

Cryptography for  
**Privacy-Preserving Machine Learning**

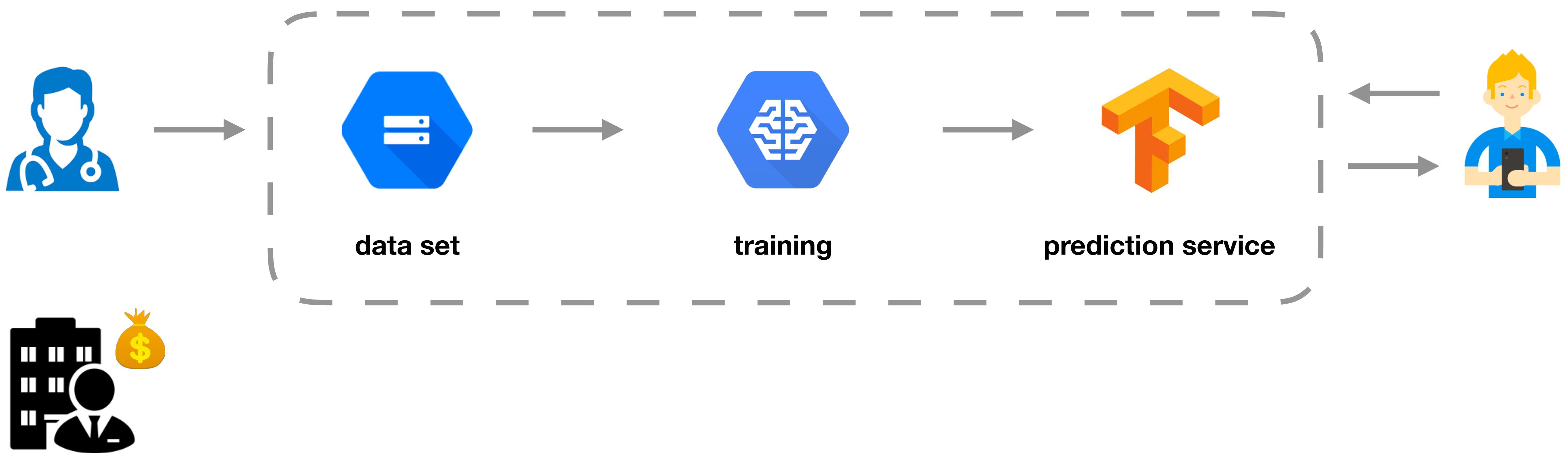
Morten Dahl

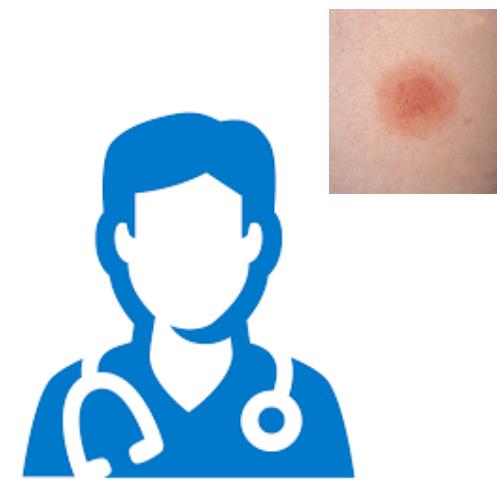
*Zero Knowledge TLV meetup, Tel Aviv, January 2019*

**Why?**

# Machine Learning Process

IMAGENET



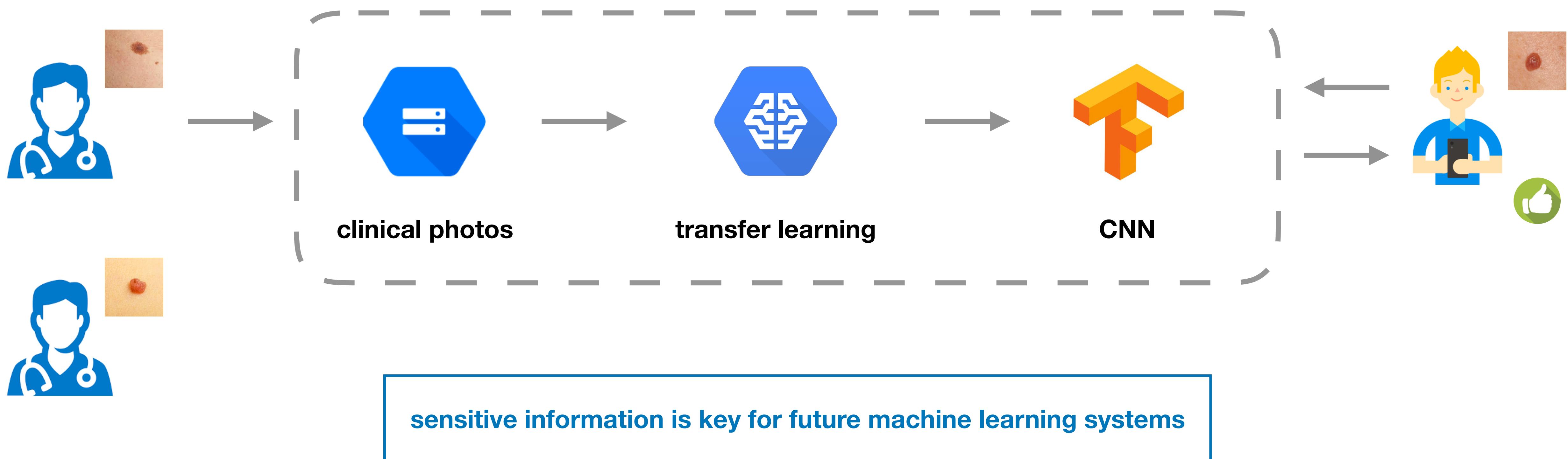


# Skin Cancer Image Classification

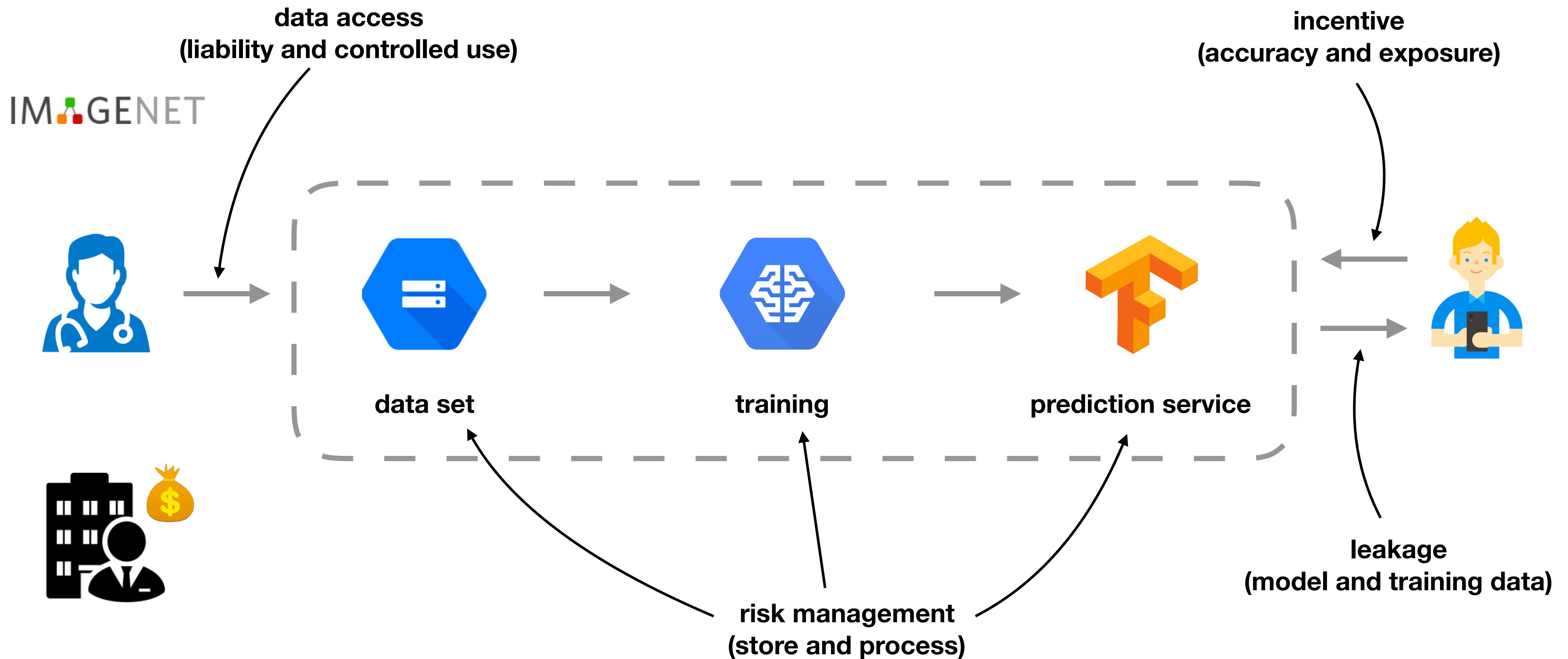
Brett Kuprel

12:30-12:40pm

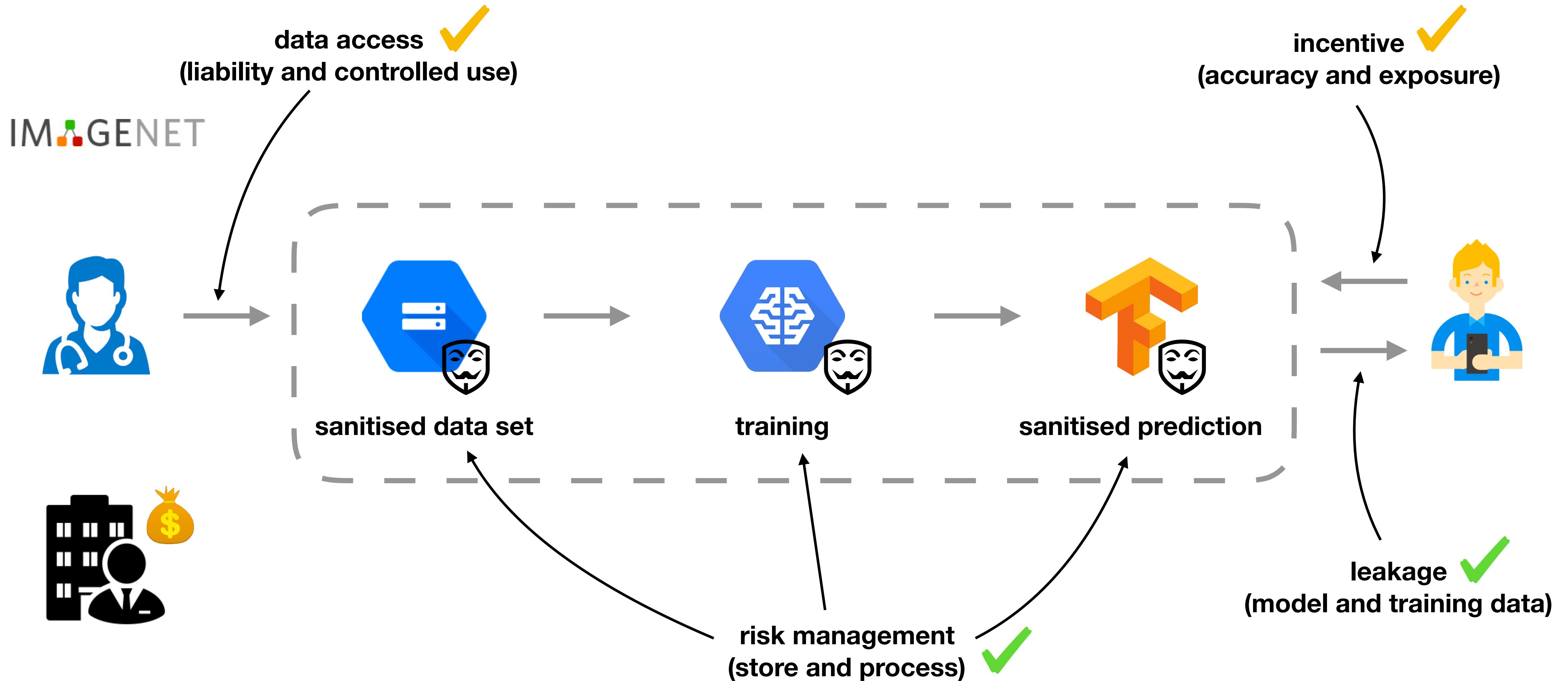
Join Brett Kuprel, and see how TensorFlow was used by the artificial intelligence lab and medical school of Stanford to classify skin cancer images. He'll describe the project steps: from acquiring a dataset, training a deep network, and evaluating of the results. To wrap up, Brett will give his take on the future of skin cancer image classification.



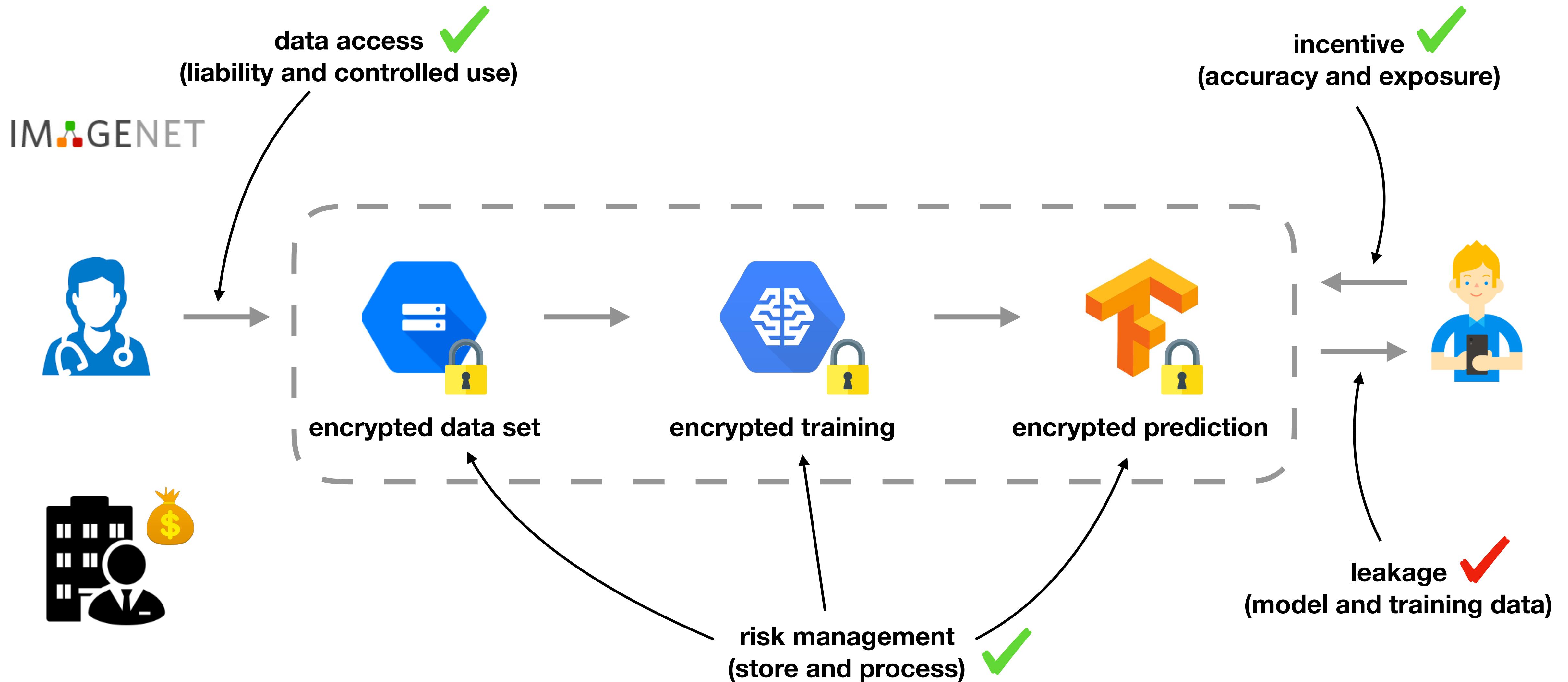
# Potential Bottlenecks



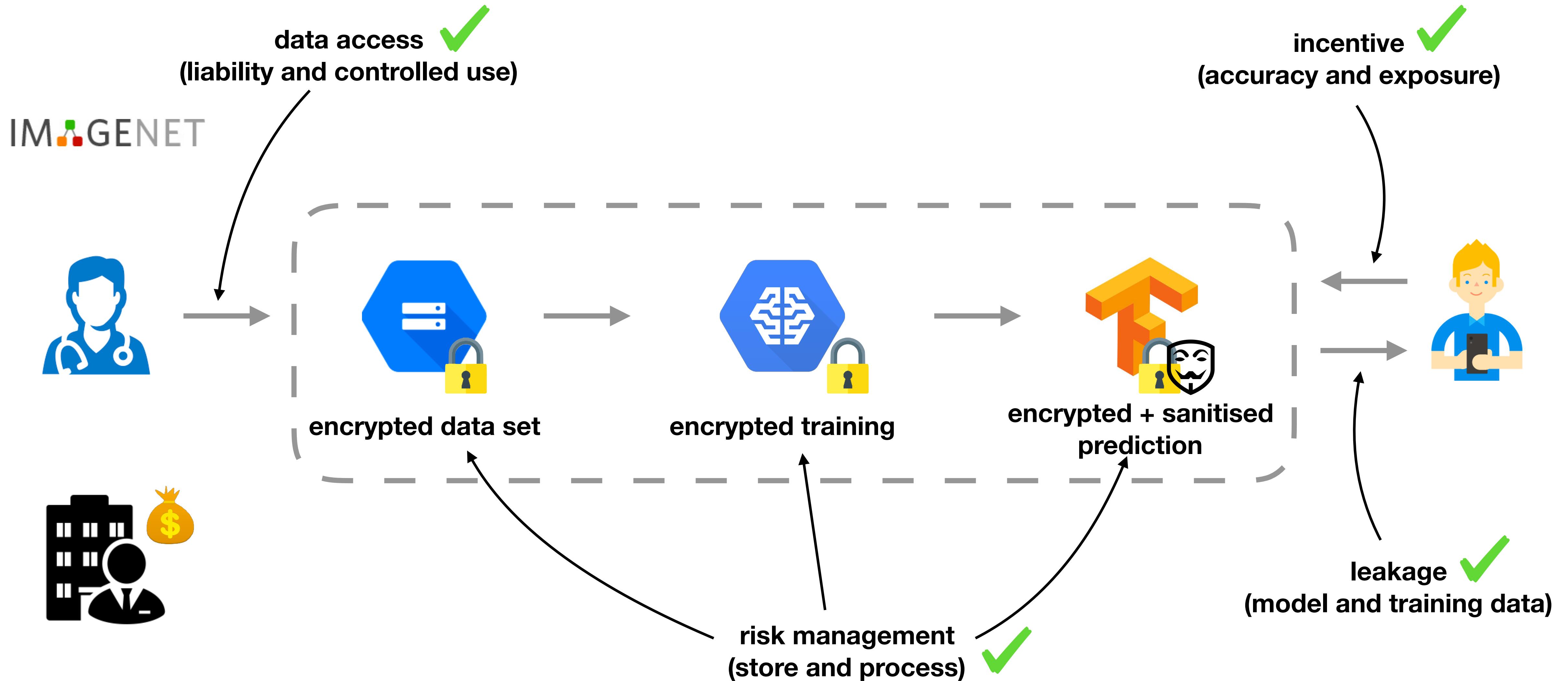
# Differential Privacy



# Secure Computation



# Hybrid



# Secure Computation

**fixed-point encoding**

$$\lfloor \pi * 10^4 \rfloor = 31415$$

## Homomorphic Encryption

Computation

*heavy*

Communication

*light*

Data types

*integers + bits*

Platform

*software*

## Secret Sharing

*light*

*heavy*

*integers + bits*

*software*

## Garbled Circuits

*medium*

*medium*

*bits*

*software*

## Secure Enclaves

*light*

*light*

*any*

*hardware*

CryptoNets'16



SecureML'17



DeepSecure'17



Gazelle'18



ABY3'18



SecureNN'18



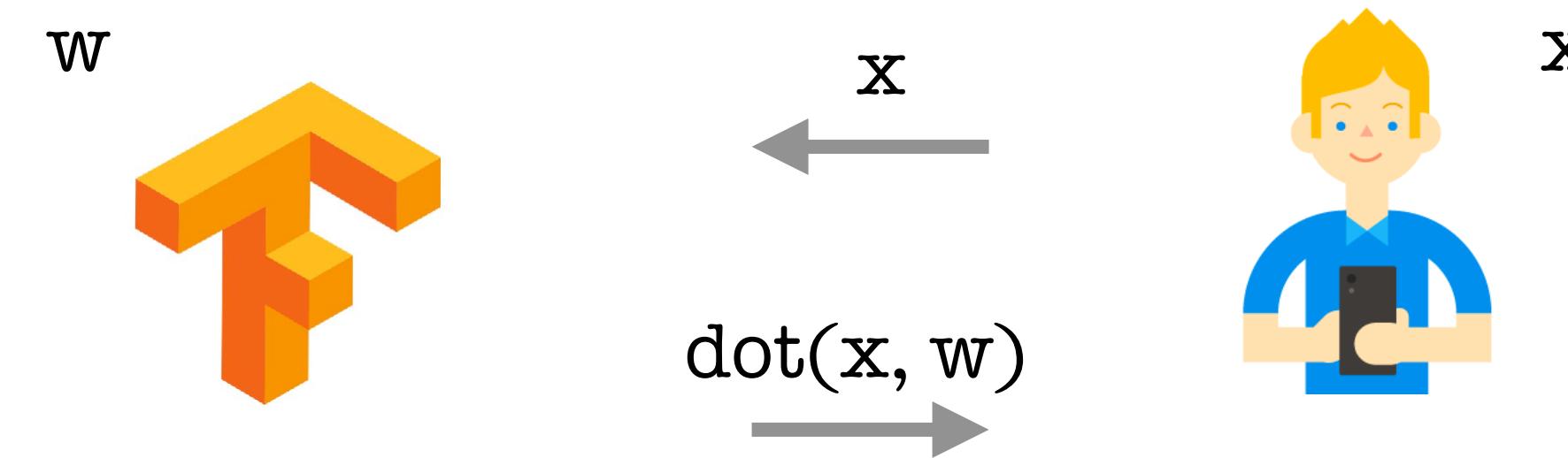
Slalom'18



tendency to mix techniques

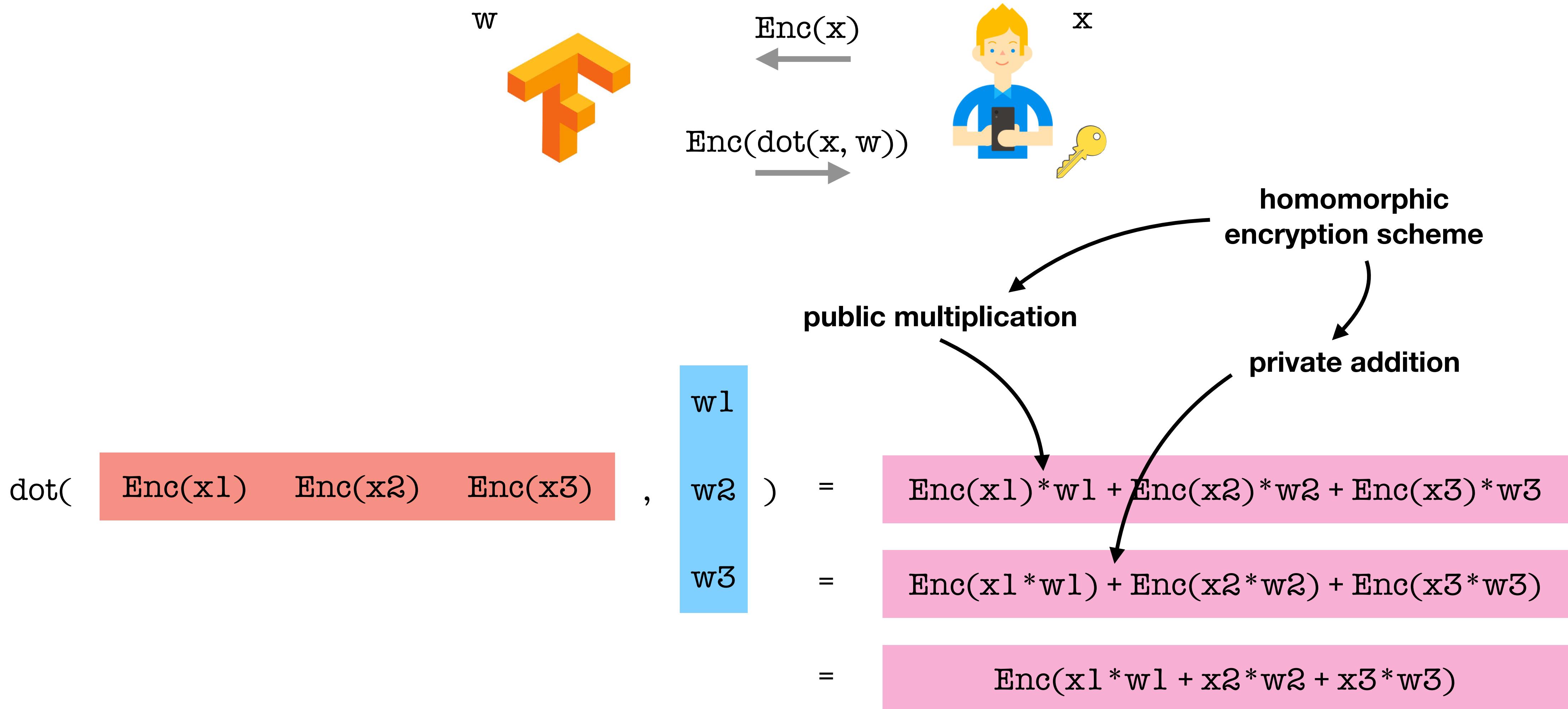
# Linear Regression

# Prediction with Linear Regression Model



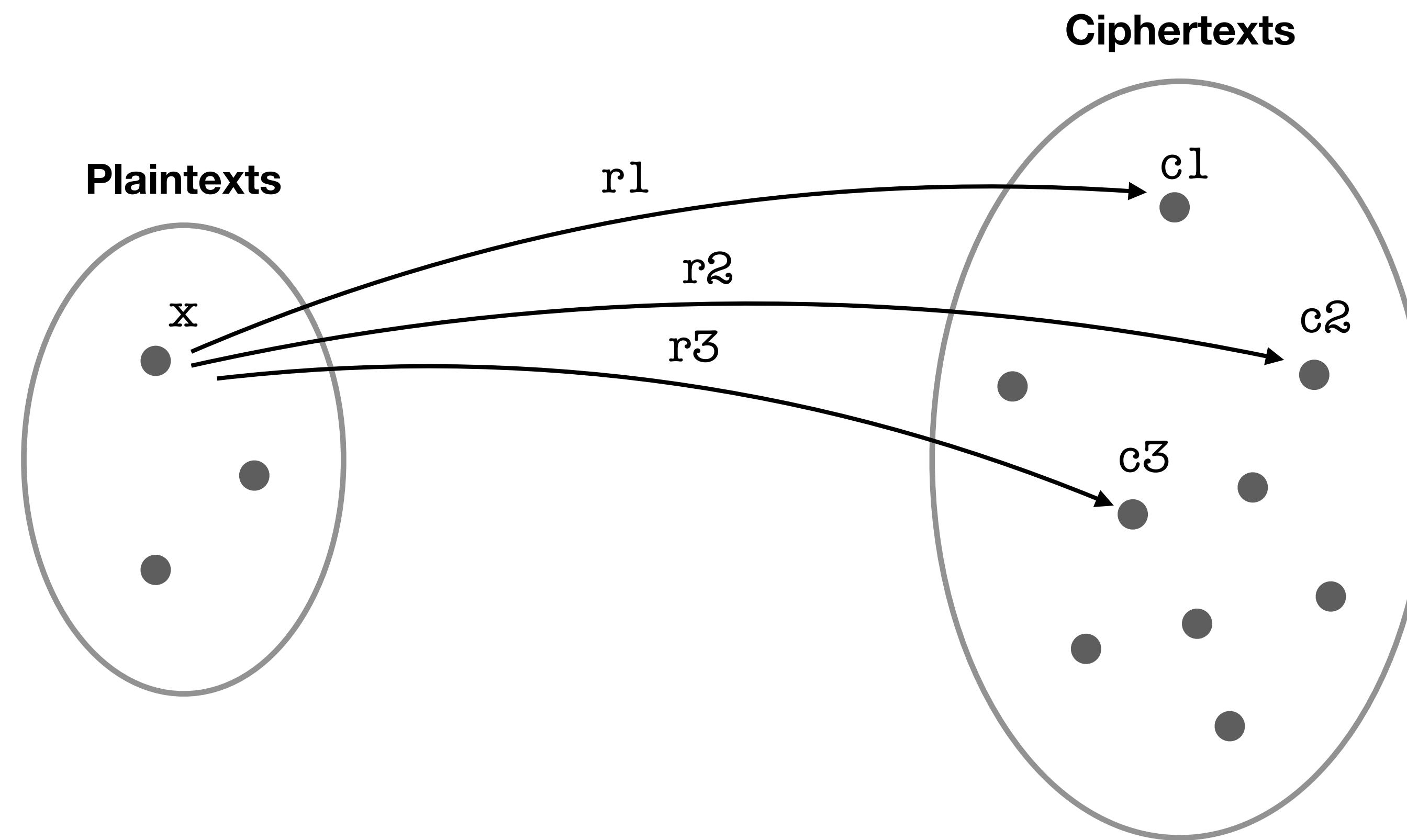
$$\text{dot}(\begin{matrix} x_1 & x_2 & x_3 \end{matrix}, \begin{matrix} w_1 \\ w_2 \\ w_3 \end{matrix}) = x_1 * w_1 + x_2 * w_2 + x_3 * w_3$$

# Using Homomorphic Encryption



# Probabilistic Encryption

$$c = \text{Enc}(x, r)$$



# Paillier Homomorphic Encryption

typically ~4000 bits:  
computation is significantly  
more expensive

$$c = \text{Enc}(x, r) = g^x * r^n \bmod n^2$$

public encryption key

public encryption key

$$\begin{aligned} g &= 36 \\ n &= 35 \\ n^2 &= 1225 \end{aligned}$$

$$\begin{aligned} \text{Enc}(5, 2) &= 36^5 * 2^{35} \bmod 1225 = 718 \\ \text{Enc}(5, 4) &= 36^5 * 4^{35} \bmod 1225 = 674 \end{aligned}$$

# Private Addition in Paillier

$$\begin{aligned} & \text{Enc}(\mathbf{x}, \mathbf{r}) * \text{Enc}(\mathbf{y}, \mathbf{s}) \\ &= (\mathbf{g}^{\mathbf{x}} * \mathbf{r}^n \bmod n^2) * (\mathbf{g}^{\mathbf{y}} * \mathbf{s}^n \bmod n^2) \\ &= \mathbf{g}^{\mathbf{x+y}} * (\mathbf{r} * \mathbf{s})^n \bmod n^2 \\ &= \text{Enc}(\mathbf{x+y}, \mathbf{r*s}) \end{aligned}$$

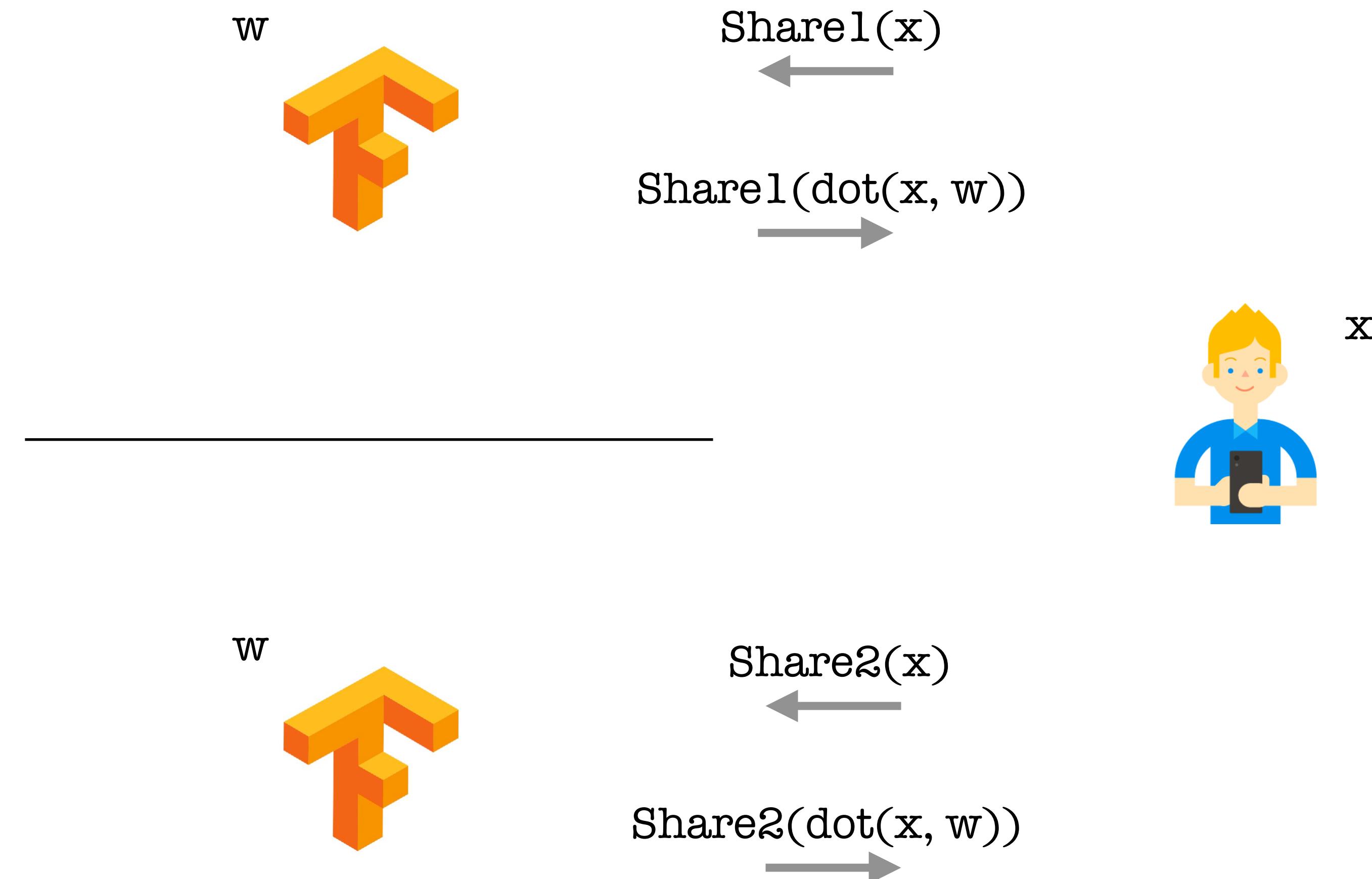
$$\begin{aligned} & \text{Enc}(5, 2) * \text{Enc}(5, 4) \\ &= 718 * 674 \\ &= 57 \\ &= 36^{10} * 8^{35} \\ &= \text{Enc}(10, 8) \end{aligned}$$

# Public Multiplication in Paillier

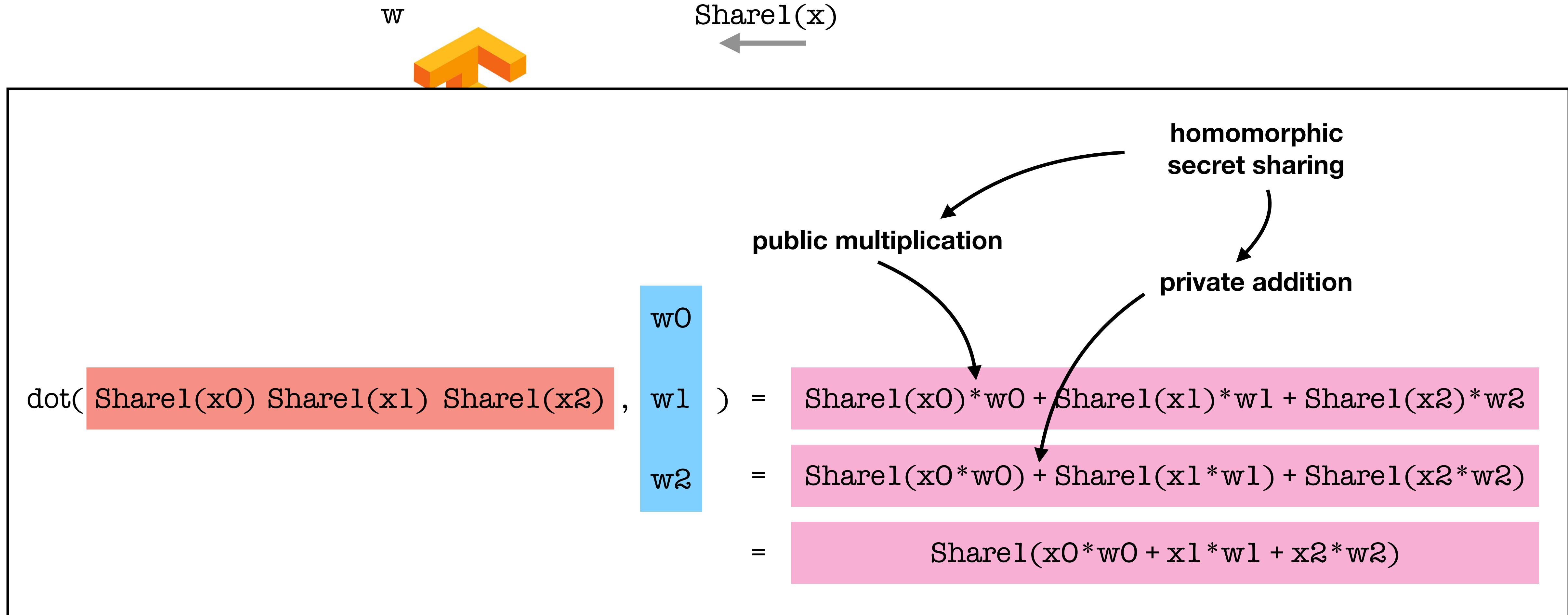
$$\begin{aligned} & \text{Enc}(\text{x}, \text{r})^{\wedge w} \\ &= (\text{g}^{\wedge x} * \text{r}^{\wedge n} \bmod \text{n}^2)^{\wedge w} \\ &= \text{g}^{\wedge(x * w)} * (\text{r}^{\wedge w})^{\wedge n} \bmod \text{n}^2 \\ &= \text{Enc}(\text{x} * w, \text{r}^{\wedge w}) \end{aligned}$$

$$\begin{aligned} & \text{Enc}(5, 2)^{\wedge 2} \\ &= 718 * 718 \\ &= 1024 \\ &= 36^{\wedge 10} * 4^{\wedge 35} \\ &= \text{Enc}(10, 4) \end{aligned}$$

# Using Secret Sharing



# Using Secret Sharing



# SPDZ Secret Sharing

$$\text{Share1}(x, r) = r \bmod m$$

public parameter

$$\text{Share2}(x, r) = x - r \bmod m$$

$$x = \text{Share1}(x, r) + \text{Share2}(x, r) \bmod m$$

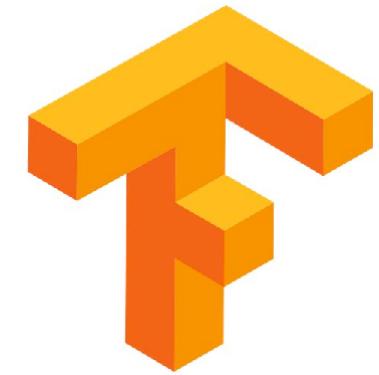
$$m = 10$$

$$\text{Share1}(5, 7) = 7 \bmod 10 = 7$$

$$\text{Share2}(5, 7) = 5 - 7 \bmod 10 = 8$$

$$7 + 8 = 15 = 5 \bmod 10$$

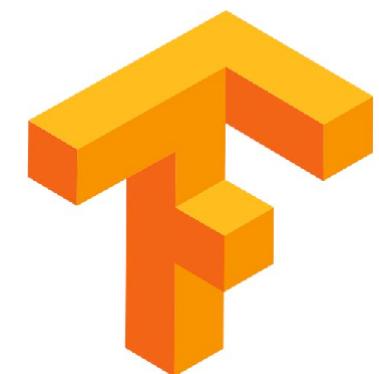
# Private Addition in SPDZ



x<sub>1</sub>

y<sub>1</sub>

z<sub>1</sub> = x<sub>1</sub> + y<sub>1</sub>



x<sub>2</sub>

y<sub>2</sub>

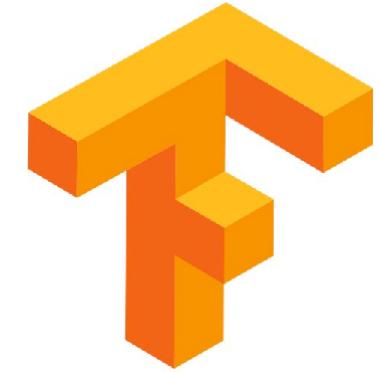
z<sub>2</sub> = x<sub>2</sub> + y<sub>2</sub>

$$x = x_1 + x_2$$

$$y = y_1 + y_2$$

$$\begin{aligned}x + y &= (x_1 + x_2) + (y_1 + y_2) \\&= (x_1 + y_1) + (x_2 + y_2) \\&= z_1 + z_2\end{aligned}$$

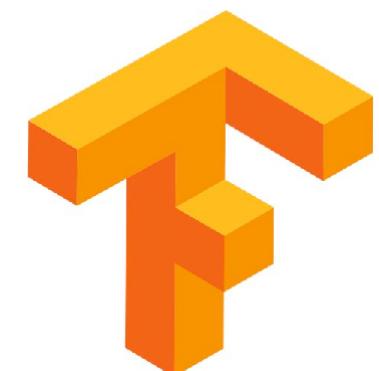
# Public Multiplication in SPDZ



x1

w

z1 = x1 \* w



x2

w

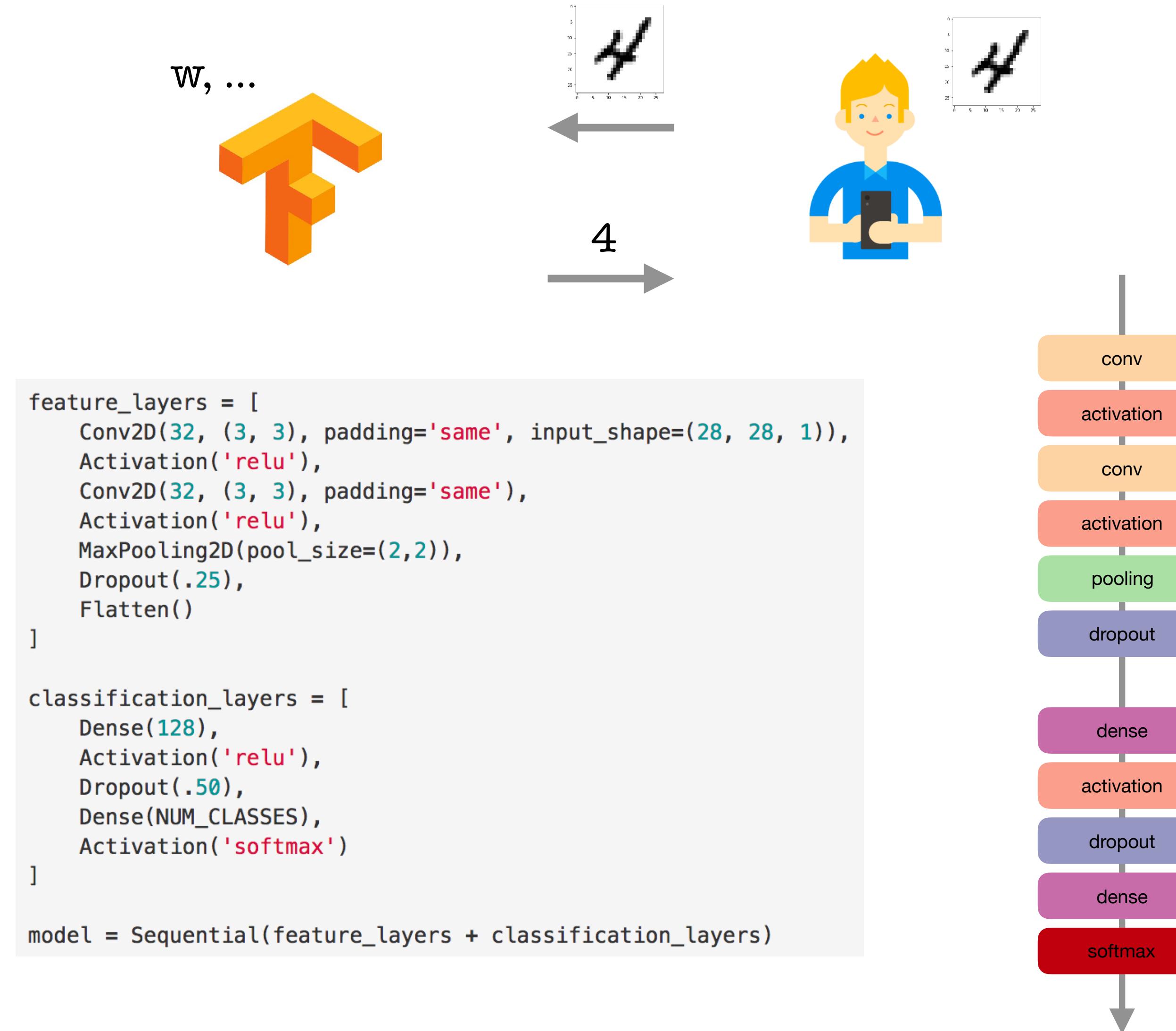
z2 = x2 \* w

$$x = x1 + x2$$

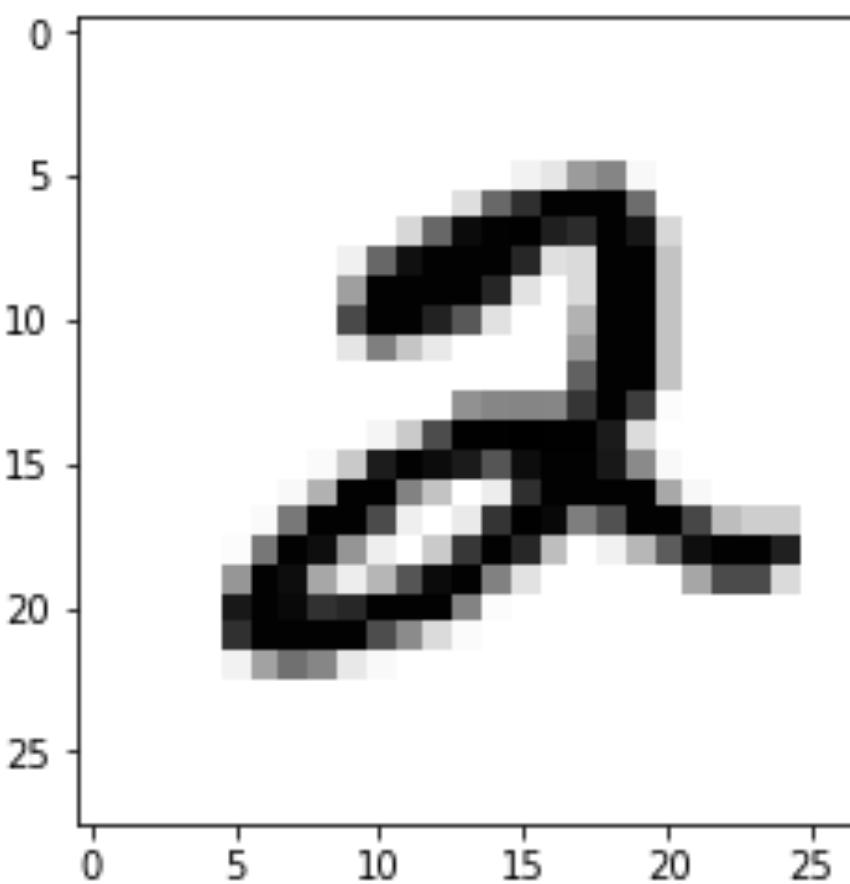
$$\begin{aligned} x * w &= (x1 + x2) * w \\ &= (x1 * w) + (x2 * w) \\ &= z1 + z2 \end{aligned}$$

# **Convolutional Neural Networks**

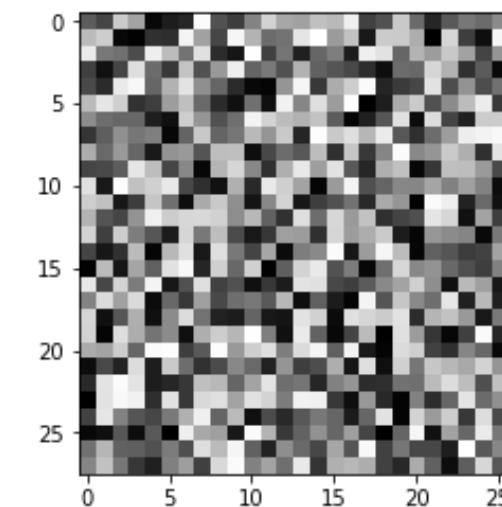
# Digit Classification with CNNs



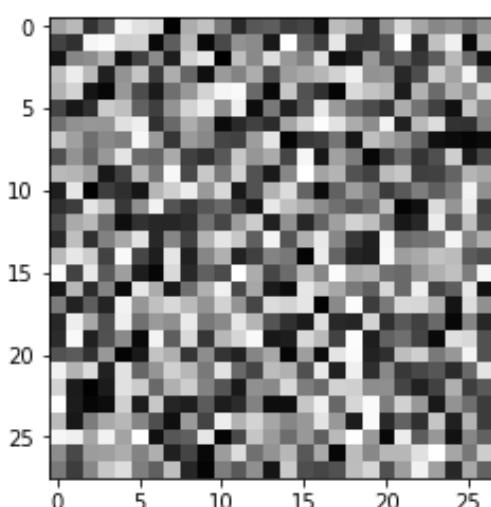
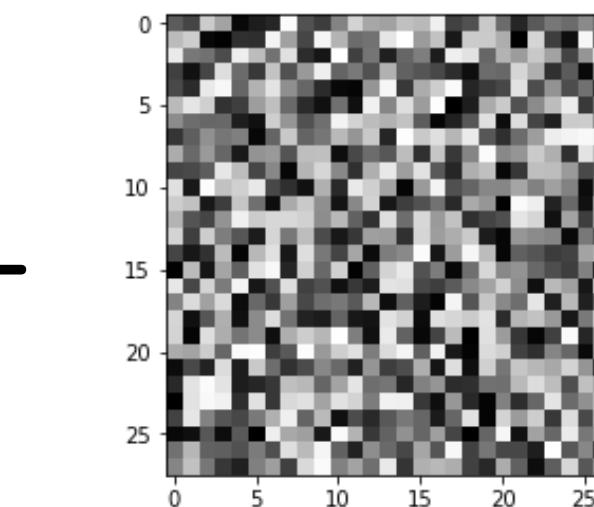
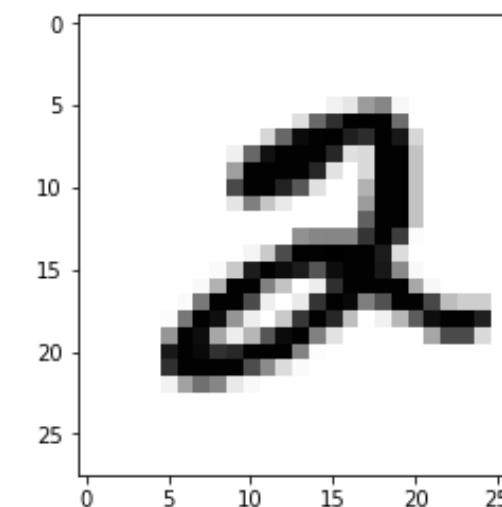
# Secret Sharing Images



$$\text{Share}_1(x, r) = r \quad =$$

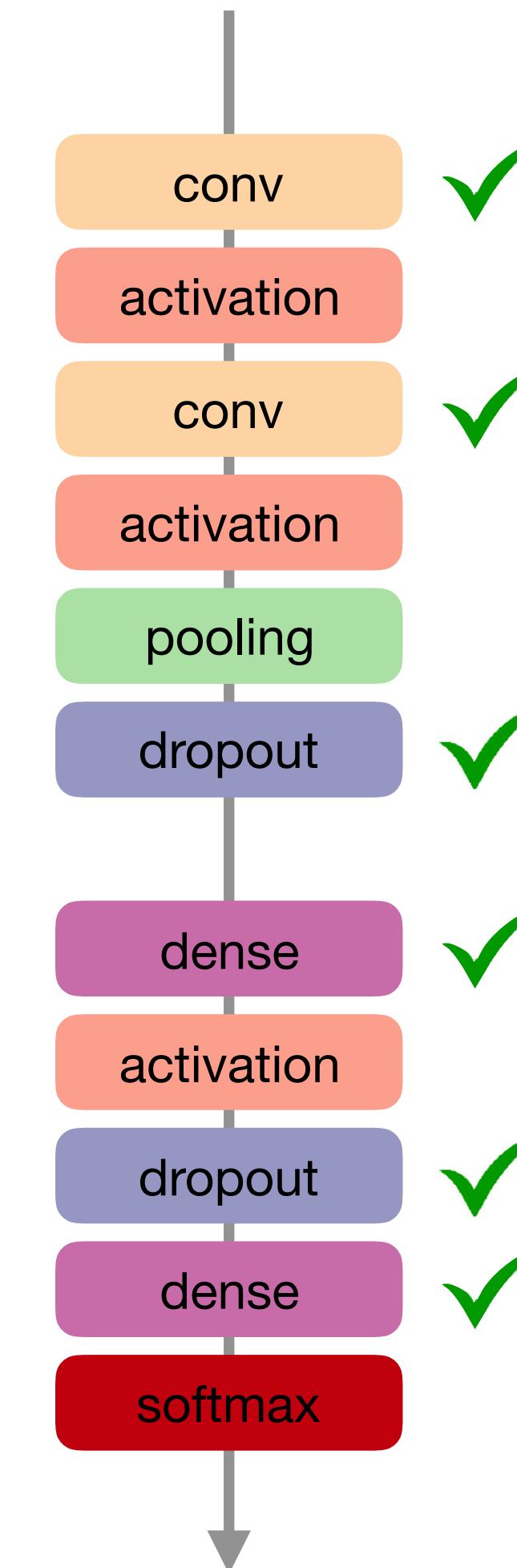


$$\text{Share2}(x, r) = x - r =$$

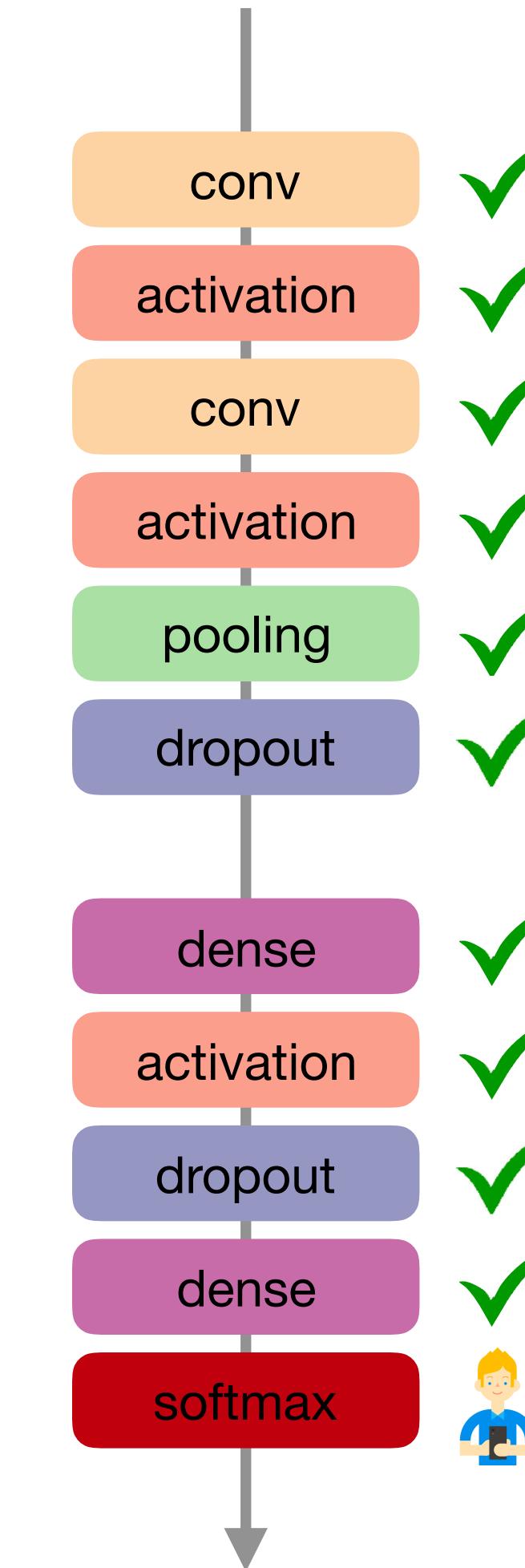
# Digit Classification with Secret Sharing

```
feature_layers = [  
    Conv2D(32, (3, 3), padding='same', input_shape=(28, 28, 1)),  
    Activation('relu'),  
    Conv2D(32, (3, 3), padding='same'),  
    Activation('relu'),  
    MaxPooling2D(pool_size=(2,2)),  
    Dropout(.25),  
    Flatten()  
]  
  
classification_layers = [  
    Dense(128),  
    Activation('relu'),  
    Dropout(.50),  
    Dense(NUM_CLASSES),  
    Activation('softmax')  
]  
  
model = Sequential(feature_layers + classification_layers)
```



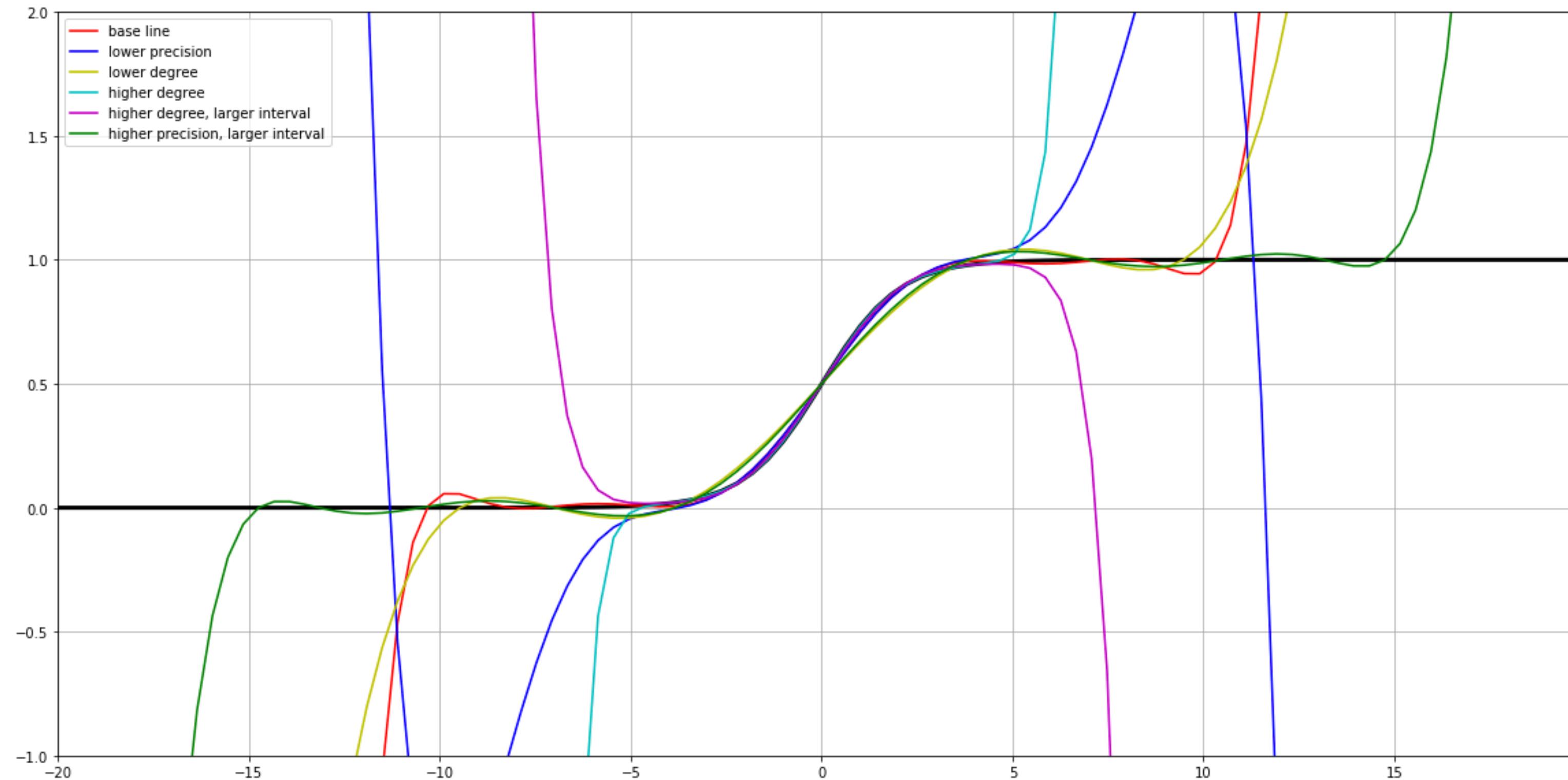
# Digit Classification with Secret Sharing

```
feature_layers = [
    feature_layers = [
        Conv2D(32, (3, 3), padding='same', input_shape=(28, 28, 1)),
        Activation('sigmoid'),
        Conv2D(32, (3, 3), padding='same'),
        Activation('sigmoid'),
        AveragePooling2D(pool_size=(2,2)),
        Dropout(.25),
    ] Flatten()
]
c1
classification_layers = [
    Dense(128),
    Activation('sigmoid'),
    Dropout(.50),
    Dense(5),
] Activation('softmax')
]
mc
model = Sequential(feature_layers + classification_layers)
```



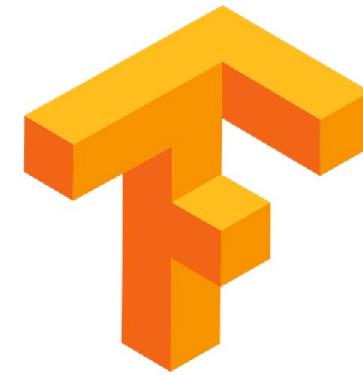
model adaptation can improve performance significantly

# Sigmoid via Polynomial Approximation



$$f(x) = c_7 * x^7 + c_5 * x^5 + \dots + c_1 * x + c_0$$

# Private Multiplication in SPDZ



random triple  
(  $a_1$  ,  $b_1$  ,  $c_1$  )

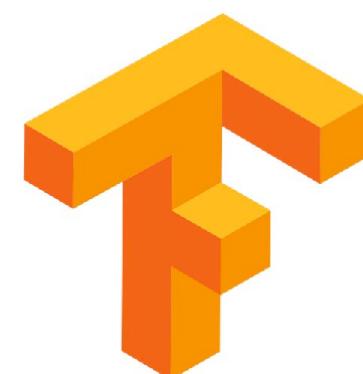
$x_1$

$y_1$

alpha

beta

$$z_1 = \text{alpha} * \text{beta} + \text{alpha} * b_1 + \text{beta} * a_1 + c_1$$



(  $a_2$  ,  $b_2$  ,  $c_2$  )

$x_2$

$y_2$

alpha

beta

$$z_2 = \text{alpha} * b_2 + \text{beta} * a_2 + c_2$$

$$\begin{aligned} a &= a_1 + a_2 \\ b &= b_1 + b_2 \\ c &= a * b = c_1 + c_2 \end{aligned}$$

$$x = x_1 + x_2 \quad y = y_1 + y_2$$

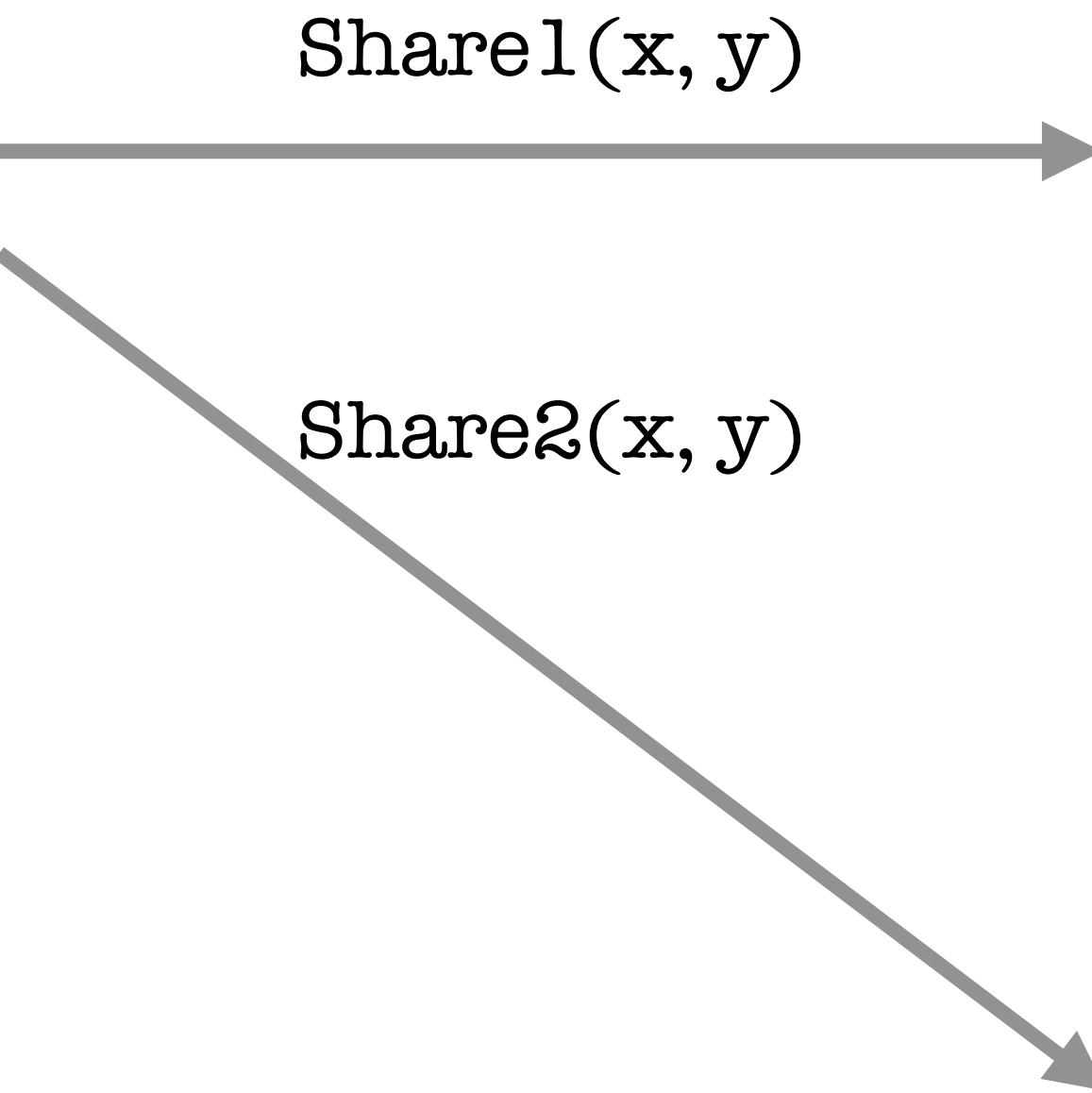
$$\text{alpha} = x - a$$

$$\text{beta} = y - b$$

$$\begin{aligned} x * y &= \dots \\ &= z_1 + z_2 \end{aligned}$$

# Training

# Participants



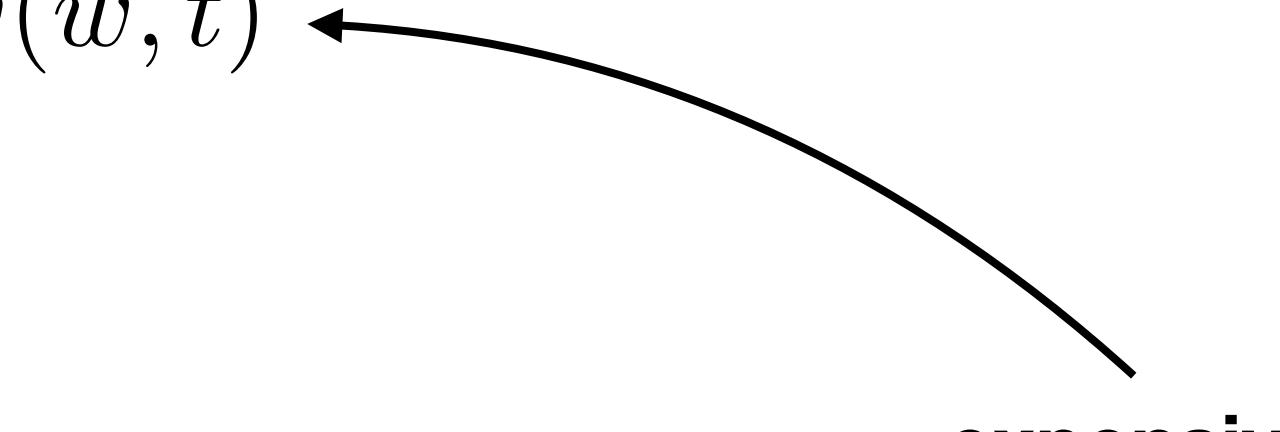
# SoftMax Alternatives

$$f(x_i) = \frac{e^{-x_i}}{\sum_j e^{-x_j}}$$

$$g(x_i) = \frac{\text{ReLU}(x_i)}{\sum_j \text{ReLU}(x_j)}$$

# Simpler Optimisation Strategies

RMSProp:

$$w = w - \frac{\eta}{\sqrt{v(w, t)}} \nabla Q(w)$$


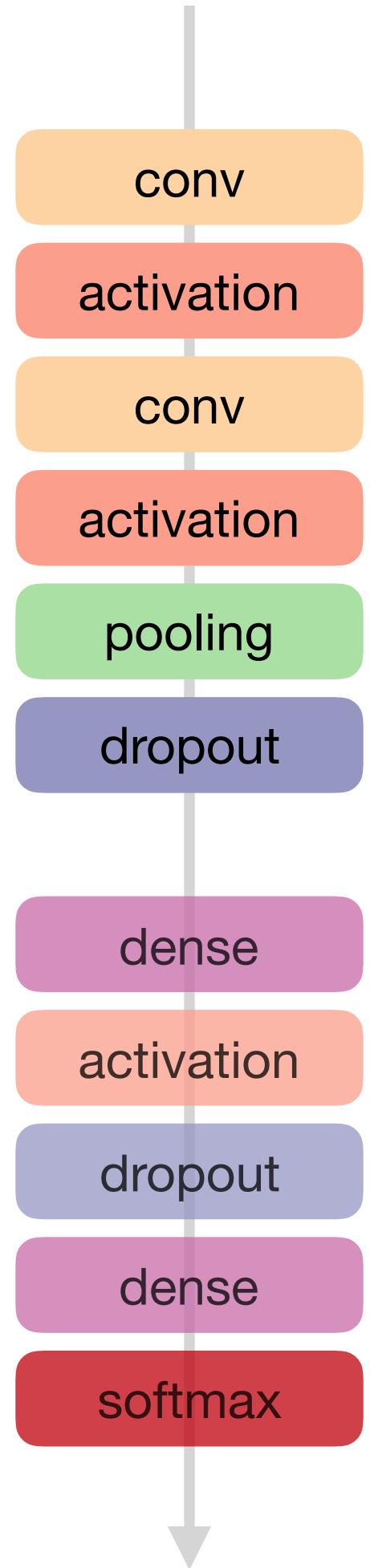
**expensive**

SGD:

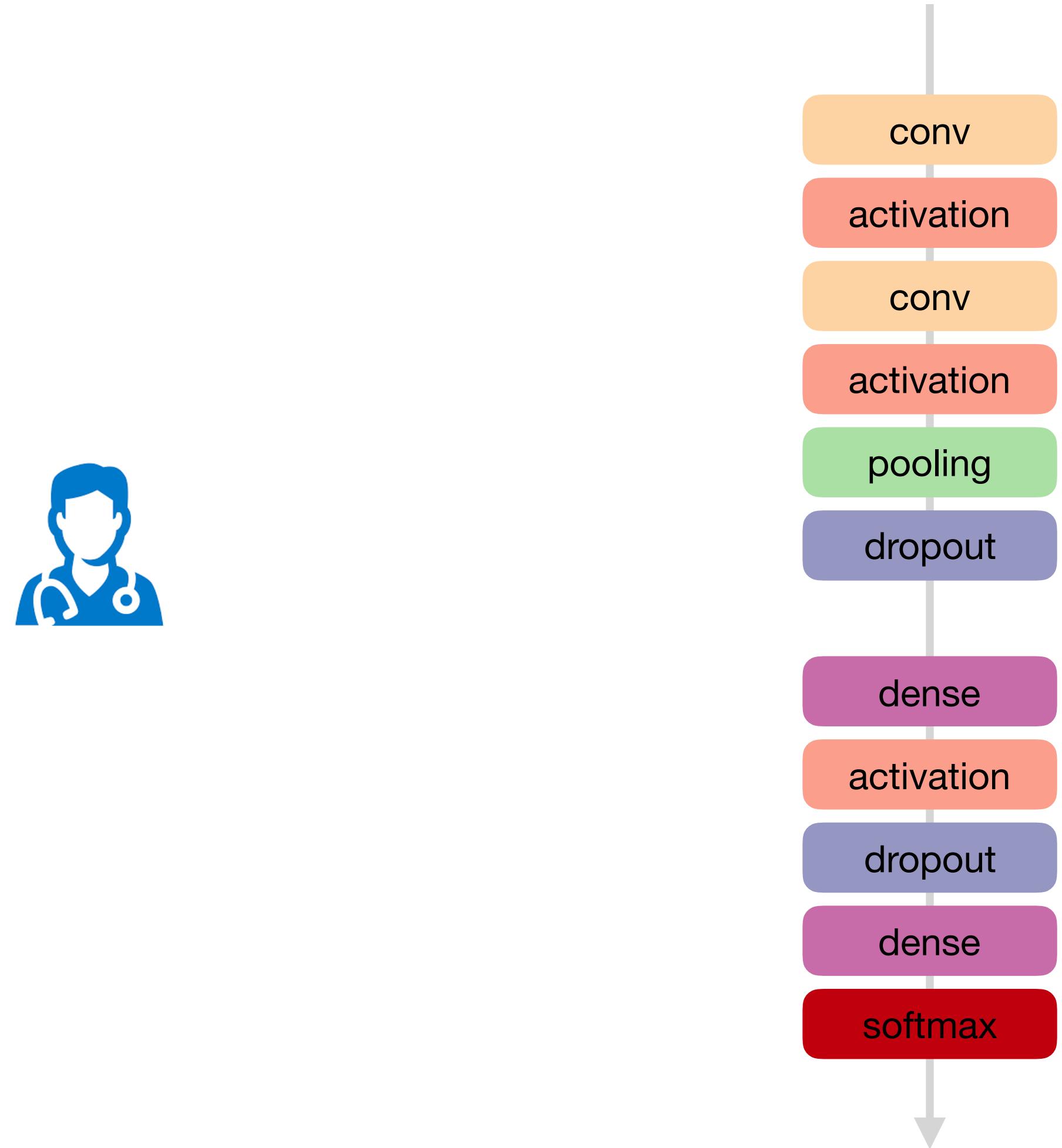
$$w = w - \eta \nabla Q(w)$$

# Transfer Learning

IMAGENET



# Transfer Learning



# Making It Accessible

# Projects and Literature

## Recent research papers using secure computation

**CryptoNets**: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy, Dowlin et al.

**SecureML**: A System for Scalable Privacy-Preserving Machine Learning, Mohassel and Zhang

**DeepSecure**: Scalable Provably-Secure Deep Learning, Rouhani et al.

**Gazelle**: A Low Latency Framework for Secure Neural Network Inference, Juvekar et al.

**ABY3**: A Mixed Protocol Framework for Machine Learning, Mohassel and Rindal

**SecureNN**: Efficient and Private Neural Network Training, Wagh et al.

(great summary in <https://eprint.iacr.org/2017/1190>)

## Secure computation frameworks

**SCALE-MAMBA** (<https://homes.esat.kuleuven.be/~nsmart/SCALE/>)

**MP-SPDZ** (<https://github.com/n1analytics/MP-SPDZ>)

**ABY** (<https://github.com/encryptogroup/ABY>)

**OblivC** (<http://oblivc.org/>)

**tf-encrypted** (<https://github.com/mortendahl/tf-encrypted>)

(much more at <https://github.com/rdragos/awesome-mpc>)

**PySyft** (<https://github.com/OpenMined/PySyft>)

# Multidisciplinary Challenge

Data science  
(use-cases, workflow, monitoring)

Cryptography  
(techniques, protocols, guarantees)

Machine learning  
(models, approx, quantisation)

Engineering  
(distributed, multi-core, readability)

**useful to have common framework**

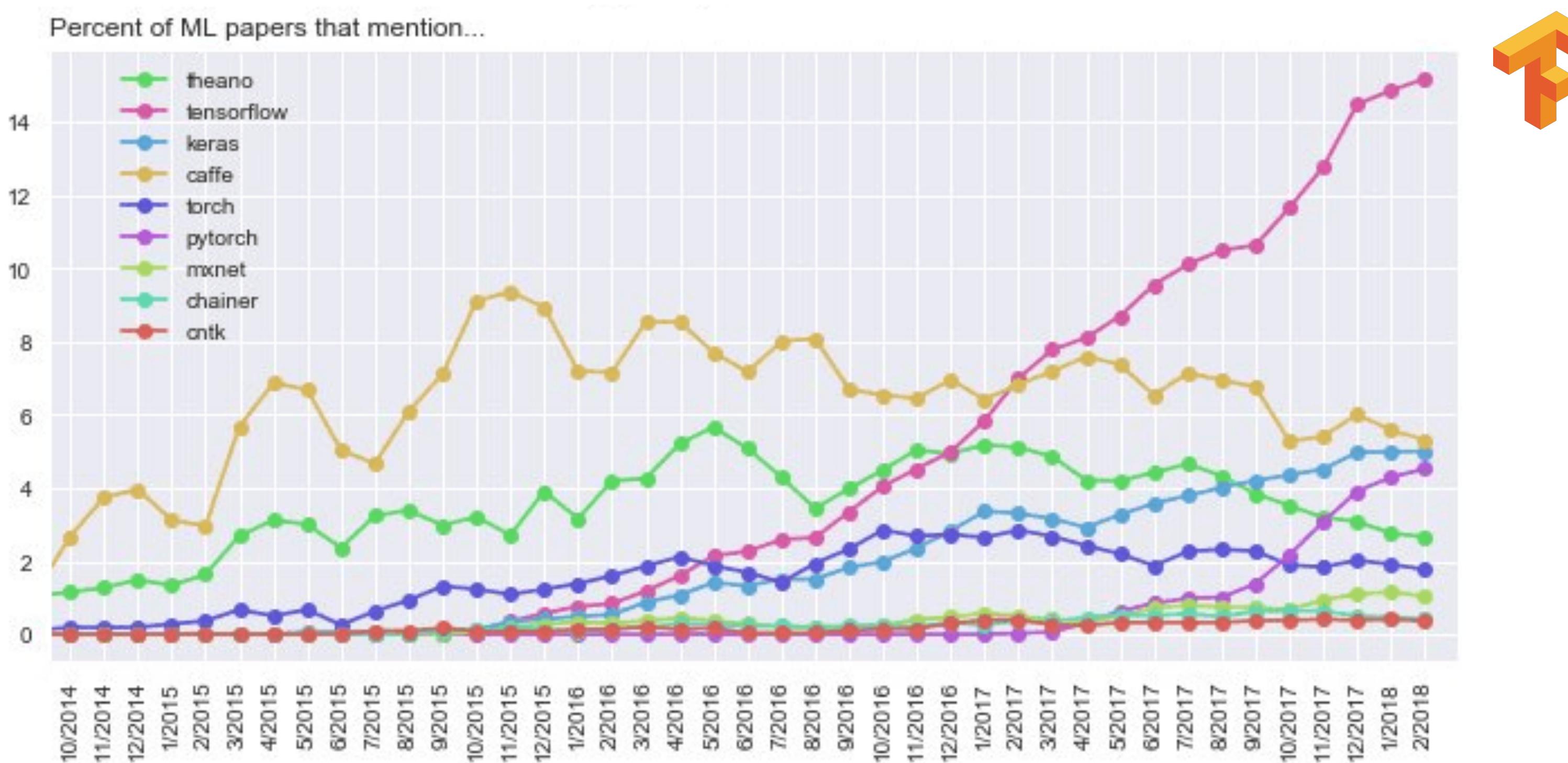
# **tf-encrypted**

open source project for exploring and experimenting with  
privacy-preserving machine learning in TensorFlow

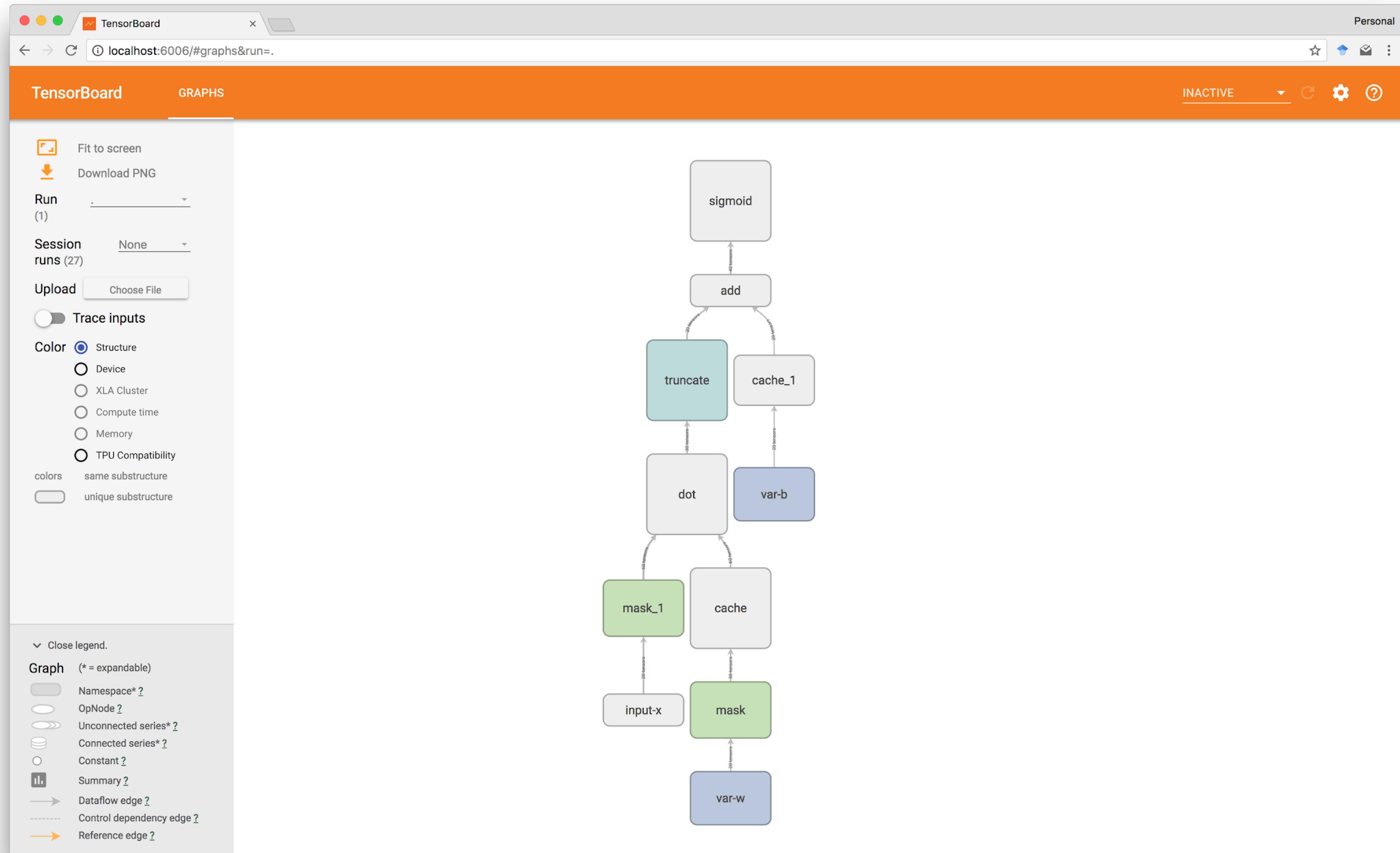
# TensorFlow

platform for production-level training and deployment of models

popular and backed by Google



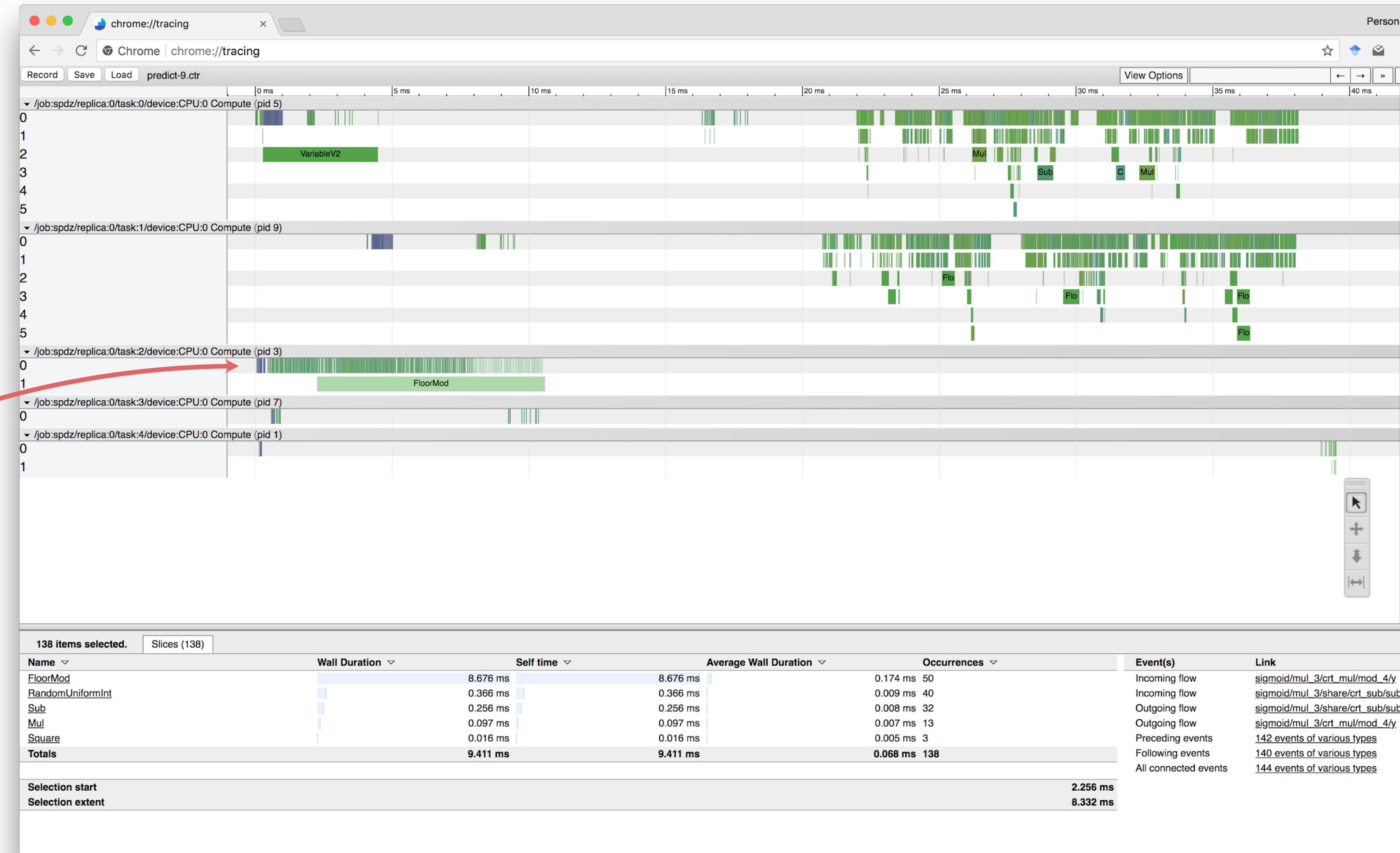
# Dataflow Graphs



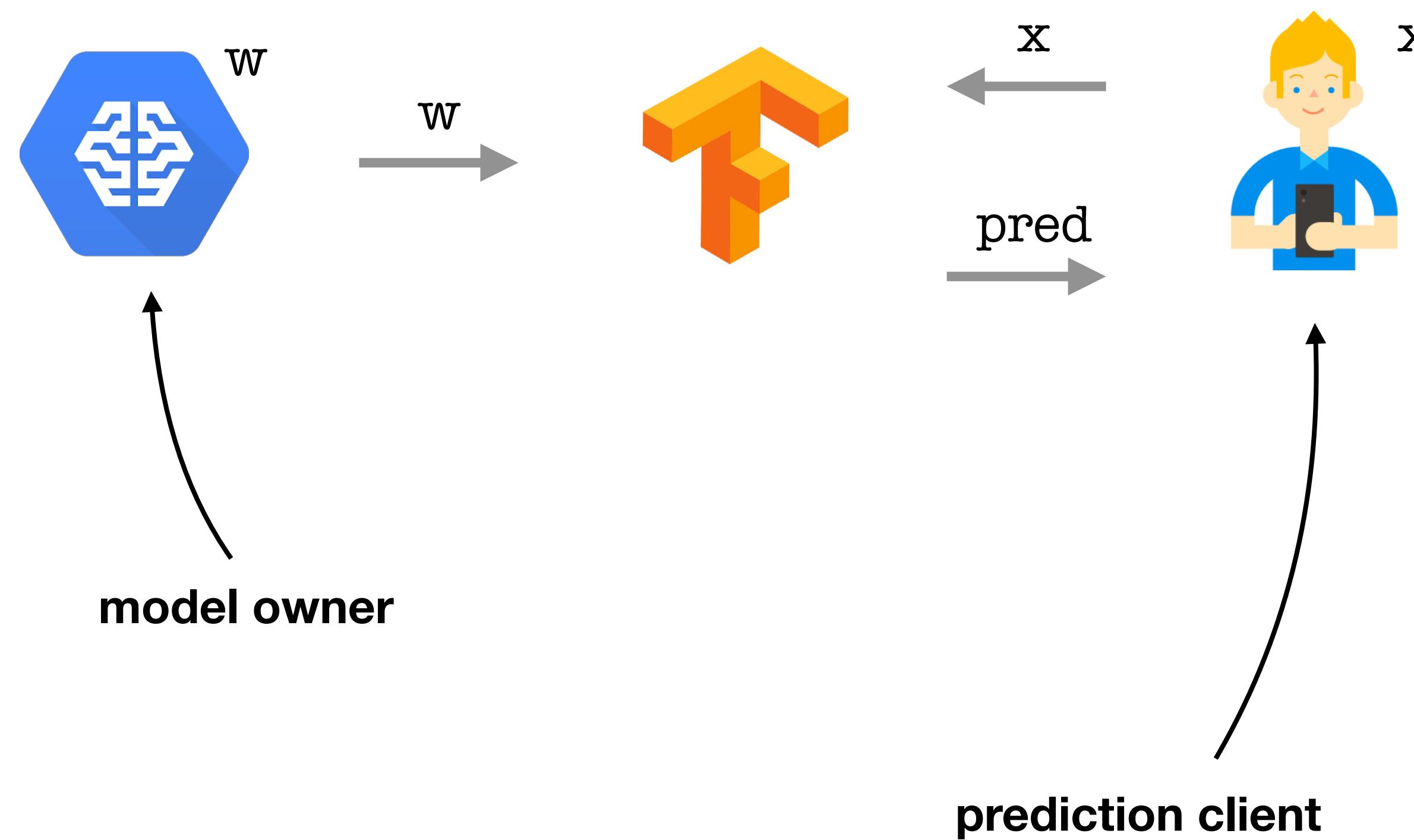
# Optimized Distributed Execution

Concurrent

Batching

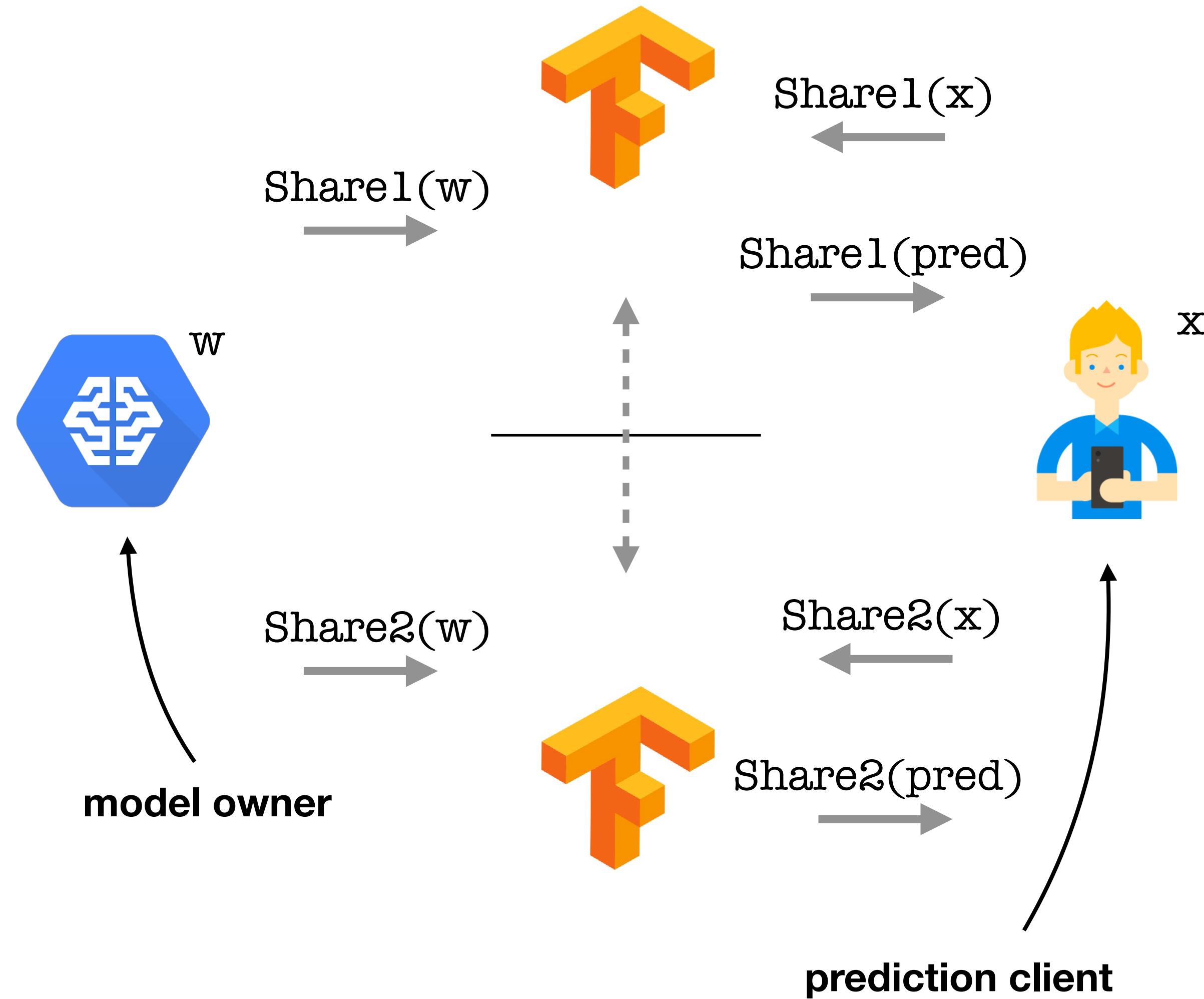


# Public Prediction Example (TensorFlow)



```
1 import tensorflow as tf
2
3 def provide_weights(): """ Load from disk """
4 def provide_input(): """ Pre-process """
5 def receive_output(logits): return tf.Print([], [tf.argmax(logits)])
6
7 # get model weights
8 w0, b0, w1, b1, w2, b2 = provide_weights()
9
10 # get prediction input
11 x = provide_input()
12
13 # compute prediction
14 layer0 = tf.nn.relu(tf.matmul(x, w0) + b0)
15 layer1 = tf.nn.relu(tf.matmul(layer0, w1) + b1)
16 logits = tf.matmul(layer2, w2) + b2
17
18 # process result of prediction
19 prediction_op = receive_output(logits)
20
21 # run graph execution in a tf.Session
22 with tf.Session() as sess:
23     sess.run(tf.global_variables_initializer())
24     sess.run(prediction_op)
```

# Private Prediction Example (tf-encrypted)



```
1 import tensorflow as tf
2 import tf_encrypted as tfe
3
4 def provide_weights(): """ Load from disk """
5 def provide_input(): """ Pre-process """
6 def receive_output(logits): return tf.Print([], [tf.argmax(logits)])
7
8 # get model weights as private tensors from owner
9 w0, b0, w1, b1, w2, b2 = tfe.define_private_input("model-owner", provide_weights)
10
11 # get prediction input as private tensors from client
12 x = tfe.define_private_input("prediction-client", provide_input)
13
14 # compute private prediction on servers
15 layer0 = tfe.relu((tfe.matmul(x, w0) + b0))
16 layer1 = tfe.relu((tfe.matmul(layer0, w1) + b1))
17 logits = tfe.matmul(layer1, w2) + b2
18
19 # process result of prediction on client
20 prediction_op = tfe.define_output("prediction-client", logits, receive_output)
21
22 # run secure graph execution in a tf.Session()
23 with tfe.Session() as sess:
24     sess.run(tf.global_variables_initializer())
25     sess.run(prediction_op)
```

# Wrap-Up

Privacy-preserving technologies are enablers for accessing sensitive information

Secure computation is complementary, not a replacement

Multidisciplinary field benefitting from adaptations on both sides

Bonus:  
some interest from safety (governance)  
and fairness (validation)

**Thank you!**