Cryptography for

Privacy-Preserving Machine Learning

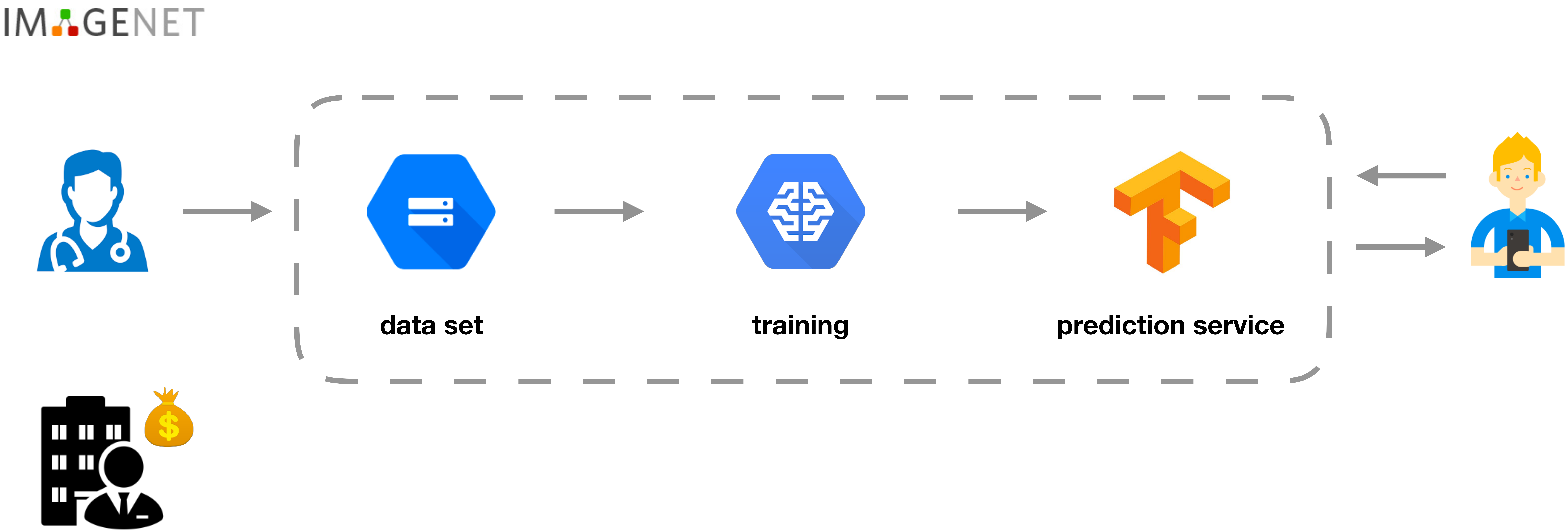
Morten Dahl

The tf-encrypted Project and Dropout Labs

Applied ML Days, AI & Trust track, EPFL, January 2019

Why?

Machine Learning Process

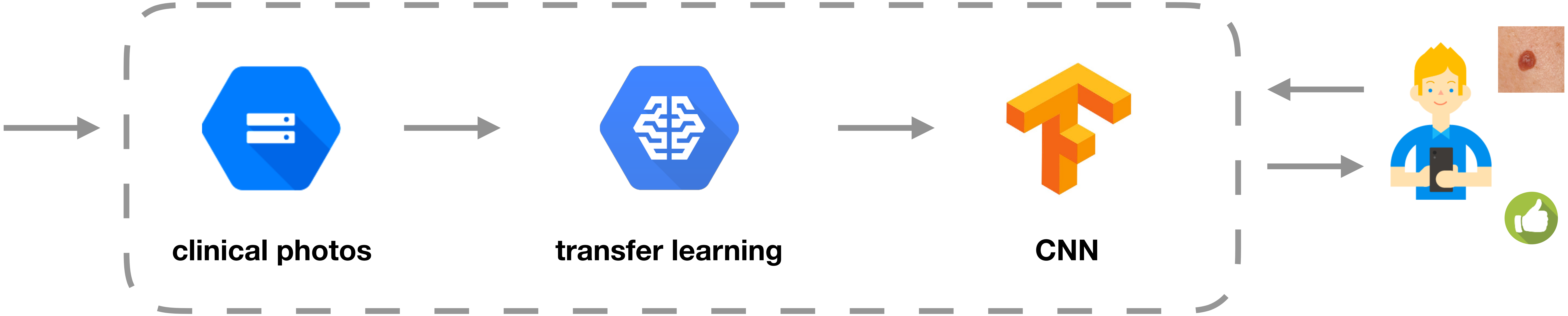


Skin Cancer Image Classification

Brett Kuprel

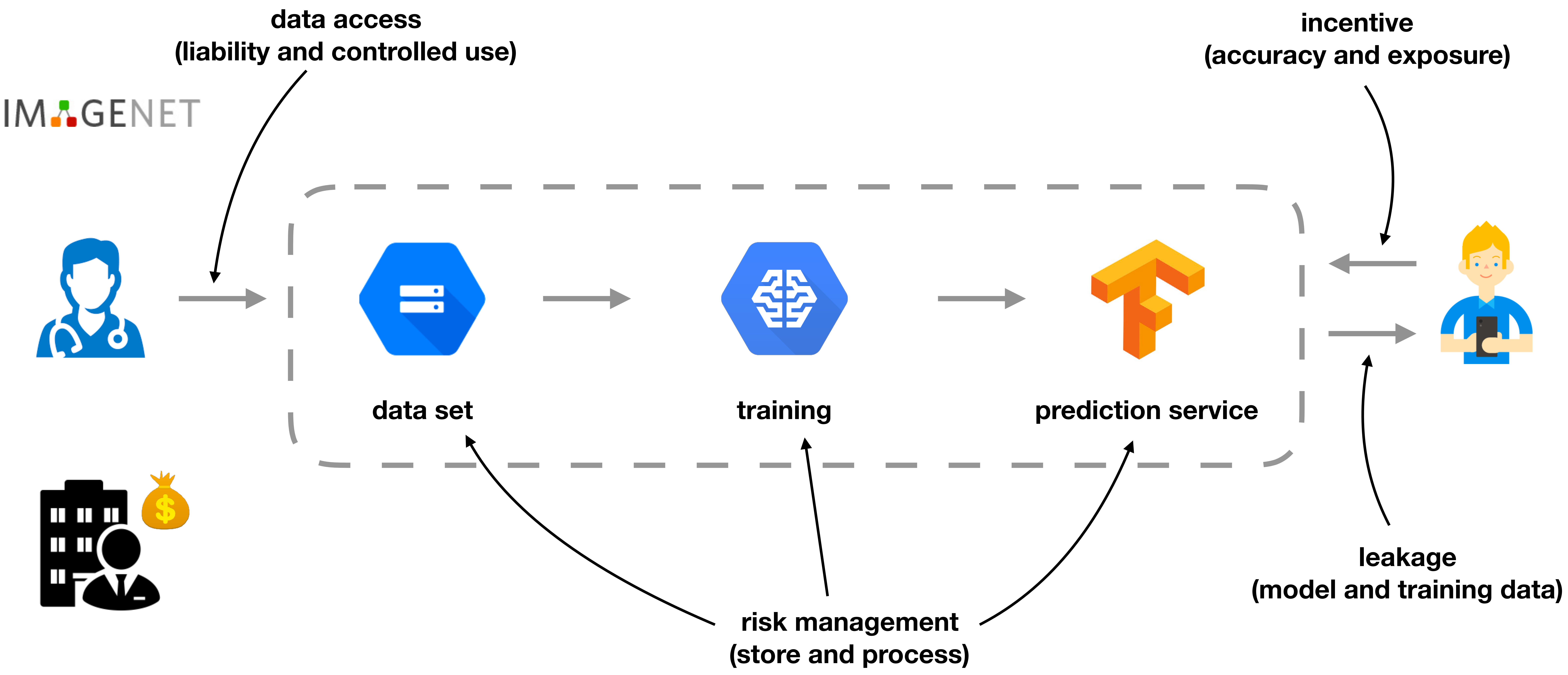
12:30-12:40pm

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.

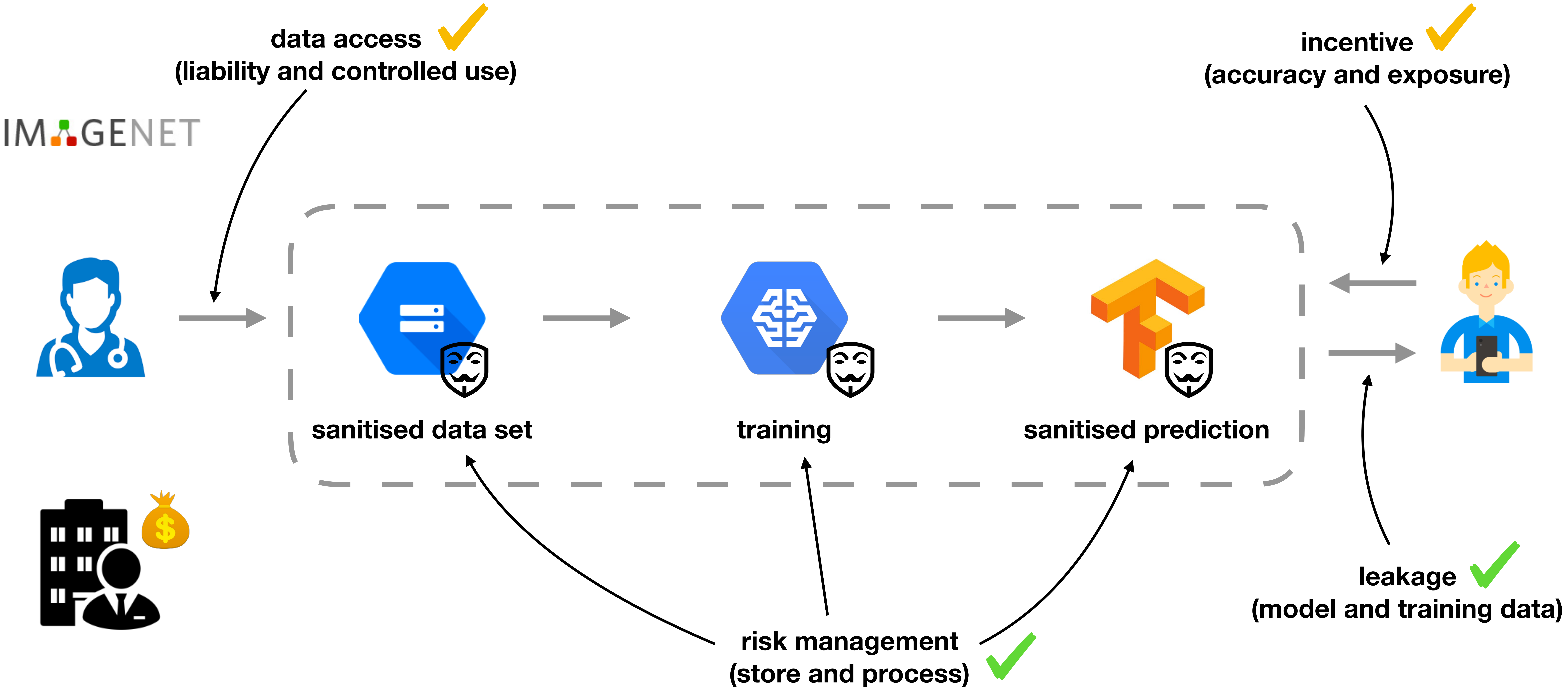


machine learning positioned to have huge impact on health care

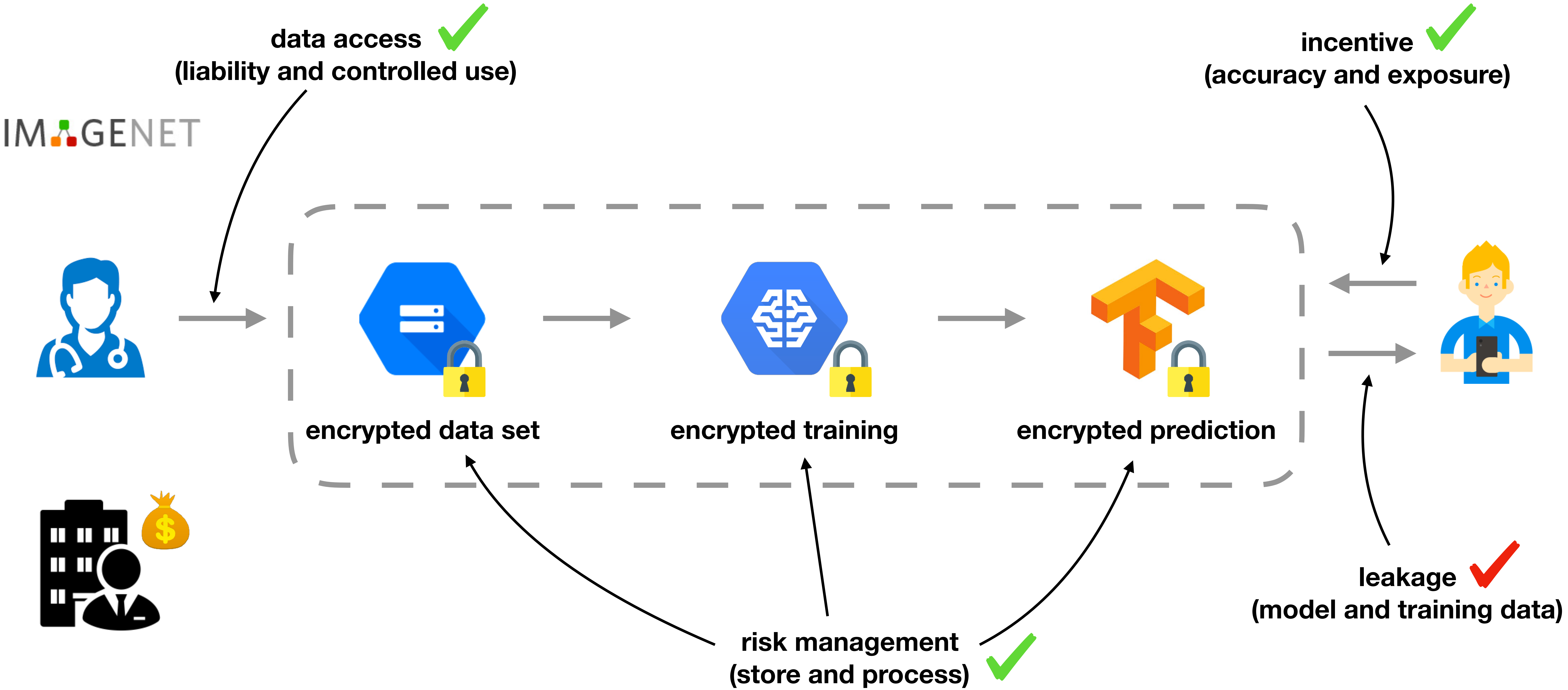
Potential Bottlenecks



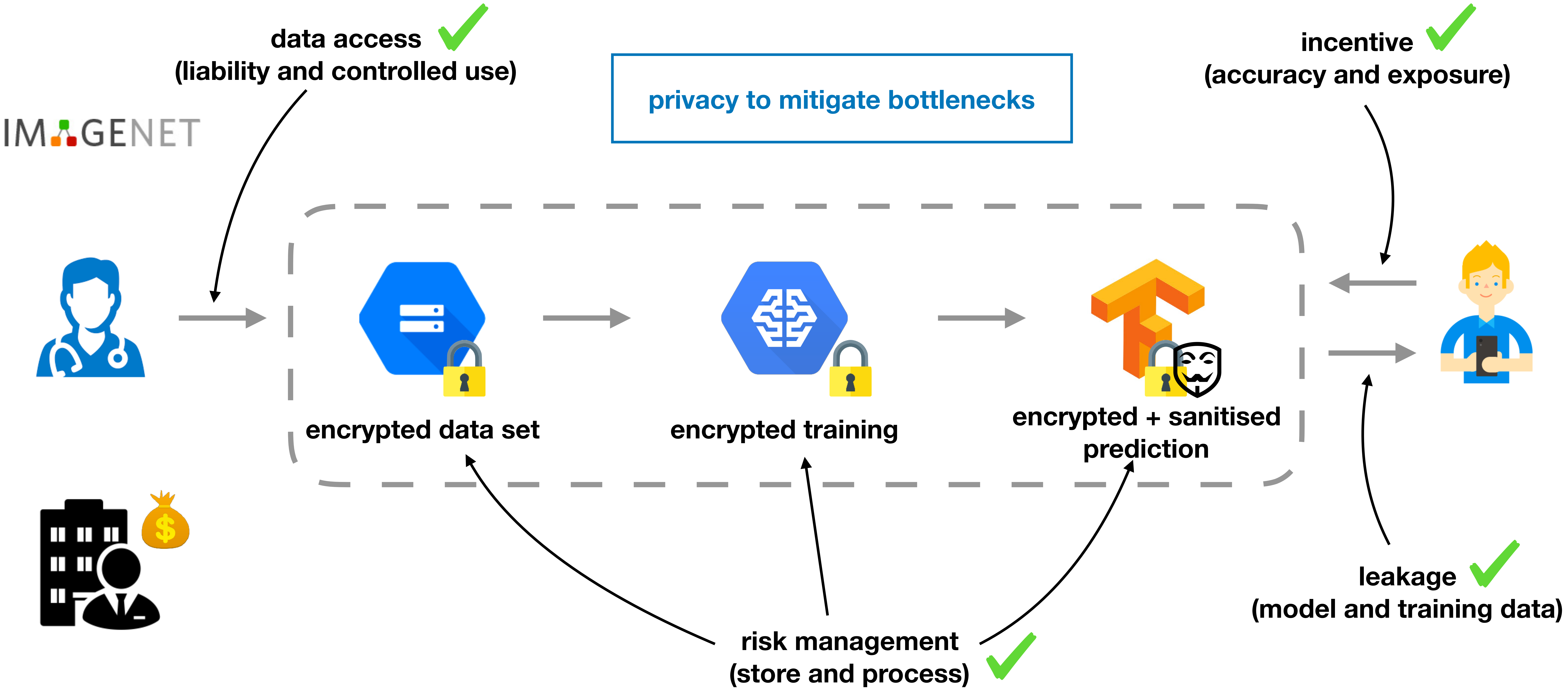
Differential Privacy



Secure Computation

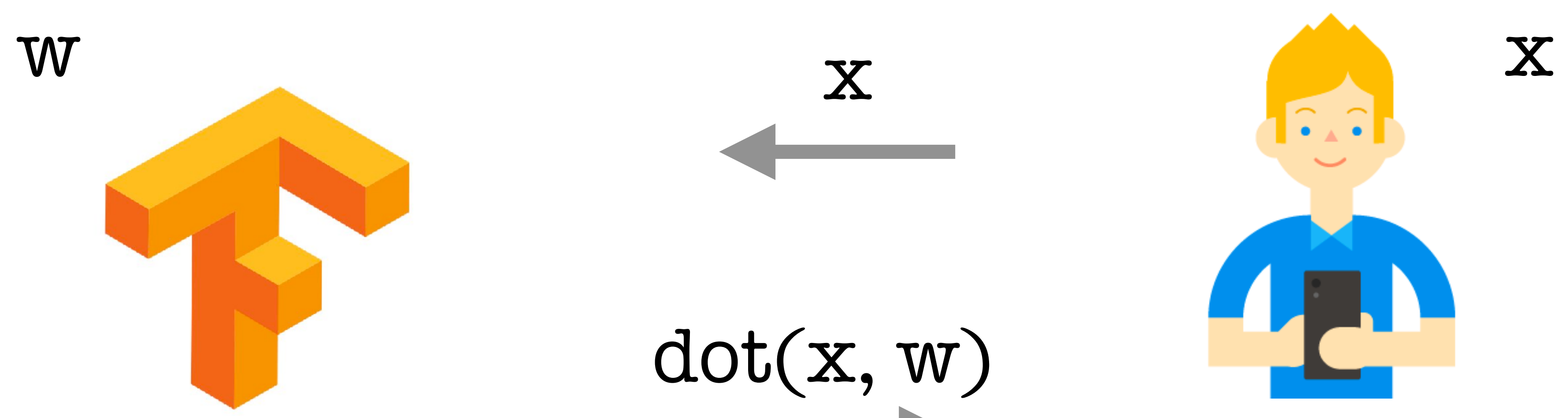


Hybrid


data access
(liability and controlled use)
privacy to mitigate bottlenecks
incentive
(accuracy and exposure)
IMAGENET
encrypted data set
encrypted training
encrypted + sanitised prediction
risk management
(store and process)
leakage
(model and training data)

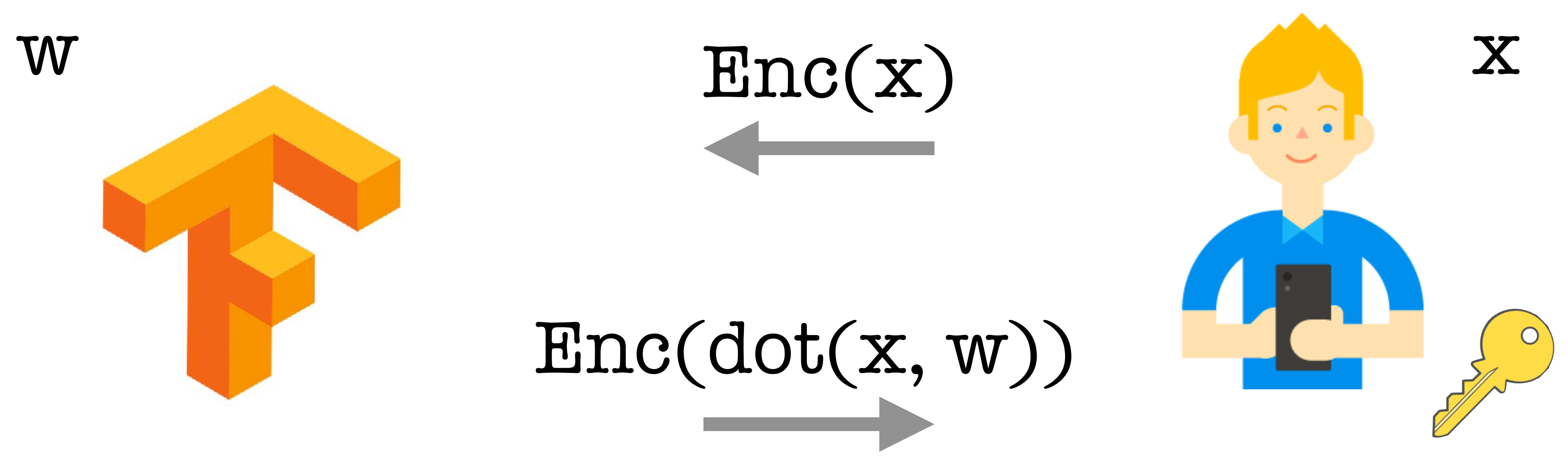
Prediction

Prediction with Linear Model



$$\mathrm{dot}\left(\begin{bmatrix} x1 & x2 & x3 \end{bmatrix}, \begin{bmatrix} w1 \\ w2 \\ w3 \end{bmatrix}\right) = x1 * w1 + x2 * w2 + x3 * w3$$

… using Homomorphic Encryption



homomorphic encryption scheme

public multiplication

private addition

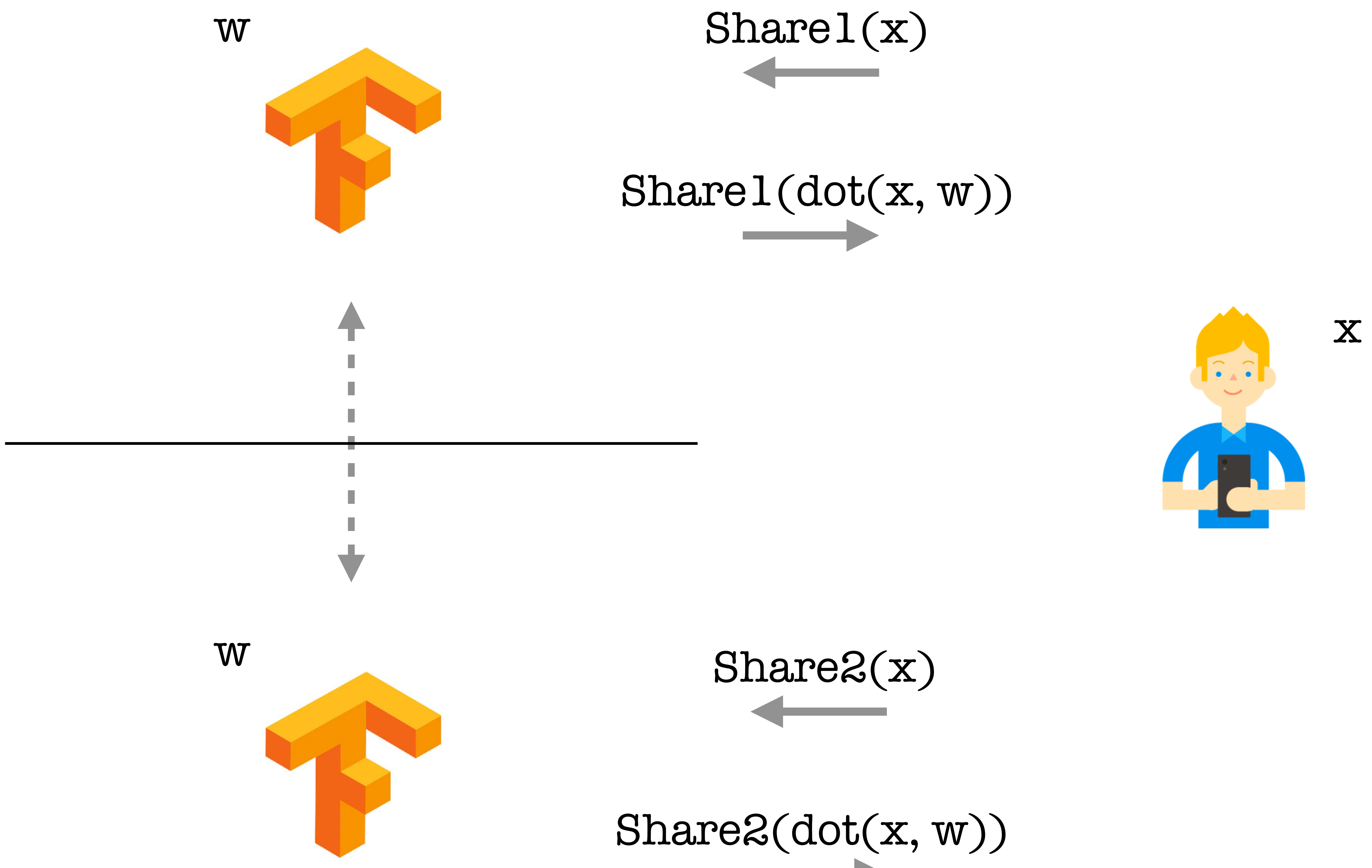
$$\mathrm{dot}\left(\begin{bmatrix}\mathrm{Enc}(x1) & \mathrm{Enc}(x2) & \mathrm{Enc}(x3)\end{bmatrix}, \begin{bmatrix} w1 \\ w2 \\ w3 \end{bmatrix}\right)$$

$$= \mathrm{Enc}(x1)*w1 + \mathrm{Enc}(x2)*w2 + \mathrm{Enc}(x3)*w3$$

$$= \mathrm{Enc}(x1*w1) + \mathrm{Enc}(x2*w2) + \mathrm{Enc}(x3*w3)$$

$$= \mathrm{Enc}(x1*w1 + x2*w2 + x3*w3)$$

… using Secret Sharing



… using Secret Sharing

w

Share1(x)

Share1(dot(x,w))

homomorphic secret sharing

public multiplication

private addition

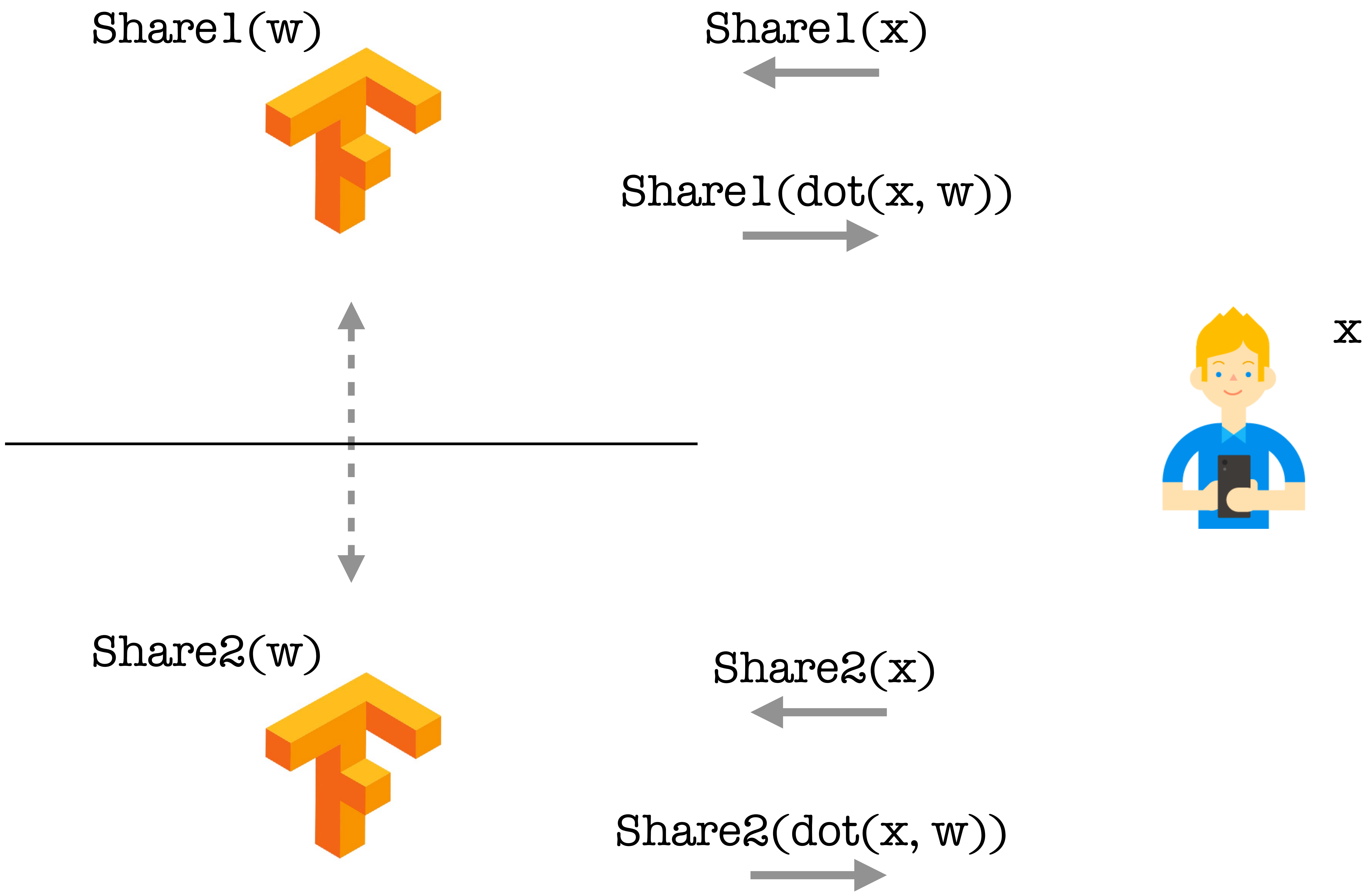
dot(Share1(x0) Share1(x1) Share1(x2) , [w0, w1, w2])

= Share1(x0)*w0 + Share1(x1)*w1 + Share1(x2)*w2

= Share1(x0*w0) + Share1(x1*w1) + Share1(x2*w2)

= Share1(x0*w0 + x1*w1 + x2*w2)

… using Secret Sharing



… using Secret Sharing

Share1(w)

Share1(x)

Share1(dot(x, w))

x

private multiplication

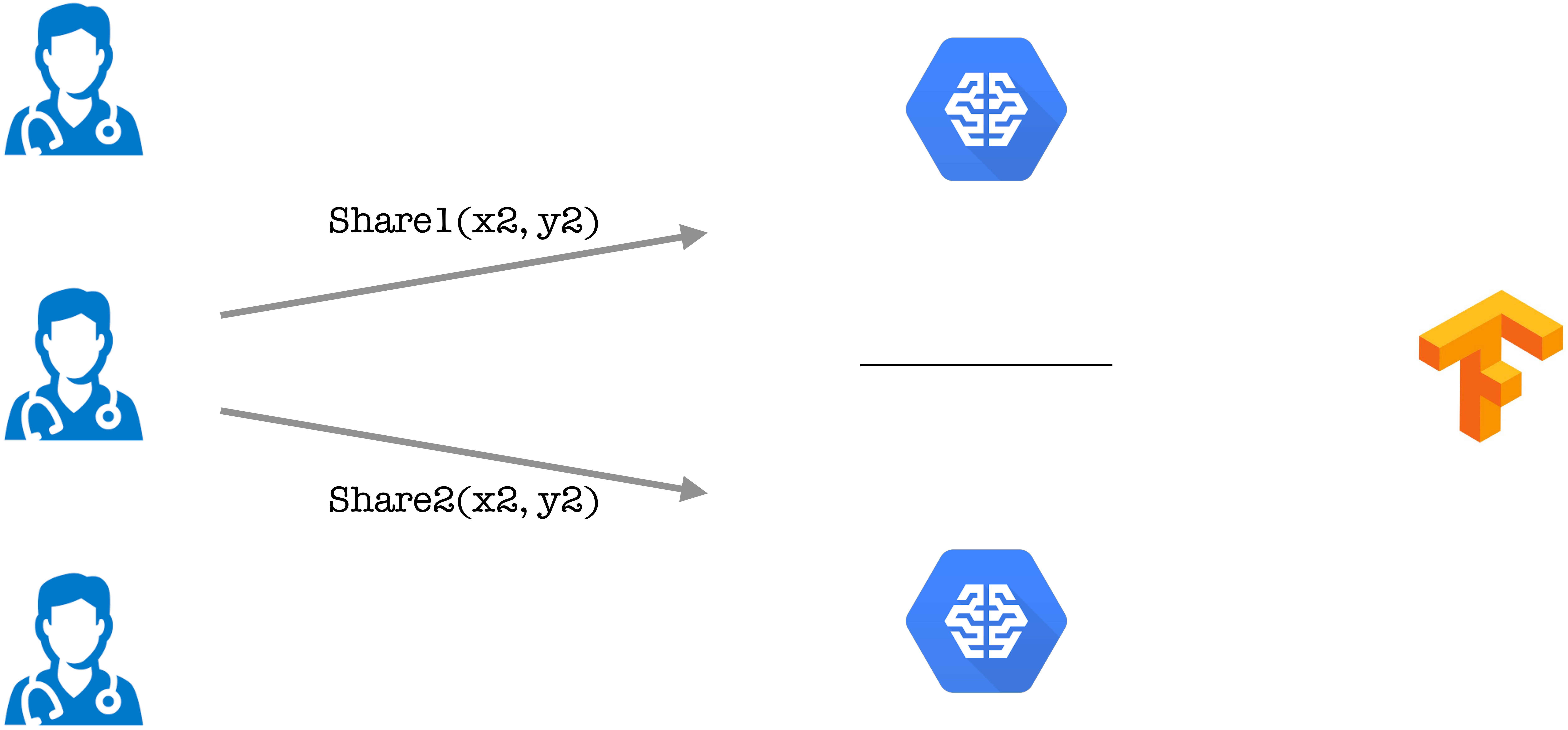
dot(Share1(x0) Share1(x1) Share1(x2) , Share(w0) Share(w1) Share(w2)) = Share1(x0)*Share(w0) + …

= Share1(x0*w0) + …

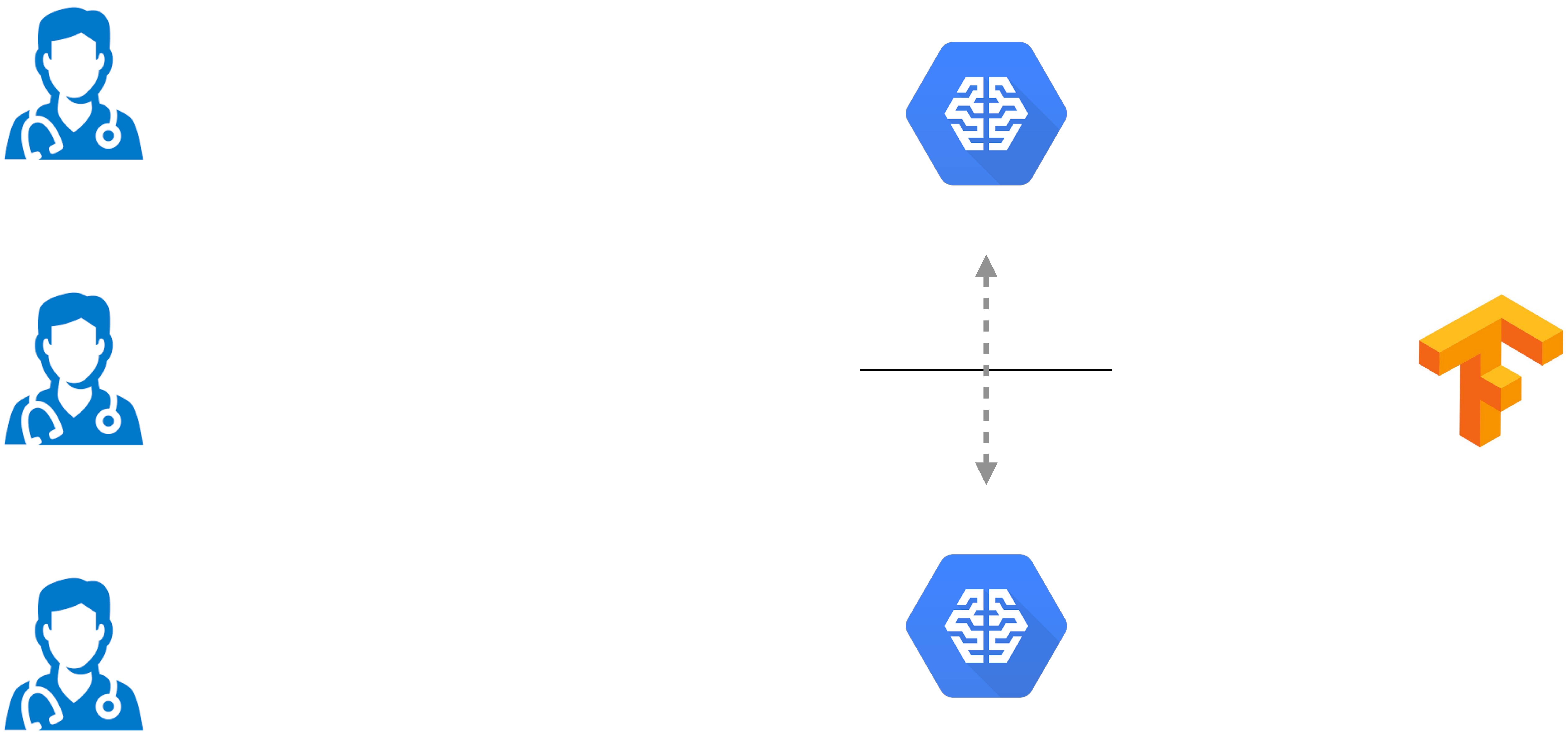
= Share1(x0*w0 + x1*w1 + x2*w2)

Training

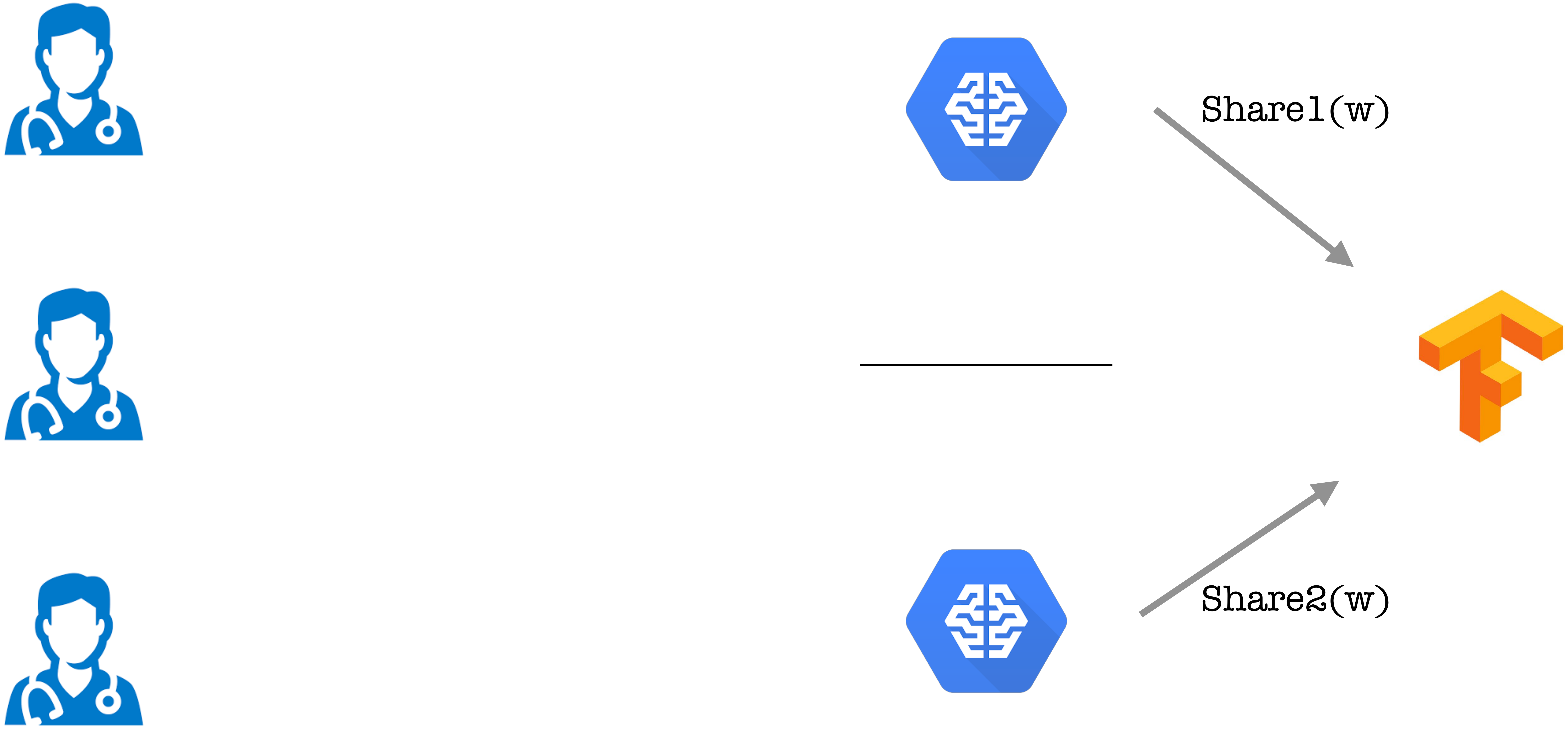
Server-Aided



Server-Aided

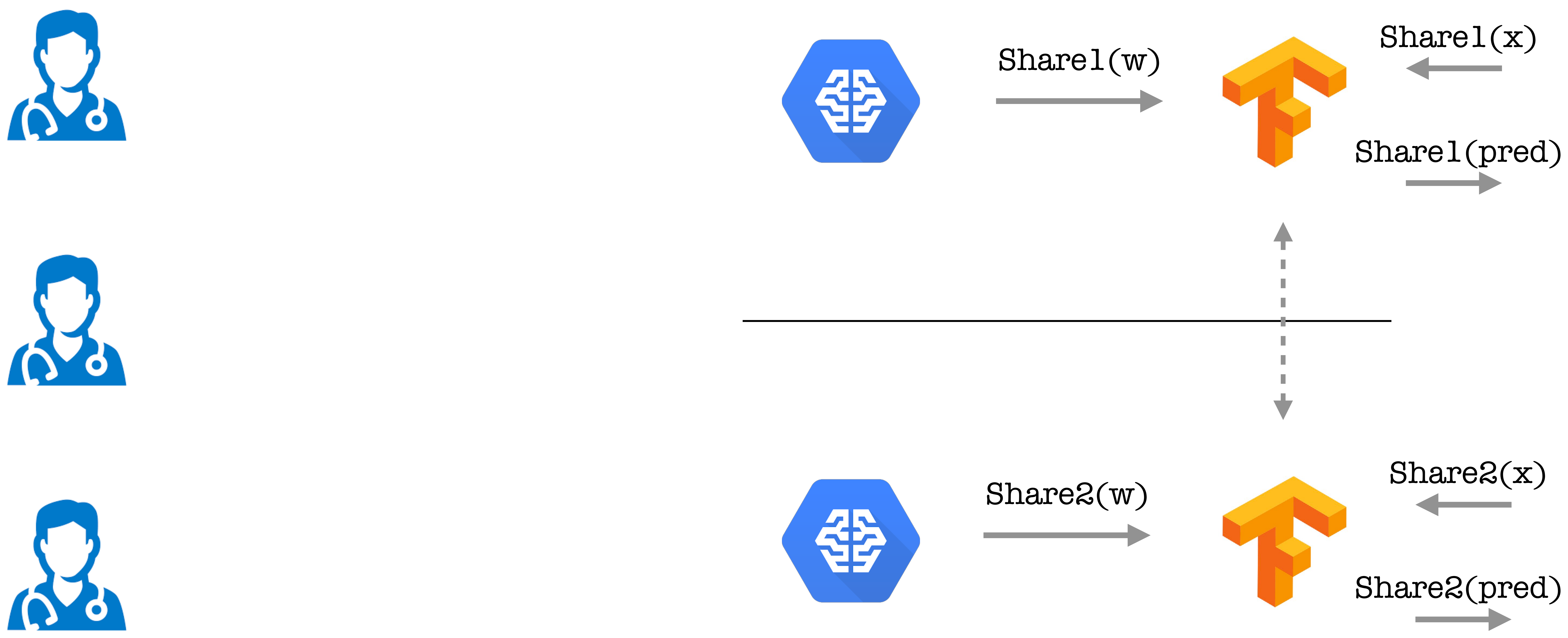


Server-Aided

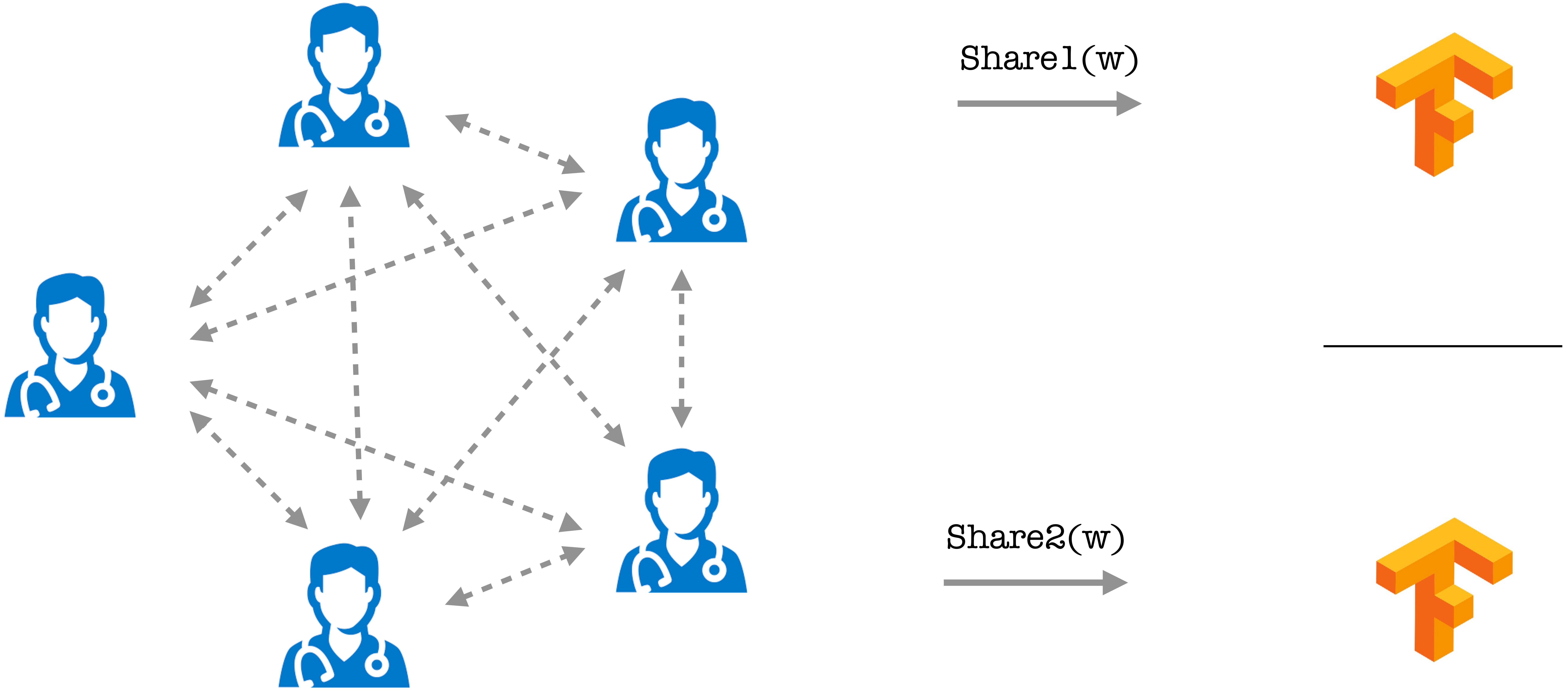


Server-Aided

Server-Aided



Multi-Party



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.

Blind Justice*: Fairness with Encrypted Sensitive Attributes*, Kilbertus et al.

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

Specialised projects

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

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

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/)*

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

Multidisciplinary Challenge

Data science
(use-cases, workflow, monitoring)

Cryptography
(techniques, protocols, trust)

Machine learning
(models, approx, precision)

Engineering
(distributed, multi-core, readability)

need for common language

tf-encrypted

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

separate concerns, take expertise out of equation,
and provide tight integration with ecosystem

Private Prediction with tf-encrypted

```python
import tensorflow as tf

def provide_weights():""" Load from disk """
def provide_input(): """ Pre-process """
def receive_output(logits): return tf.Print([], [tf.argmax(logits)])

# get model weights
w0, b0, w1, b1, w2, b2 = provide_weights()

# get prediction input
x = provide_input()

# compute prediction
layer0 = tf.nn.relu((tf.matmul(x, w0) + b0))
layer1 = tf.nn.relu((tf.matmul(layer0, w1) + b1))
logits = tf.matmul(layer2, w2) + b2

# process result of prediction
prediction_op = receive_output(logits)

# run graph execution in a tf.Session
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    sess.run(prediction_op)
```

```python
import tensorflow as tf
import tf_encrypted as tfe

def provide_weights():""" Load from disk """
def provide_input(): """ Pre-process """
def receive_output(logits): return tf.Print([], [tf.argmax(logits)])

# get model weights as private tensors from owner
w0, b0, w1, b1, w2, b2 = tfe.define_private_input("model-owner", provide_weights)

# get prediction input as private tensors from client
x = tfe.define_private_input("prediction-client", provide_input)

# compute private prediction on servers
layer0 = tfe.relu((tfe.matmul(x, w0) + b0))
layer1 = tfe.relu((tfe.matmul(layer0, w1) + b1))
logits = tfe.matmul(layer1, w2) + b2

# process result of prediction on client
prediction_op = tfe.define_output("prediction-client", logits, receive_output)

# run secure graph execution in a tf.Session
with tfe.Session() as sess:
    sess.run(tf.global_variables_initializer())
    sess.run(prediction_op)
```

Private Prediction with tf-encrypted

```python
import tensorflow as tf

def provide_weights():""" Load from disk """
def provide_input(): """ Pre-process """
def receive_output(logits): return tf.Print([], [tf.argmax(logits)])

# get model weights
w0, b0, w1, b1, w2, b2 = provide_weights()

# get prediction input
x = provide_input()

# compute prediction
layer0 = tf.nn.relu((tf.matmul(x, w0) + b0))
layer1 = tf.nn.relu((tf.matmul(layer0, w1) + b1))
logits = tf.matmul(layer2, w2) + b2

# process result of prediction
prediction_op = receive_output(logits)

# run graph execution in a tf.Session
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    sess.run(prediction_op)
```

```python
import tensorflow as tf
import tf_encrypted as tfe

def provide_weights():""" Load from disk """
def provide_input(): """ Pre-process """
def receive_output(logits): return tf.Print([], [tf.argmax(logits)])

# get model weights as private tensors from owner
w0, b0, w1, b1, w2, b2 = tfe.define_private_input("model-owner", provide_weights)

# get prediction input as private tensors from client
x = tfe.define_private_input("prediction-client", provide_input)

# compute private prediction on servers
layer0 = tfe.relu((tfe.matmul(x, w0) + b0))
layer1 = tfe.relu((tfe.matmul(layer0, w1) + b1))
logits = tfe.matmul(layer1, w2) + b2

# process result of prediction on client
prediction_op = tfe.define_output("prediction-client", logits, receive_output)

# run secure graph execution in a tf.Session
with tfe.Session() as sess:
    sess.run(tf.global_variables_initializer())
    sess.run(prediction_op)
```

Wrap-Up

You can **compute on encrypted data**,
without the ability to decrypt

Privacy-preserving ML mitigate **bottlenecks** and
enable access to sensitive information

Secure computation **distributes trust and control**,
and is complementary to e.g. differential privacy

Privacy-preserving ML is a multidisciplinary field
benefitting from **adaptations** on both sides

Focus on **usability** and **integration**

Thank you!