Data Privacy in The Digital Age

for businesses, citizens and data scientists

Changrong Ji

A3.AI

04/2020

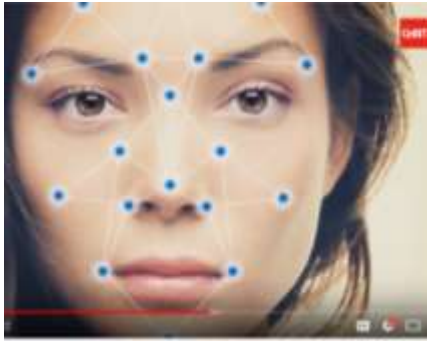
TOPICS

1. Data Privacy is a societal problem Business, technology, regulations, ethics implications
2. Privacy vs. Utility vs. Efficiency
3. Anonymization is not enough
4. State-of-the-art privacy preserving techniques
5. Opportunities exist in the intersection of disciplines

Resources for deeper dive

AI is Addressing Humanities' Big Challenges

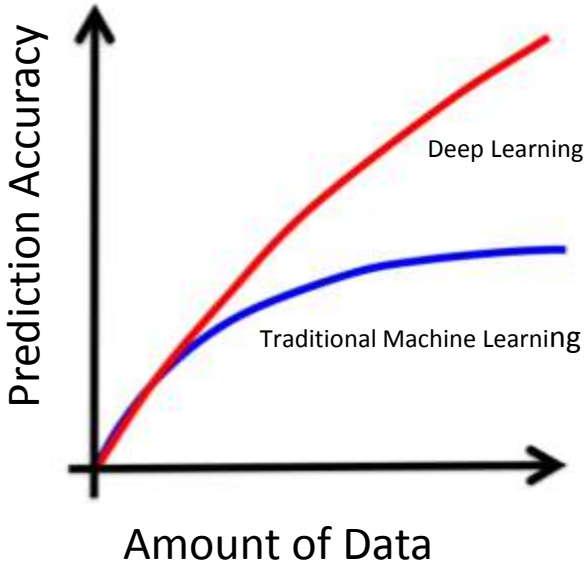
6 Deep Genomics



AI and Big Data Helping to Fight COVID-19

1. Epidemiological modeling
2. Drug Discovery and Repurposing
3. Personalized Infection Risk and Severity Prediction
4. Contact Tracing
5. More...

Deep Learning is Data Hungry



More Attributes

More Samples

Data Linkage Attack

The State of Data

- ImageNet
- Wikipedia
- Canadian Book Corpus
- IMDB Movie Reviews
- Twitter
- …

Big Public Data propelling AI adoptions in many areas

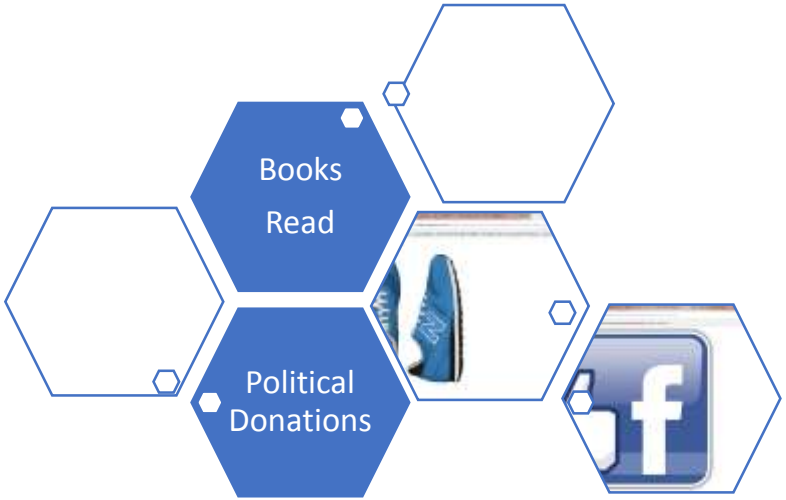
- Medical records
- Biometrics
- Financial data
- Locations
- Communications
- …

Data Privacy Challenges limiting progress in others

‘Anonymized Data’ Isn’t so Anonymous. Why?

Anonymized data is not as anonymous as we thought. Credit: Getty Images

Data Scientists Knows All Your Secrets



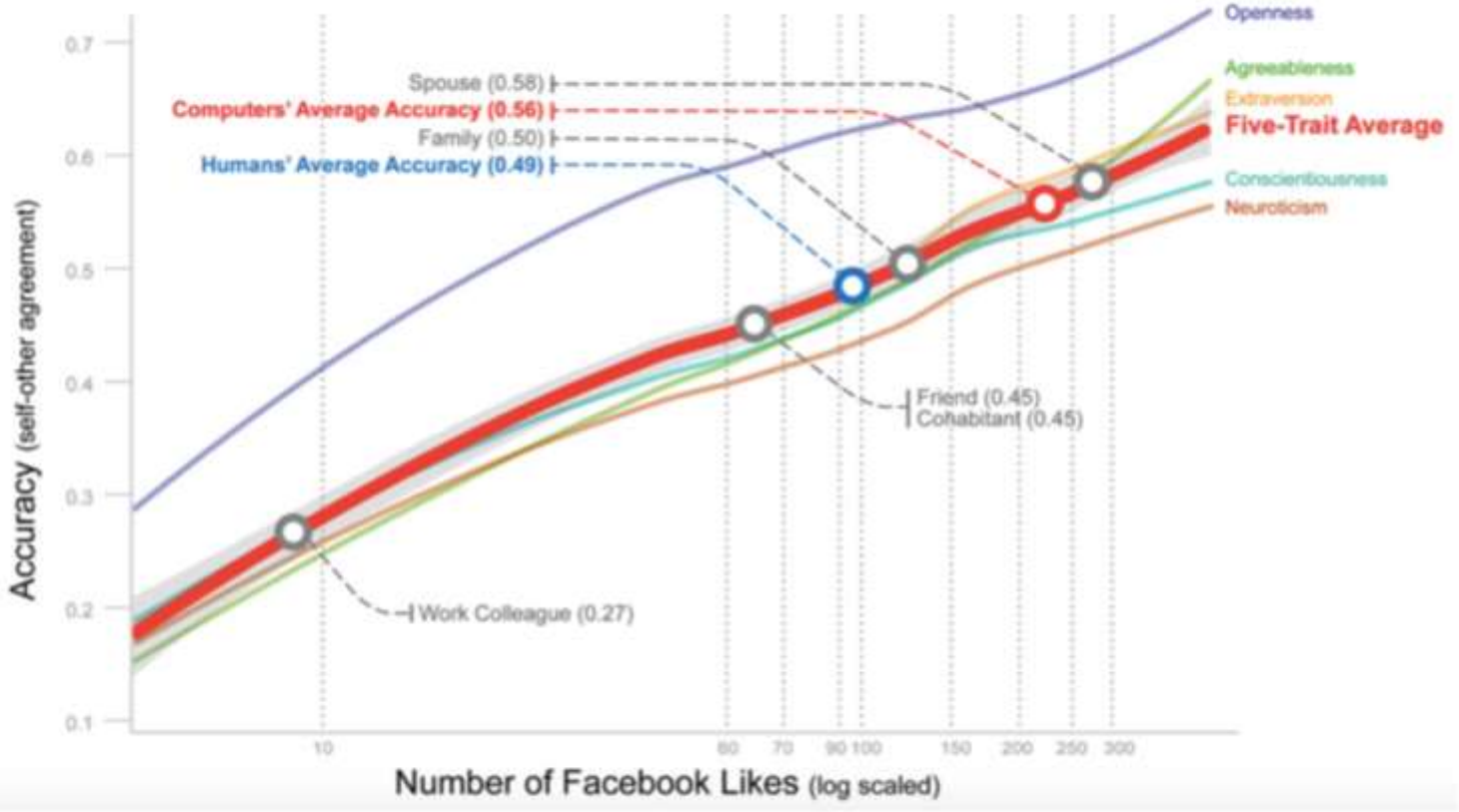
Political Views

Sexual Orientations

Personalities

….

“Facebook Likes” Predict Personality



“Ask Siri, Dictation & Privacy”

…
When you use Siri and Dictation, the things you say and dictate will be sent to Apple to process your requests… your device will send other Siri Data, such as:

- contact names, nicknames, and relationships (e.g., “my dad”),
- music, books and podcasts you enjoy
- names of your and your Family Sharing members’ devices;
- names of devices and members of a shared home in the Home app; and
- the names of your photo albums, apps installed on your device, and shortcuts you added through Siri.
- your approximate GPS location…

Requests are associated with a random identifier, not your Apple ID
…

Privacy Regulations

EU: General Data Protection Regulation (GDPR)

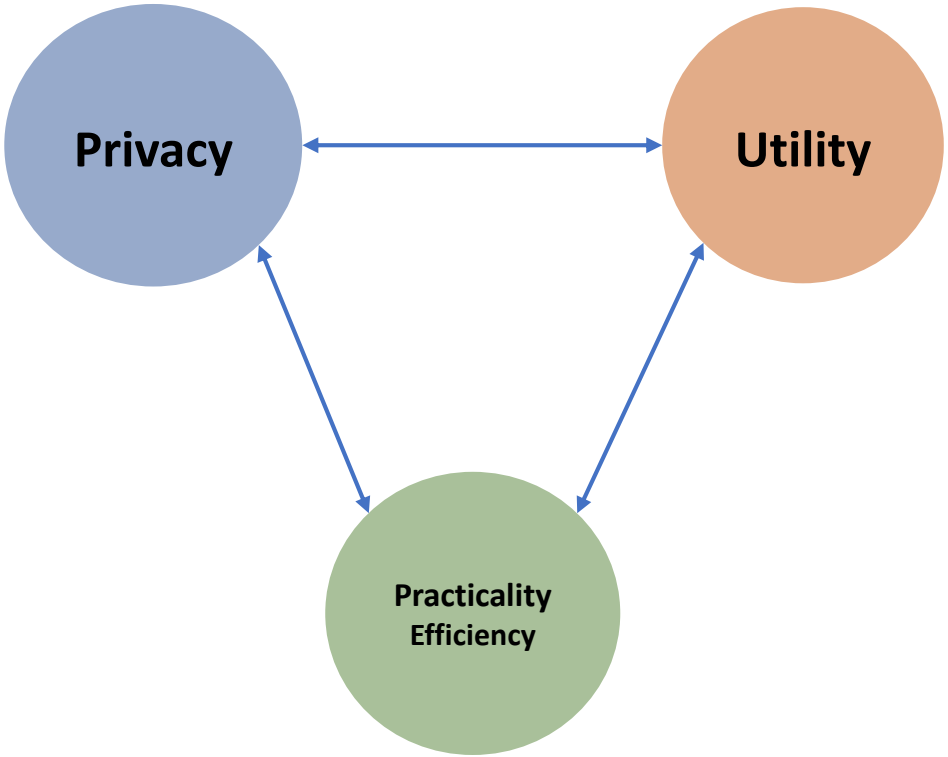
US: hundreds of federal, state, sector level laws and regulations

- Healthcare HIPPA
- Financial Payment Card Industry Data Security Standard
- California Consumer Privacy Act of 2020
- A few other states likely to follow soon

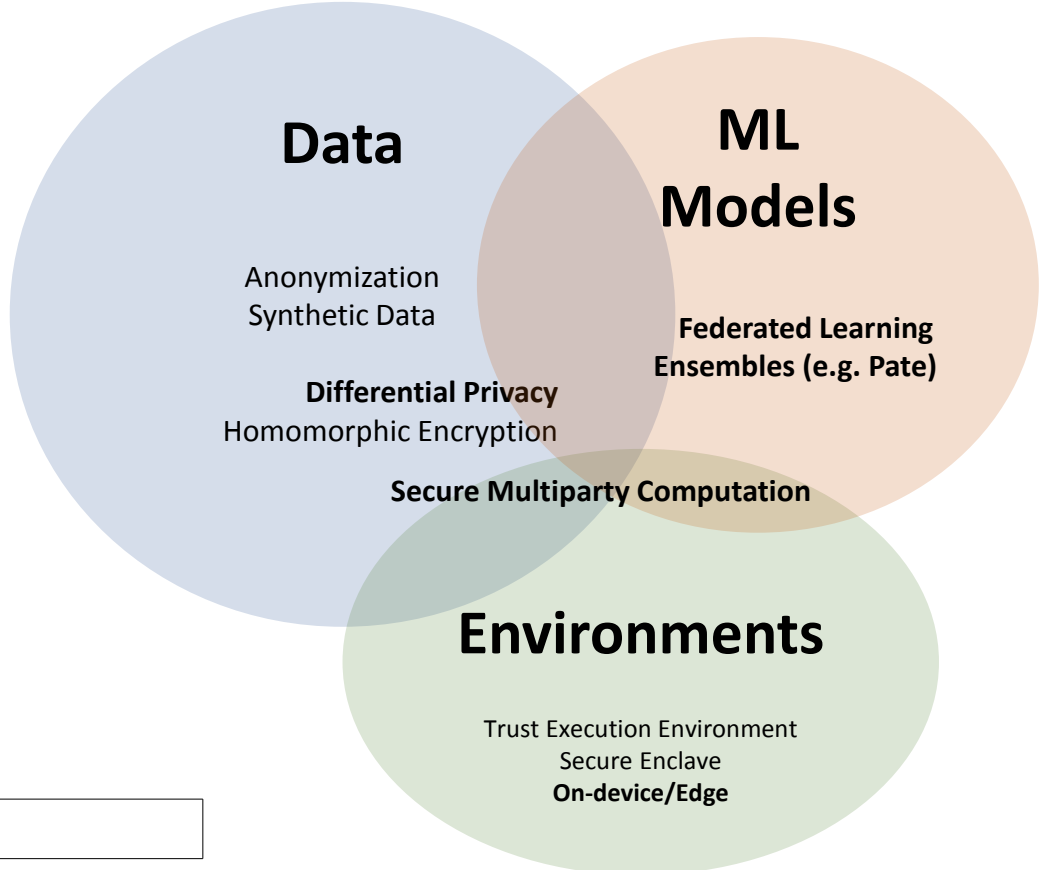
…

Can you learn from data that you don’t own, and can’t see?

Trade-offs



Privacy Preserving Techniques

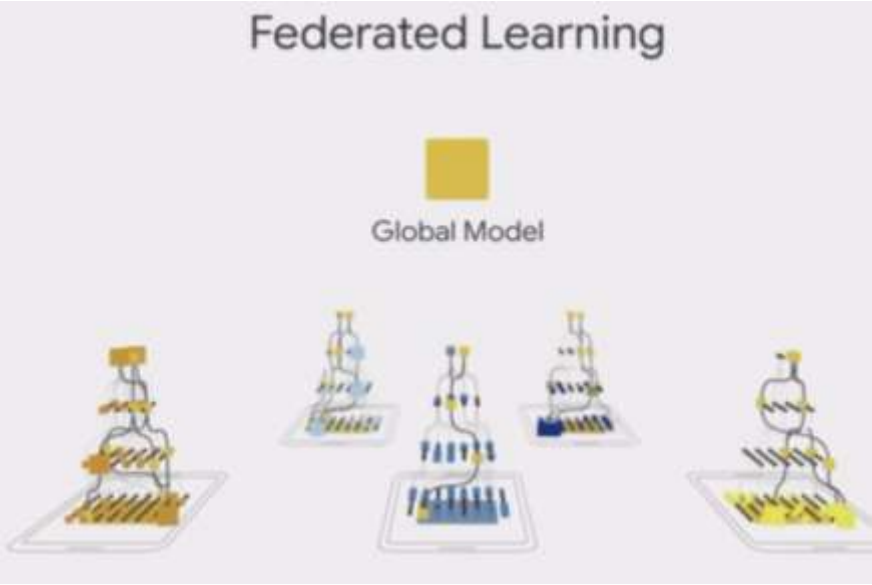


Differential Privacy

NIST on What is Differential Privacy

Differential Privacy Simply Explained

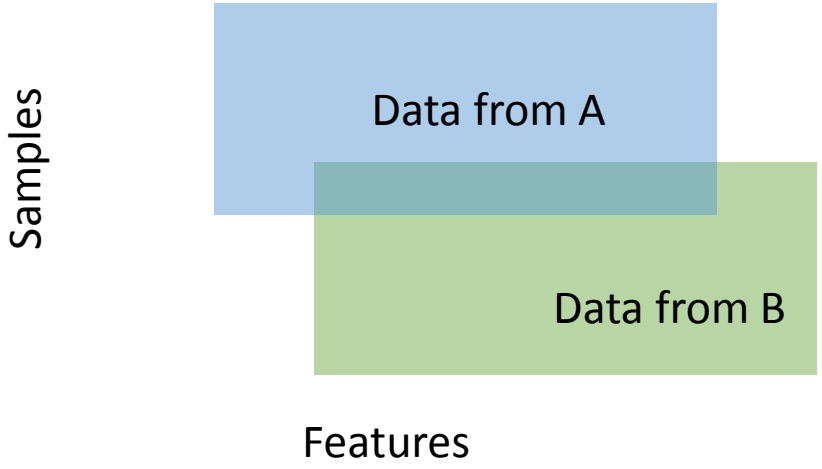
Federated Learning



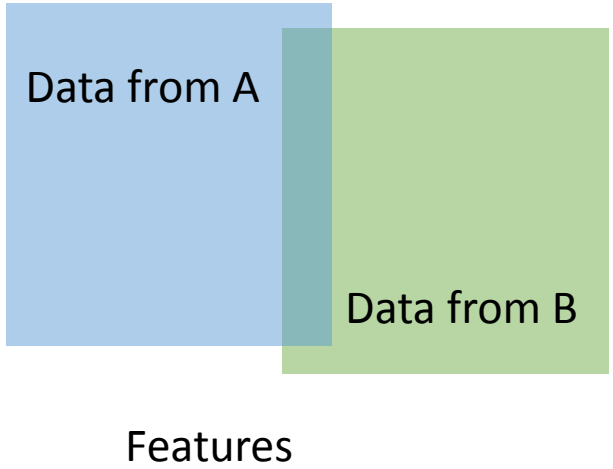
Google CEO Video

Federated Learning

Google CEO on Privacy and Federated Learning



Horizontal

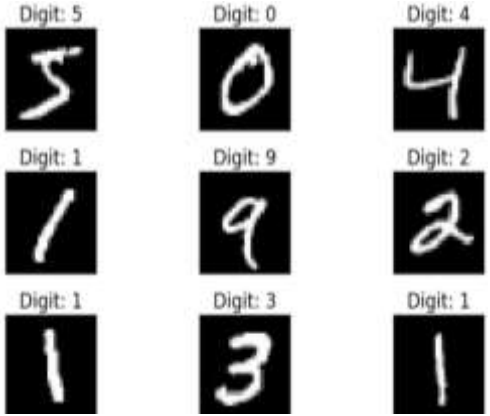


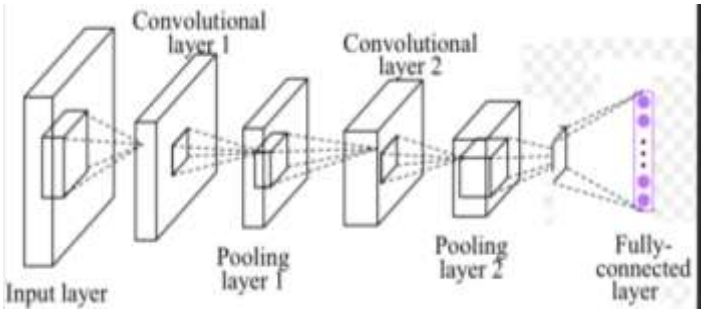
Vertical

Federated Learning Frameworks

- TensorFlow Federated (Google) - in production
- Pysyft (Openmined)
- FATE (Webank, China)
- Clara Federated Learning (NVIDIA)

MNIST Code Example with Federated Learning





...

Classification Result

0
1
2
3
4
5
6
7
8
9

Imports and model specifications

First we make the official imports

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms
```

And than those specific to PySyft. In particular we define remote workers `alice` and bob.

```
import syft as sy  # <-- NEW: import the Pysyft library
hook = sy.TorchHook(torch)  # <-- NEW: hook PyTorch ie add extra functionalities to support Federated Lear
ning
bob = sy.VirtualWorker(hook, id="bob")  # <-- NEW: define remote worker bob
alice = sy.VirtualWorker(hook, id="alice")  # <-- NEW: and alice
```

We define the setting of the learning task

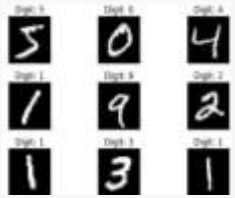
No Change

```
class Arguments():
    def __init__(self):
        self.batch_size = 64
        self.test_batch_size = 1000
        self.epochs = epochs
        self.lr = 0.01
        self.momentum = 0.5
        self.no_cuda = False
        self.seed = 1
        self.log_interval = 30
        self.save_model = False

args = Arguments()

use_cuda = not args.no_cuda and torch.cuda.is_available()

torch.manual_seed(args.seed)

device = torch.device("cuda" if use_cuda else "cpu")

kwargs = {'num_workers': 1, 'pin_memory': True} if use_cuda else {}
```

Data loading and sending to workers

We first load the data and transform the training Dataset into a Federated Dataset split across the workers using the `.federate` method. This federated dataset is now given to a Federated DataLoader. The test dataset remains unchanged.

```
federated_train_loader = sy.FederatedDataLoader( # <-- this is now a FederatedDataLoader
    datasets.MNIST('../data', train=True, download=True,
                   transform=transforms.Compose([
                       transforms.ToTensor(),
                       transforms.Normalize((0.1307,), (0.3081,))
                   ]))
    .federate((bob, alice)), # <-- NEW: we distribute the dataset across all the workers, it's now a Federatedataset
    batch_size=args.batch_size, shuffle=True, **kwargs)

test_loader = torch.utils.data.DataLoader(
    datasets.MNIST('../data', train=False, transform=transforms.Compose([
                       transforms.ToTensor(),
                       transforms.Normalize((0.1307,), (0.3081,))
                   ])),
    batch_size=args.test_batch_size, shuffle=True, **kwargs)
```



Set Hyperparamters: No Change

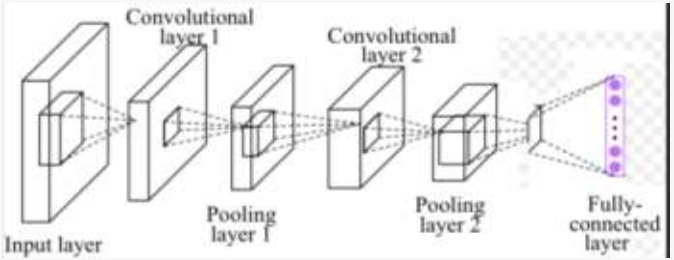
In [ ]:
```
class Arguments():
    def __init__(self):
        self.batch_size = 64
        self.test_batch_size = 1000
        self.epochs = epochs
        self.lr = 0.01
        self.momentum = 0.5
        self.no_cuda = False
        self.seed = 1
        self.log_interval = 30
        self.save_model = False

args = Arguments()

use_cuda = not args.no_cuda and torch.cuda.is_available()

torch.manual_seed(args.seed)

device = torch.device("cuda" if use_cuda else "cpu")

kwargs = {'num_workers': 1, 'pin_memory': True} if use_cuda else {}
```

Construct Neural Net: No Change

CNN specification

Here we use exactly the same CNN as in the official example.

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 20, 5, 1)
        self.conv2 = nn.Conv2d(20, 50, 5, 1)
        self.fc1 = nn.Linear(4*4*50, 500)
        self.fc2 = nn.Linear(500, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x, 2, 2)
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, 2, 2)
        x = x.view(-1, 4*4*50)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)
```



...

Define the train and test functions

For the train function, because the data batches are distributed across `alice` and `bob`, you need to send the model to the right location for each batch. Then, you perform all the operations remotely with the same syntax like you're doing local PyTorch. When you're done, you get back the model updated and the loss to look for improvement.

```
def train(args, model, device, federated_train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(federated_train_loader): # <-- now it is a distributed data
set
        model.send(data.location) # <-- NEW: send the model to the right location
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        model.get() # <-- NEW: get the model back
        if batch_idx % args.log_interval == 0:
            loss = loss.get() # <-- NEW: get the loss back
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * args.batch_size, len(federated_train_loader) * args.batch_size,
                100. * batch_idx / len(federated_train_loader), loss.item()))
```

The test function does not change!

No Change

```
def test(args, model, device, test_loader):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            test_loss += F.nll_loss(output, target, reduction='sum').item() # sum up batch loss
            pred = output.argmax(1, keepdim=True) # get the index of the max log-probability
            correct += pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(test_loader.dataset)

    print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
        test_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))
```

Launch the training !

```
%%time
model = Net().to(device)
optimizer = optim.SGD(model.parameters(), lr=args.lr) # TODO momentum is not supported at the moment

for epoch in range(1, args.epochs + 1):
    train(args, model, device, federated_train_loader, optimizer, epoch)
    test(args, model, device, test_loader)

if (args.save_model):
    torch.save(model.state_dict(), "mnist_cnn.pt")
```

Et voilà! Here you are, you have trained a model on remote data using Federated Learning!

Federated Learning with Pysyft Summary

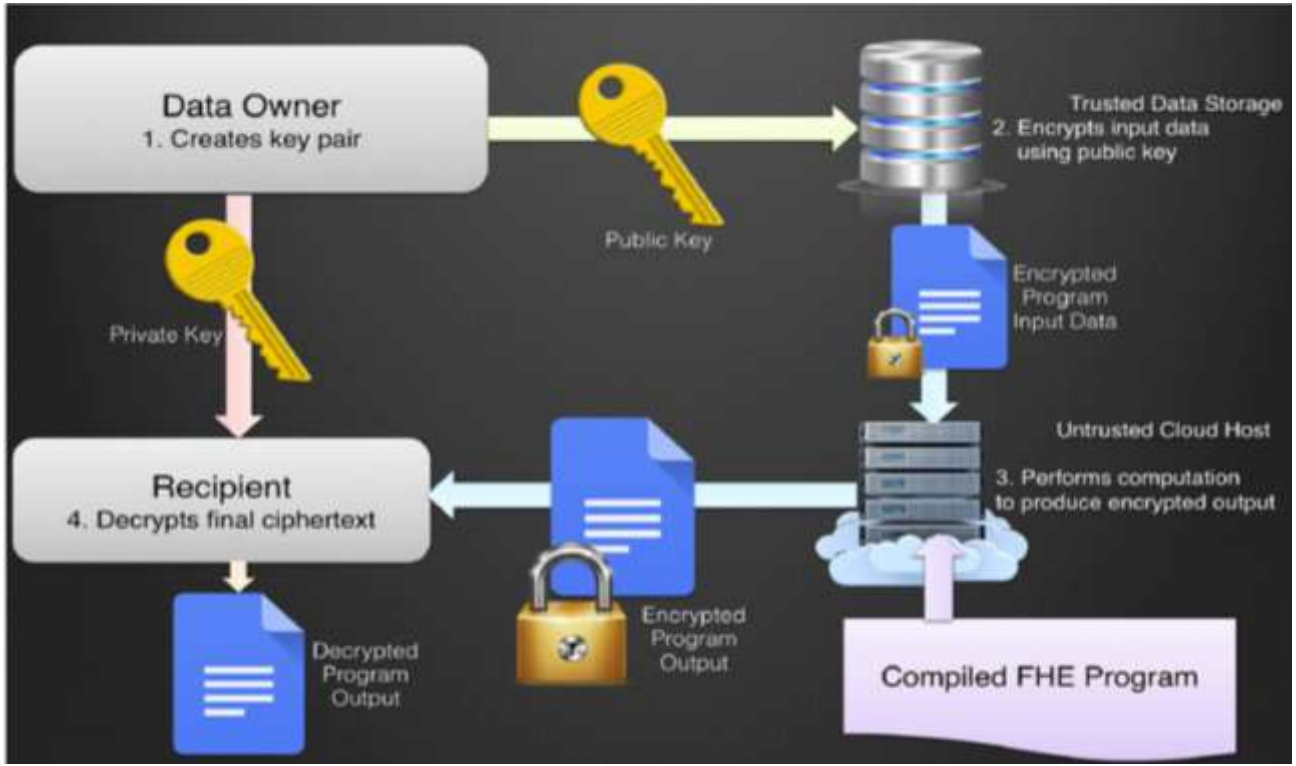
- 10 lines of code changes
- Twice the compute time over non-federated
- More features/optimization needed to be production ready

Differential Privacy, Transfer Learning: PATE

Private Aggregation of Teacher Ensembles

- Train Teacher models with individual private datasets
- Train a public Student with a public dataset
  - Teachers make inferences and "vote" on a given sample
  - The voted result is used to train a student model which is public.
- Synergy between privacy and accuracy

Homomorphic Encryption: Compute on Encrypted Data



[California’s new privacy law creates $55-billion gold rush for start-ups]

Resources

The End of Privacy: Data Scientists Know All your Secrets

Frameworks and Code

OpenMined
TensorFLow Federated
TensorFlow Privacy
FATE
Udacity Secure and Private AI class

General Deep Learning Course:
Fast.ai
Deeplearning.ai

Privacy Preserving Deep Learning Techniques

1. Homomorphic Encryption based
    - Training on encrypted data
    - Encrypt the model.
    - Both the privacy of the data owners and the IP of model owners are protected.
2. Secure Multi-party Computation based
3. Differential Privacy based
4. Hybrid

Secure MPC-based Deep Learning Techniques

Scenario	Proposed schemes	DL technique	Accuracy (%)	Run time (s)	Data transfer (Mbytes)	PoC	PoM
Cloud Service	DeepSecure [40]	CNN	Good (98.95)	Bad (10,649)	Bad (722,000)	No	Yes
Image Recogni-tion	SecureML [36]	DNN	Bad (93.40)	–	–	No	Yes
PaaS	MiniONN [42]	NN	Good (98.95)	Good (1.04)	Good (47.60)	No	Yes
	ABY3 [46]	NN	Bad (94.00)	Good (0.01)	Good (5.20)	No	Yes

HE-based Deep Learning Techniques

Scenario	Proposed schemes	DL technique	Accuracy (%)	Run time (s)	Data transfer (Mbytes)	PoC	PoM
Cloud Service	ML Confidential [31]	DNN	Bad (95.00)	Bad (255.7)	–	Yes	No
	Cryptonets [34]	CNN	Good (98.95)	Bad (697)	Bad (595.5)	Yes	No
	PP on DNN [35]	CNN	Good (99.30)	–	–	Yes	No
	E2DM [47]	CNN	Good (98.10)	Good (28.59)	Good (17.48)	Yes	Yes
	PPDL via Additively HE [48]	CNN	Good 97.00	Good (120)	–	Yes	Yes
Image Recogni-tion	CryptoDL [29]	CNN	Good (99.52)	Bad (320)	Bad (336.7)	Yes	No
	PP All Convolutional Net [30]	CNN	Good (98.97)	Bad (477.6)	Bad (361.6)	Yes	No
Content Sharing	Distributed PP Multi-Key FHE [39]	CNN	Good (99.73)	–	–	Yes	No
PaaS	Gazelle [43]	CNN	–	Good (0.03)	Good (0.5)	Yes	Yes
	Tapas [44]	BNN	Good (98.60)	Good (147)	–	Yes	Yes
	FHE-DNN [45]	DiNN	Bad (96.35)	Good (1.64)	–	Yes	Yes