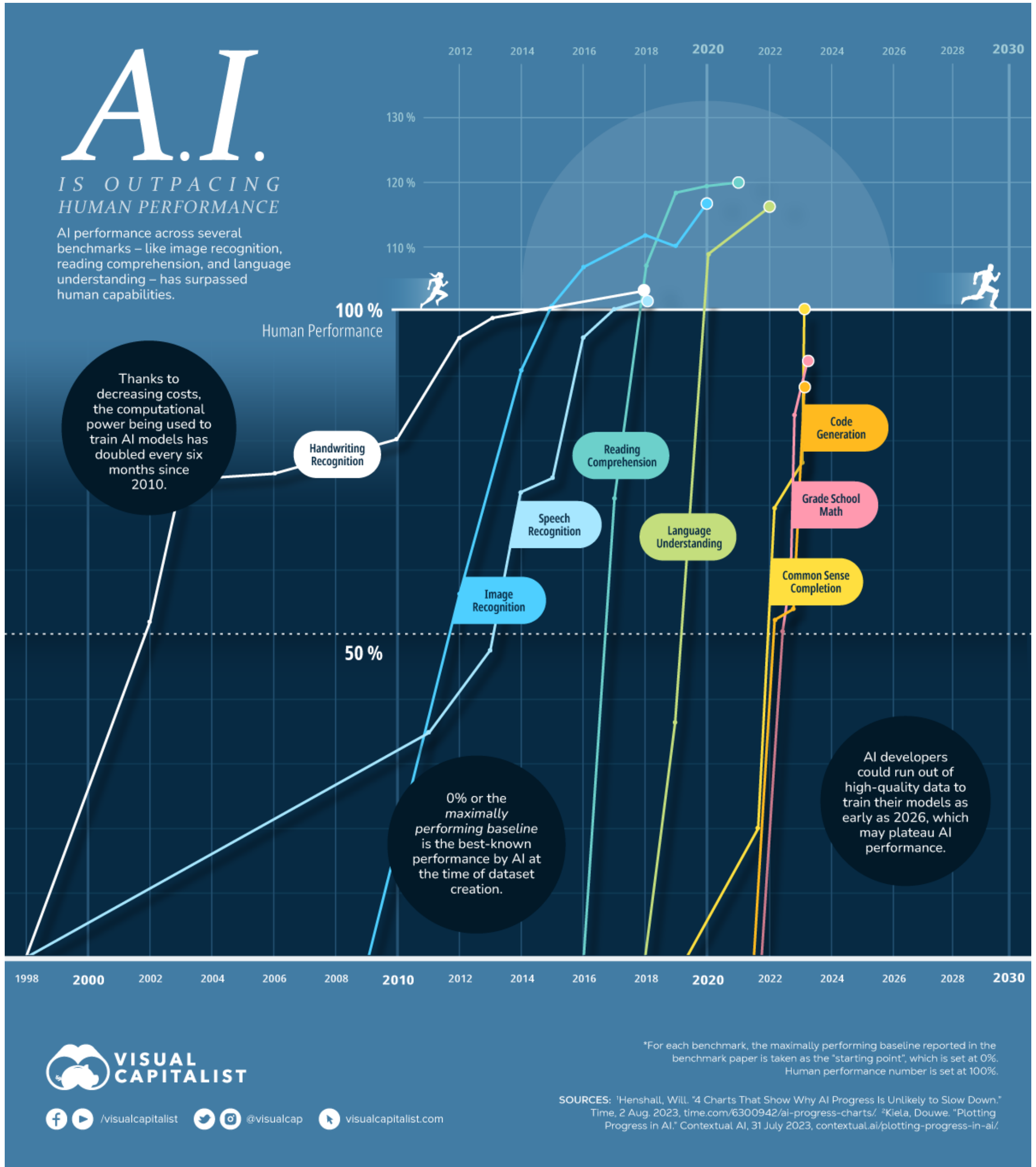
Privacy and Security

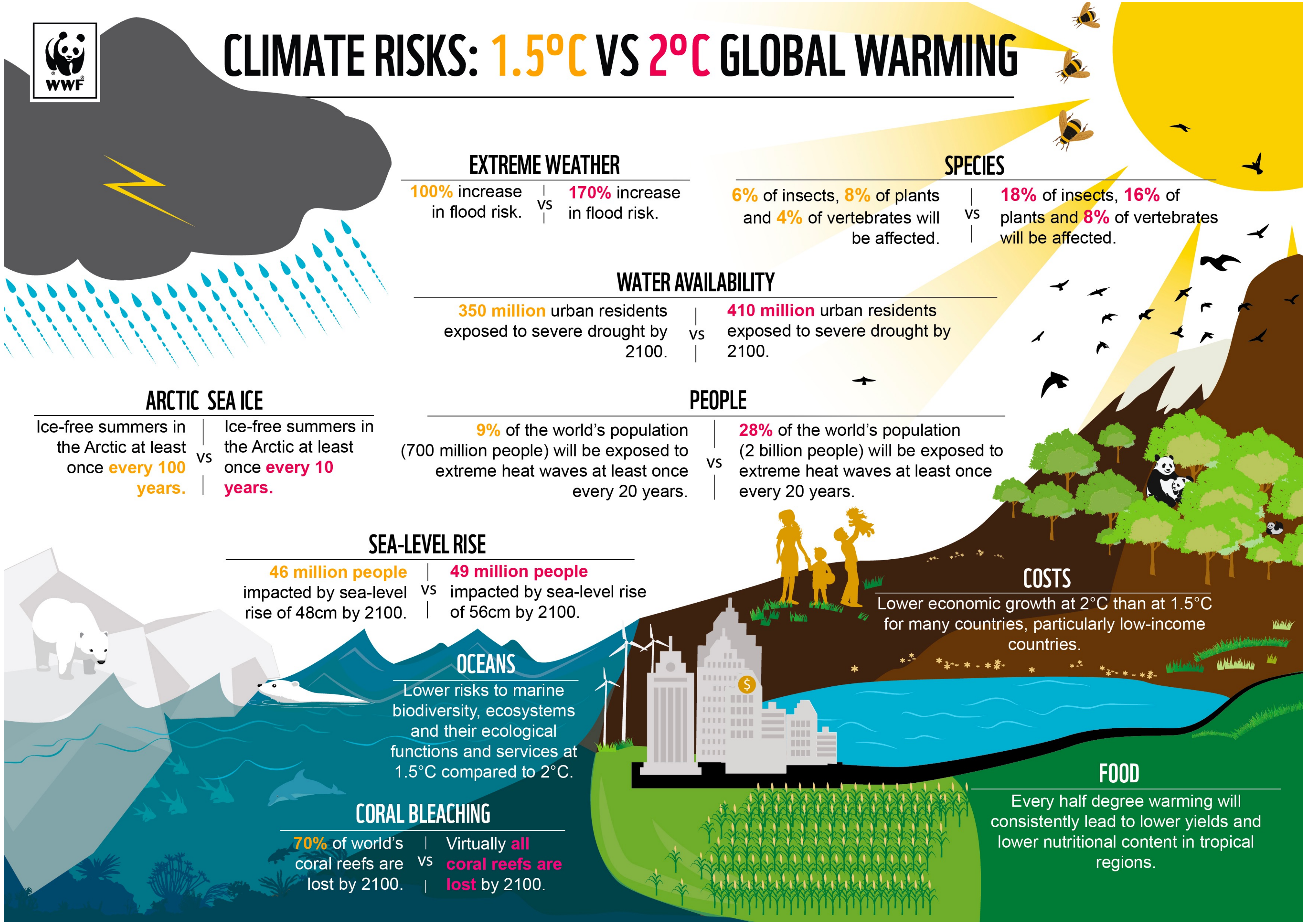
ICS 491

Nice data visualizations from last time

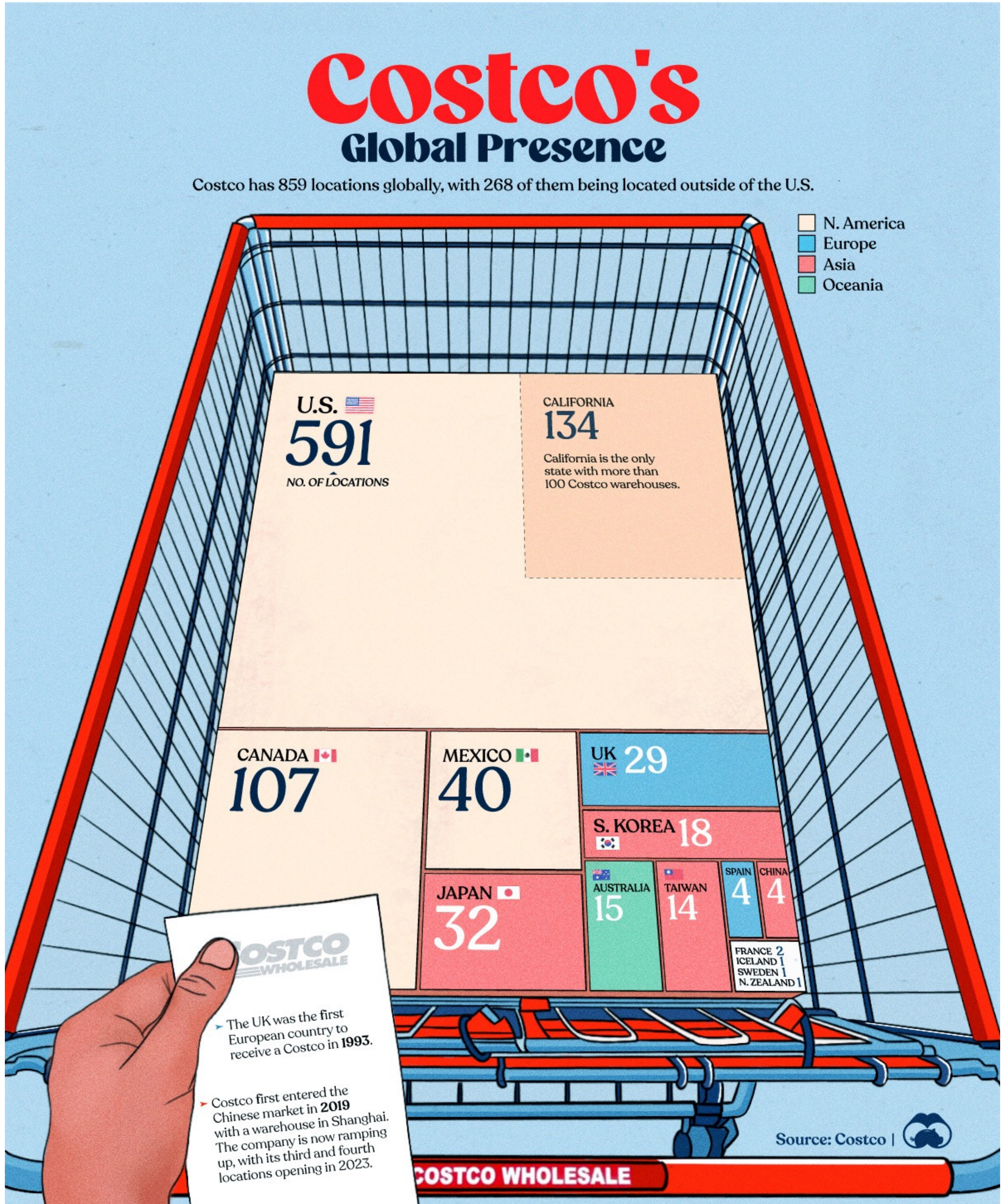


Nice data visualizations from last time

Nice data visualizations from last time



Nice data visualizations from last time



Updated Course Schedule

Tue Nov 14	Digital Therapeutics	
Thu Nov 16	Natural Language Processing	Coding Notebook #3: Machine Learning Studies
Tue Nov 21	Social Network Analysis (Class on Zoom)	
Thu Nov 23	Thanksgiving Holiday (No Class)	
Tue Nov 28	Watch final project videos (No Class)	Project Milestone #6: Final Presentation
Thu Nov 30	Watch final project videos (No Class)	
Tue Dec 5	Multimedia Analytics	
Thu Dec 7	Course Overview	
Fri Dec 15		Final Project Infographic and Code

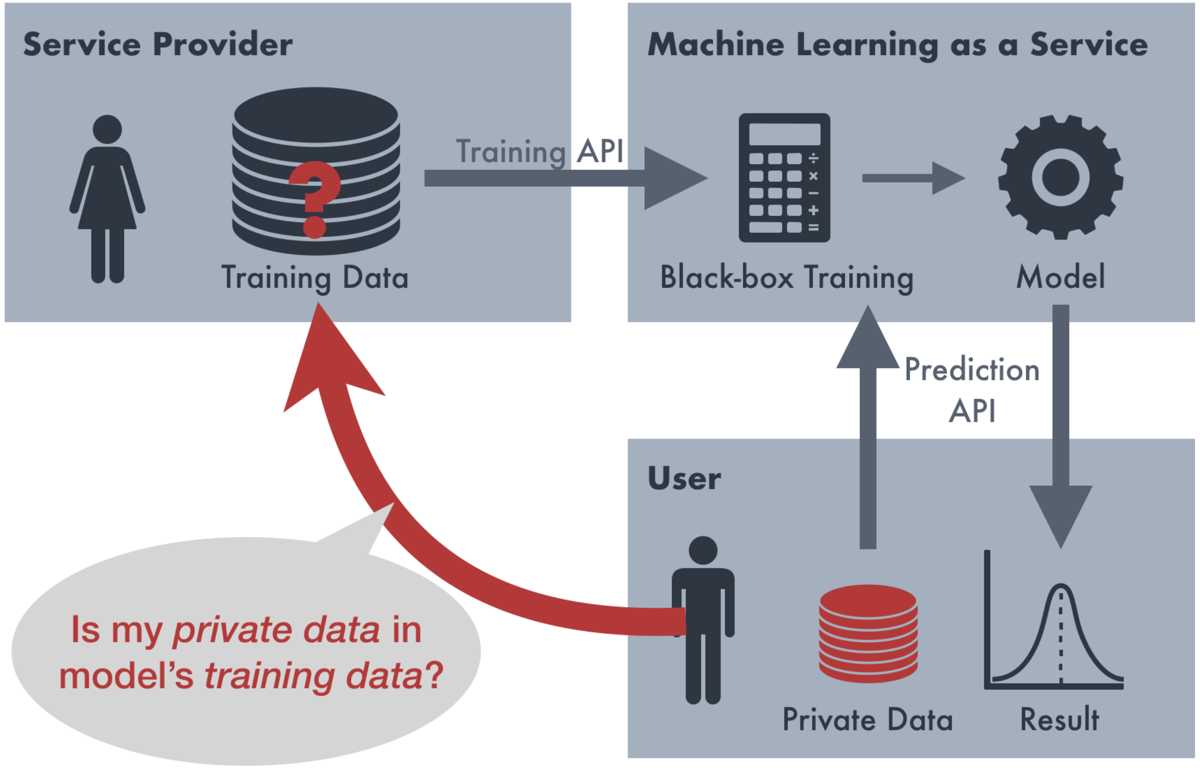
Final Project Infographic and Code

https://docs.google.com/document/d/1_L0Yszy7XKC4b9vtZx7sgT4LgyViUwIysr1Oxmo1KWs/edit?usp=sharing

Privacy <-> Accuracy Tradeoff

“hacking” ML systems

Membership Inference Attack



Hisamoto, Sorami, Matt Post, and Kevin Duh. "Membership inference attacks on sequence-to-sequence models: Is my data in your machine translation system?." *Transactions of the Association for Computational Linguistics* 8 (2020): 49-63.

“hacking” ML systems

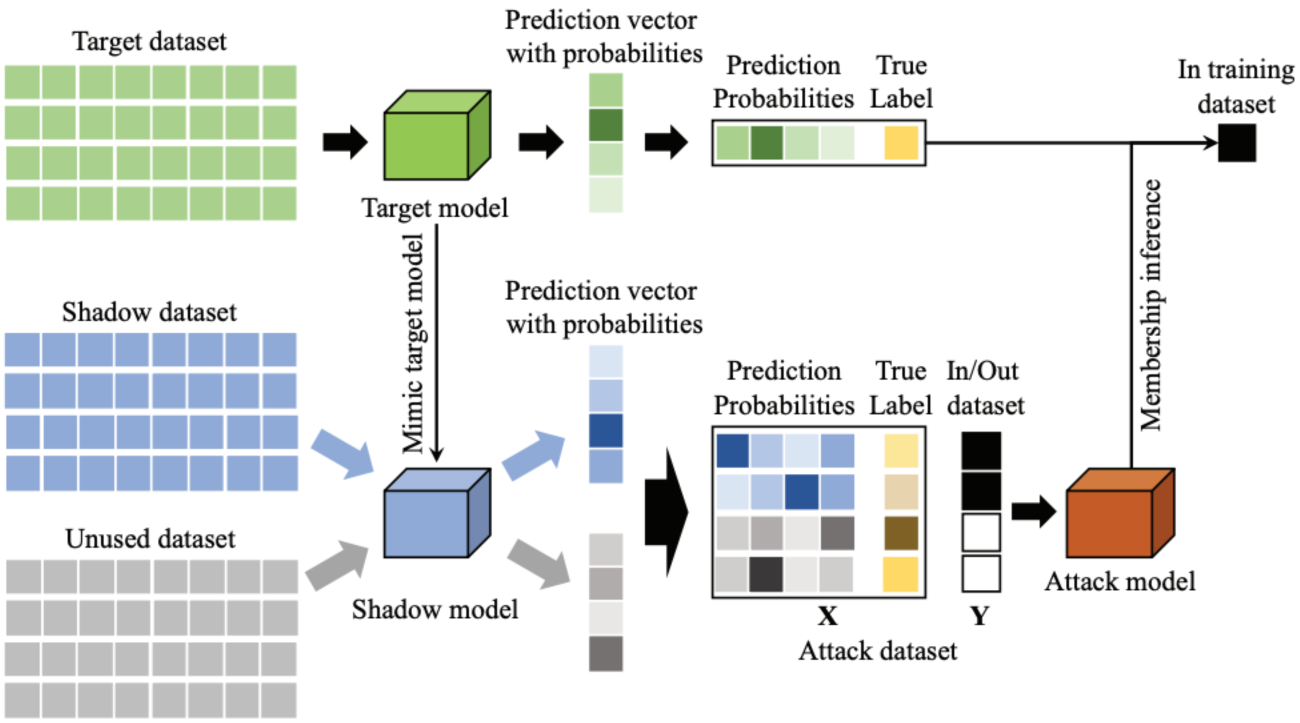
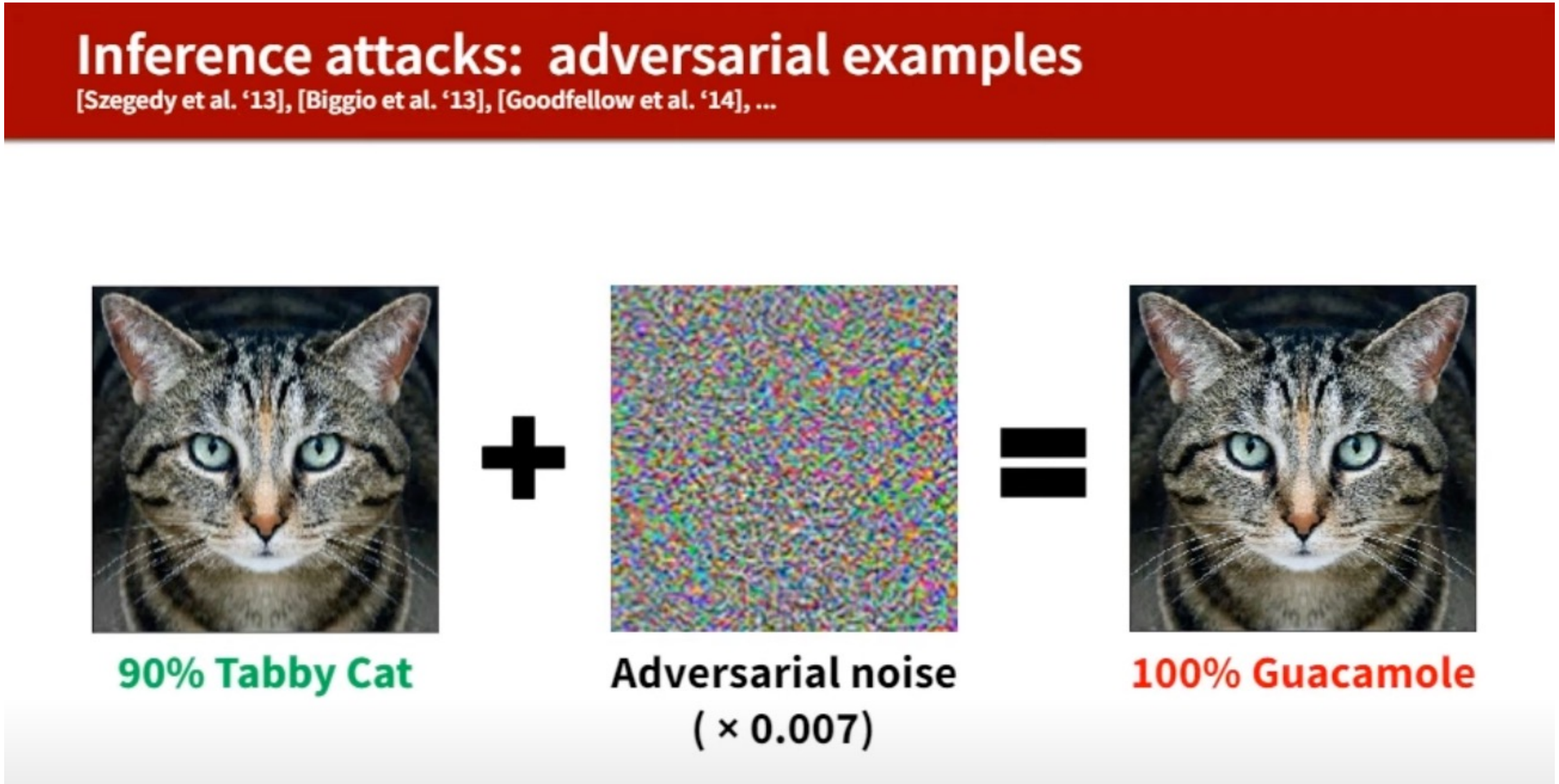


Fig. 1. **An illustration of membership inference attack.**

Today’s paper

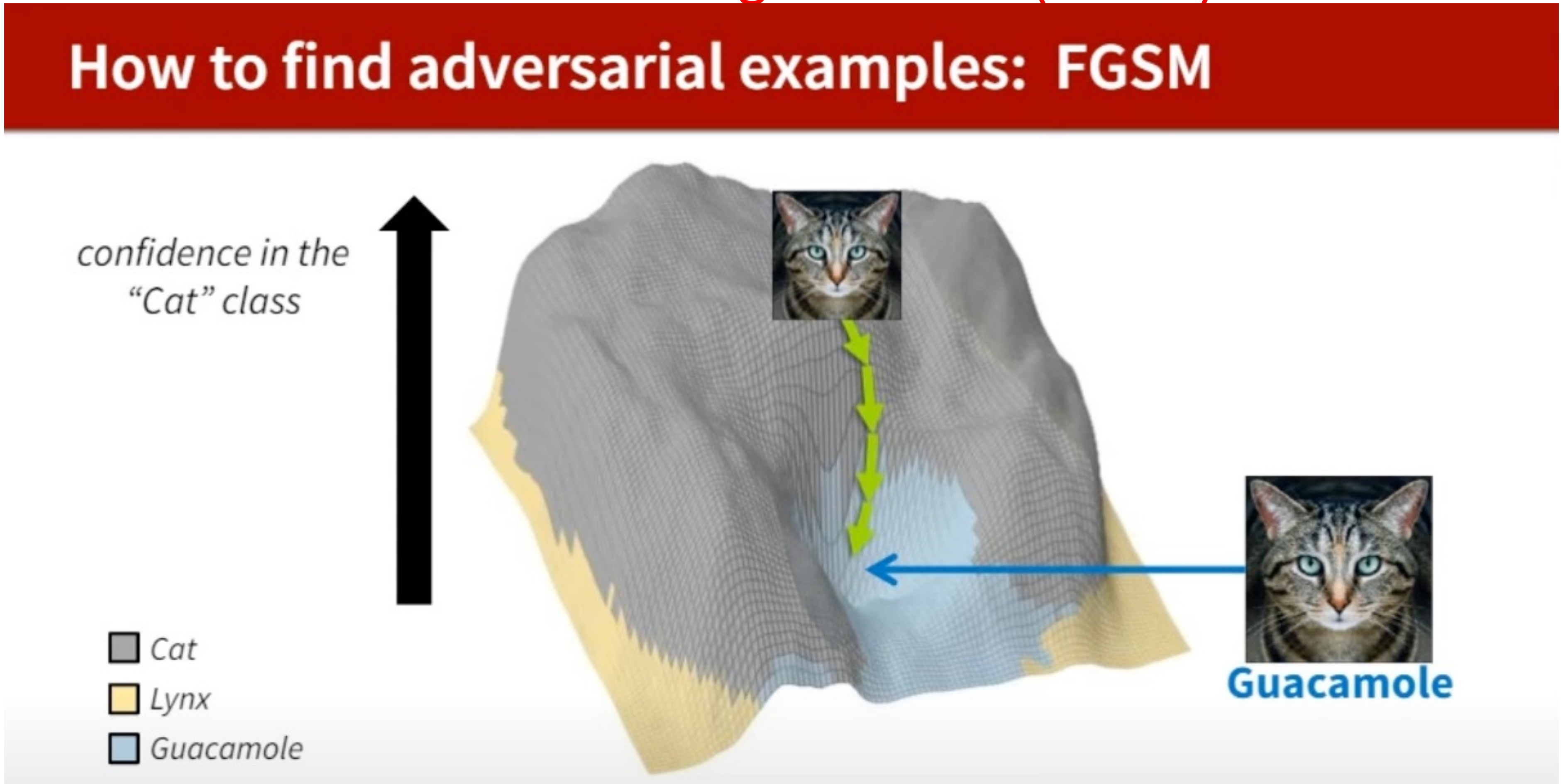
“hacking” ML systems

Adversarial noise



“hacking” ML systems

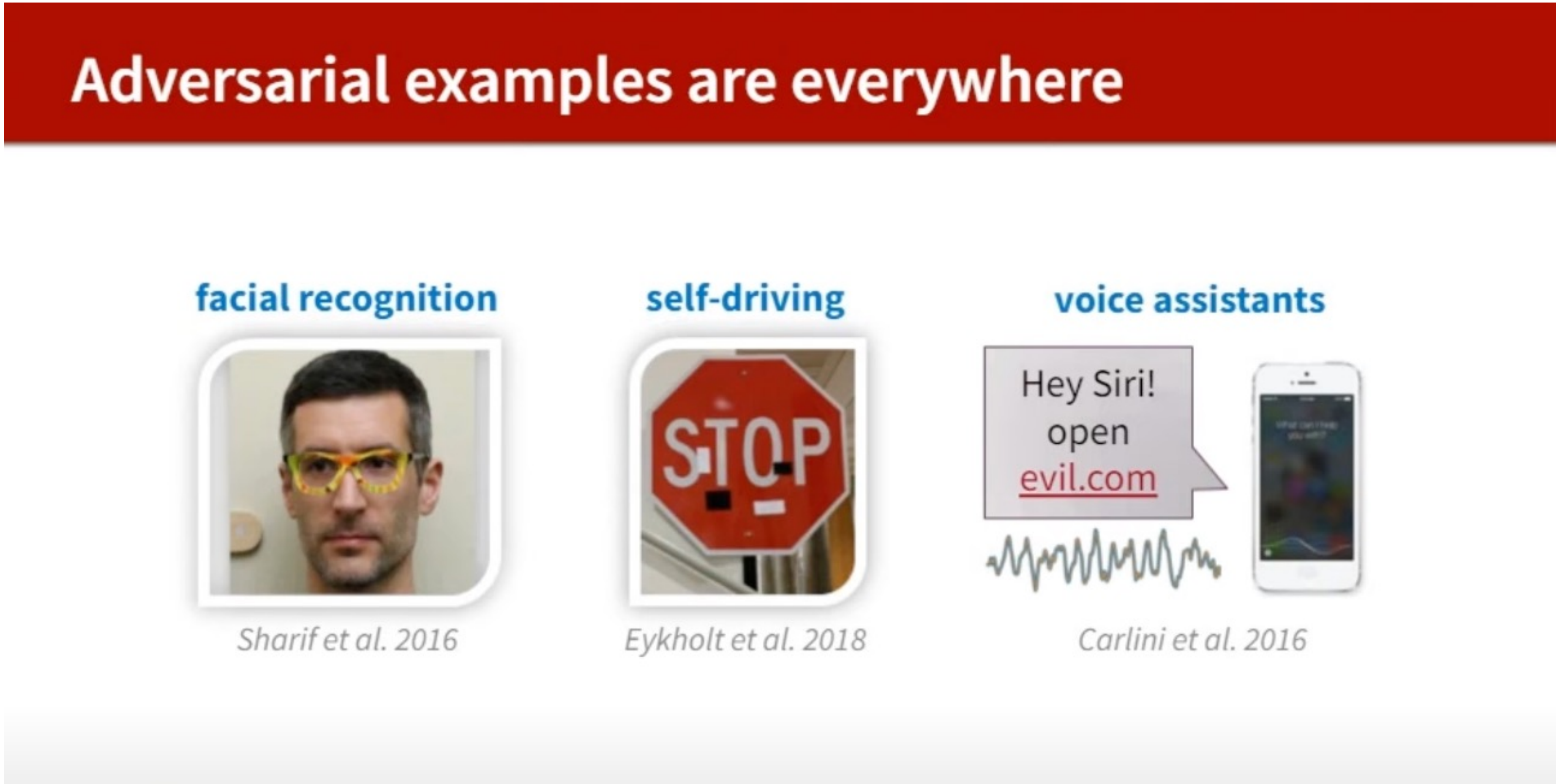
Fast Gradient Sign Method (FGSM)



Stanford Webinar with Dan Boneh - Hacking AI: Security & Privacy of Machine Learning Models

“hacking” ML systems

Adversarial examples



Stanford Webinar with Dan Boneh - Hacking AI: Security & Privacy of Machine Learning Models

“hacking” ML systems

Cat and mouse game: paper proposing defense followed

Most proposed defenses are broken

[Carlini & Wagner ’17], [Athalye et al. ’18], [Tramer, Carlini, Brendel, Mądry (NeurIPS 2020)], …

- *denoising*
- *randomization*
- *dimensionality reduction*
- *input transformations*
- *generative modeling*
- *Bayesian learning*
- …

Stanford Webinar with Dan Boneh - Hacking AI: Security & Privacy of Machine Learning Models

There is no 100% complete defense

- This is an active area of research

- You can build an entire research career out of:
  - Proposing ML security procedures, OR
  - Breaking recently released ML security procedures, OR
  - Doing both (breaking your defenses)

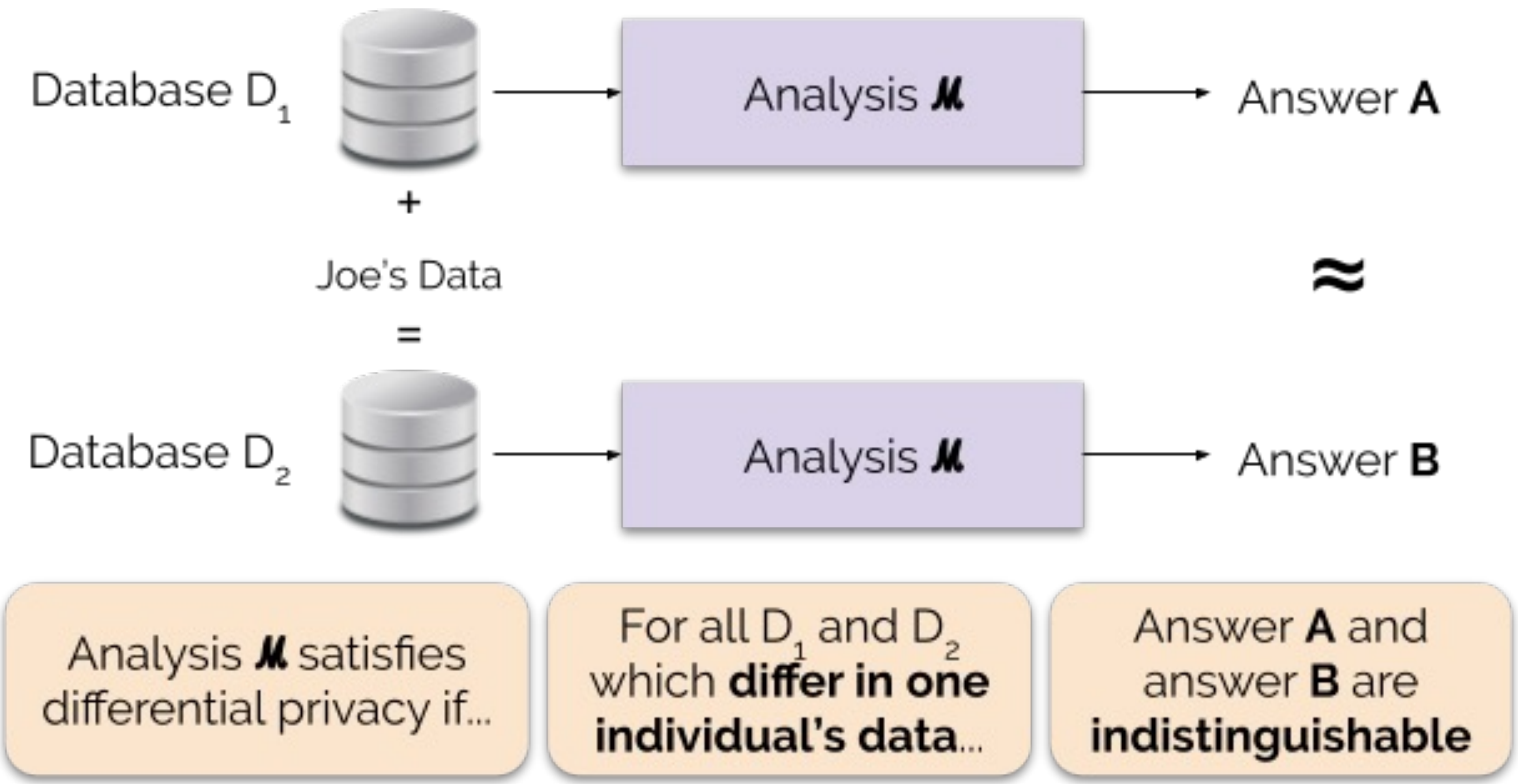
Differential privacy

Add noise to training dataset

The central question:

How much noise to add?

Differential privacy



https://www.nist.gov/blogs/cybersecurity-insights/differential-privacy-privacy-preserving-data-analysis-introduction-our

formal definition

Probability of seeing output O on input D_1

Probability of seeing output O on input D_2

$$\frac{\mathbf{Pr}[\mathcal{M}(D_1) \in O]}{\mathbf{Pr}[\mathcal{M}(D_2) \in O]} \leq e^{\varepsilon}$$

Indistinguishability: bounded ratio of probabilities

https://www.nist.gov/blogs/cybersecurity-insights/differential-privacy-privacy-preserving-data-analysis-introduction-our

formal definition

$$\Pr[\mathcal{A}(D_1) \in S] \leq \exp(\varepsilon) \cdot \Pr[\mathcal{A}(D_2) \in S]$$

Real-world use of differential privacy

- US Census Bureau using DP with the 2020 Census Data
- Apple uses DP in iOS and macOS for personal data like emoji use, search queries, and health data
- Microsoft uses DP for collecting data from Windows devices
- Google uses DP on Chrome and has released an open source DP library

Real-world use of differential privacy

- US Census Bureau using DP with the 2020 Census Data
- Apple uses DP in iOS and macOS for personal data like emoji use, search queries, and health data
- Microsoft uses DP for collecting data from Windows devices
- Google uses DP on Chrome and has released an open source DP library

Unfortunately, DP doesn’t work without a massive dataset…

Differential privacy libraries

- TensorFlow Privacy (Python) – used in today's paper
  - implementations of TensorFlow optimizers for training machine learning models with differential privacy
- Google Differential Privacy (C++, Go, Java)
  - libraries to generate ε- and (ε, δ)-differentially private statistics over datasets
- PipelineDP (Python / Apache Spark)
  - framework for applying differentially private aggregations to large datasets using batch processing systems such as Apache Spark, Apache Beam, and more

Biggest issue: degradation of performance

steadily decreasing when ε increases.

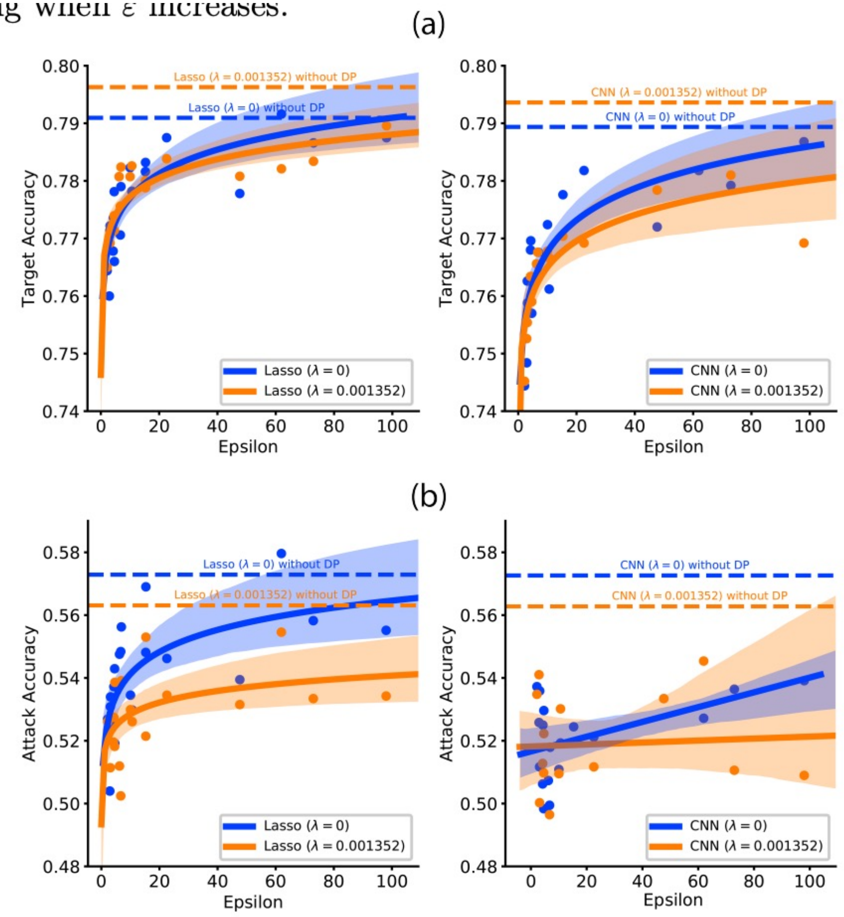


Fig. 2. **Accuracy values of the (a) target model and (b) attack model respectively under various privacy budgets (5-fold cross validation).** Curves indicate the fitted regression lines; shadow areas represent the 95% confidence intervals for corresponding regressions. Horizontal dotted lines represent model performances without DP.

Today's paper

Other Real-world issues with dp, as discussed at the us census

- “Perceptions and demography use cases involving census data often omit or downplay uncertainty measures”
  - Computer scientists argue that DP-related error due to noise can be quantified
  - Analyses using Census data often only look at means rather than standard deviations or confidence intervals
  - Census data are usually treated as point estimates with minimal statistical error

“More discussion of differential privacy at the Census” by Jessica Hullman. 2022.

Other Real-world issues with dp, as discussed at the us census

- “Developing methods for effectively using noisy data is difficult when use cases are not predetermined”
  - “sometimes demographers are put in the position of answering bizarre queries and this makes it difficult to precisely predict the use cases for which they’ll need methods for adapting noisy counts”

“More discussion of differential privacy at the Census” by Jessica Hullman. 2022.

Other Real-world issues with dp, as discussed at the us census

- “The relationship between privacy and legitimacy is complicated”
  - the Census Bureau misusing or inappropriately sharing confidential records
  - an outside party performing an attack on published census statistics

“More discussion of differential privacy at the Census” by Jessica Hullman. 2022.

Other Real-world issues with dp, as discussed at the us census

- “Courts may weigh the Census Bureau’s mandates differently and interpret DP noise as conceptually distinct from other forms of error”
  - It is currently unclear how the courts will interpret DP
  - “One legal expert noted that the enumeration of the population is constitutionally mandated, whereas the Census Bureau’s mandate to keep responses confidential does not appear in the Constitution”
  - “courts may consider DP noise to be of a different ‘flavor,’ since there is something conceptually different about intentionally injected noise compared to other sources of error that are not intentional or widely known”

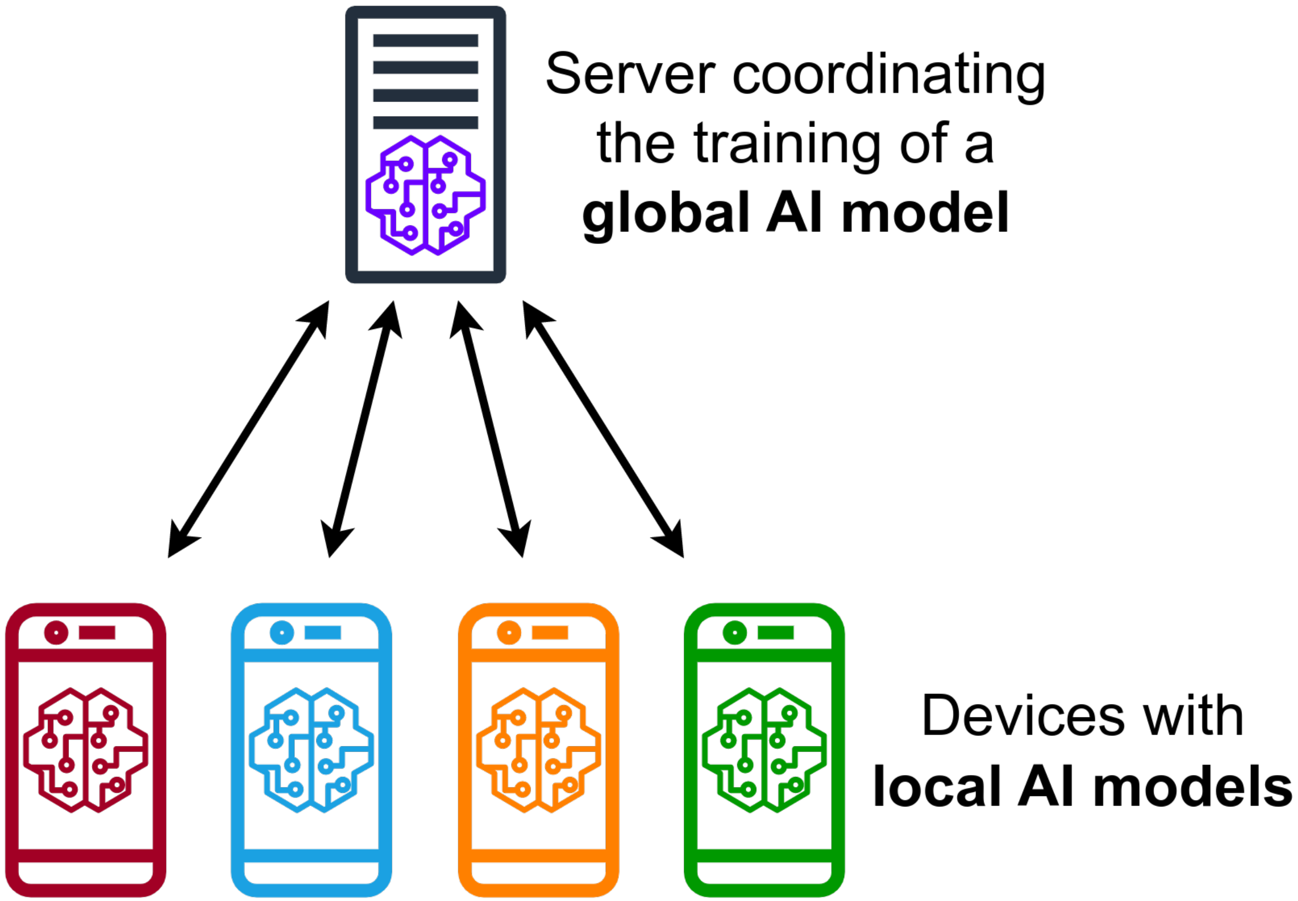
“More discussion of differential privacy at the Census” by Jessica Hullman. 2022.

Other Real-world issues with dp, as discussed at the us census

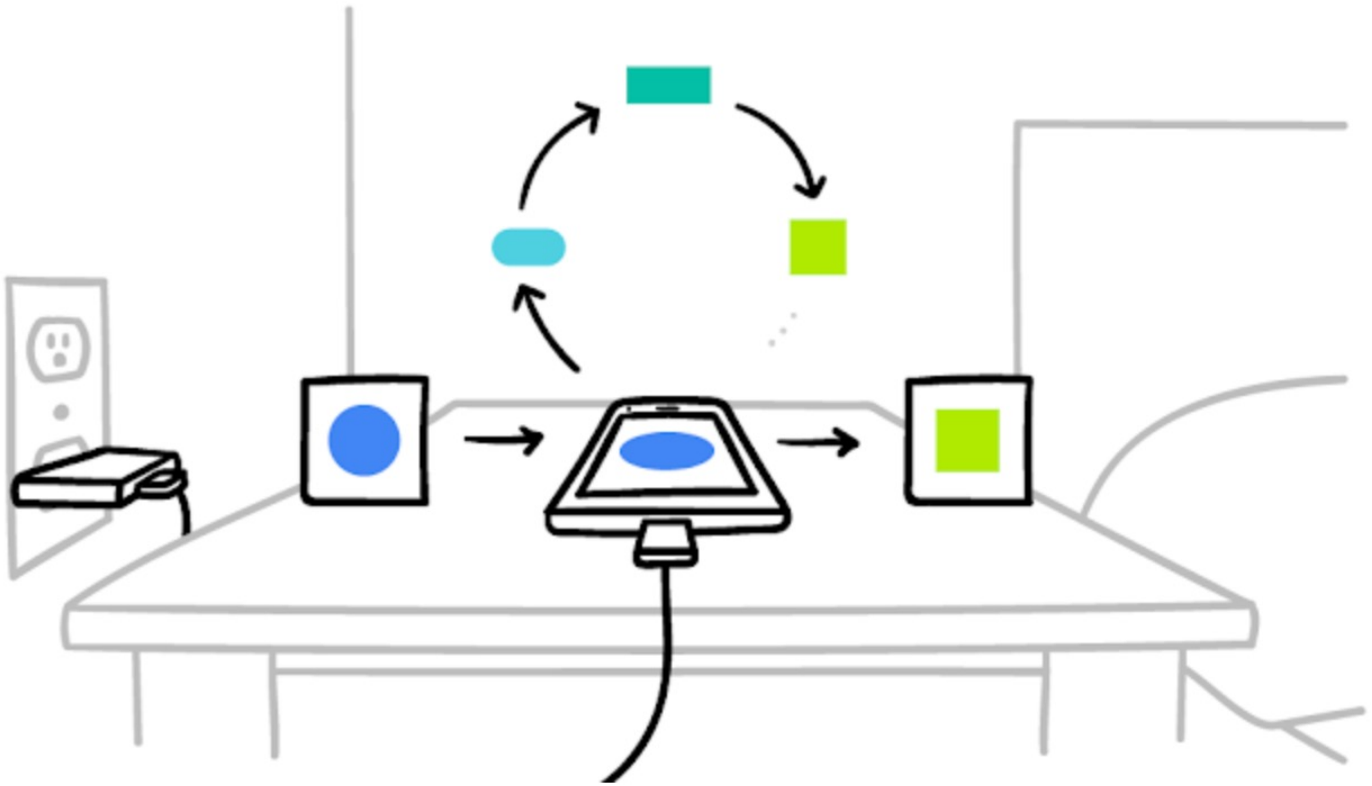
- “The trade-off is not just between privacy and accuracy—there are other dimensions, too”
  - Third dimension: legitimacy and trust of Census data
  - People may be alarmed to see low quality data given the amount of tax dollars that go into the US Census

“More discussion of differential privacy at the Census” by Jessica Hullman. 2022.

Federated Data Science



Example use case: Android



Your phone participates in Federated Learning only
when it won't negatively impact your experience.

https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

Example use cases on Mobile devices

- Query suggestions when typing
- Photo rankings based on types of photos someone views/shares/deletes
- Autocorrect
- …

https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

Other real-world use cases

- Autonomous vehicles
- Internet of Things
- Digital Health
- Mobile Apps
- Websites

Client-server model

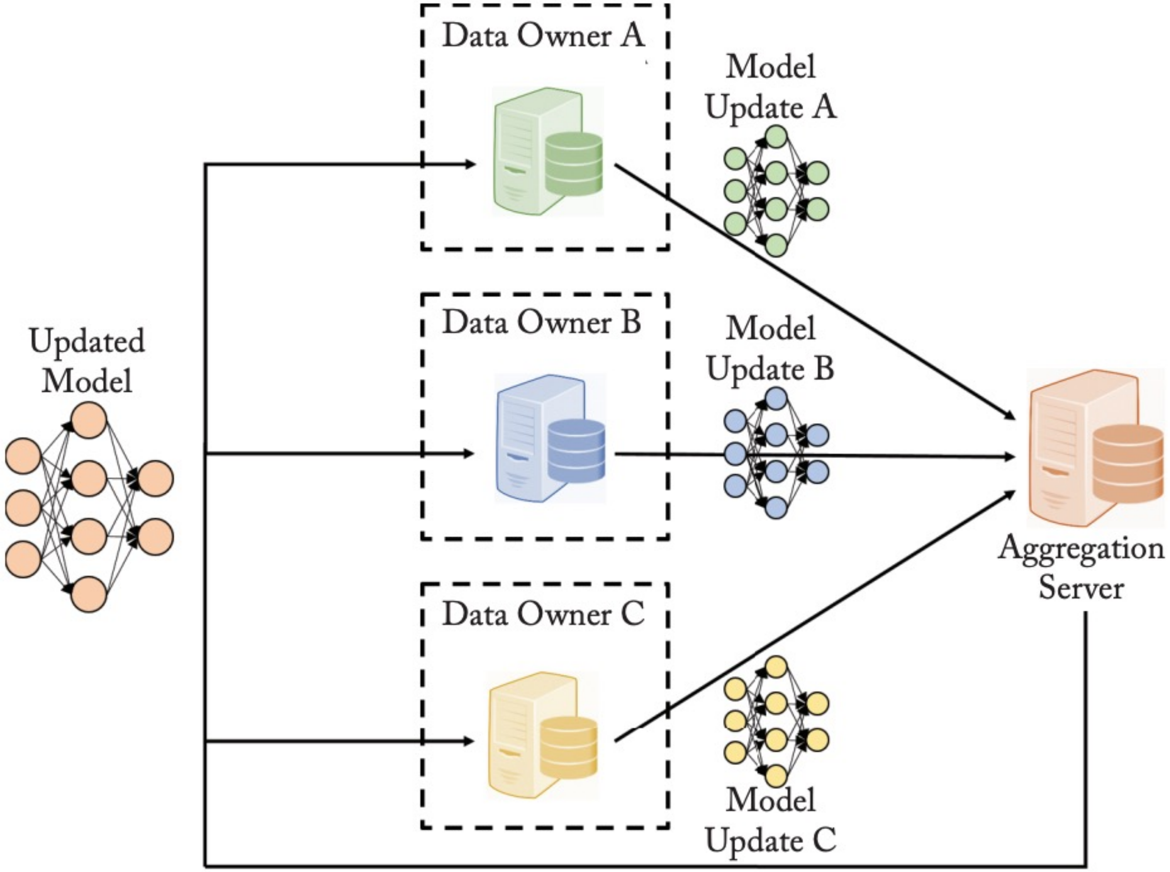


Figure 1.1: An example federated learning architecture: client-server model.

Yang Qiang et al. “Federated Learning.” Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

Peer-to-peer model

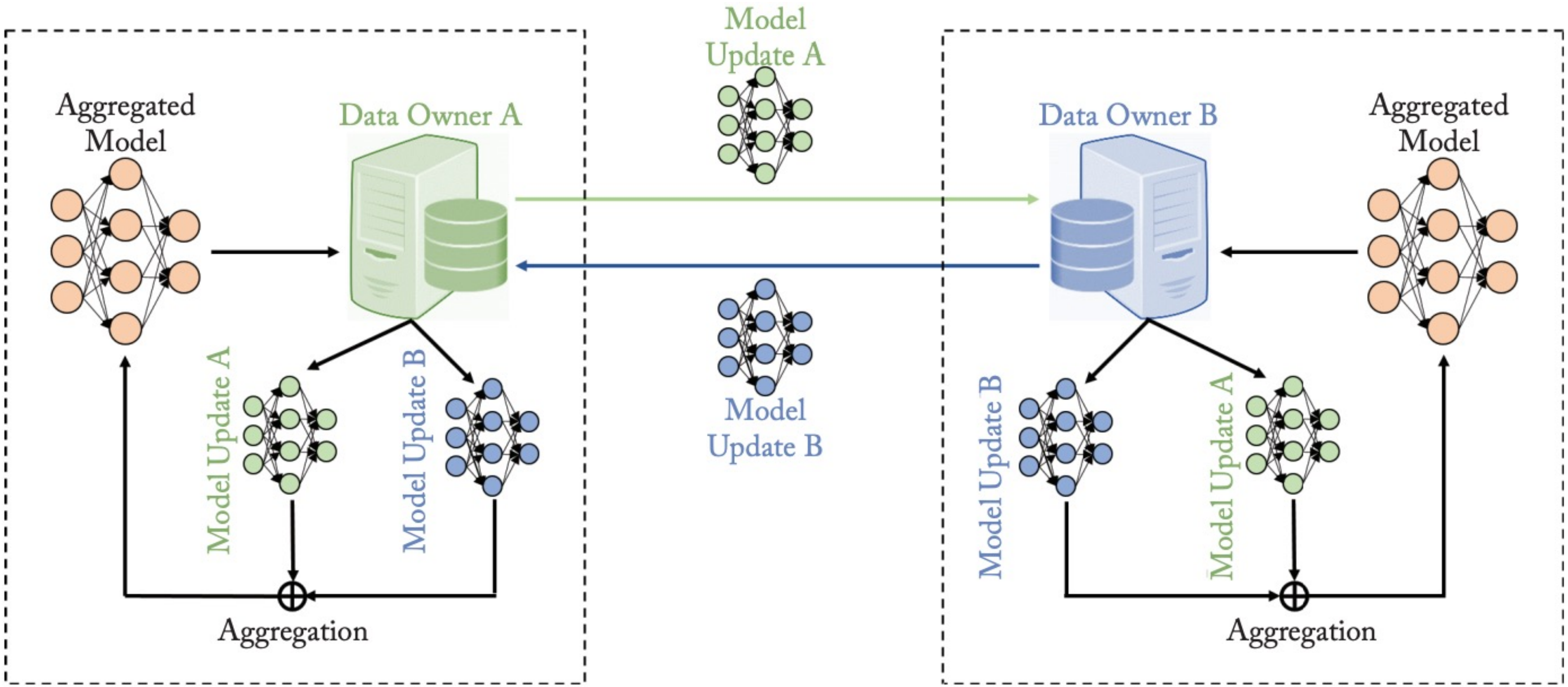


Figure 1.2: An example federated learning architecture: peer-to-peer model.

Yang Qiang et al. "Federated Learning." Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

Distributed machine learning

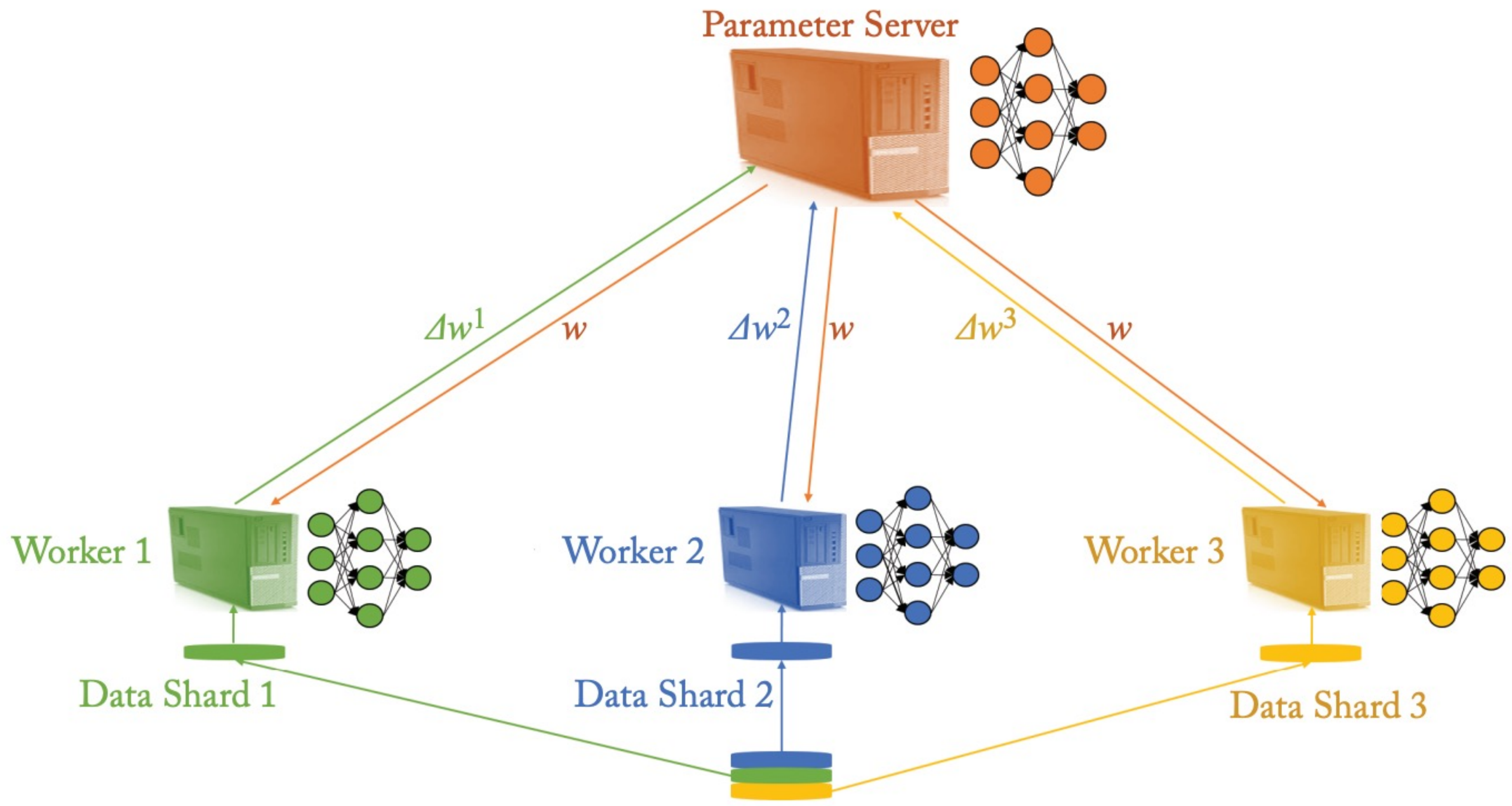


Figure 3.1: Illustration of a distributed machine learning (DML) system.

Yang Qiang et al. "Federated Learning." Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

Comparison of model aggregation techniques

Method	Advantage	Disadvantage
Gradient averaging	Accurate gradient information Guaranteed convergence	Heavy communication Require reliable connection
Model averaging	Not bound to SGD Tolerance of update loss Infrequent synchronization	No guarantee of convergence Performance loss

Yang Qiang et al. "Federated Learning." Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

“Horizontal” federated learning

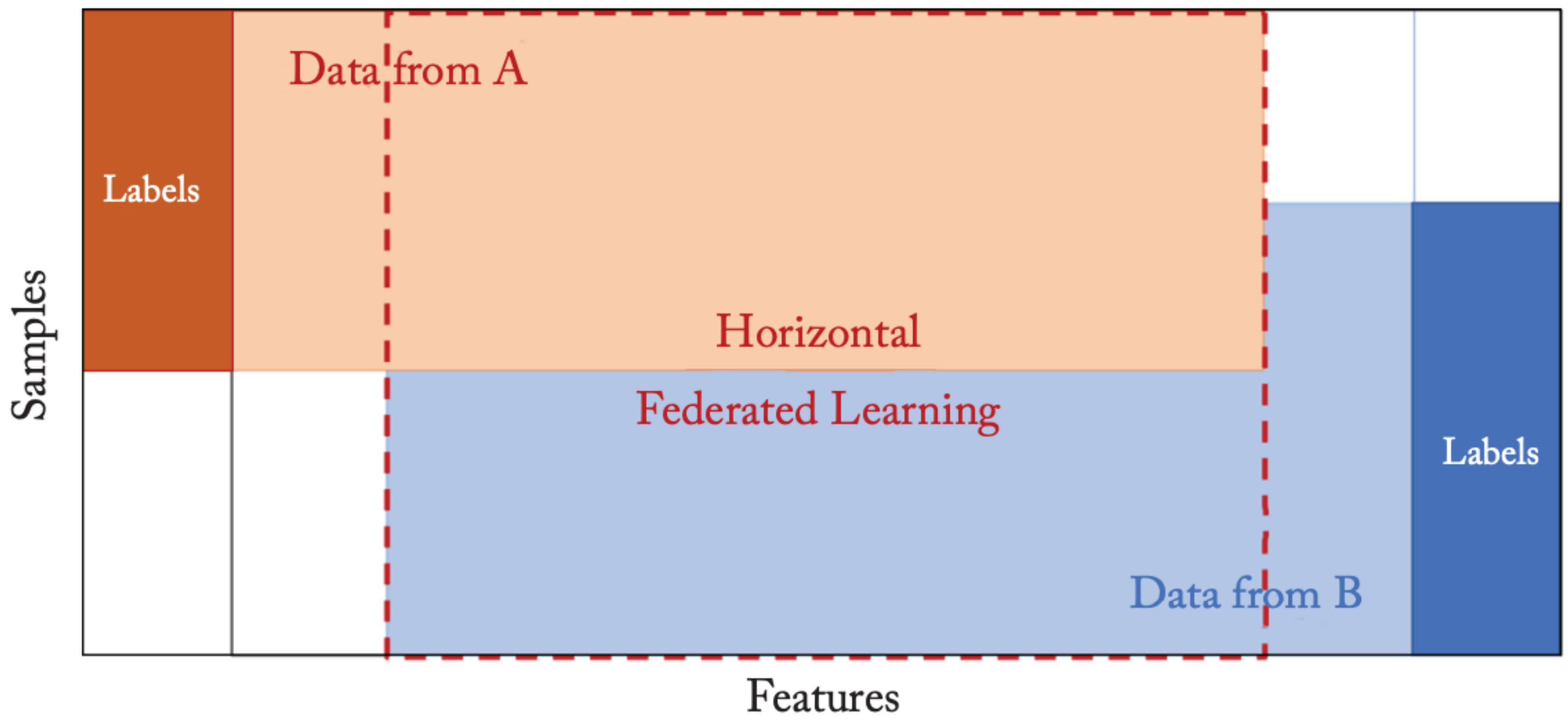


Figure 1.3: Illustration of HFL, a.k.a. sample-partitioned federated learning where the overlapping features from data samples held by different participants are taken to jointly train a model [Yang et al., 2019].

Yang Qiang et al. “Federated Learning.” Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

“Horizontal” federated learning

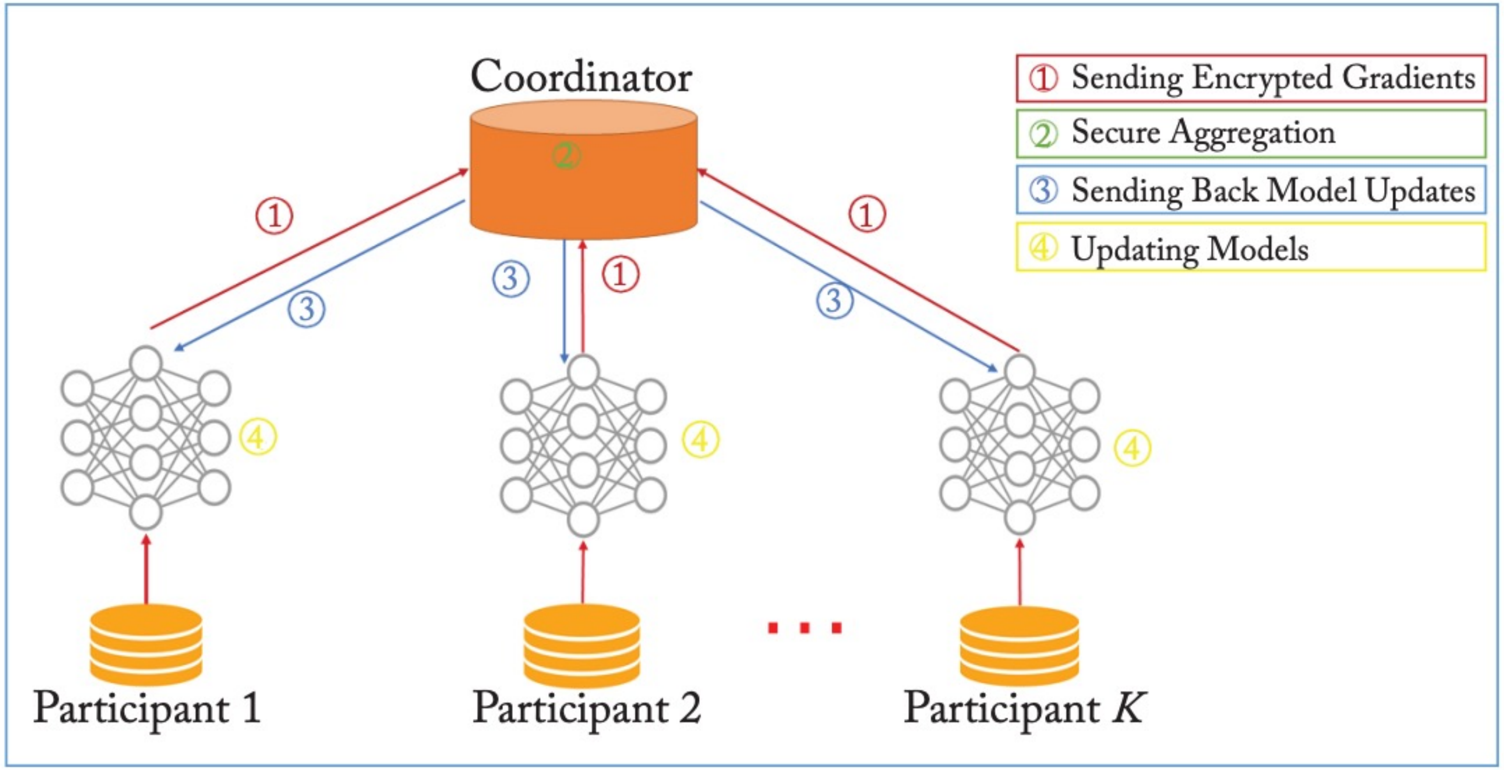


Figure 4.2: Exemplary client-server architecture for an HFL system [Yang et al., 2019].

Yang Qiang et al. “Federated Learning.” Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

General algorithm: Fedavg

Algorithm 1. Example of a FL algorithm[16] via Hub & Spoke (Centralised topology) with `FedAvg` aggregation[9].

Require: num_federated_rounds T

procedure AGGREGATING
Initialise global model: $W^{(0)}$
for $t \leftarrow 1 \cdots T$ **do**
for *client* $k \leftarrow 1 \cdots K$ **do** $\triangleright$ *Run in parallel*
Send $W^{(t-1)}$ to client k
Receive model updates and number of local training iterations $(\Delta W_k^{(t-1)}, N_k)$ from client's local training with $\mathcal{L}_k(X_k; W^{(t-1)})$
end for
$W^{(t)} \leftarrow W^{(t-1)} + \frac{1}{\sum_k N_k} \sum_k (N_k \cdot W_k^{(t-1)})$
end for
return $W^{(t)}$
end procedure

Today's paper

General algorithm: Fedavg

Algorithm 1. Example of a FL algorithm[16] via Hub & Spoke (Centralised topology) with `FedAvg` aggregation[9].

Require: num_federated_rounds T

1: **procedure** AGGREGATING

2: Initialise global model: $W^{(0)}$

3: **for** $t \leftarrow 1 \cdots T$ **do** Perform several rounds of client training and server aggregation

4: **for** *client* $k \leftarrow 1 \cdots K$ **do** ▷ *Run in parallel*

5: Send $W^{(t-1)}$ to client k

6: Receive model updates and number of local training iterations $(\Delta W_k^{(t-1)}, N_k)$ from client's local training with $\mathcal{L}_k(X_k; W^{(t-1)})$

7: **end for**

8: $W^{(t)} \leftarrow W^{(t-1)} + \frac{1}{\sum_k N_k} \sum_k (N_k \cdot W_k^{(t-1)})$

9: **end for**

10: **return** $W^{(t)}$

11: **end procedure**

Today's paper

General algorithm: Fedavg

Algorithm 1. Example of a FL algorithm[16] via Hub & Spoke (Centralised topology) with `FedAvg` aggregation[9].

Require: num_federated_rounds T

1: **procedure** AGGREGATING

2: Initialise global model: $W^{(0)}$

3: **for** $t \leftarrow 1 \cdots T$ **do** Perform several rounds of client training and server aggregation

4: **for** *client* $k \leftarrow 1 \cdots K$ **do** ▷ *Run in parallel*

5: Send $W^{(t-1)}$ to client k Each client gets global model weights so far

6: Receive model updates and number of local training iterations $(\Delta W_k^{(t-1)}, N_k)$ from client's local training with $\mathcal{L}_k(X_k; W^{(t-1)})$

7: **end for**

8: $W^{(t)} \leftarrow W^{(t-1)} + \frac{1}{\sum_k N_k} \sum_k (N_k \cdot W_k^{(t-1)})$

9: **end for**

10: **return** $W^{(t)}$

11: **end procedure**

Today's paper

General algorithm: Fedavg

Algorithm 1. Example of a FL algorithm[16] via Hub & Spoke (Centralised topology) with `FedAvg` aggregation[9].

Require: num_federated_rounds T

1: **procedure** AGGREGATING
2: Initialise global model: $W^{(0)}$
3: **for** $t \leftarrow 1 \cdots T$ **do** Perform several rounds of client training and server aggregation
4: **for** *client* $k \leftarrow 1 \cdots K$ **do** $\triangleright$ *Run in parallel*
5: Send $W^{(t-1)}$ to client k Each client gets global model weights so far
6: Receive model updates and number of local training iterations $(\Delta W_k^{(t-1)}, N_k)$ from client's local training with $\mathcal{L}_k(X_k; W^{(t-1)})$ Client updates weights using its local data and sends to server
7: **end for**
8: $W^{(t)} \leftarrow W^{(t-1)} + \frac{1}{\sum_k N_k} \sum_k (N_k \cdot W_k^{(t-1)})$
9: **end for**
10: **return** $W^{(t)}$
11: **end procedure**

Today's paper

General algorithm: Fedavg

Algorithm 1. Example of a FL algorithm[16] via Hub & Spoke (Centralised topology) with `FedAvg` aggregation[9].

Require: num_federated_rounds T

1: **procedure** AGGREGATING
2: Initialise global model: $W^{(0)}$
3: **for** $t \leftarrow 1 \cdots T$ **do** Perform several rounds of client training and server aggregation
4: **for** *client* $k \leftarrow 1 \cdots K$ **do** ▷ *Run in parallel*
5: Send $W^{(t-1)}$ to client k Each client gets global model weights so far
6: Receive model updates and number of local training iterations $(\Delta W_k^{(t-1)}, N_k)$ from client's local training with $\mathcal{L}_k(X_k; W^{(t-1)})$ Client updates weights using its local data and sends to server
7: **end for**
8: $W^{(t)} \leftarrow W^{(t-1)} + \frac{1}{\sum_k N_k} \sum_k (N_k \cdot W_k^{(t-1)})$ Global model is updated as weighted sum of each client model's weights
9: **end for**
10: **return** $W^{(t)}$
11: **end procedure**

Today's paper

“vertical” federated learning

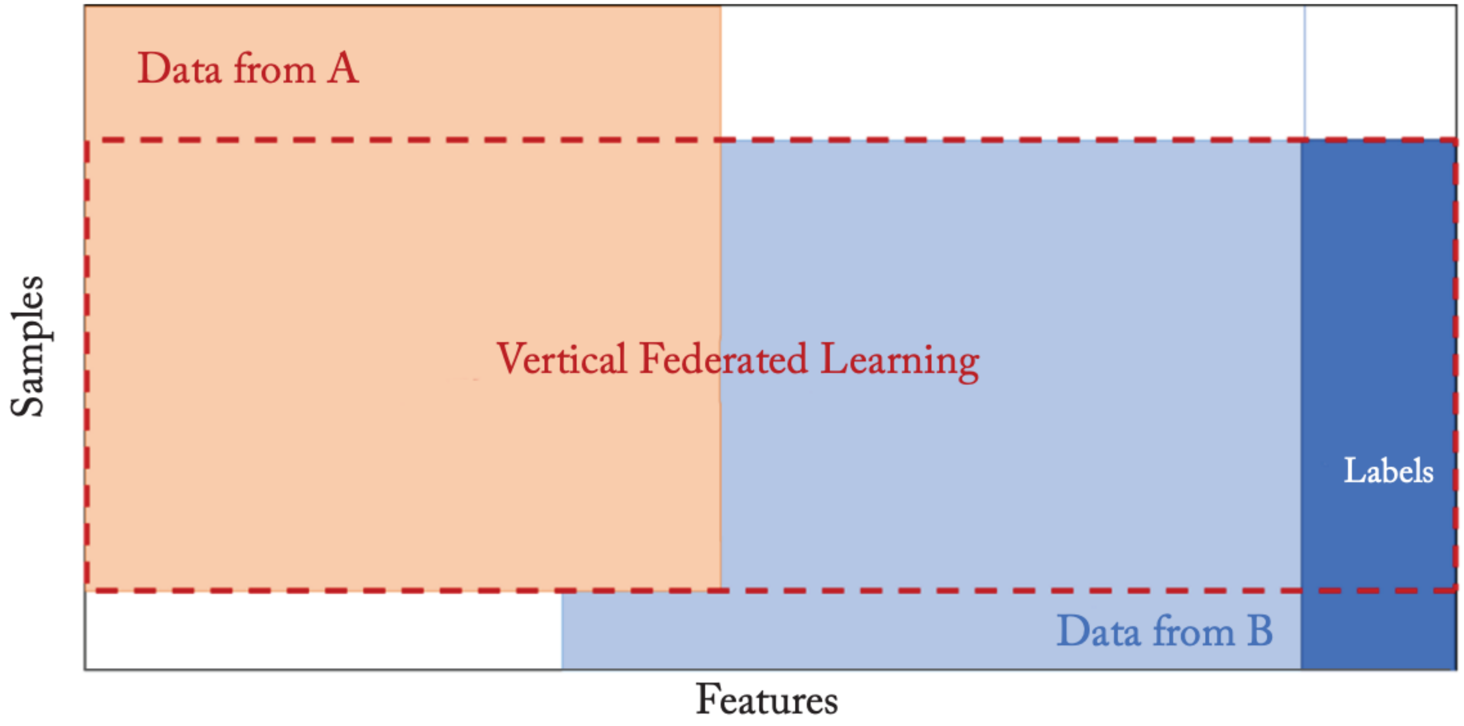


Figure 1.4: Illustration of VFL, a.k.a feature-partitioned federated learning where the overlapping data samples that have non-overlapping or partially overlapping features held by multiple participants are taken to jointly train a model [Yang et al., 2019].

Yang Qiang et al. “Federated Learning.” Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

“vertical” federated learning

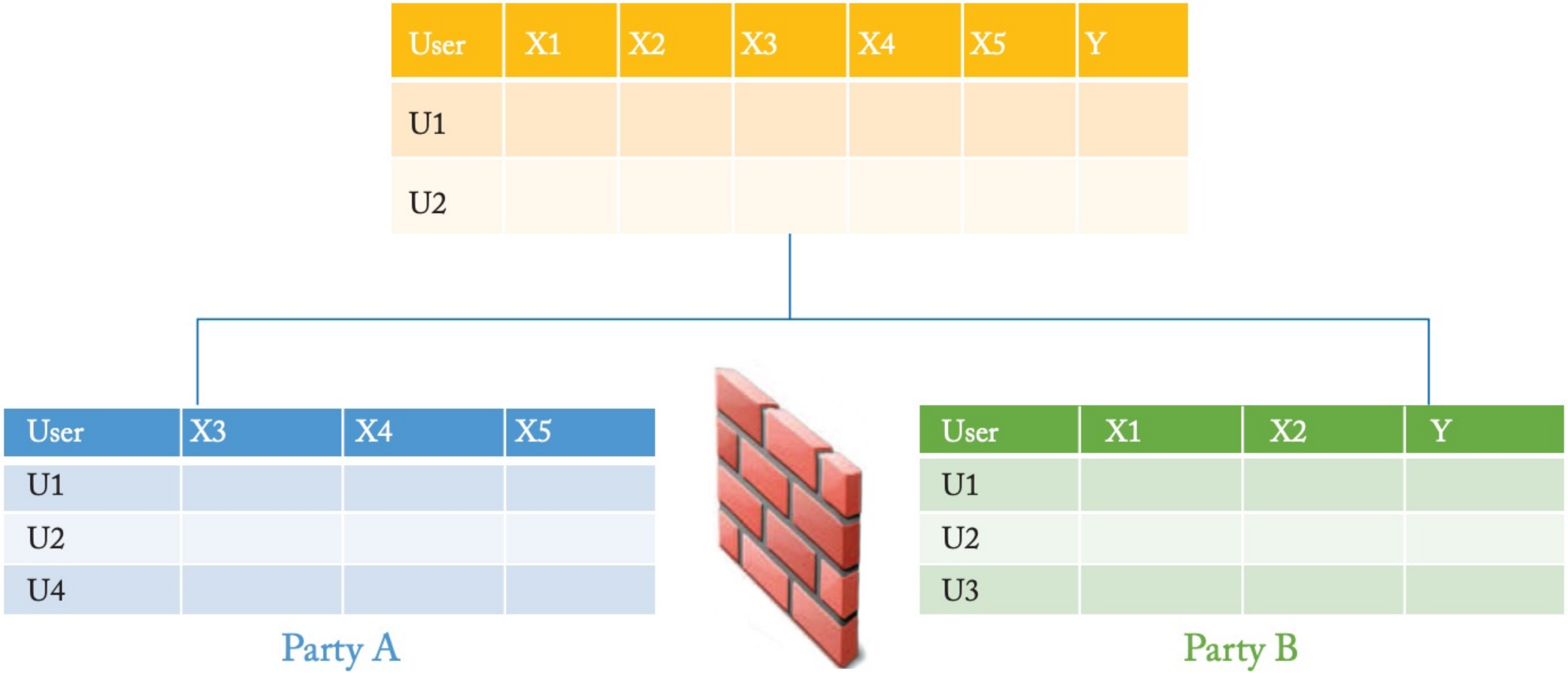


Figure 5.3: Illustration of encrypted entity alignment [Cheng et al., 2019].

Yang Qiang et al. “Federated Learning.” Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

“vertical” federated learning

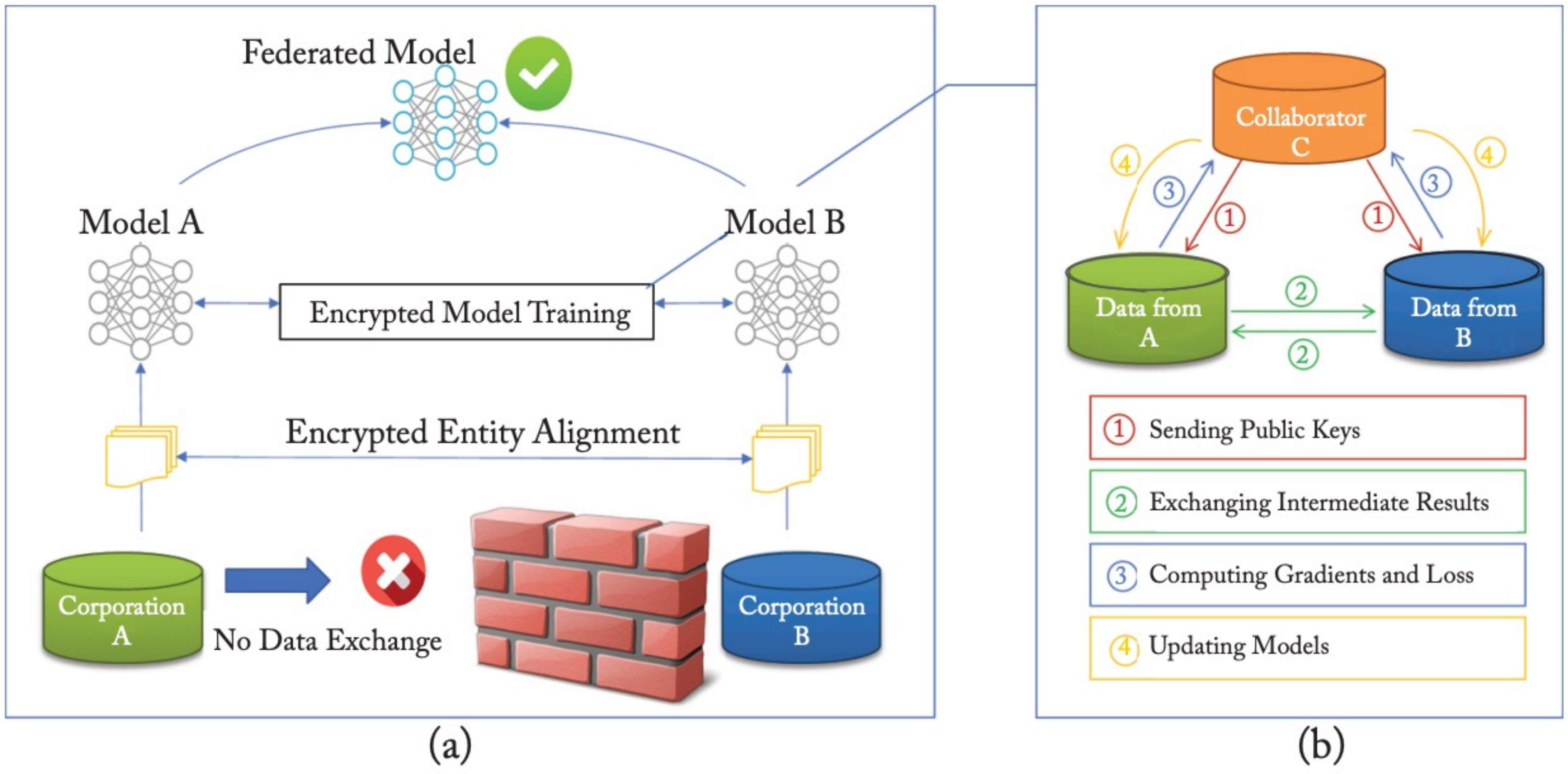


Figure 5.2: Architecture for a vertical federated learning system [Yang et al., 2019].

Yang Qiang et al. “Federated Learning.” Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

Example: secure linear regression

The Goal, simplified for only 2 clients:

$$\min_{\Theta_A,\Theta_B} \sum_i \left\| \Theta_A x_i^A + \Theta_B x_i^B - y_i \right\|^2 + \frac{\lambda}{2}\left(\|\Theta_A\|^2 + \|\Theta_B\|^2 \right).$$

Yang Qiang et al. “Federated Learning.” Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

secure linear regression: Training

	Party A	Party B	Party C
Step 1	Initializes Θ_A	Initializes Θ_B	Creates an encryption key pair and sends public key to A and B
Step 2	Computes $[[u_i^A]]$, $[[\mathcal{L}_A]]$ and sends to B	Computes $[[u_i^B]]$, $[[d_i^B]]$, $[[\mathcal{L}]]$, and sends $[[d_i^B]]$ to A, and sends $[[\mathcal{L}]]$ to C	
Step 3	Initializes R_A, computes $[[\frac{\partial \mathcal{L}}{\partial \Theta_A}]] + [[R_A]]$ and sends to C	Initializes R_B, computes$[[\frac{\partial \mathcal{L}}{\partial \Theta_B}]] + [[R_B]]$ and sends to C	Decrypts $[[\mathcal{L}]]$ and sends $[[\frac{\partial \mathcal{L}}{\partial \Theta_A}]] + R_A$ to A, $[[\frac{\partial \mathcal{L}}{\partial \Theta_B}]] + R_B$ to B
Step 4	Updates Θ_A	Updates Θ_B	
What is obtained?	Θ_A	Θ_B	

η	The learning rate
λ	The regularization parameter
y_i	The label space of party B
x_i^A, x_i^B	Feature space of party A and B, respectively
Θ_A, Θ_B	Local model parameters of party A and B, respectively
u_i^A	Defined as $u_i^A = \Theta_A x_i^A$
u_i^B	Defined as $u_i^B = \Theta_B x_i^B$
$[[d_i]]$	Defined as $[[d_i]] = [[u_i^A]] + [[u_i^B - y_i]]$
$\{x_i^A\}_{i \in \mathcal{D}_A}$	The local dataset of party A
$\{x_i^B, y_i\}_{i \in \mathcal{D}_B}$	The local dataset and labels of party B
$[[\cdot]]$	Additive homomorphic encryption (AHE)
R_A and R_B	The random masks of party A and party B, respectively

Yang Qiang et al. "Federated Learning." Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

secure linear regression: Training

Party C only learns masked gradients

Party A learns its gradient at each step but nothing about Party B

Party B learns its gradient at each step but nothing about Party A

Yang Qiang et al. "Federated Learning." Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

secure linear regression: Predicting

	Party A	Party B	Evaluator C
Step 0			Sends user ID i to A and B
Step 1	Computes u_i^A and sends to C	Computes u_i^B and sends to C	Computes the result of $u_i^A + u_i^B$

Yang Qiang et al. “Federated Learning.” Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

Secure examples of the other ML algorithms we have discussed in class have also been published

See here if interested:

Yang Qiang et al. “Federated Learning.” Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

federated Transfer learning

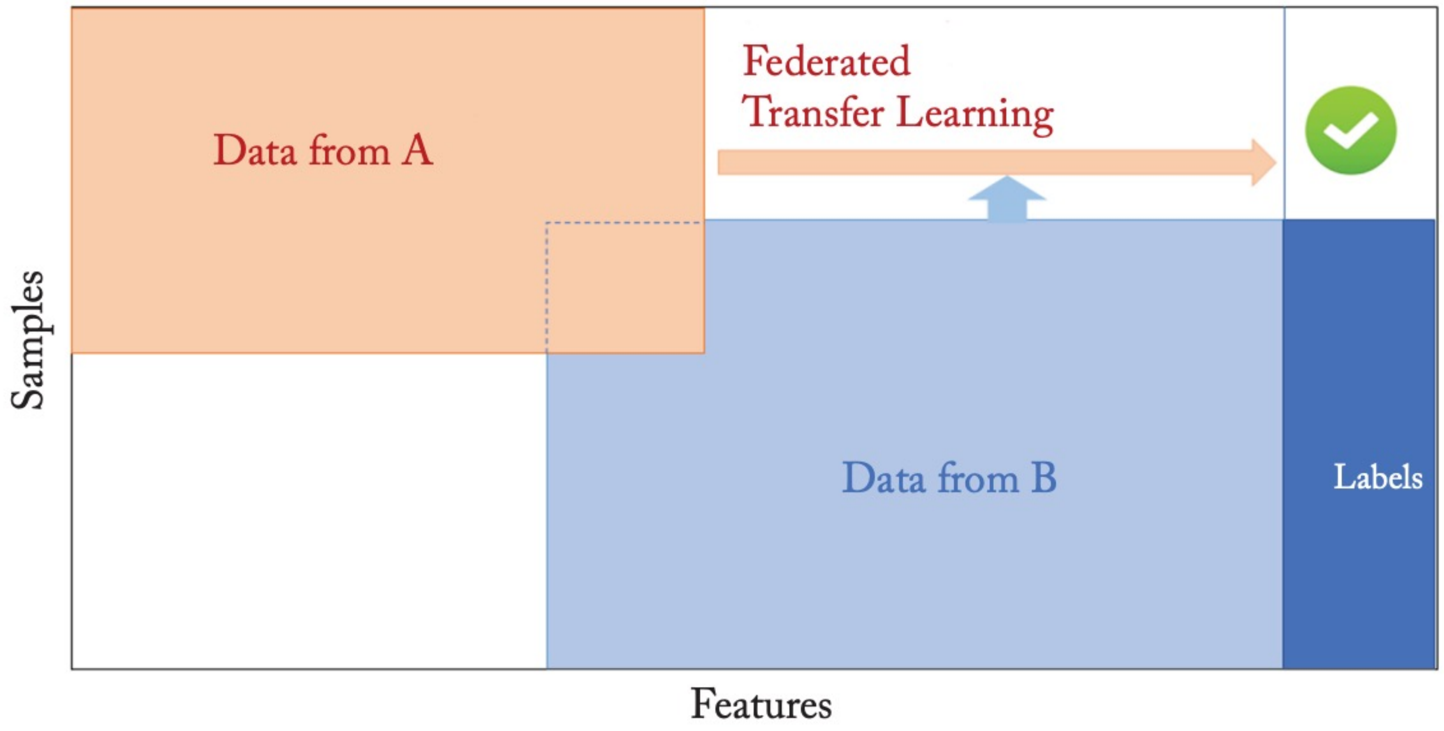


Figure 1.5: Federated transfer learning (FTL) [Yang et al., 2019]. A predictive model learned from feature representations of aligned samples belonging to party A and party B is utilized to predict labels for unlabeled samples of party A.

Yang Qiang et al. “Federated Learning.” Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.

Practical implementation considerations

- Ethics: How do we choose how much to weight each local federation’s data?
- How to efficiently aggregate the data?
- How to train models on relatively small processors on mobile devices?
- How to account for long upload speeds?
- How to handle heterogeneous data sources?
- How to handle limited connectivity/bandwidth of local devices?
- How to handle asynchrony of device updates?

Ongoing research questions

- How to handle differently distributed data sources? (Big issue in the real world)
- How to handle when one party is malicious and provides “poison” data?
- How to perform federated learning in reinforcement learning settings?
- How to perform model training on resource-constrained mobile devices?

Yang Qiang et al. “Federated Learning.” Synthesis Lectures on Artificial Intelligence and Machine Learning. 2020.