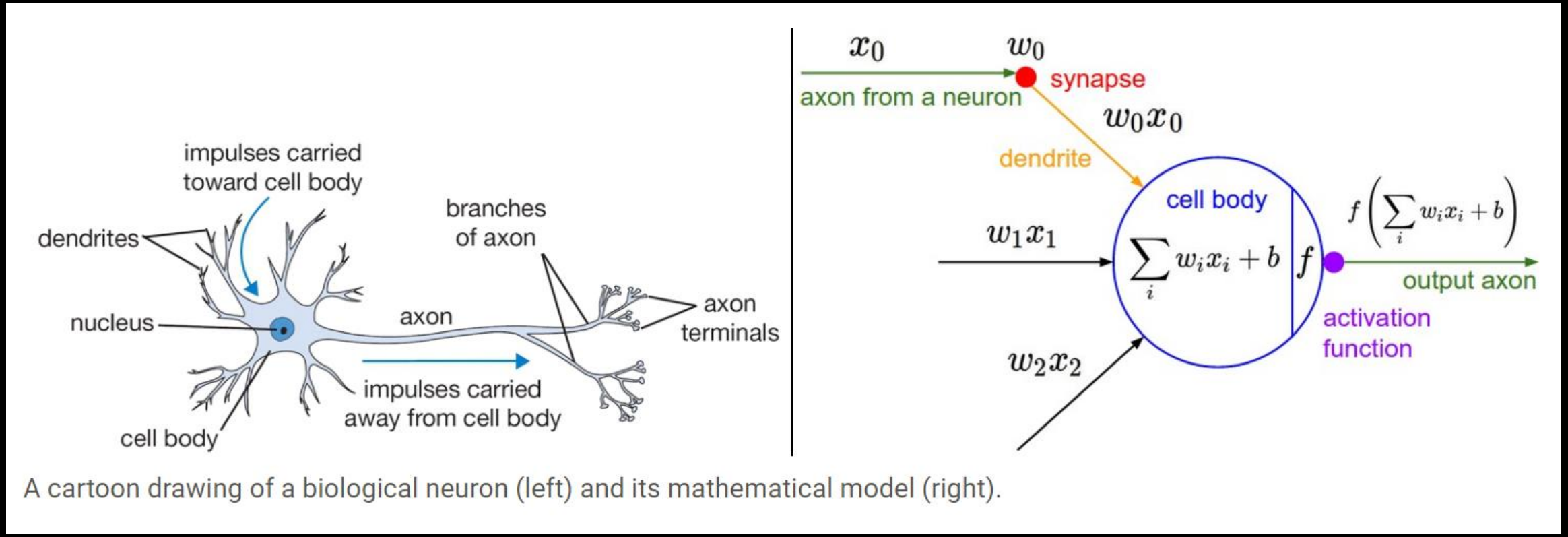
Deep Learning with Tensorflow

Speaker: Xingjun (Daniel) Ma

School of Computing and Information Systems,
The University of Melbourne

Personal page: *xingjunma.com*

Deep Neural Networks

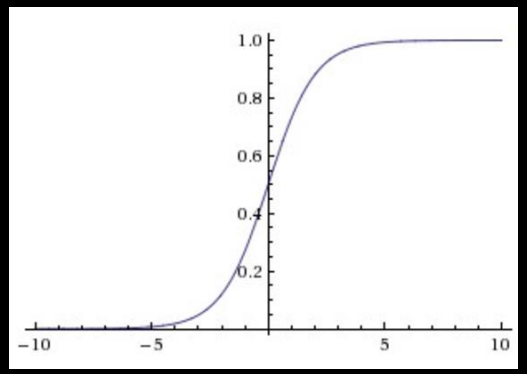


A cartoon drawing of a biological neuron (left) and its mathematical model (right).

1. Many neurons stacked in layers: input → hidden (…) → output
2. A transformation graph with many nodes

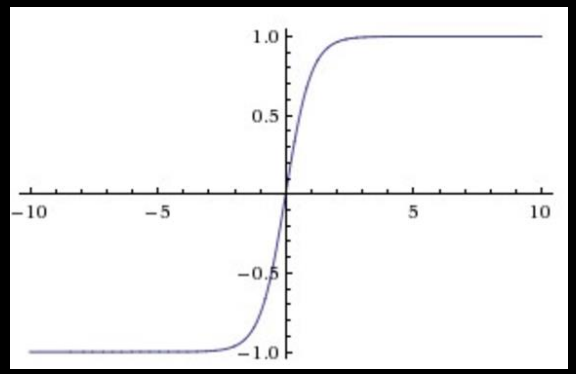
Design Deep Neural Networks?

Step 1: Need a Transformation Unit – which type of neuron to use?



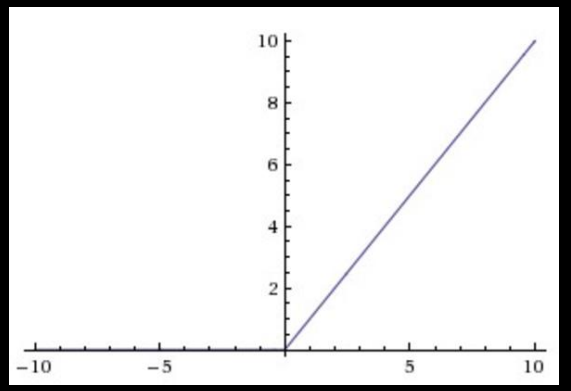
sigmoid

1. shallow networks
2. not efficient
3. gradient problem:
   too flat on both sides



Tanh

1. **symmetric: stronger gradients**
2. not efficient:
   $\tanh(x)=2\cdot\text{sigmoid}(2x)-1$
3. gradient problem



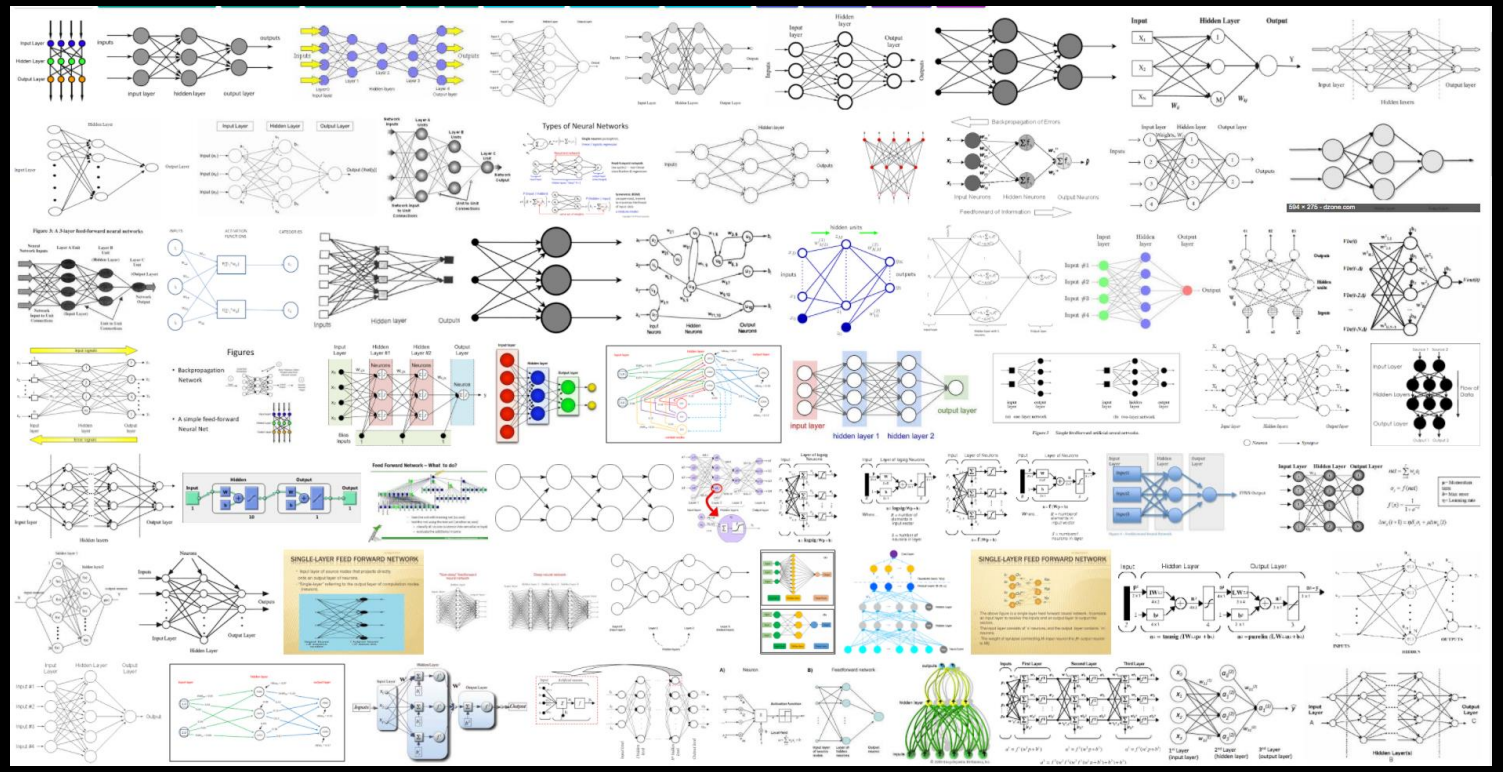
ReLU √

1. **efficient: deep networks**
2. no gradient problem
3. sparsity

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks." *ICAIS*. 2011.

Design Deep Neural Networks?

Step 2: wire up!

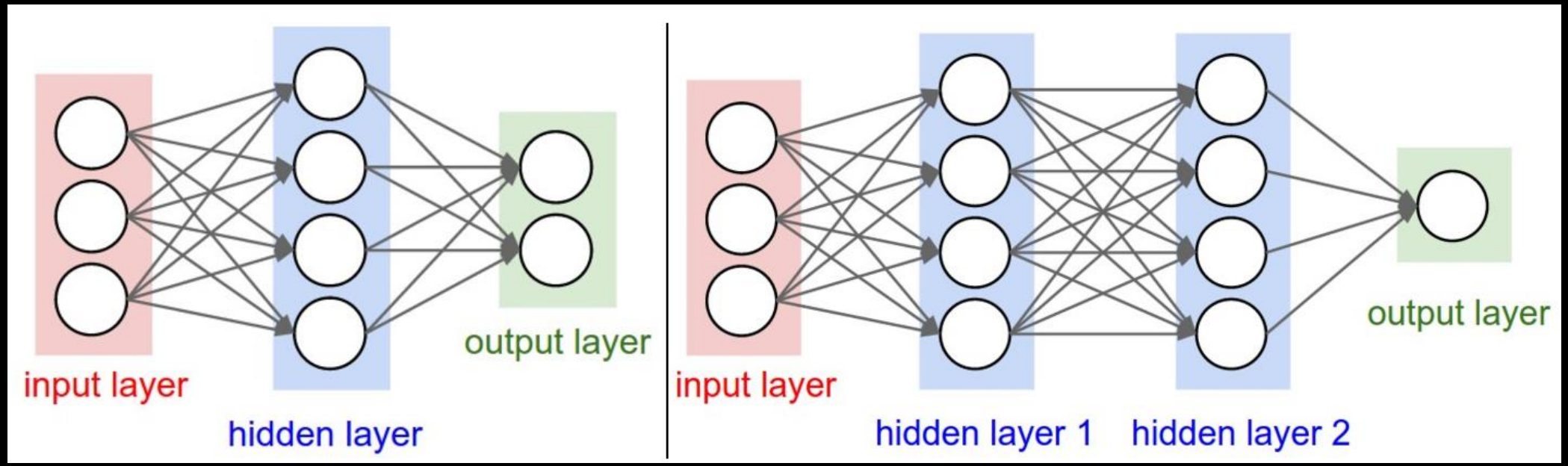


So many ways!

Google Search: deep neural networks

Popular Architectures

- Feed-forward Networks

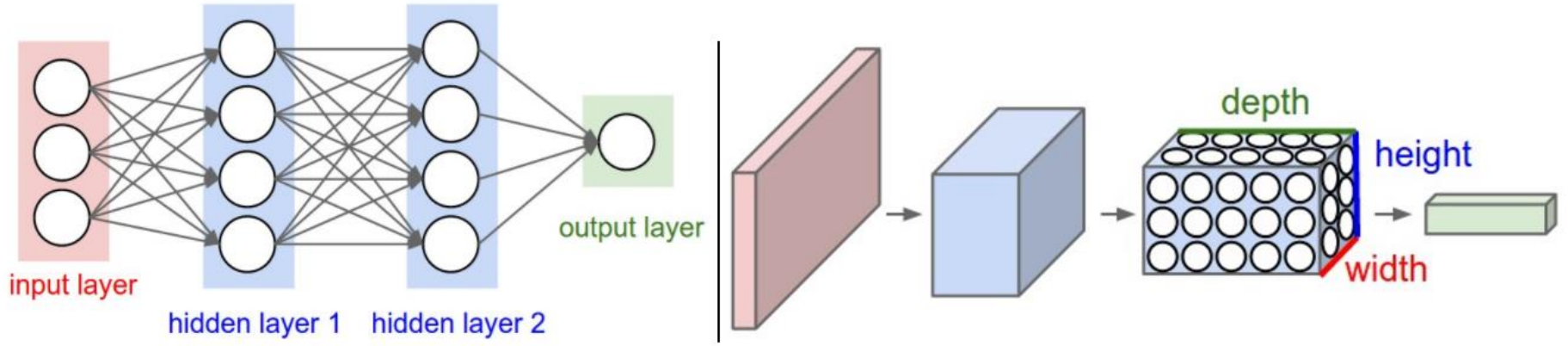


1. fully connected between layers
2. data that has NO temporal or spatial order

Bebis, George, and Michael Georgiopoulos. "Feed-forward neural networks." *IEEE Potentials* 13.4 (1994): 27-31.

Popular Architectures

- Convolutional Networks



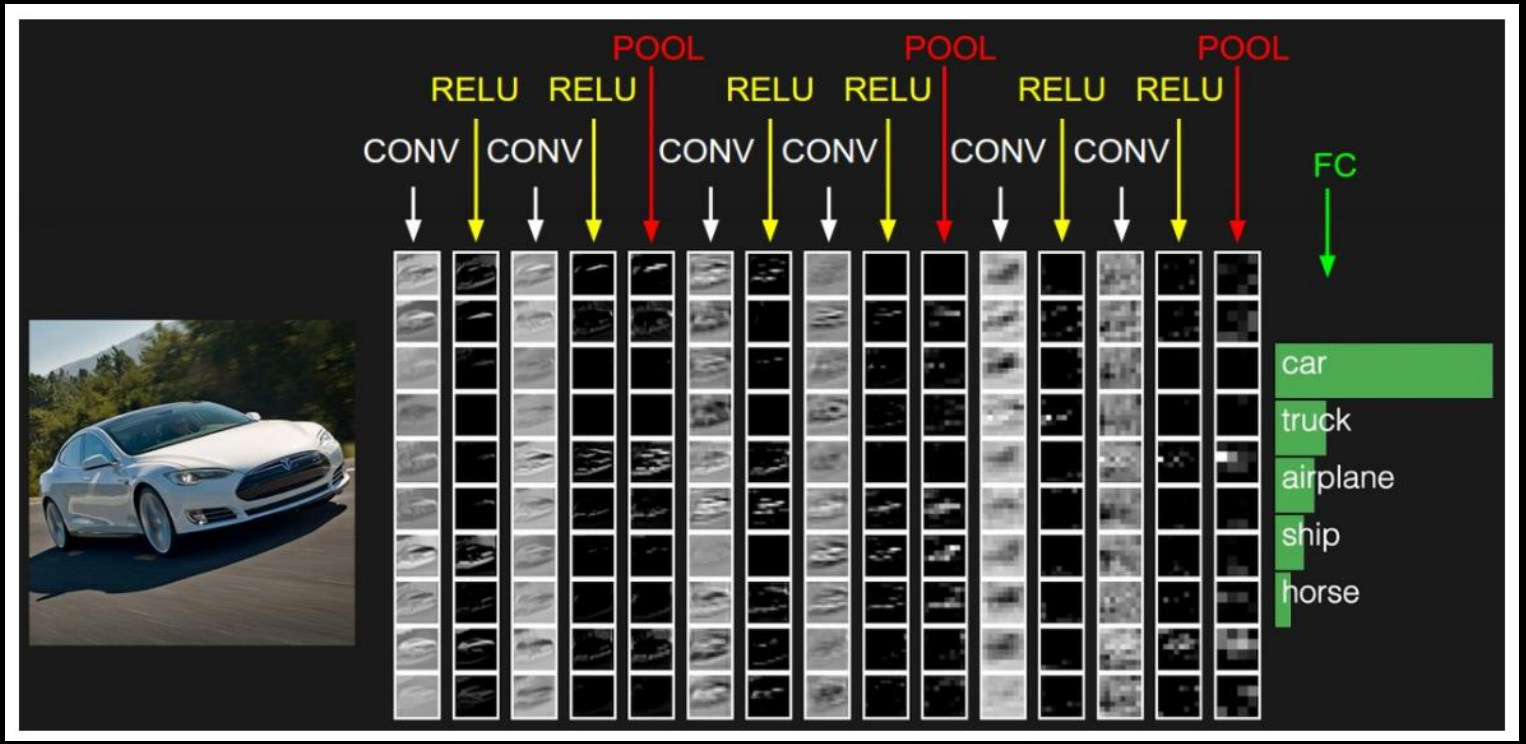
Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

1. For computer vision tasks: images, videos → spatial order

http://cs231n.stanford.edu/

Popular Architectures

- Convolutional Networks



- **LeNet, 1990's**
- **AlexNet, 2012**
- **ZF Net, 2013**
- **GoogLeNet, 2014**
- **VGGNet, 2014**
- **ResNet, 2015**
- **Inception V4, 2016**

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *NIPS*. 2012.

Implementation

Define your architecture in Tensorflow

- Import Tensorflow to use:

```
import tensorflow as tf
```

- Input:

```
# input X: 28x28 grayscale images
X = tf.placeholder(tf.float32, [None, 28, 28, 1])
```

- Output:

```
# 10 classes
Y_ = tf.placeholder(tf.float32, [None, 10])
```

Implementation

- Define neural network

```
# weights W[28,28,10]
W = tf.Variable(tf.zeros([28, 28, 10]))
# biases b[10]
b = tf.Variable(tf.zeros([10]))
# The model
Y = tf.nn.softmax(tf.matmul(X, W) + b)
```

- Define loss or objective

```
# cross entropy loss
cross_entropy = -tf.reduce_mean(Y_ * tf.log(Y))
```

Implementation

- Define an optimizer to minimize the loss

```
# training, learning rate = 0.005
train_step =
tf.train.GradientDescentOptimizer(0.005).minimize(cross_entropy)
```

- Create a session to train

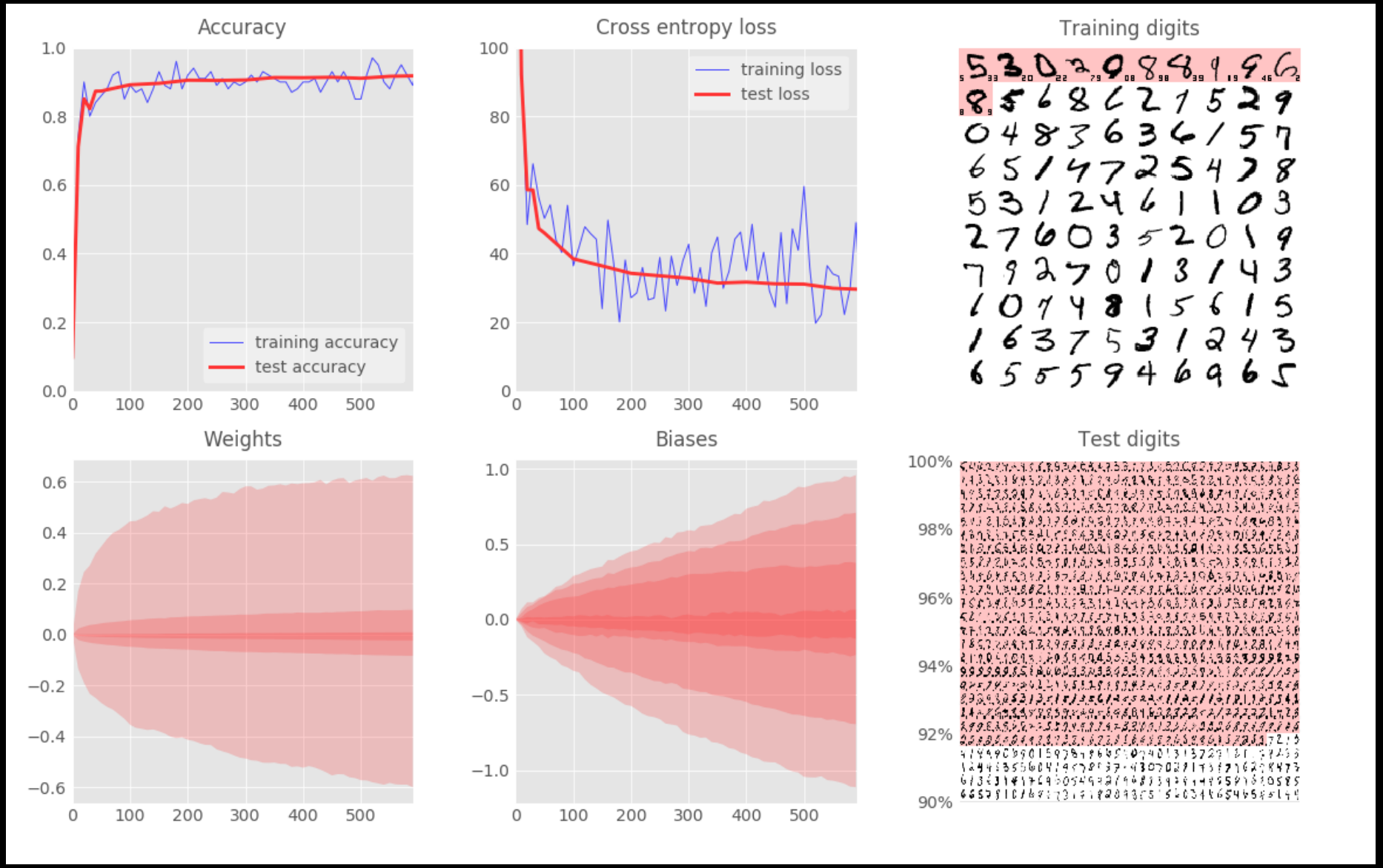
```
# init
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
```

Implementation

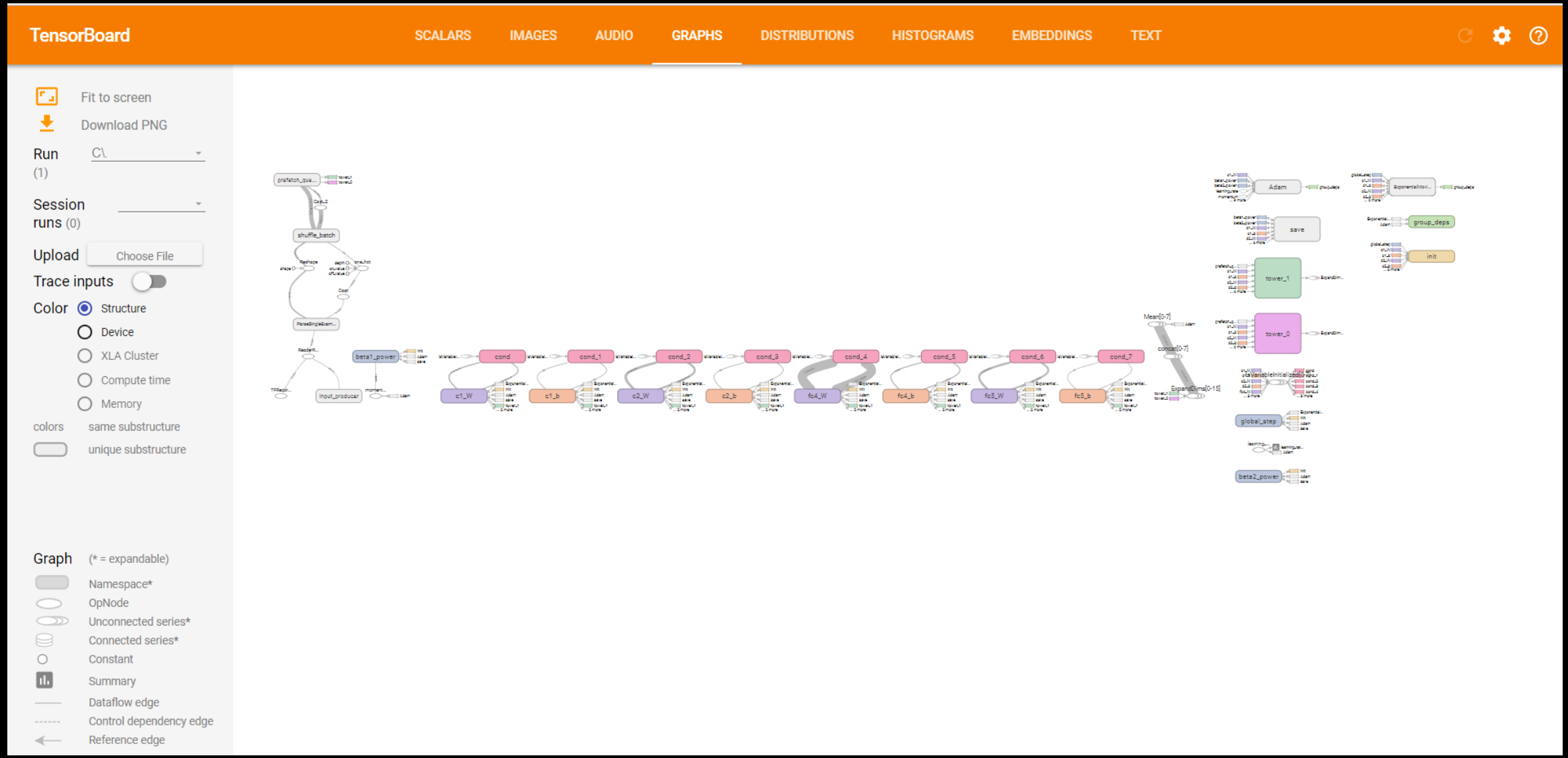
- Feed the data into the model in batch

```
for step in range(10000):
        # training on batches of 100 images with 100 labels
        batch_X, batch_Y = mnist.train.next_batch(100)
        # the backpropagation training step
        sess.run(train_step, feed_dict={X: batch_X, Y_: batch_Y})
```

Run the Code!


Accuracy
training accuracy
test accuracy
Cross entropy loss
training loss
test loss
Training digits
Weights
Biases
Test digits

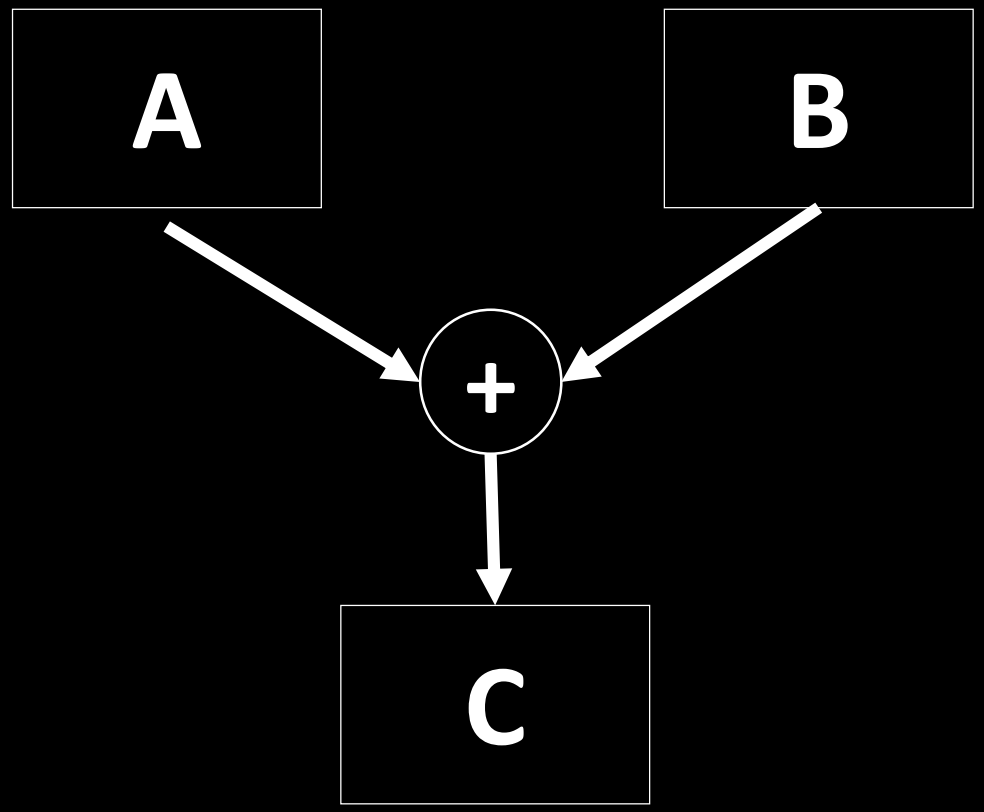
Dive into your model with Tensorboard!



What is Tensorflow anyway?

A + B = C

What is Tensorflow anyway?



Variable A, B, C:

1D: number

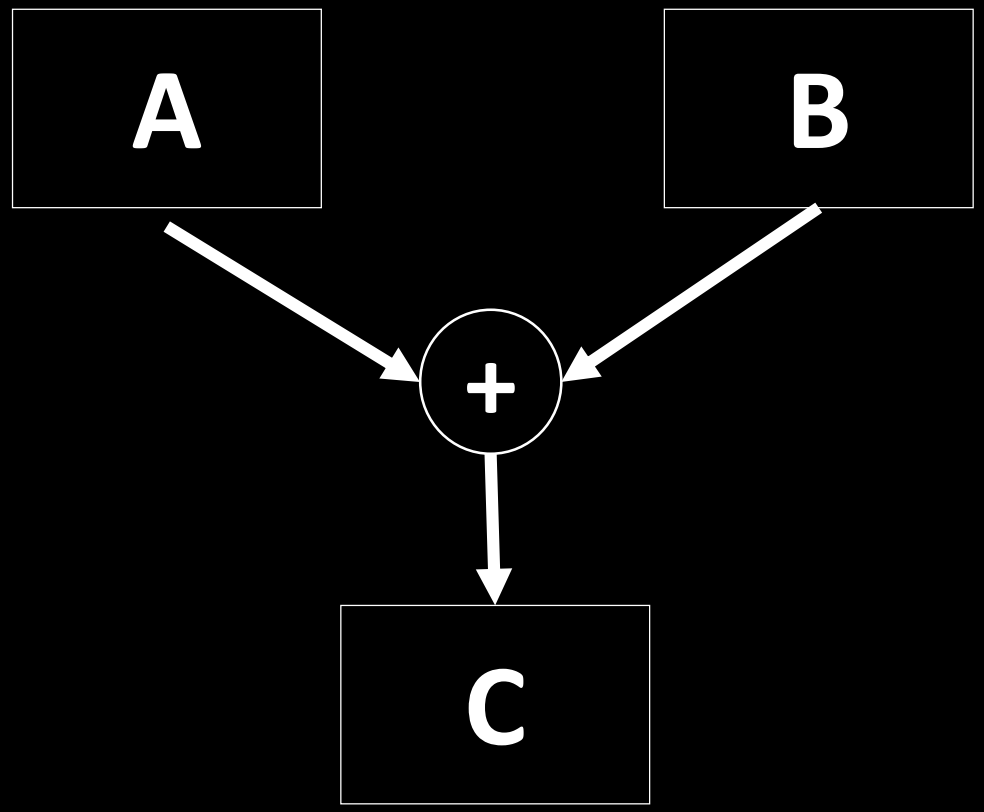
2D: vector

3D: matrix

…

nD: Tensor!

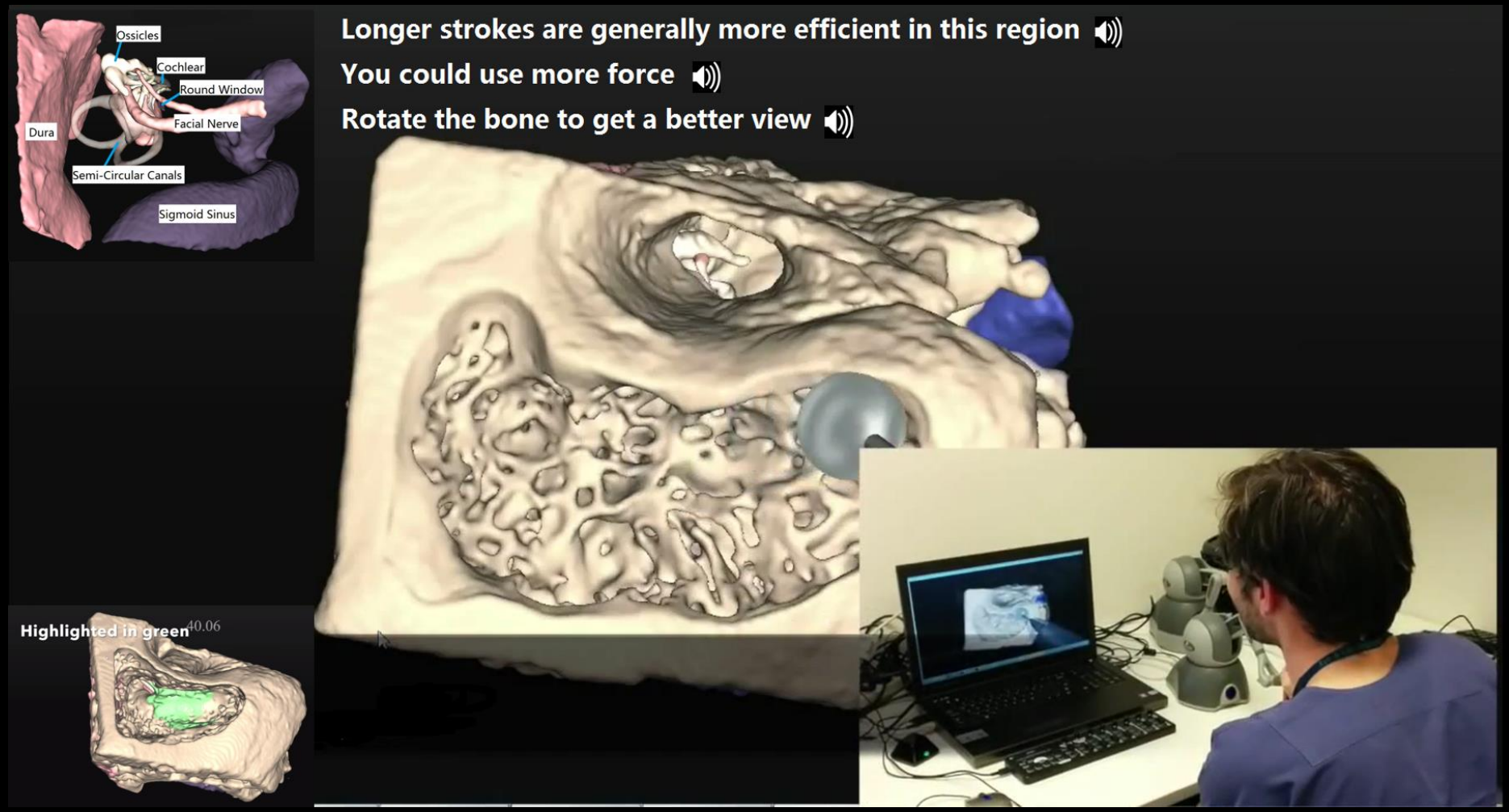
What is Tensorflow anyway?



Tensor ? Flow ?

Let nD variable flow in a graph!

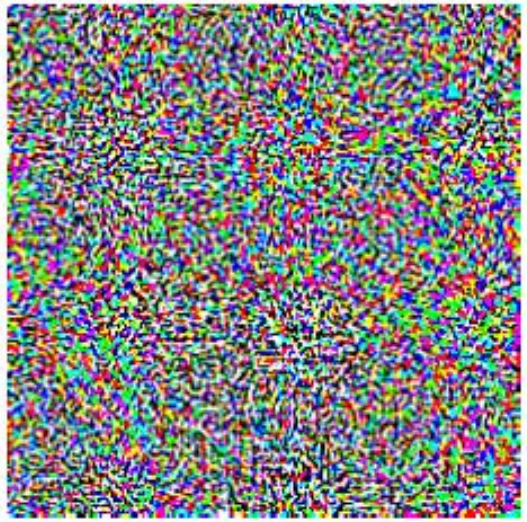
My research – medical training



Teach medical students to do surgery!

My research – adversarial learning

$+ .007 \times$



$=$

$\boldsymbol{x}$

"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

My research – adversarial learning

https://www.youtube.com/watch?v=9Yq67CjDqvw

https://www.youtube.com/watch?v=EFHyzuqjaok

My research – adversarial examples

Potential security threats to:

1. Identity verification
2. Face/fingerprint recognition
3. Autonomous cars
4. **forensic** analysis