ML Exercise 3

Secure Multi-party computation for Image Classification

Group 8:
Alexander Leitner, 01525882
Mario Hiti, 01327428
Peter Holzner, 01426733

Outline

- Introduction
  - Overview
  - Task
- Frameworks
  - What is (Secure) MPC?
  - MPyC
  - CrypTen
- Models
  - CNN
  - MLP (ReLU vs Sign activation)
  - Fixed-Point Encoding
- (Datasets)
  - MNIST
  - Fashion MNIST
- Benchmarks
- Summary
  - Experiences with frameworks
  - Secure MPC for Image Classification - usable?
  - Verdict

Overview

Task

Evaluate Secure MPC for Image Classification

Utilising e.g. the library mpyc for Python (https://github.com/lschoe/mpyc) and the implementation of a secure computation for a binarised multi-layer perceptron, first try to recreate the results reported in Abspoel et al, “Fast Secure Comparison for Medium-Sized Integers and Its Application in Binarized Neural Networks”, i.e. train a baseline CNN to estimate a potential upper limit of achievable results, and then train the binarized network, as a simplified but still rather performant version, in a secure way. If needed, you can use a subset of the MNIST dataset.

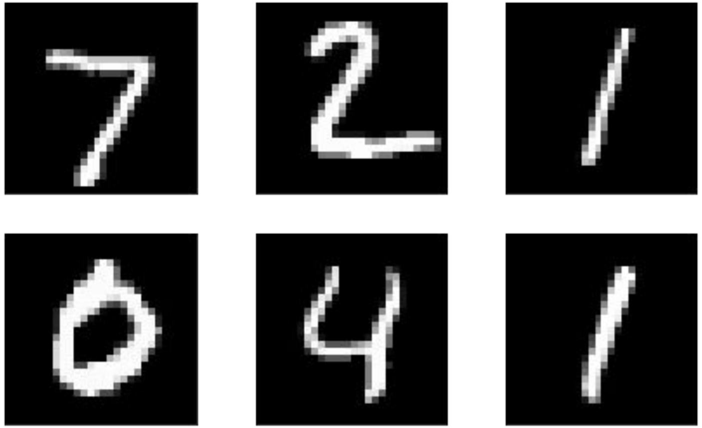
Then, try to perform a similar evaluation on another small dataset, either already available in grayscale, or converted to grayscale, e.g. using (a subset of) the AT&T faces dataset. Specifically, evaluate the final result in terms of effectiveness, but also consider efficiency aspects, i.e. primarily runtime, but also other resource consumption.

Datasets

- MNIST
- MNIST_Fashion

Classification Models

- CNN
- MLP (with Sign and ReLU activation)



60000 + 10000 images,

grayscale,

28x28

Language: Python >= 3.7 (3.7.9 recommended)

OS: Linux or MacOS

Packages/Frameworks: mpyc0.7, crypten0.1, torch1.4.0, (see requirements.txt)

Frameworks

Secure MPC

10 + 30 = ?

Without telling the others what my number is...

10

A

30

B

C

Secure MPC

10 + 30 = ?

Without telling the others what my number is...

10

A

3

30

B

5

C

Secure MPC

***10* + *30* = ?**

Without telling the others what my number is...

10

A

3 + 6

30

B

2 + 8

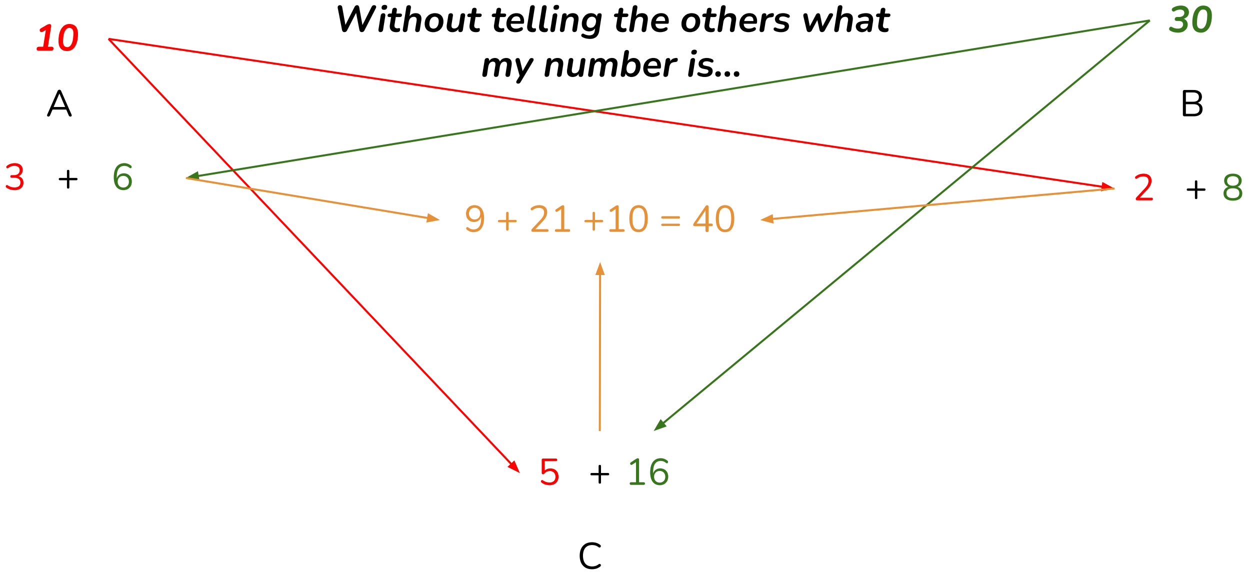
5 + 16

C

Secure MPC

***10* + *30* = ?**

Without telling the others what my number is…



Secure MPC

***10* + *30* = ?**

Without telling the others what my number is...

10

A

3 + 6

30

B

2 + 8

9 + 21 +10 = 40

More complex in practice.

Different operations require different encoding/secret sharing.

Slow + additional resources required

5 + 16

C

MPyC

- Developed by Abspoel et. al. at Centrum Wiskunde & Informatica (CWI), Amsterdam
- Pros:
  - simple
  - small (60MB on Github)
  - no third party dependencies
- Cons
  - offers only a set of mathematical functions
  - optimizers, data structures, evaluation have to be coded by hand

CrypTen

- Privacy Preserving ML via Secure MPC
- assumes a semi-honest threat model
- Developed by Facebook
- Pros:
  - uses Torch as a Back-End
  - easy syntax, fast development of models
  - efficient
- Cons:
  - developed by Facebook
  - lots of dependencies (complete env: 1.6GB)
  - still in early development
  - lots of Bugs 😡

CrypTen

CrypTen is a Privacy Preserving Machine Learning framework written using PyTorch

```python
import torch.nn as nn

PIXEL_CNT = 28

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(PIXEL_CNT, PIXEL_CNT)
        self.fc2 = nn.Linear(PIXEL_CNT, PIXEL_CNT)
        self.fc3 = nn.Linear(PIXEL_CNT, PIXEL_CNT)
        self.fc4 = nn.Linear(PIXEL_CNT, PIXEL_CNT)
        self.fc5 = nn.Linear(PIXEL_CNT, 10)

    def forward(self, x):
        x = x.view(-1, PIXEL_CNT)
        x = self.fc1(x)
        x = x.relu()
        x = self.fc2(x)
        x = x.relu()
        x = self.fc3(x)
        x = x.relu()
        x = self.fc4(x)
        x = x.relu()
        x = self.fc5(x)
        return x
```

Secure MPC

```python
import crypten.nn as nn

PIXEL_CNT = 28

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(PIXEL_CNT, PIXEL_CNT)
        self.fc2 = nn.Linear(PIXEL_CNT, PIXEL_CNT)
        self.fc3 = nn.Linear(PIXEL_CNT, PIXEL_CNT)
        self.fc4 = nn.Linear(PIXEL_CNT, PIXEL_CNT)
        self.fc5 = nn.Linear(PIXEL_CNT, 10)

    def forward(self, x):
        x = x.view(-1, PIXEL_CNT)
        x = self.fc1(x)
        x = x.relu()
        x = self.fc2(x)
        x = x.relu()
        x = self.fc3(x)
        x = x.relu()
        x = self.fc4(x)
        x = x.relu()
        x = self.fc5(x)
        return x
```

CrypTen

```
x = crypten.cryptensor([1, 2, 3])
x
```

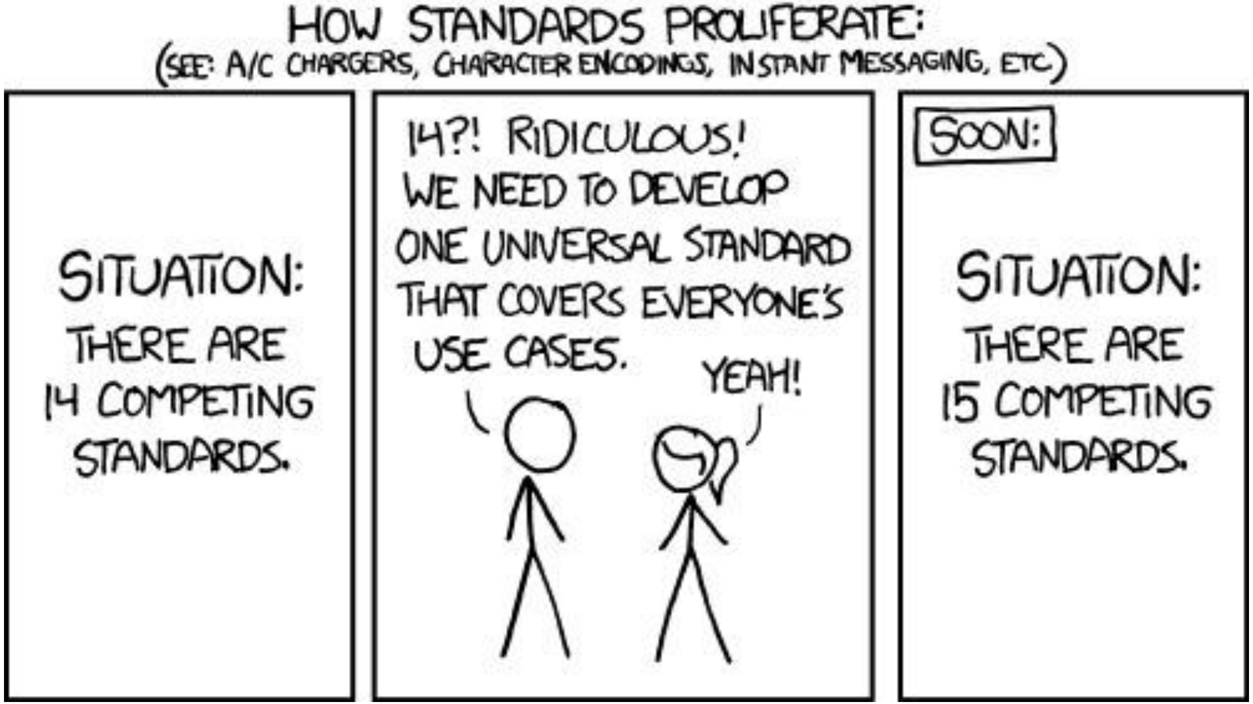
```
MPCTensor(
        _tensor=tensor([ 65536, 131072, 196608])
        plain_text=HIDDEN
        ptype=ptype.arithmetic
)
```

```
# Make it readable
x.get_plain_text()
```

```
tensor([1., 2., 3.])
```

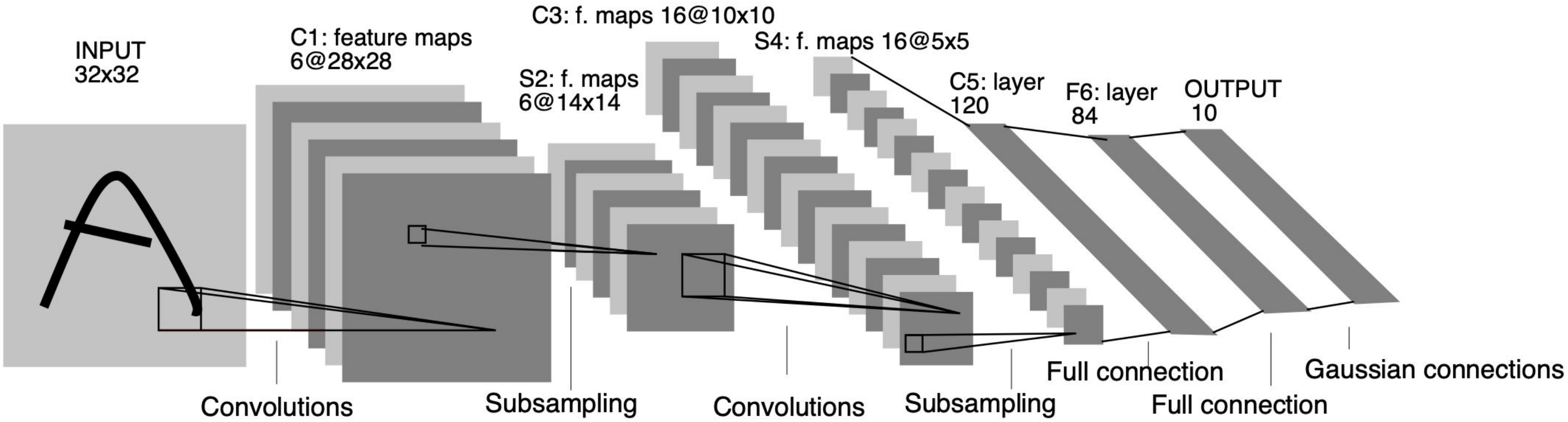
Other Frameworks

List of other privacy preserving ML frameworks:
https://awesomeopensource.com/project/rdragos/awesome-mpc



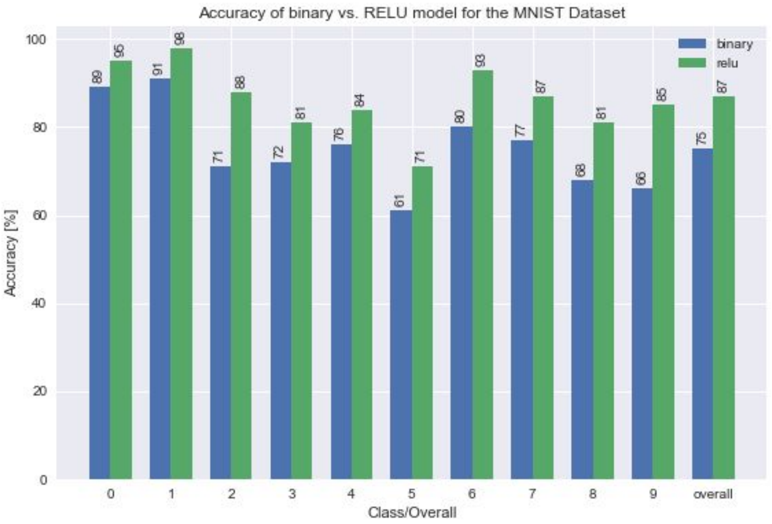
Models

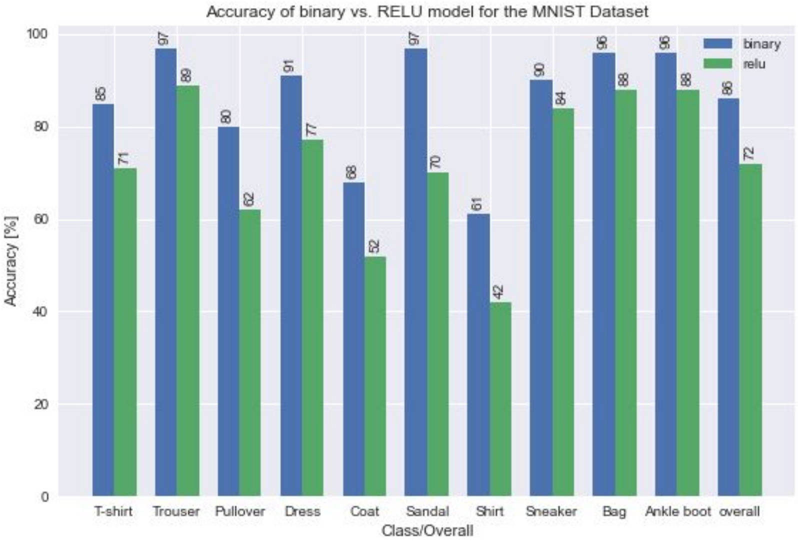
CNN *(Reference)*



- Convolution Layer (Conv2d) kernelsize, padding, stride
- Activation-function RELU
- Adam optimizer and adaptive learning rate
- max-pooling

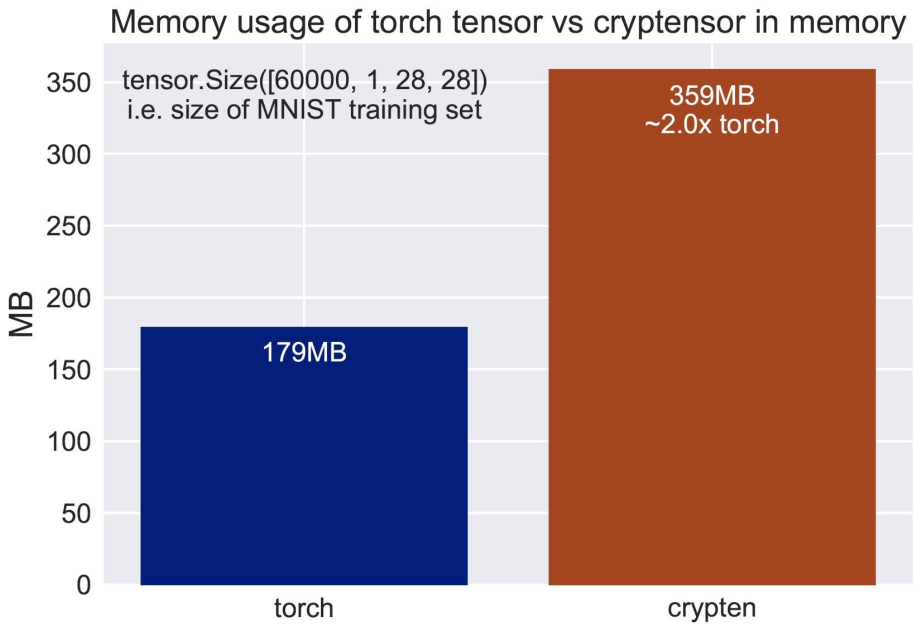
MLP - RELU vs. binary activation

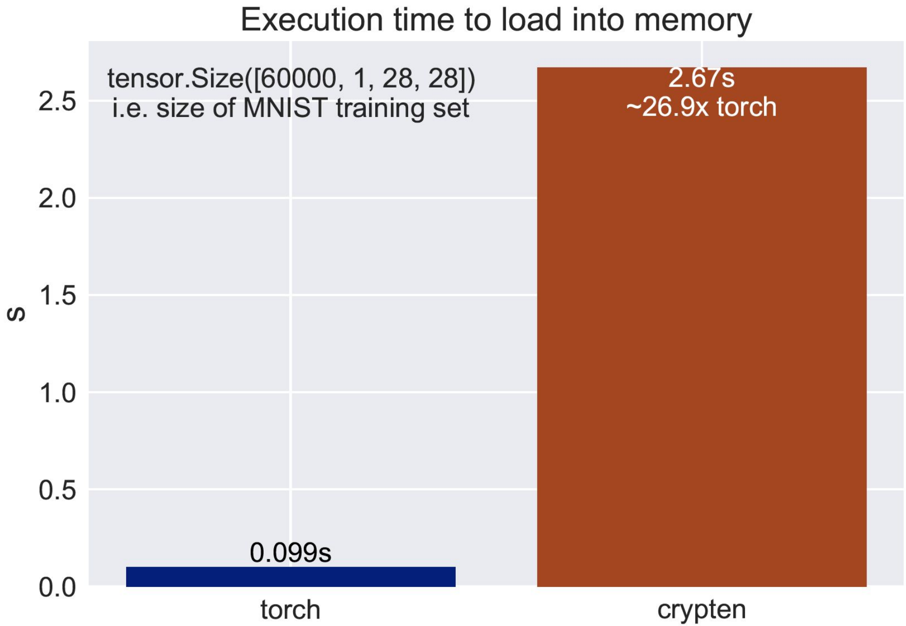




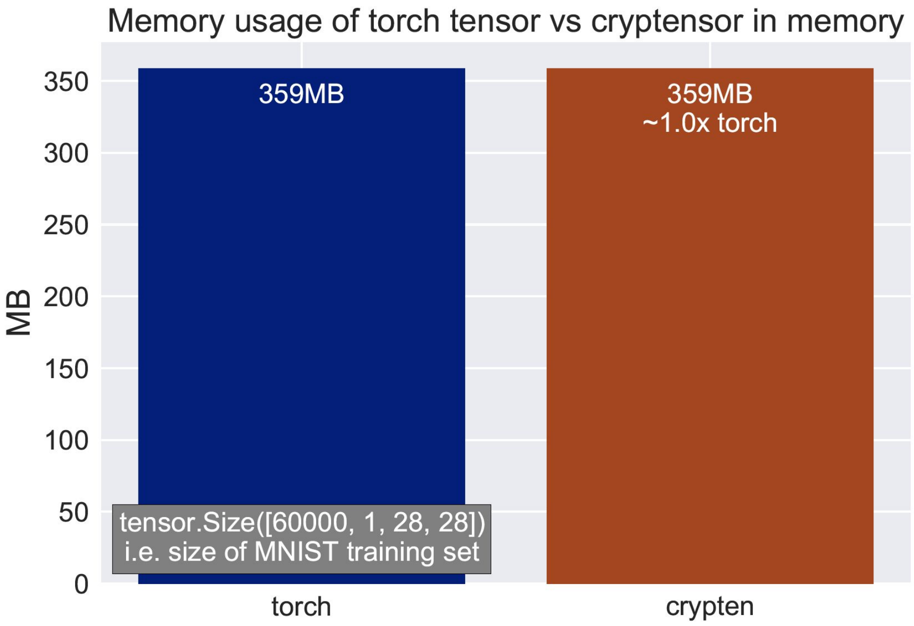
Benchmark

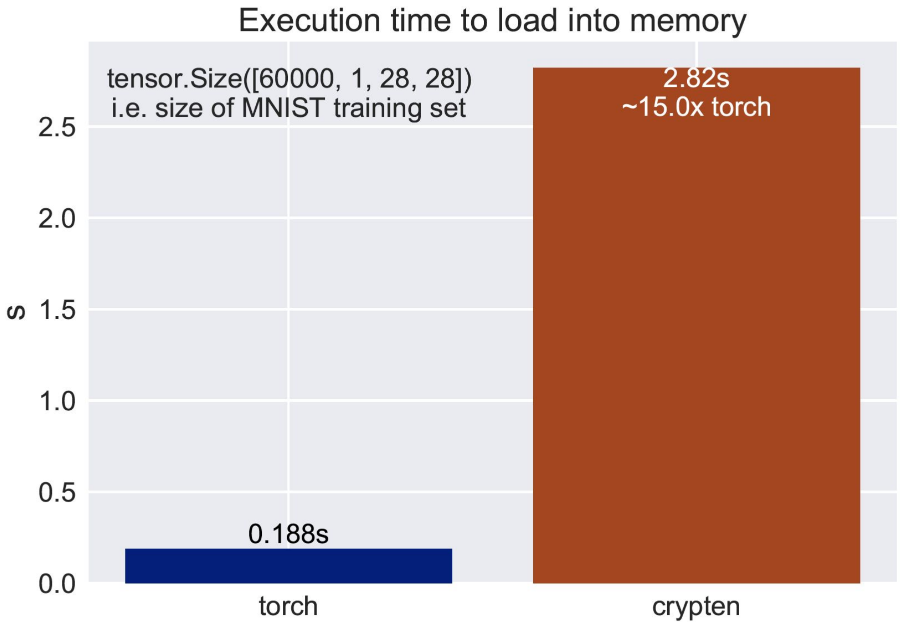
Tensor Benchmark: Torch vs CrypTen



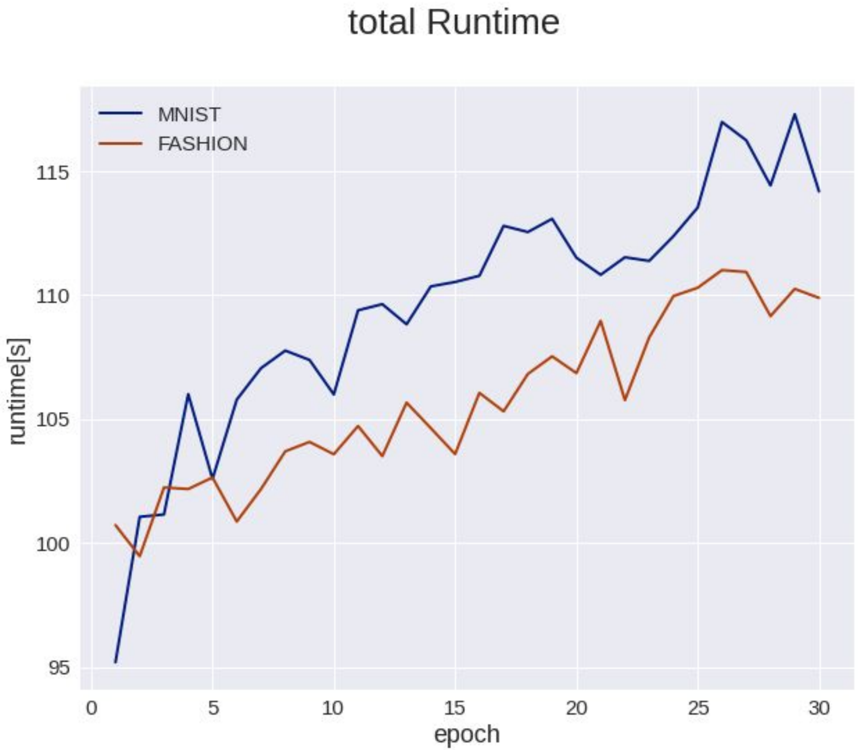


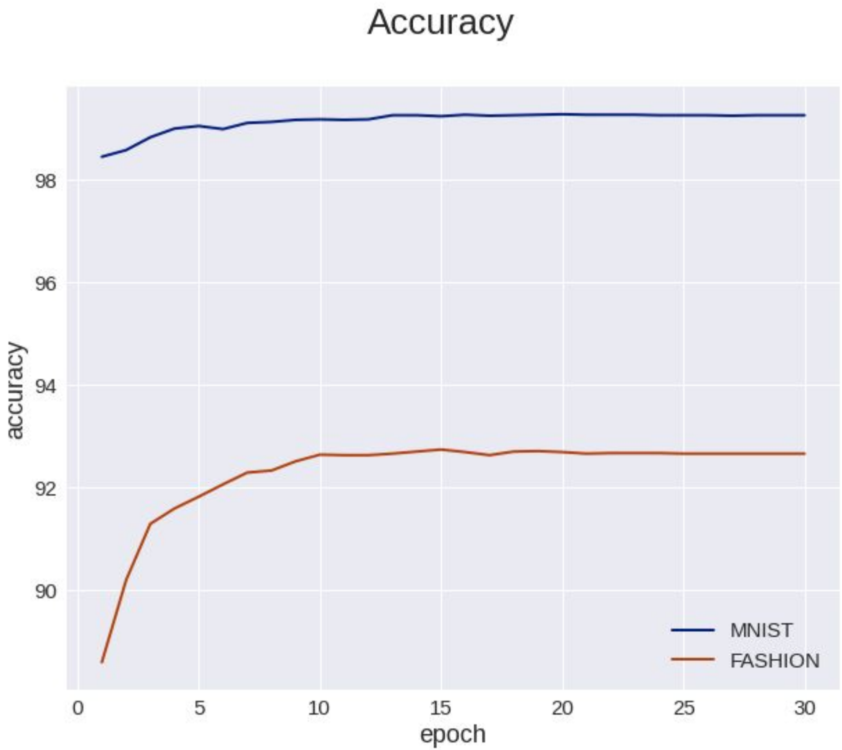
Tensor Benchmark: Torch vs CrypTen



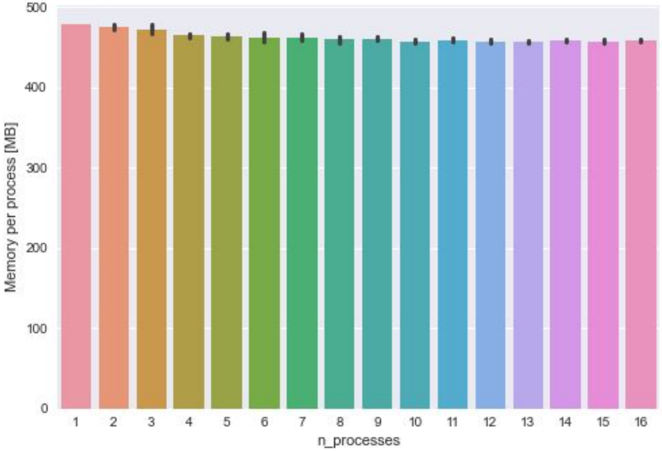


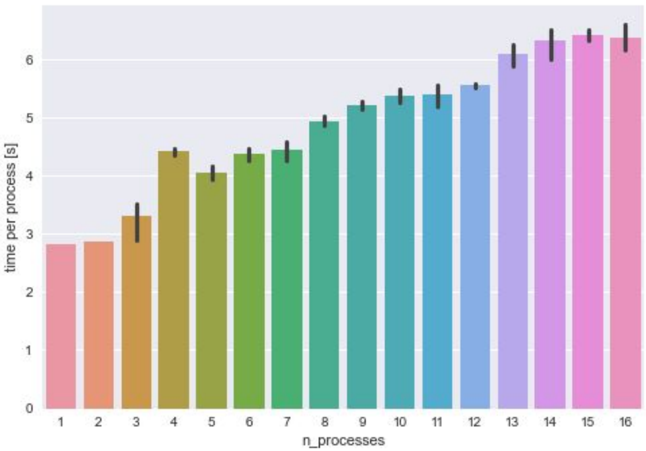
Benchmark CNN



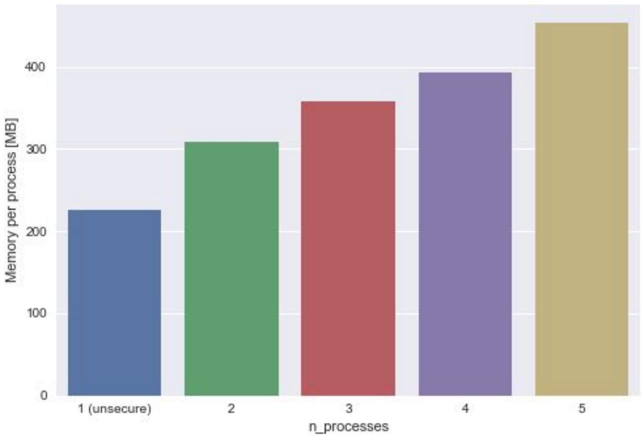


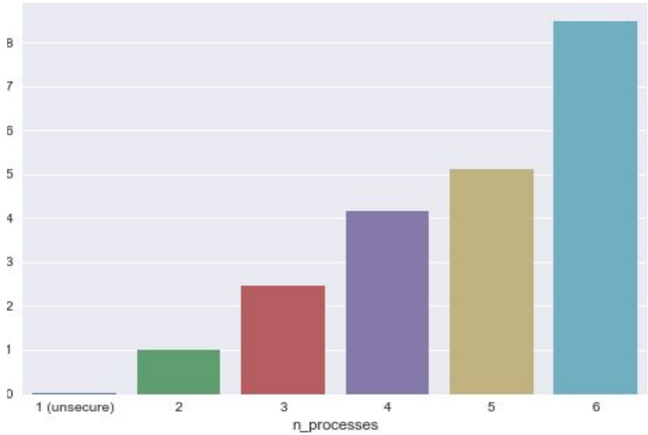
MPyC Framework - Evaluation



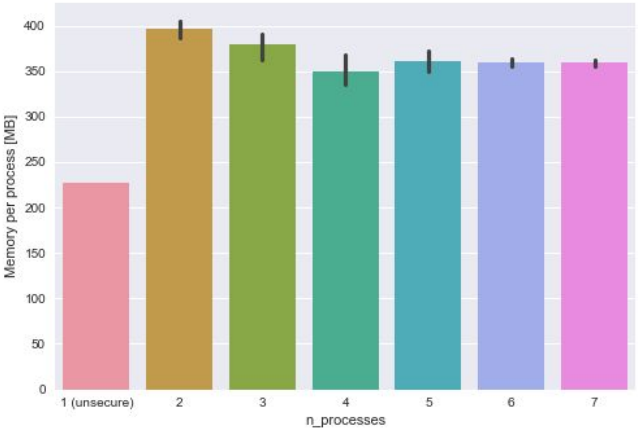


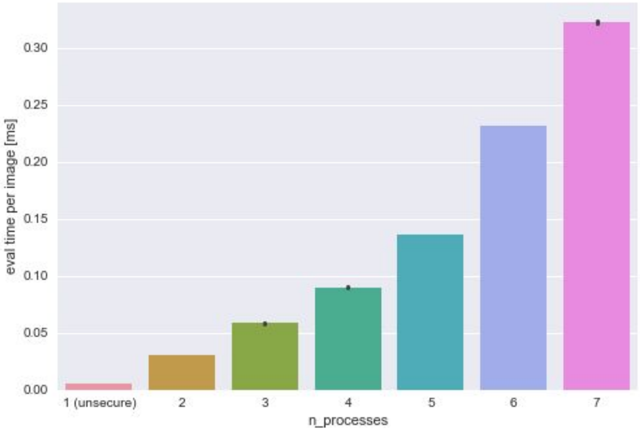
Crypten - Training





Crypten - Evaluation





Summary

Summary

CrypTen (v0.1)

- Secure training works (slow), but
- Can't save securely trained models (bug)*
- Many functions still missing*
- Relies on old PyTorch version*
- large (complete env: 1.6GB)
- dependencies (compiled code...)
- MPC setup across PCs still difficult
- fixed point encoding is fixed (64bit)
- Good prediction performance (33fps)!

MPyC (v0.7)

- good for research or learning
- no dependencies (outside std lib)
- small (complete env: 260MB)
- MPC setup across PCs still difficult
- variable fixed-point encoding
- slower (0.15 fps)*

**for now!*

Summary

Secure MPC for Image Classification

- Resource intensive:
  - Every party needs resources for the same program (PCs, RAM, ...)
  - Much slower execution - real time performance unlikely
  - Avoid secure MPC training if possible (pre-train on available clear data)
- How is data distributed?
  - One image split across multiple parties unlikely
  - Realistic: multiple parties hold multiple image-label-pairs
  - One party also wants to keep their model (know-how) private
- More complex models (e.g. object detection)
  - Untested by us
  - But yikes...