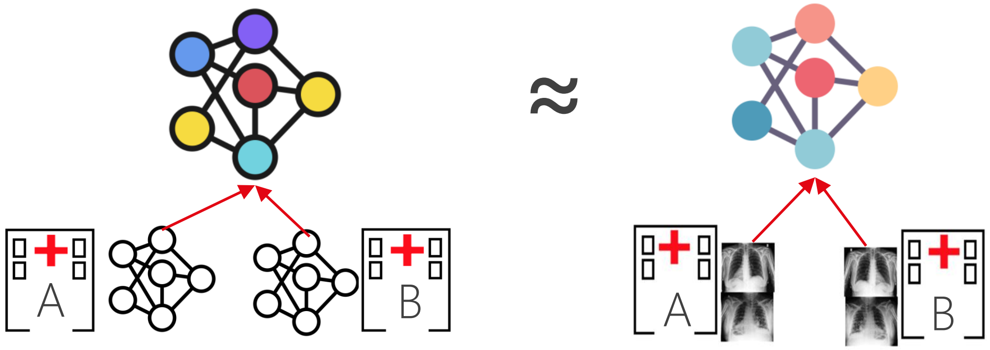
› Can the FL models outperform **any** baseline models?



› Can the FL models outperform **all** baseline models?

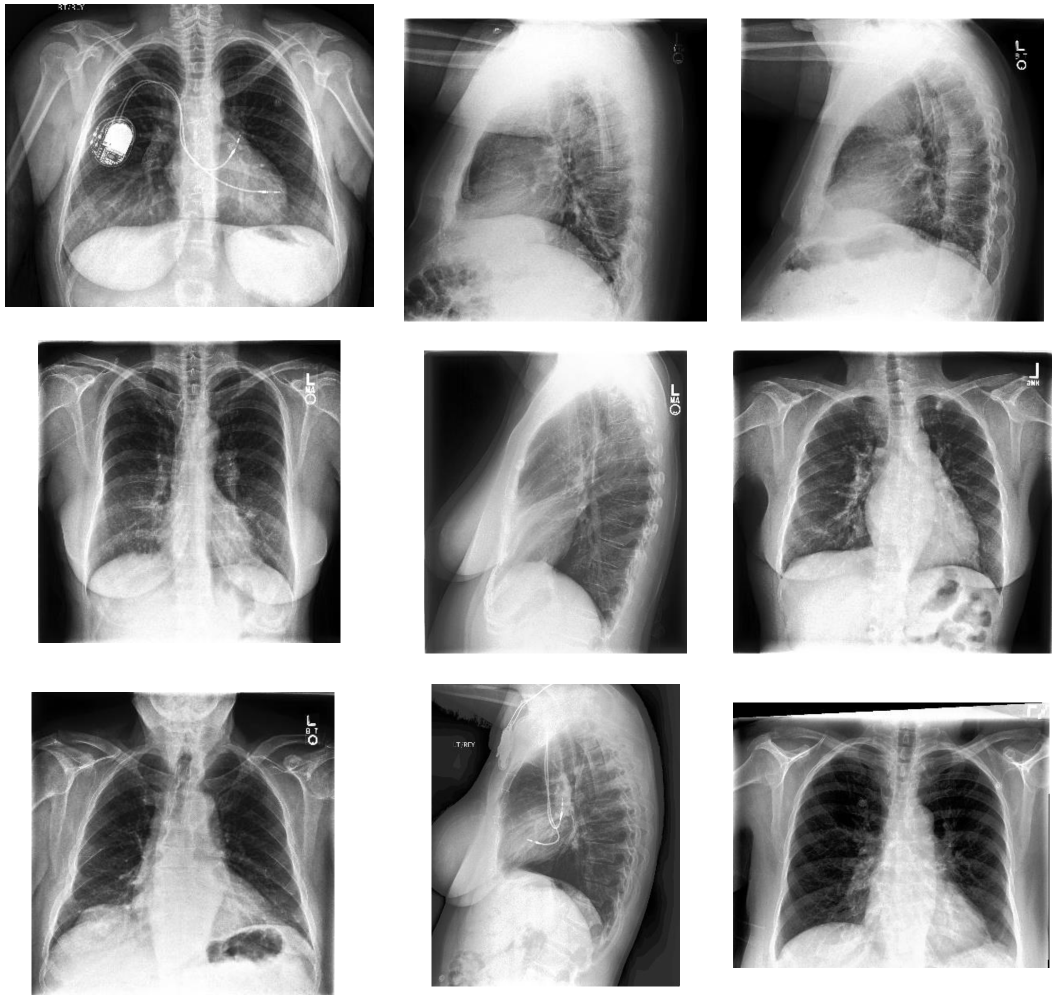


› Can the FL models perform **similarly** to the benchmark model?



Dataset

- CheXpert (downsampled)
- 224,316 multi-view radiographs
- 14 typical chest observations (focus on 5)
- Uses ROC-AUC as evaluation metric
- Report Section 3.2

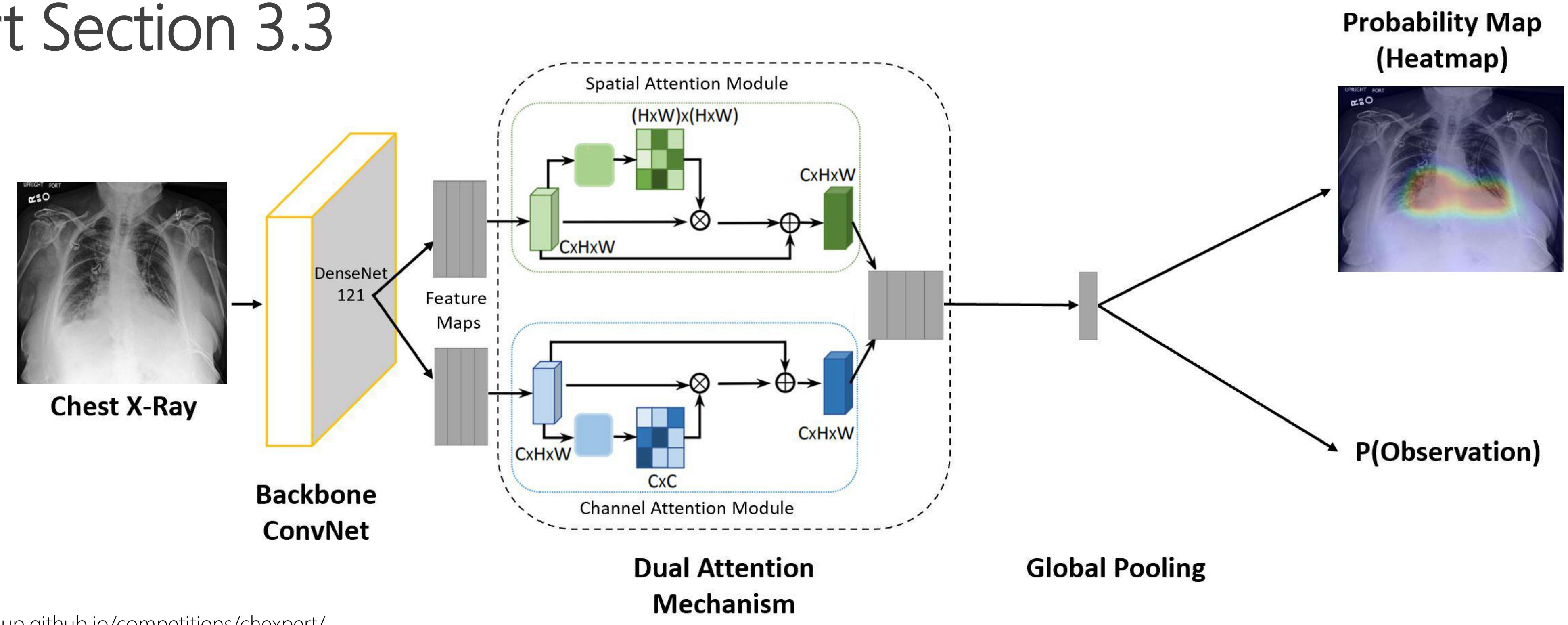


https://arxiv.org/abs/1901.07031

Implementation

Benchmark

- Off-the-shelf implementation (4th highest on CheXpert leaderboard)
- Trained on full dataset -> **0.901** mean AUC
- Report Section 3.3



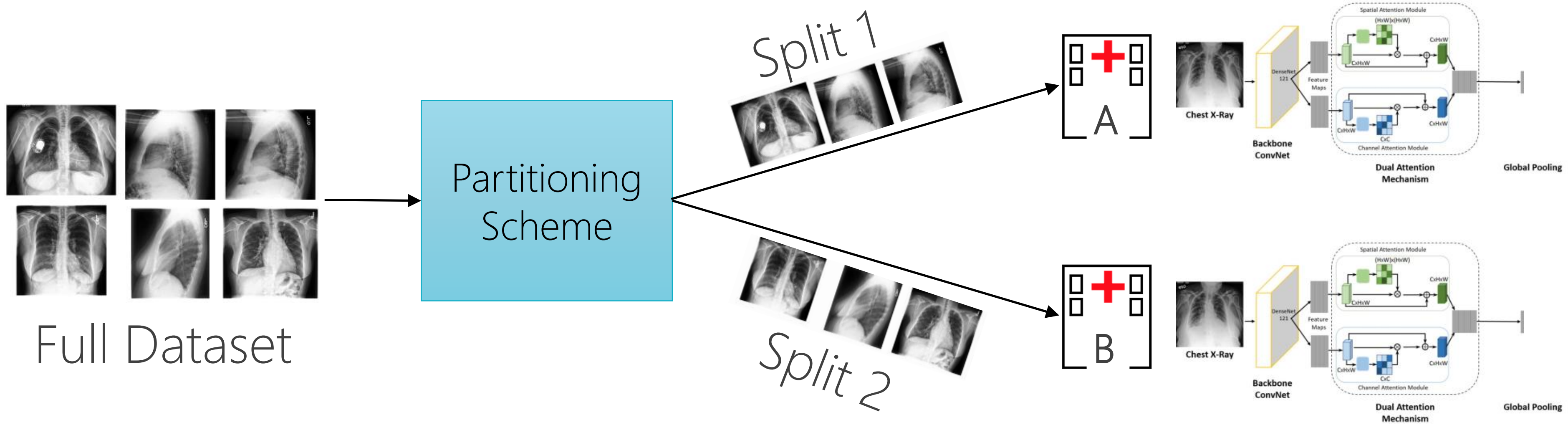
https://stanfordmlgroup.github.io/competitions/chexpert/
https://github.com/jfhealthcare/Chexpert

Baseline

› Same underlying model as benchmark

› Trained on partition of dataset

› No "collaboration" between institutions



Baseline

Data Partitioning Scheme

- Equal & unequal amounts between 2 institutions (50/50 & 75/25)
- Equal amounts between 5 and 10 institutions (stretch goal)
- Non-i.i.d. partitioning:
  - Iteratively sample from full dataset without replacement
  - Assign different weights to the observations to skew label distribution
- Report Appendix A

FL

- Various FL frameworks considered (Report Section 4.1)
- Initially proceeded with PySyft

✓ Similar API to PyTorch

✓ Multiple available tutorials

✓ Abstractions to help with implementing ε-DP and SMPC

https://github.com/OpenMined/PySyft

FL

- Eventually abandoned PySyft
  - ❌ Tutorials not always up-to-date with API
  - ❌ Cannot assign custom / uneven data partitions to clients
  - ❌ Momentum not yet implemented
- No FL framework used

https://github.com/OpenMined/PySyft

- Manually implement training loops
  - Wrap main training procedure in external loops

```
for e in range(num_epochs):
  train(model, data)
```

FL

- Inner-loop -> local training at an institution
- First outer-loop -> local training at all institutions

```
for client in clients:
    model = initialise_model(global_weights)
    data = client.data

    for e in range(num_epochs):
      train(model, data)

    updates.append(model.weights)
```

- Outer-loop -> communication rounds (server-aggregation)

```
for cr in range(num_comm_rounds):
    updates = []

    for client in clients:
        model = initialise_model(global_weights)
        data = client.data

        for e in range(num_epochs):
          train(model, data)

        updates.append(model.weights)

    global_weights = server.aggregate(updates)
```

FL

- Manually implement training loops (Report Section 4.3)
  - Wrap main training procedure in external loops
  - Inner-loop: local rounds of training at all institutions
  - Outer-loops: communication rounds (server-aggregation)
- Manually implement aggregation code
- Institutions use data partitions from baseline approach

Approach #1: FedAvg (server-side)

- Equally weight updates according to number of participants

$$w_{t+1} \leftarrow \boxed{\frac{1}{K}} \sum_{k=1}^{K} w_{t+1}^{k}$$

- Weight updates according to proportion of data contributed by each participant

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \boxed{\frac{n_k}{n}} w_{t+1}^{k}$$

https://arxiv.org/abs/1602.05629

Approach #1: FedAvg (server-side)

$$w_{t+1} \leftarrow \frac{1}{K} \sum_{k=1}^{K} w_{t+1}^{k}$$

```
def fed_avg(weights):
    n_clients = len(weights)
    factor = float(1 / n_clients)

    # Perform averaging
    avg = weights[0]
    for k in avg.keys():      # iterate through all weight parameters
        for i in range(1, n_clients):
            avg[k] += weights[i][k]
        avg[k] = torch.mul(avg[k], factor)

    # Averaged weights to be sent back to participants
    return avg
```

https://arxiv.org/abs/1602.05629

Approach #2: FedProx (client-side)

- Used in conjunction with FedAvg on server-side
- Convergence guarantees in presence of non-i.i.d. data

$$\text{Loss} = \text{Loss} + \frac{\mu}{2}||w - w_k||^2$$

- Adds regularization term to client training loss, based on difference between global and local weights
- μ hyperparameter determines aggressiveness of regularisation

https://arxiv.org/abs/1812.06127

Approach #2: FedProx (client-side)

$$\text{Loss} = \text{Loss} + \frac{\mu}{2}||w - w_k||^2$$

```
# Get loss as usual
loss = get_loss(output, target)

# Compute FedProx regularisation term and update loss value
reg = 0.0
for param_index, param in enumerate(model.parameters()):
    reg += ((mu / 2) * torch.norm((param - global_weights[param_index]))**2)

loss += reg

# Compute gradient of loss as usual
loss.backward()
```

https://arxiv.org/abs/1812.06127

Measuring Overhead

Why track communication and computational overhead?

- FL is a distributed learning problem
- Existing research tends to focus solely on model performance
- Serves as baseline for future optimisations to be evaluated against

Measuring Overhead

Computational Overhead

- THOP: Pytorch-OpCounter library
- Institutions in centralised approach incur 0 computational overhead
- FLOPs = flops_in_batch * batches_in_epoch * total_epochs
- fl_total_epochs = local_epochs * comm_rounds

https://pypi.org/project/thop/

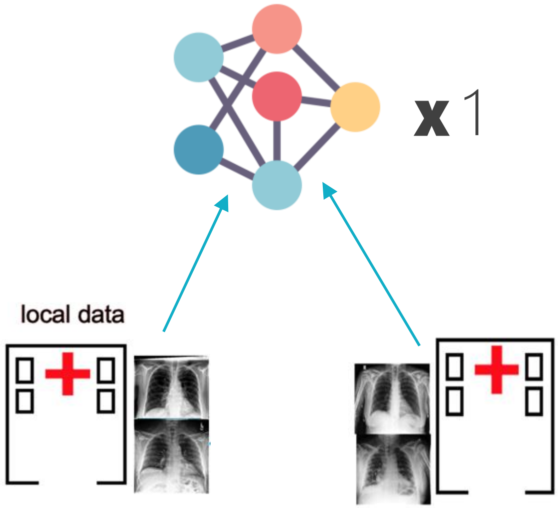
Measuring Overhead

Communication Overhead

- Implemented using our own modelling assumptions
- Institutions in baseline approach incur 0 communication overhead
- `comm_overhead = total_bytes_uploaded + total_bytes_downloaded`

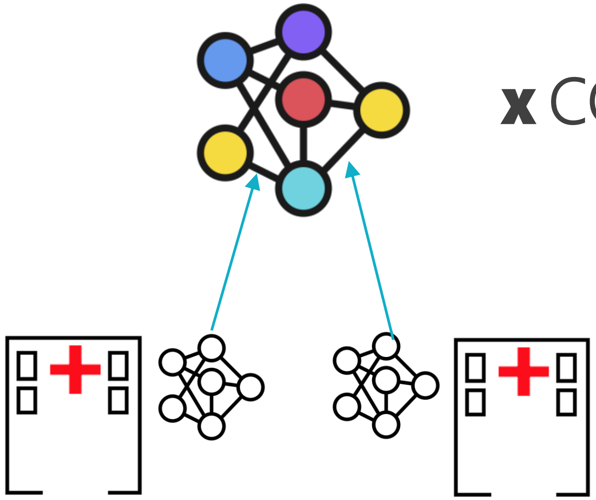
Measuring Overhead

Communication Overhead (Institution Upload)



x 1

Centralised Approach

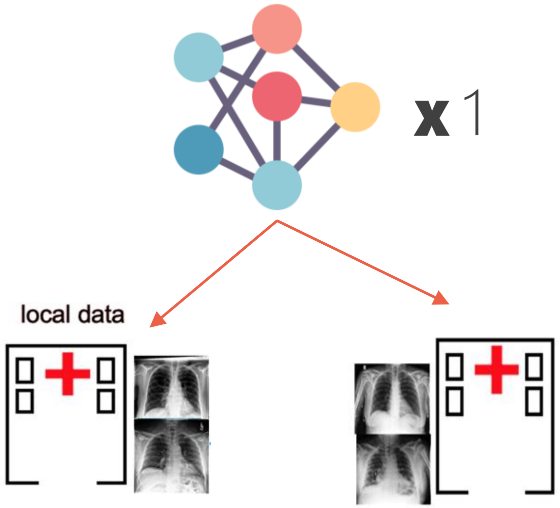


x COMM_ROUNDS

FL Approach

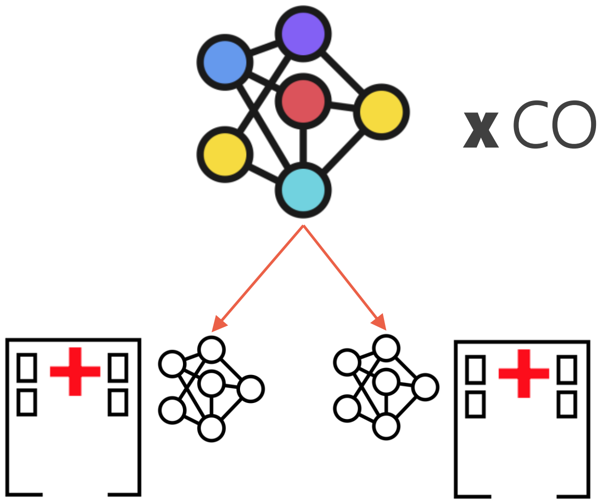
Measuring Overhead

Communication Overhead (Institution Download)



x 1

Centralised Approach

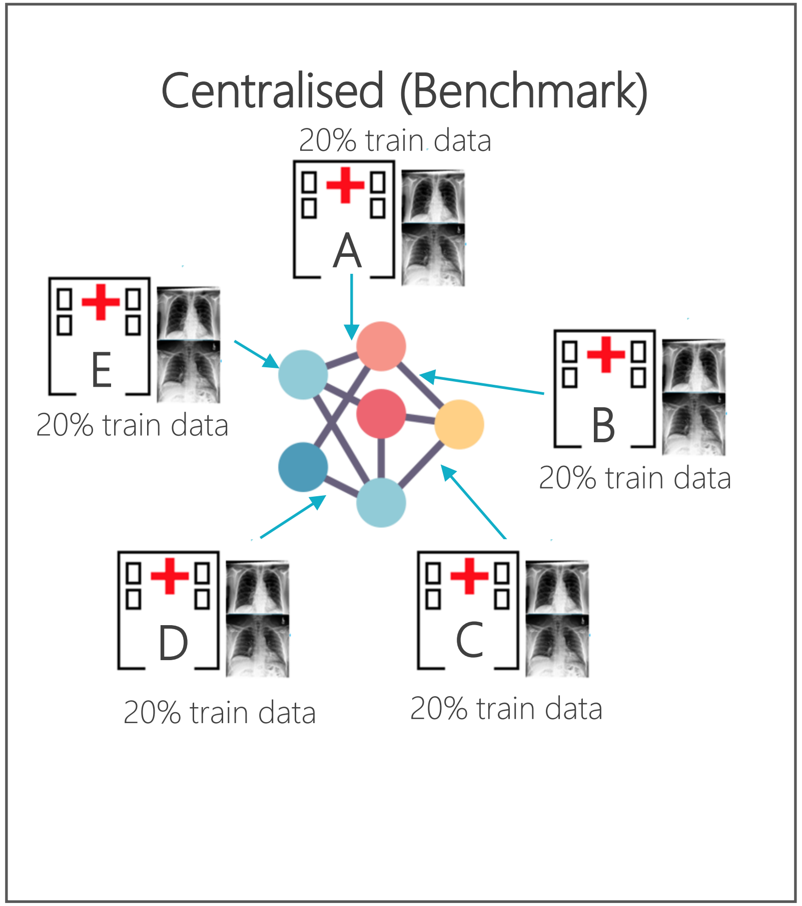


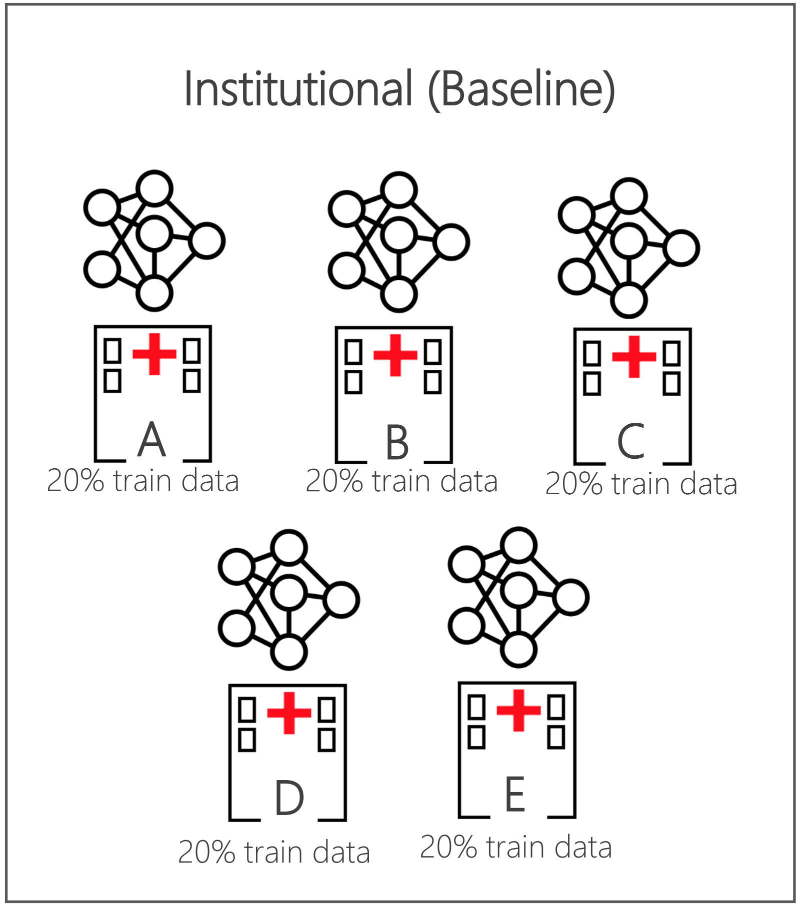
x COMM_ROUNDS + 1

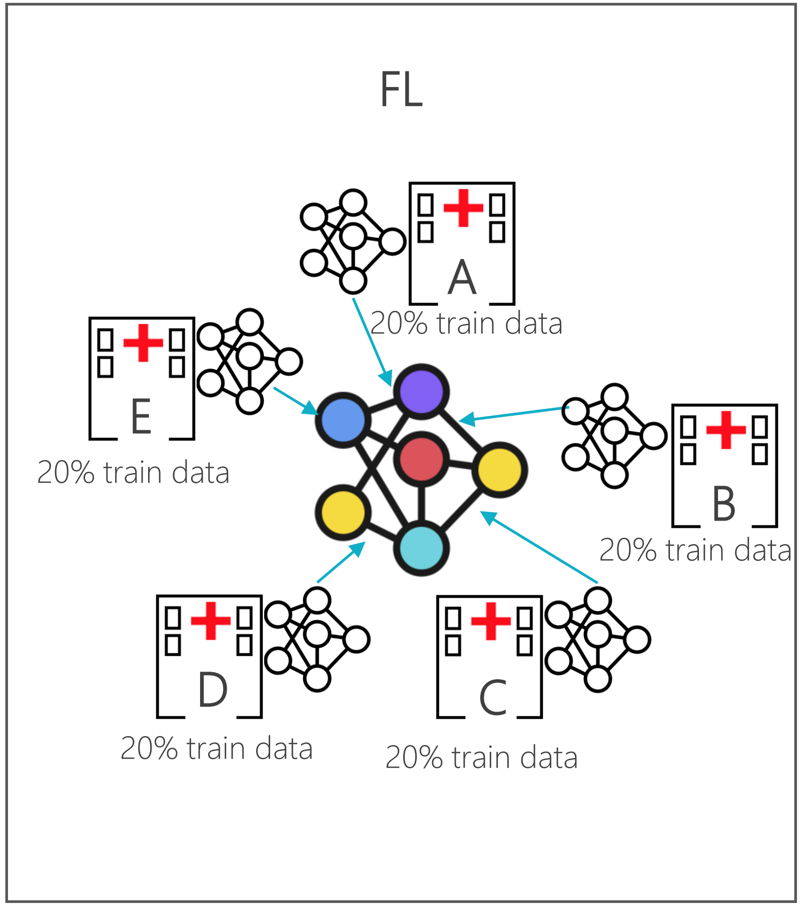
FL Approach

Evaluation

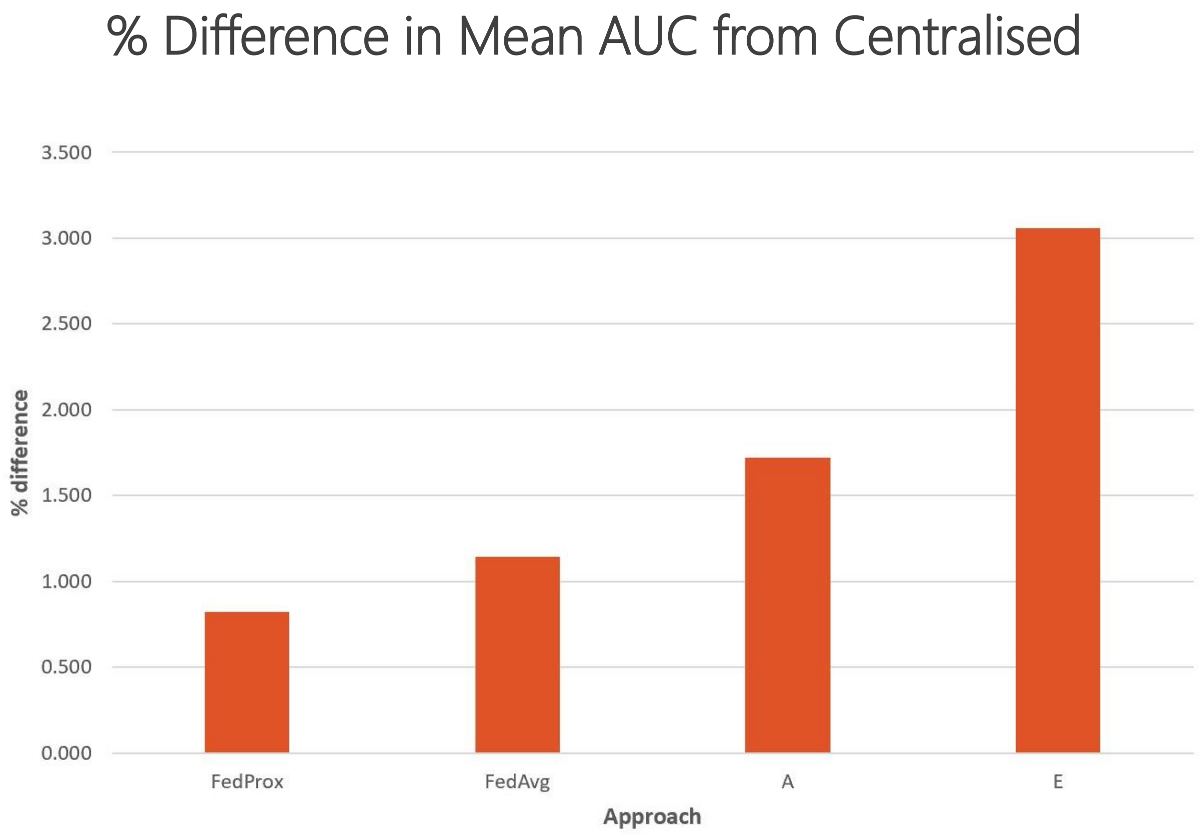
Five Institutions







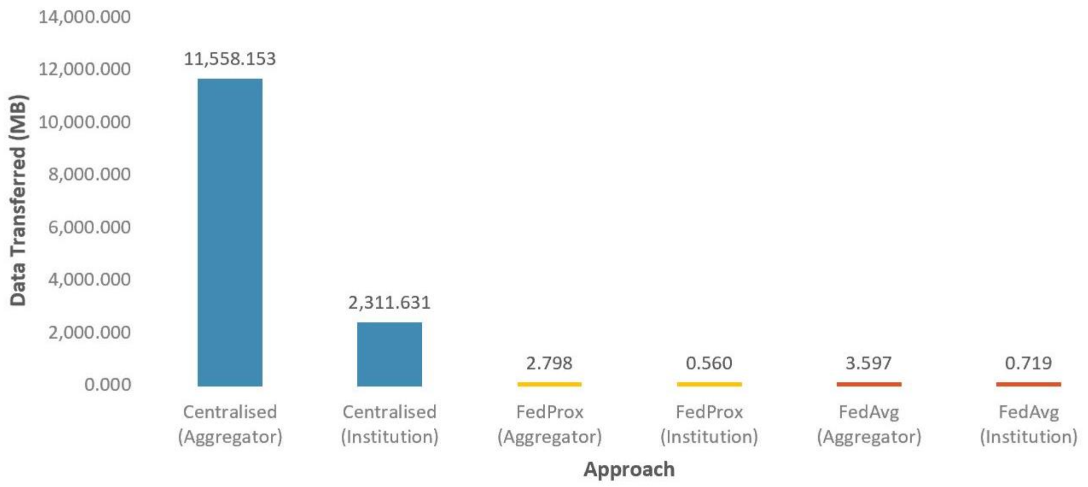
Five Institutions



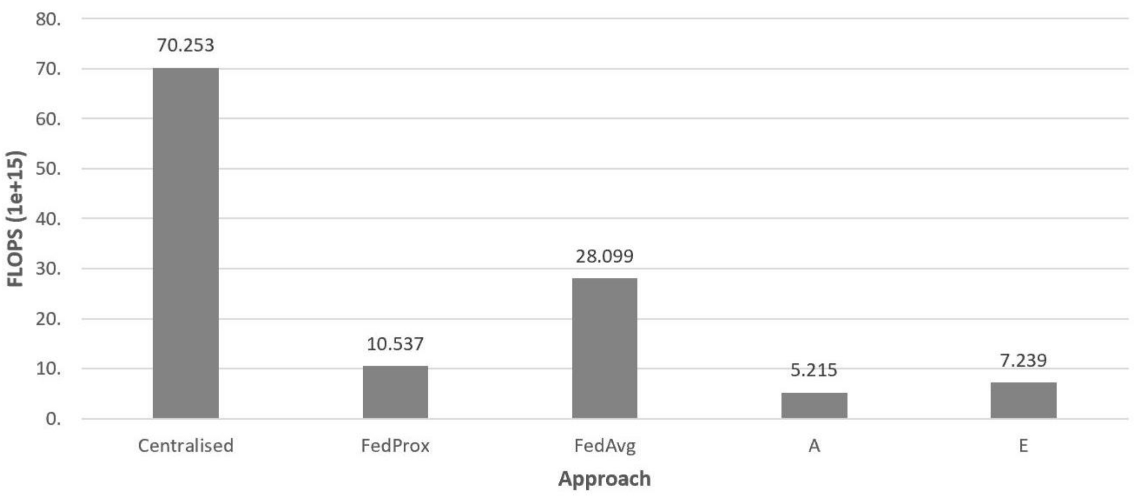
Model Performance

Five Institutions

Communication Overhead



Computational Overhead



Training Overhead

Ten Institutions

- Shows similar trends to five institution split:
  - Avg baseline model performance decreases (6.7% off benchmark)
  - FL model performance remains similar (1.4% off benchmark)
  - FedProx outperforms FedAvg (model performance & overhead)
- Report Section 5

Summary

Summary

- Implement 2 different FL training techniques (+ variant)
- Develop method to track systems overhead
- Compare model & systems performance between centralised, institutional, and FL approaches on 2 / 5 / 10 institutions
- Best FL model achieves similar performance to benchmark (SOTA) in all experiments

THANKS FOR LISTENING

Questions?