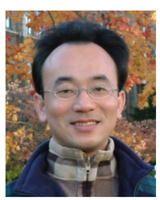
Tamkang University

Big Data Mining

Convolutional Neural Networks (CNN)

1071BDM11
TLVXM1A (M2244) (8619) (Fall 2018)
(MBA, DBETKU) (3 Credits, Required) [Full English Course]
(Master's Program in Digital Business and Economics)
Mon, 9, 10, 11, (16:10-19:00) (B206)



Min-Yuh Day, Ph.D.
Assistant Professor
Department of Information Management
Tamkang University

http://mail.tku.edu.tw/myday

2018-12-03

Course Schedule (1/2)

Tamkang University

Week	Date	Subject/Topics
1	2018/09/10	Course Orientation for Big Data Mining
2	2018/09/17	ABC: AI, Big Data, Cloud Computing
3	2018/09/24	Mid-Autumn Festival (Day off)
4	2018/10/01	Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data
5	2018/10/08	Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem
6	2018/10/15	Foundations of Big Data Mining in Python
7	2018/10/22	Supervised Learning: Classification and Prediction
8	2018/10/29	Unsupervised Learning: Cluster Analysis
9	2018/11/05	Unsupervised Learning: Association Analysis

Course Schedule (2/2)

Tamkang University

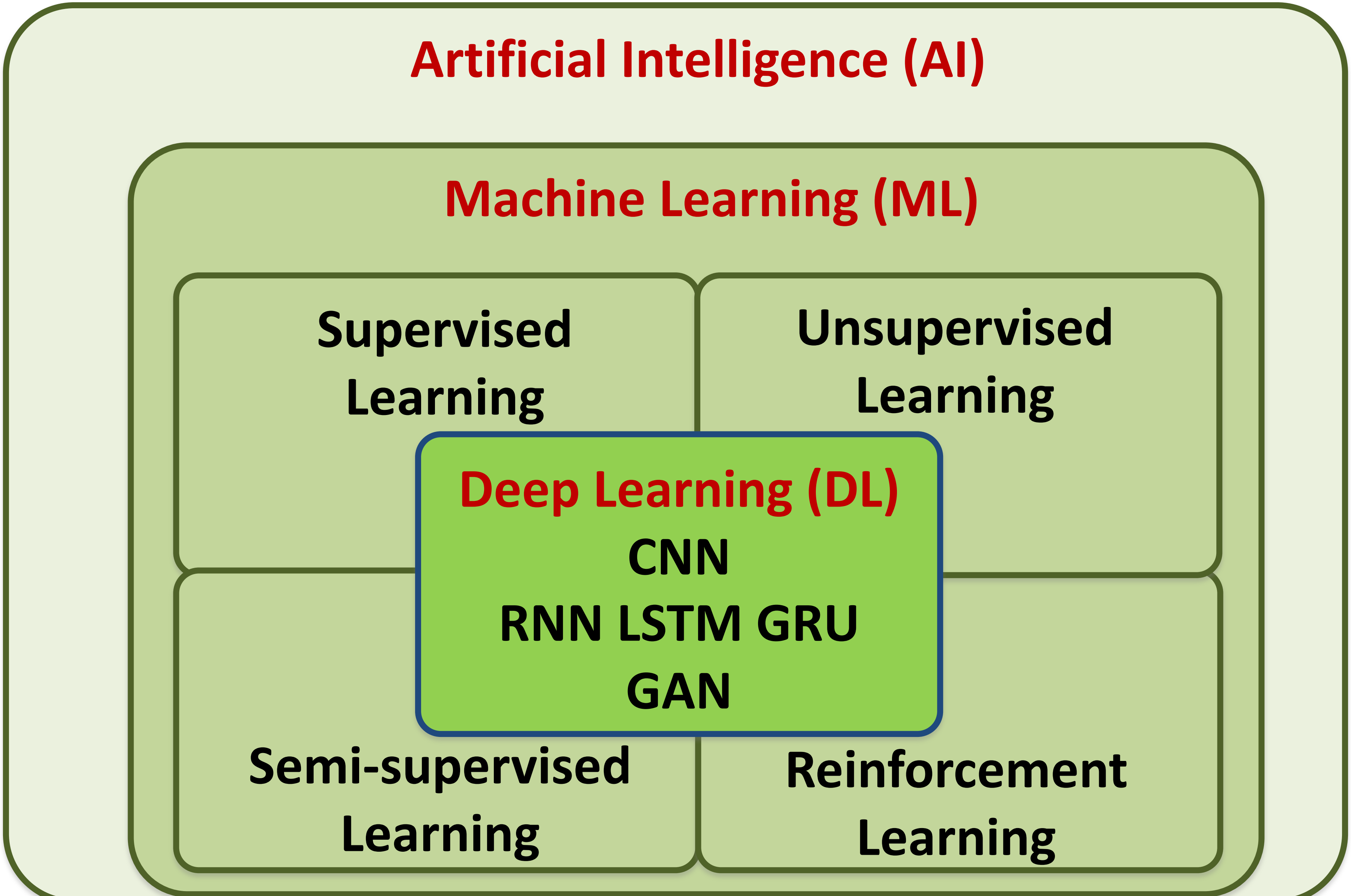
Week	Date	Subject/Topics
10	2018/11/12	Midterm Project Report
11	2018/11/19	Machine Learning with Scikit-Learn in Python
12	2018/11/26	Deep Learning for Finance Big Data with TensorFlow
13	2018/12/03	Convolutional Neural Networks (CNN)
14	2018/12/10	Recurrent Neural Networks (RNN)
15	2018/12/17	Reinforcement Learning (RL)
16	2018/12/24	Social Network Analysis (SNA)
17	2018/12/31	Bridge Holiday (Extra Day Off)
18	2019/01/07	Final Project Presentation

Convolutional Neural Networks (CNN)

Outline

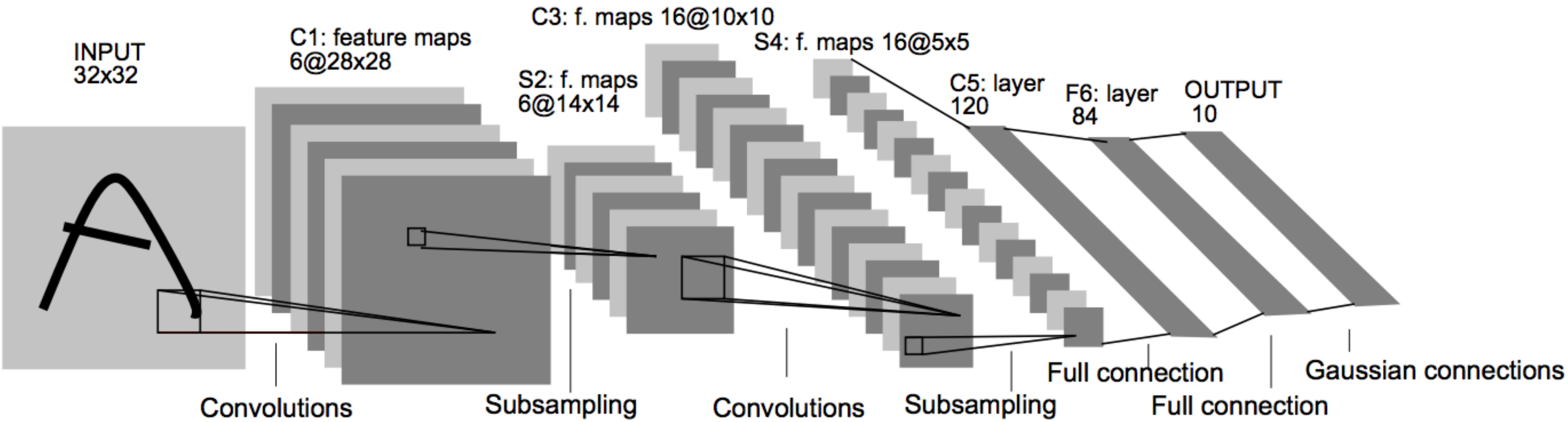
- Convolutional Neural Networks (CNN)
- TensorFlow Image Recognition

AI, ML, DL



Source: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/deep_learning.html

Convolutional Neural Networks (CNN)



Architecture of LeNet-5 (7 Layers) (LeCun et al., 1998)

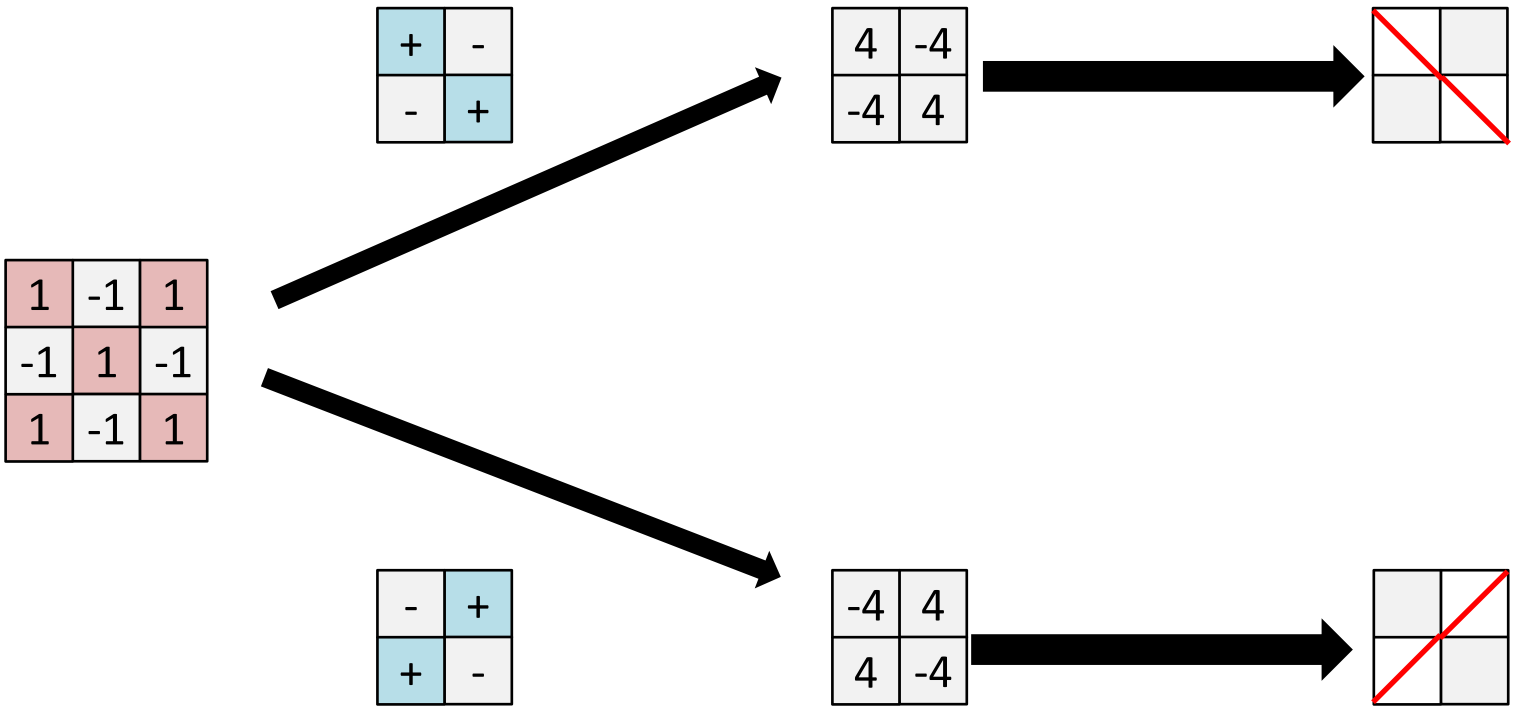
Source: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Source: LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86, no. 11 (1998): 2278-2324.

Convolutional Neural Networks (CNN)

- Convolution
- Pooling
- Fully Connection (FC) (Flattening)

A friendly introduction to Convolutional Neural Networks and Image Recognition

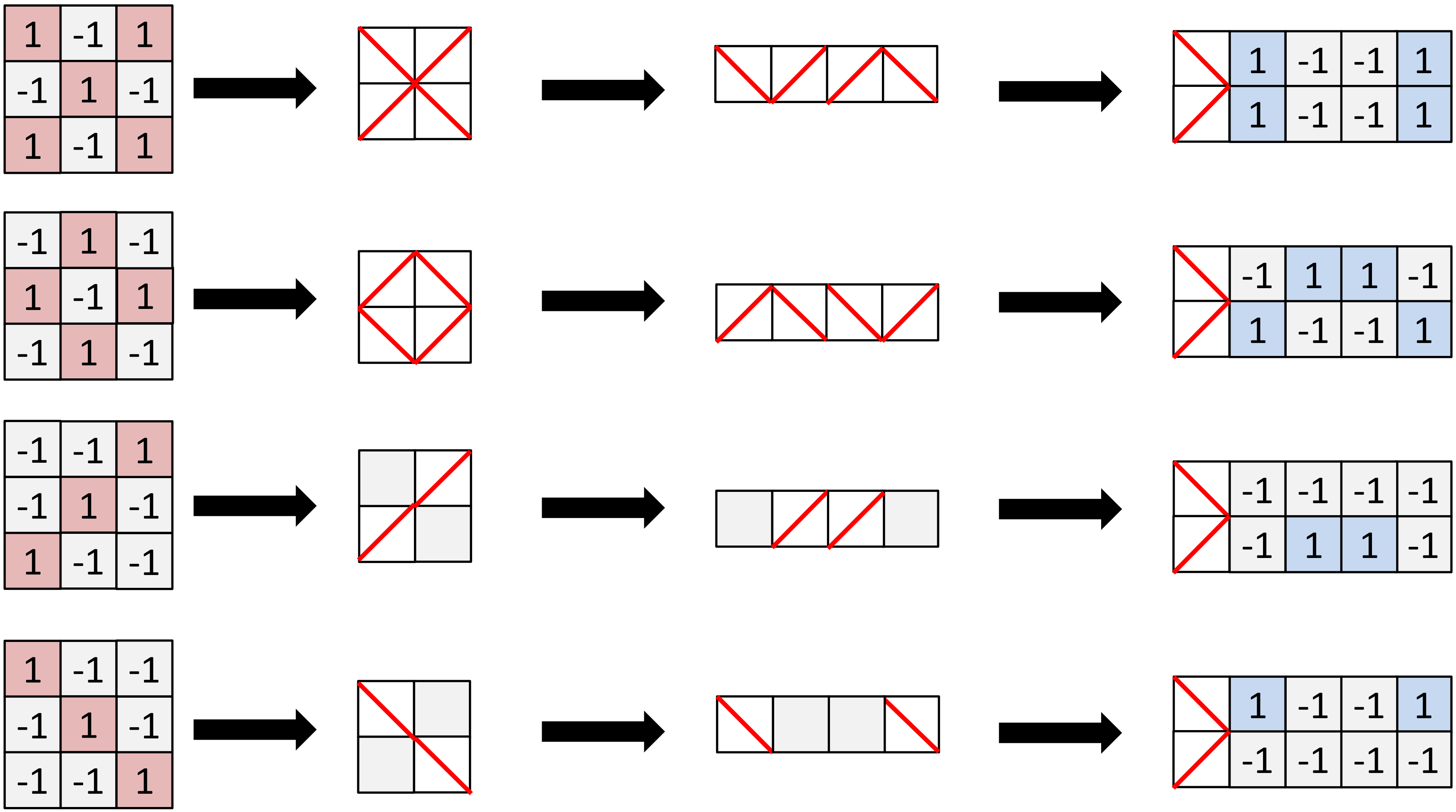


Convolution Layer

Pooling Layer

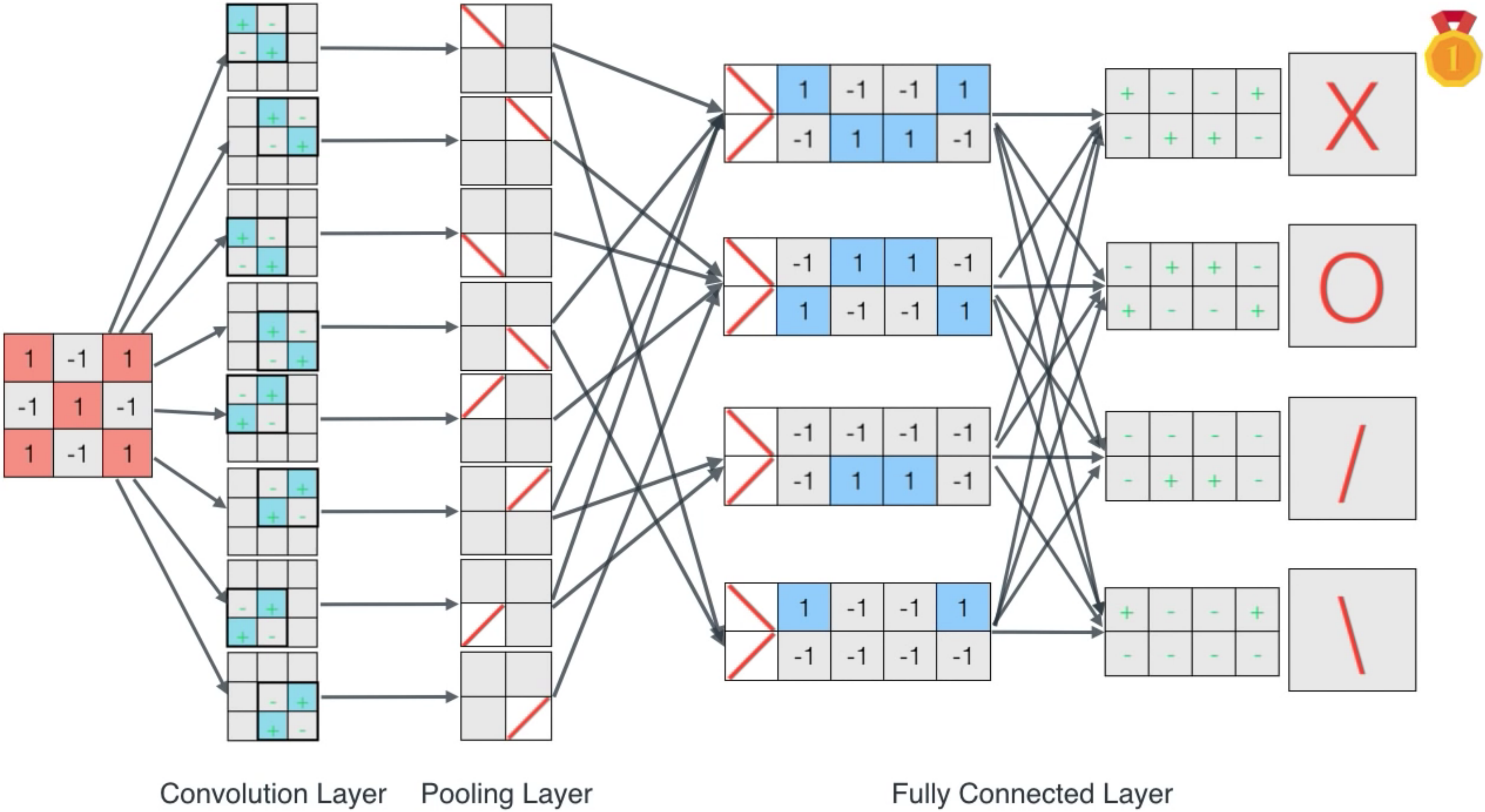
Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU

A friendly introduction to Convolutional Neural Networks and Image Recognition



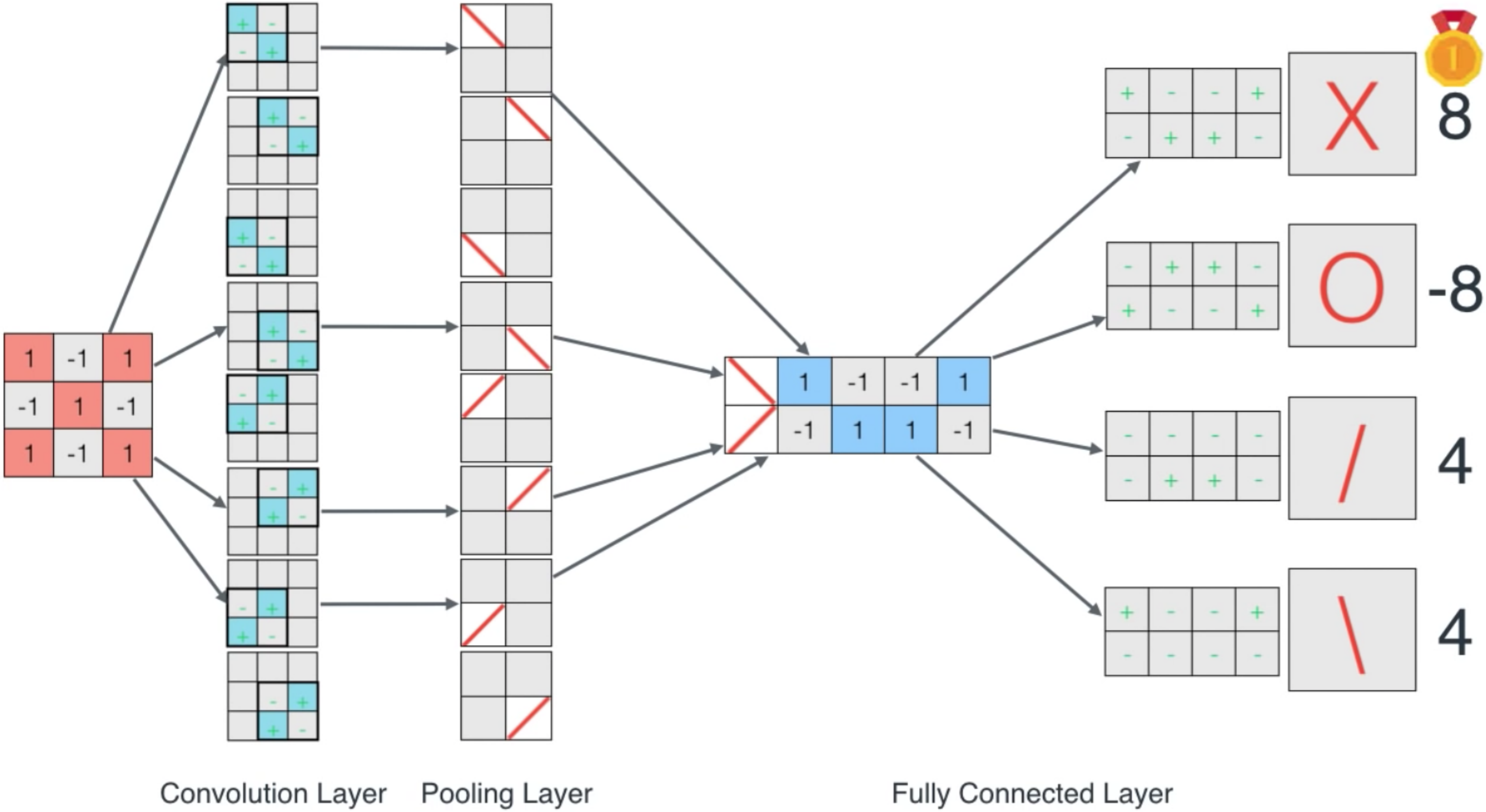
Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU

A friendly introduction to Convolutional Neural Networks and Image Recognition



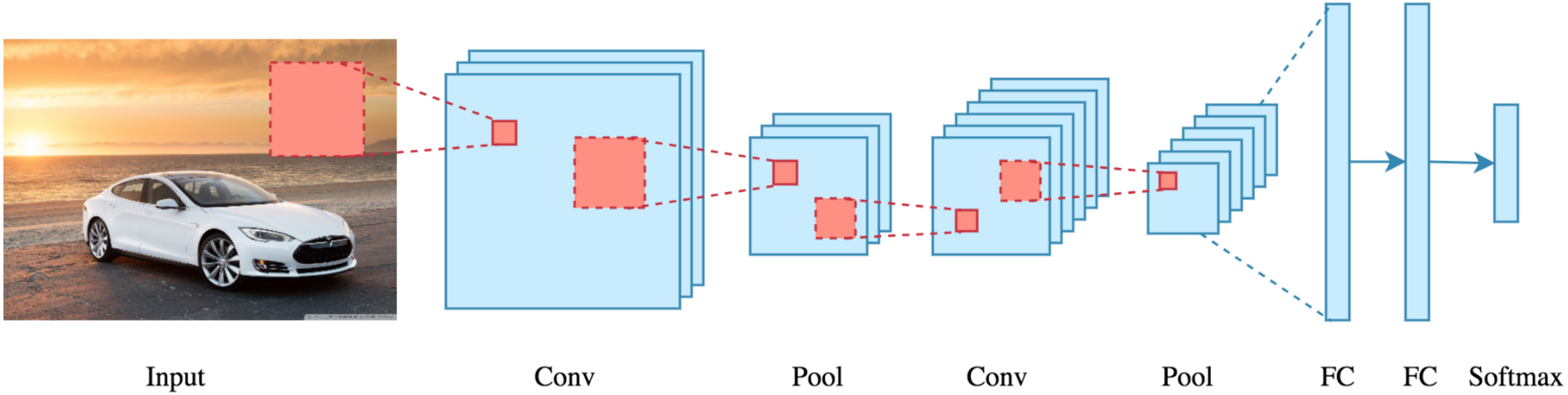
Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU

A friendly introduction to Convolutional Neural Networks and Image Recognition



Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU

CNN Architecture



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

Convolution is a **mathematical operation** to **merge two sets** of information

3x3 convolution

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

1	0	1
0	1	0
1	0	1

Filter / Kernel

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer
Input x Filter --> Feature Map

receptive field: 3x3

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

Input x Filter

4		

Feature Map

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer
Input x Filter --> Feature Map

receptive field: 3x3

1	1x1	1x0	0x1	0
0	1x0	1x1	1x0	0
0	0x1	1x0	1x1	1
0	0	1	1	0
0	1	1	0	0

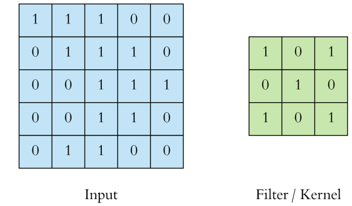
Input x Filter

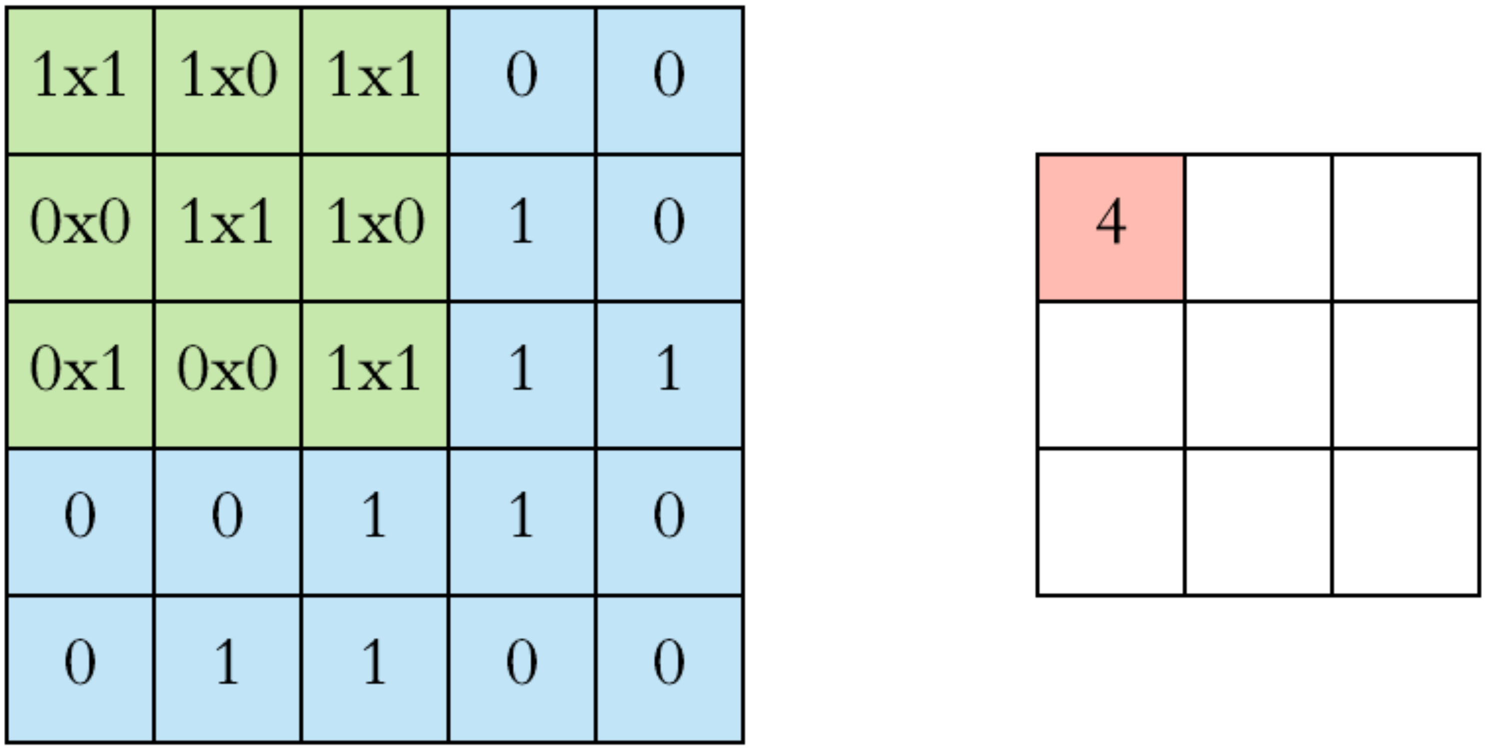
4	3	

Feature Map

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

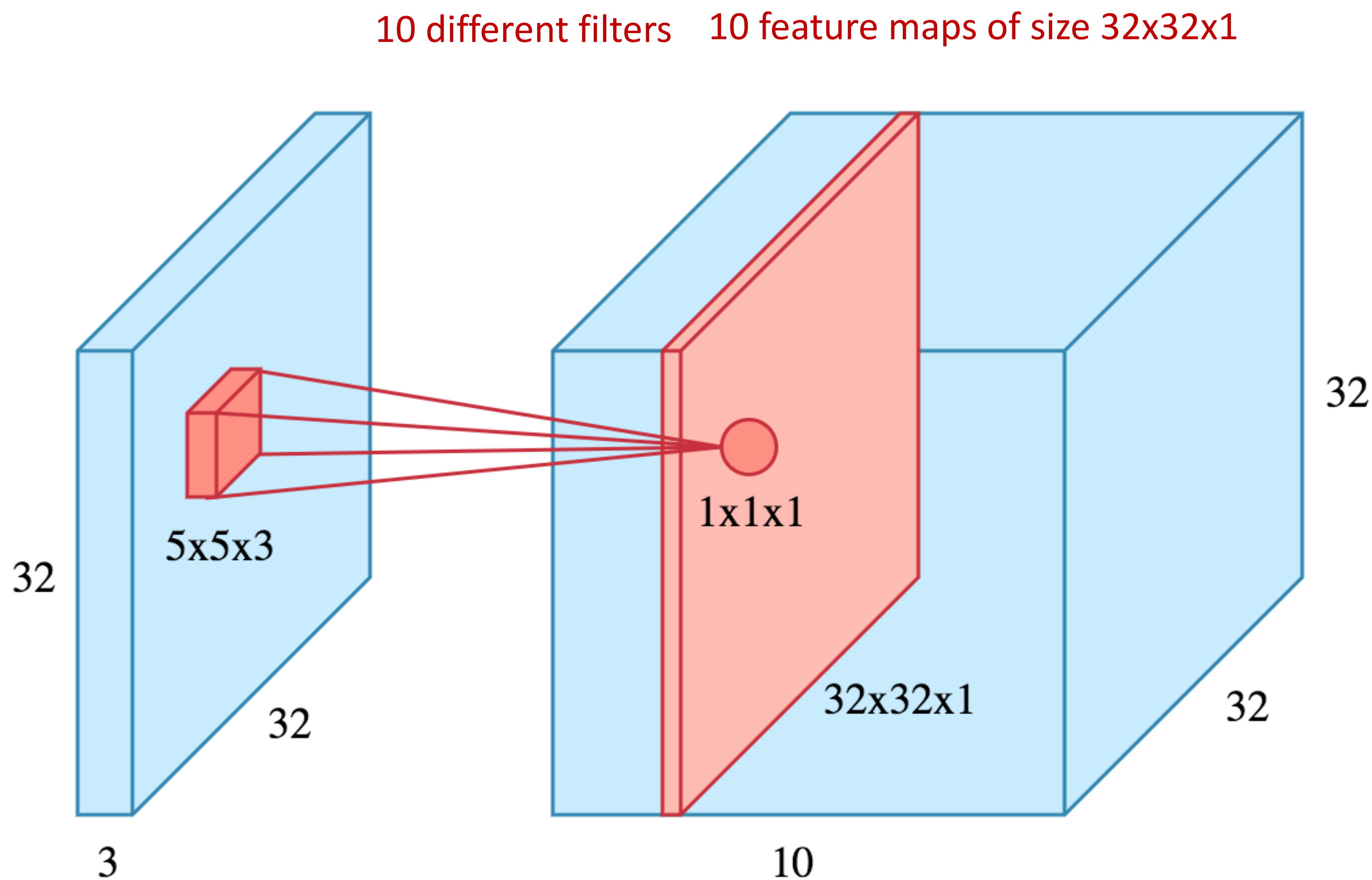




Example convolution operation shown in 2D using a 3x3 filter

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

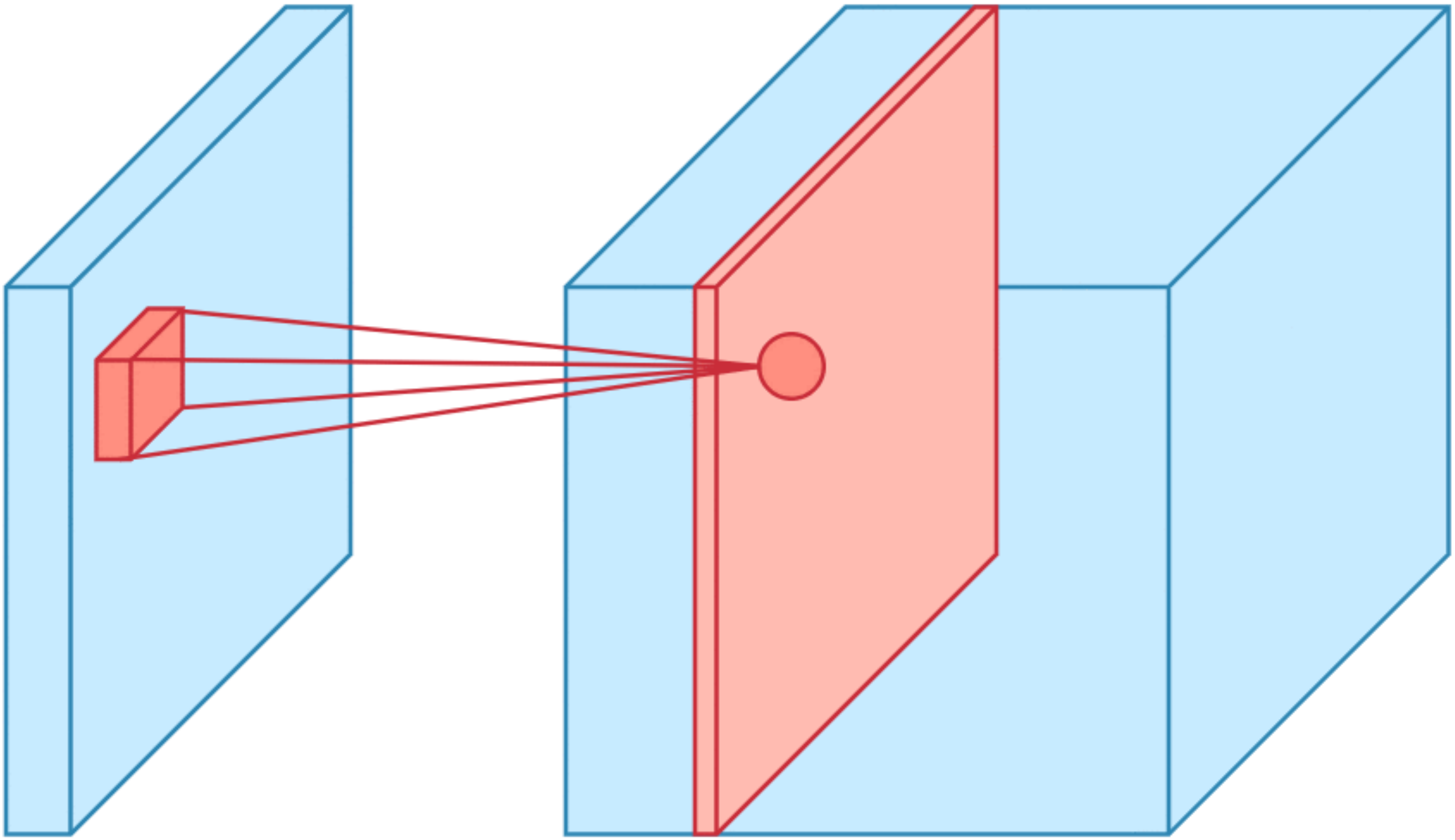
CNN Convolution Layer



final output of the convolution layer:
a volume of size 32x32x10

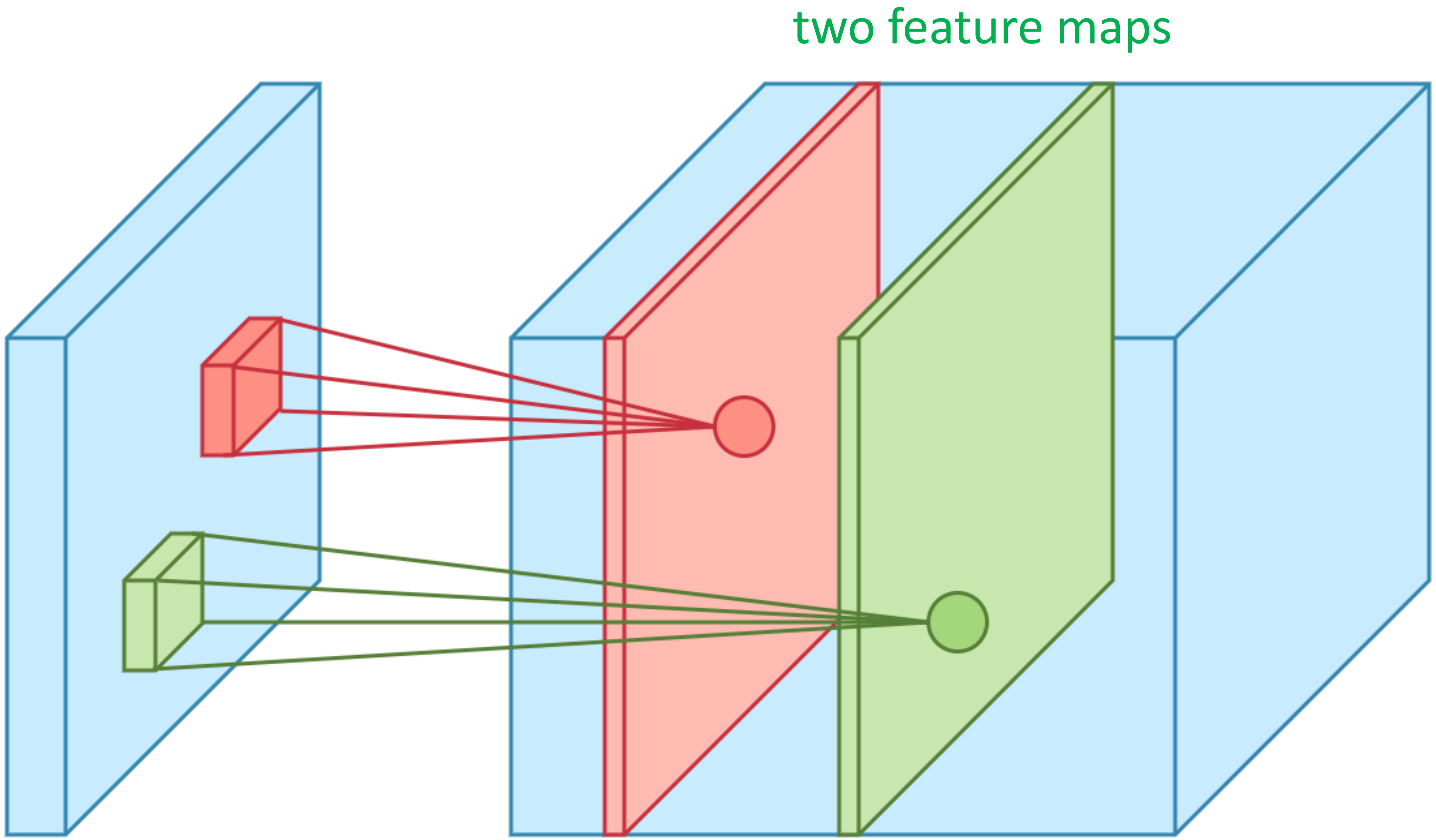
Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer
Sliding operation at 4 locations



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

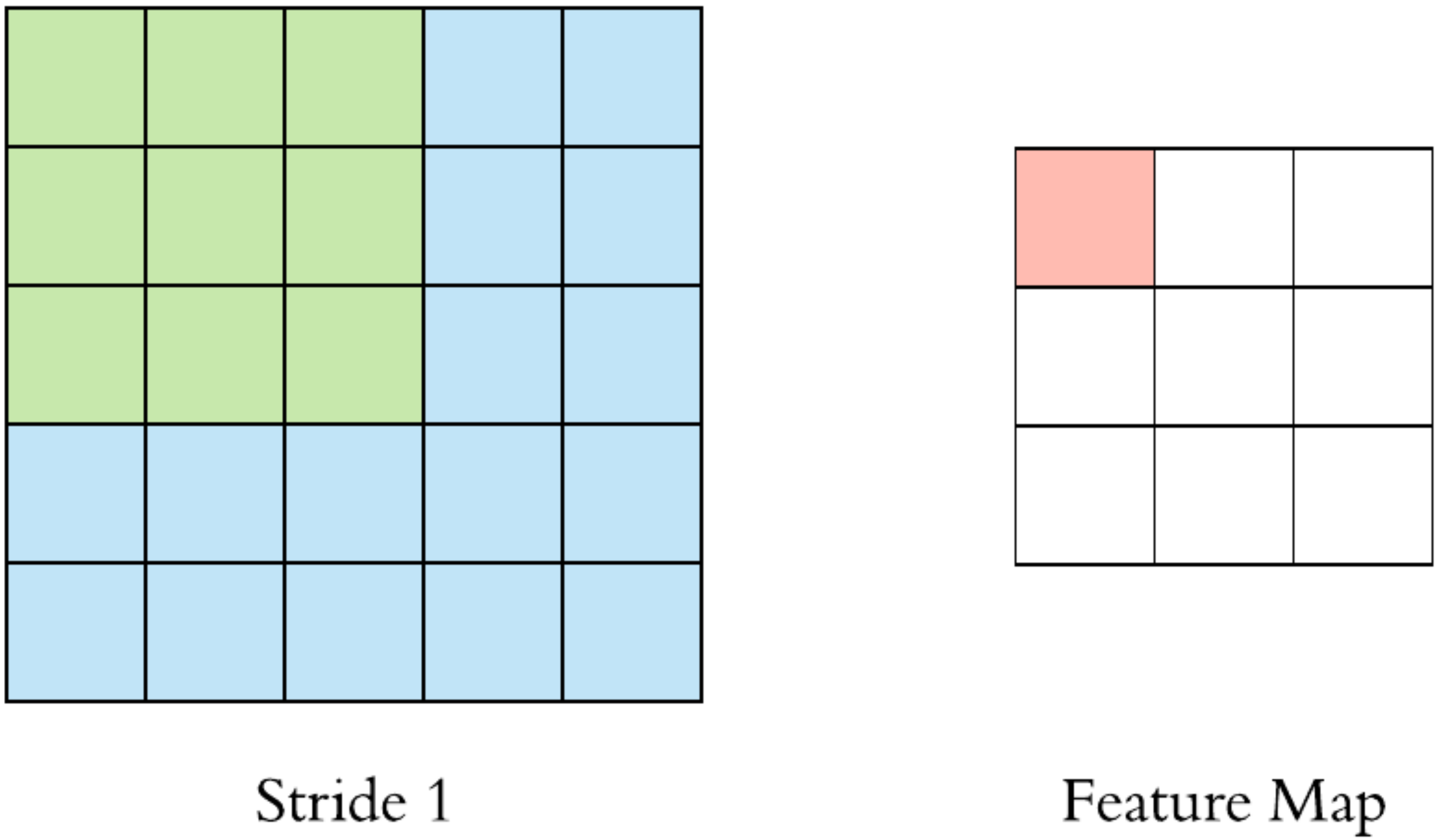
CNN Convolution Layer



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

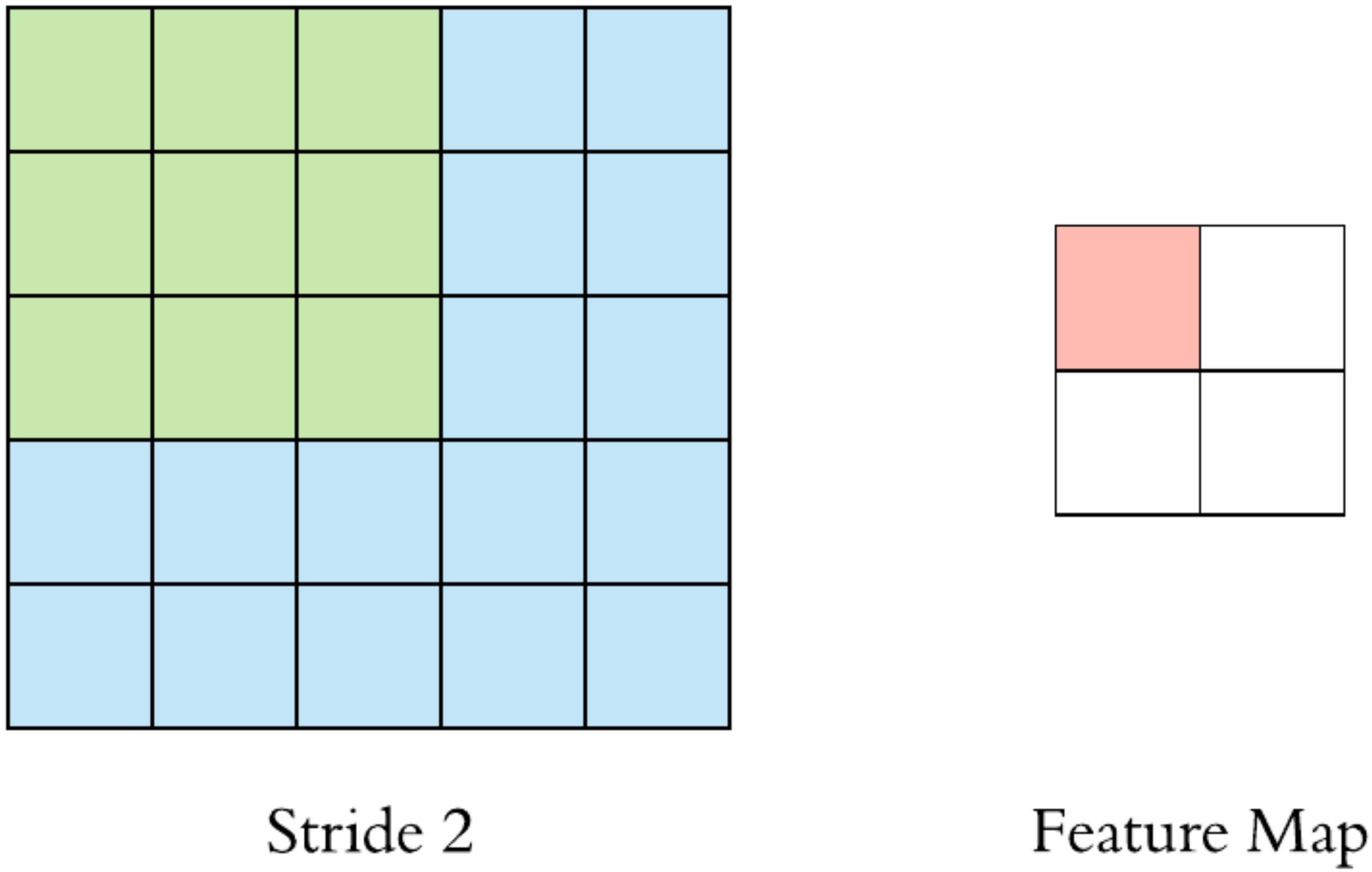
Stride specifies how much
we move the convolution filter at each step



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

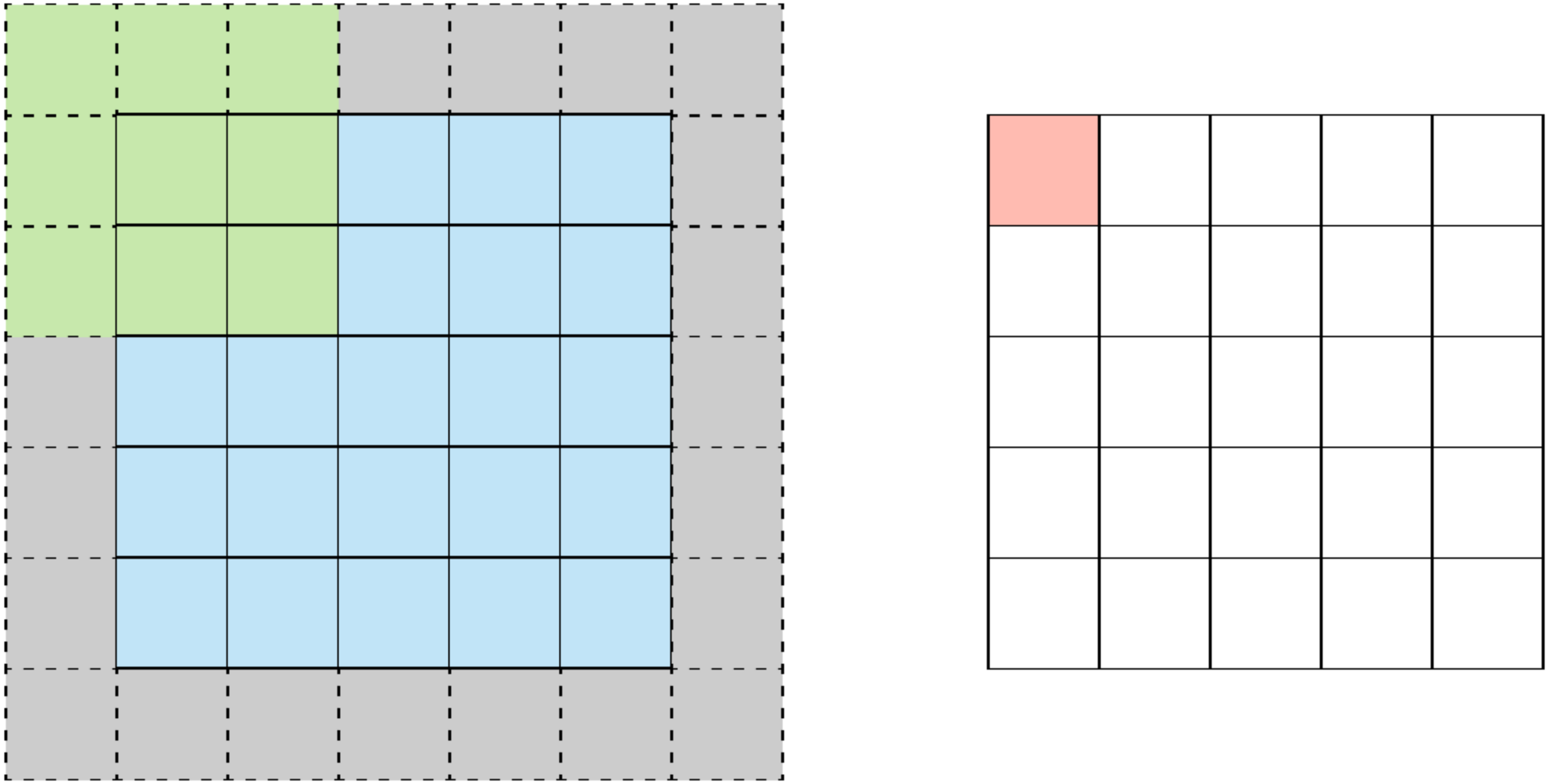
Stride specifies how much
we move the convolution filter at each step



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

Stride 1 with Padding



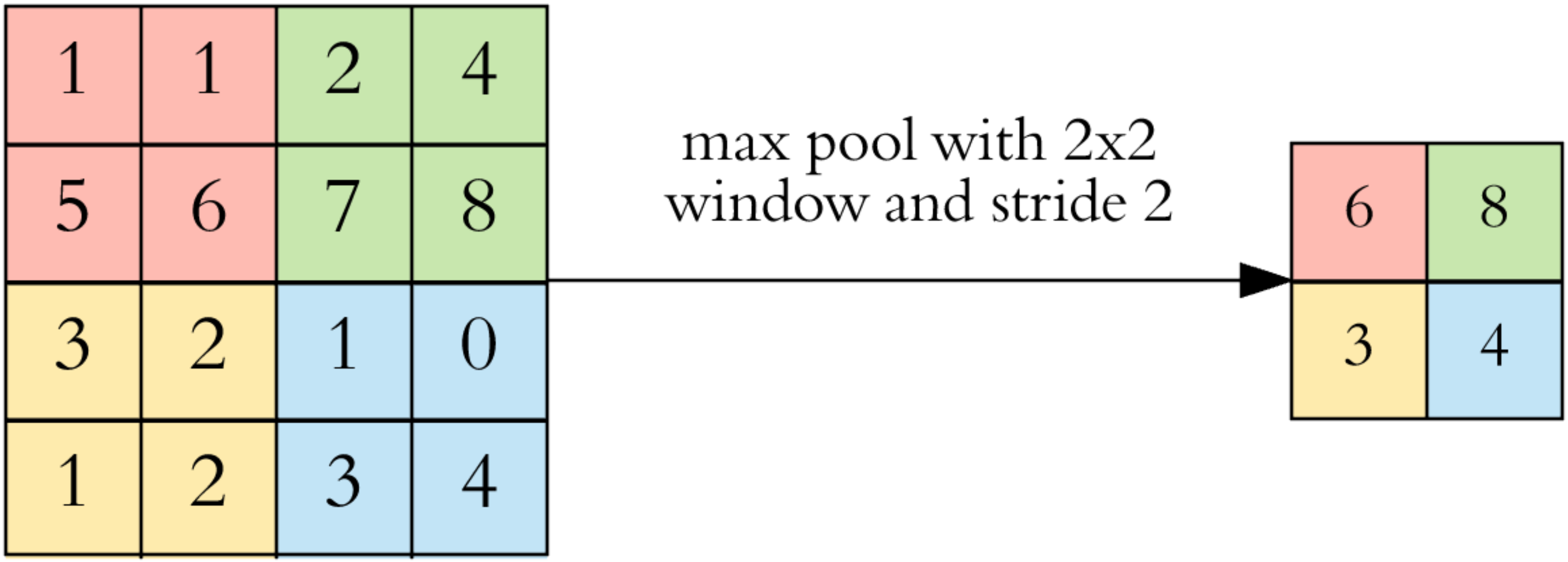
Stride 1 with Padding

Feature Map

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

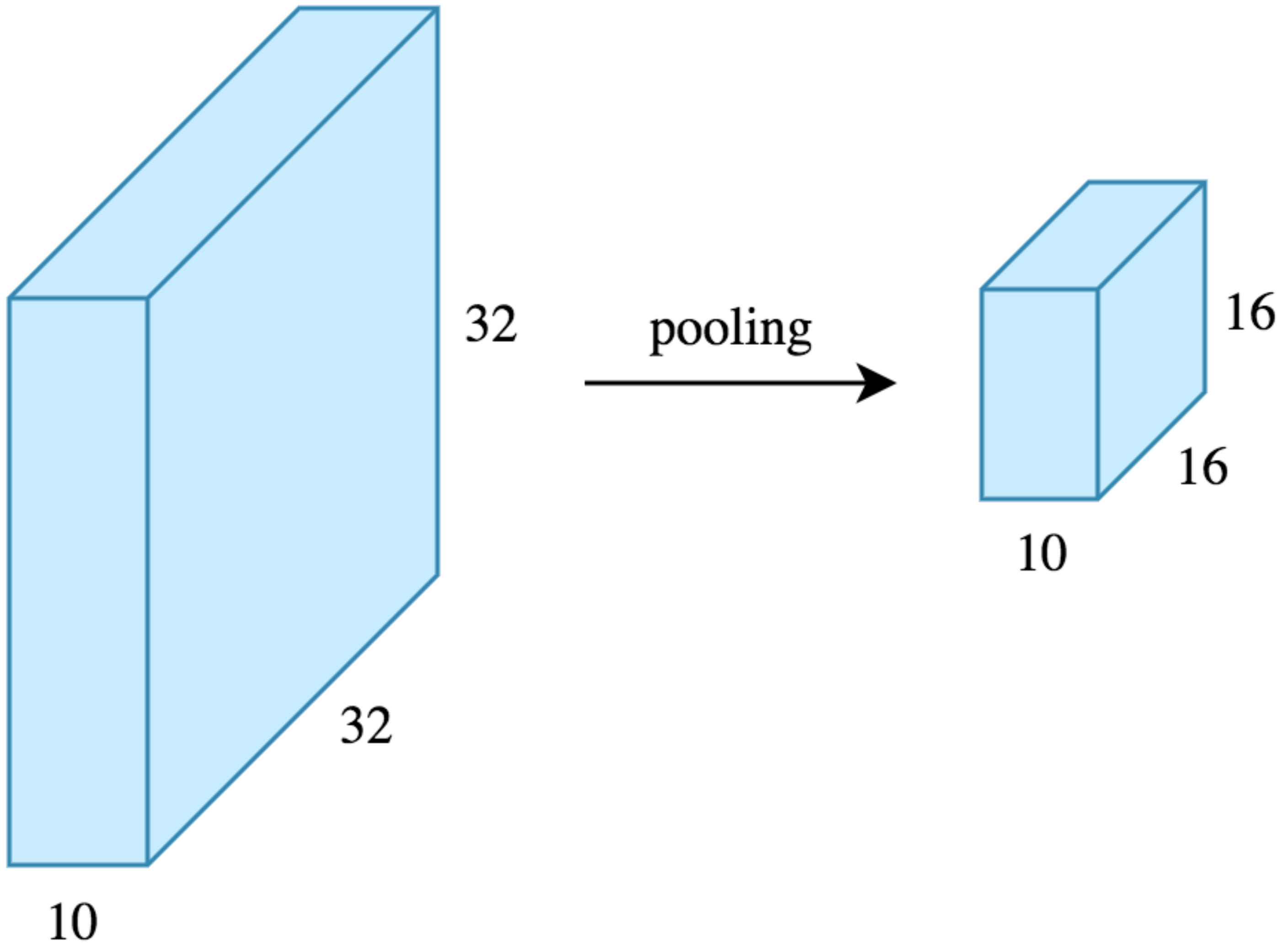
CNN Pooling Layer

Max Pooling



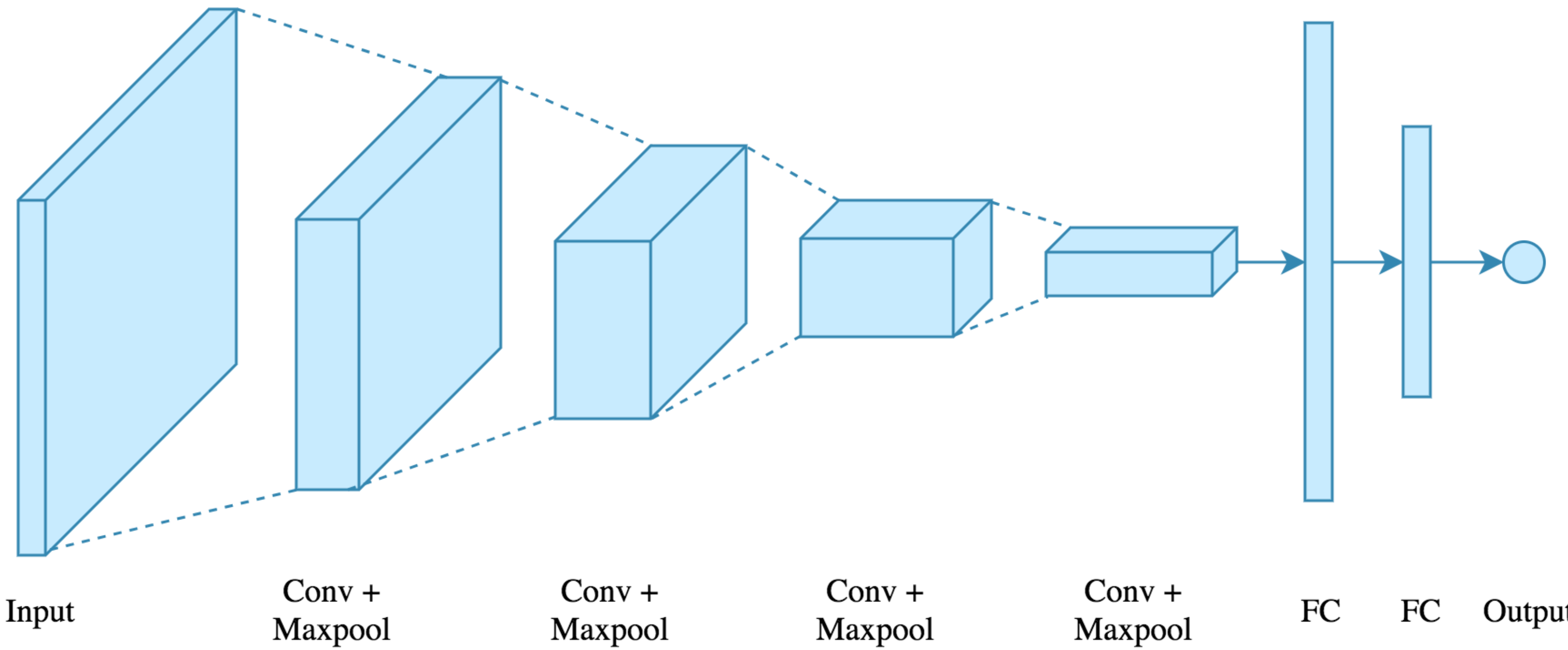
Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN **Pooling** Layer



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Architecture
4 convolution + pooling layers, followed by 2 fully connected layers



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

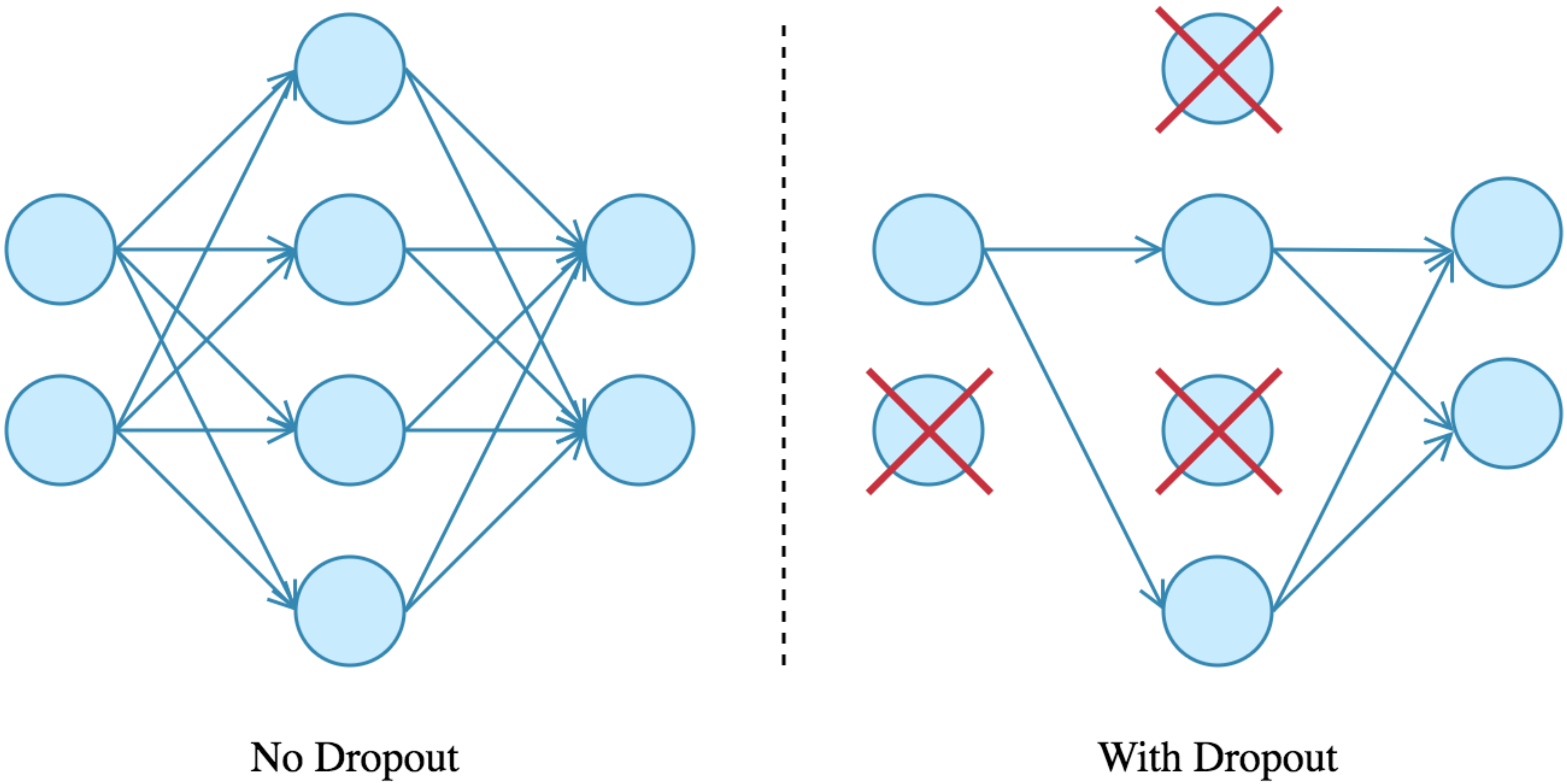
CNN Architecture
4 convolution + pooling layers, followed by 2 fully connected layers

https://gist.github.com/ardendertat/0fc5515057c47e7386fe04e9334504e3

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', name='conv_1',
                 input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2), name='maxpool_1'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', name='conv_2'))
model.add(MaxPooling2D((2, 2), name='maxpool_2'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_3'))
model.add(MaxPooling2D((2, 2), name='maxpool_3'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_4'))
model.add(MaxPooling2D((2, 2), name='maxpool_4'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu', name='dense_1'))
model.add(Dense(128, activation='relu', name='dense_2'))
model.add(Dense(1, activation='sigmoid', name='output'))
```

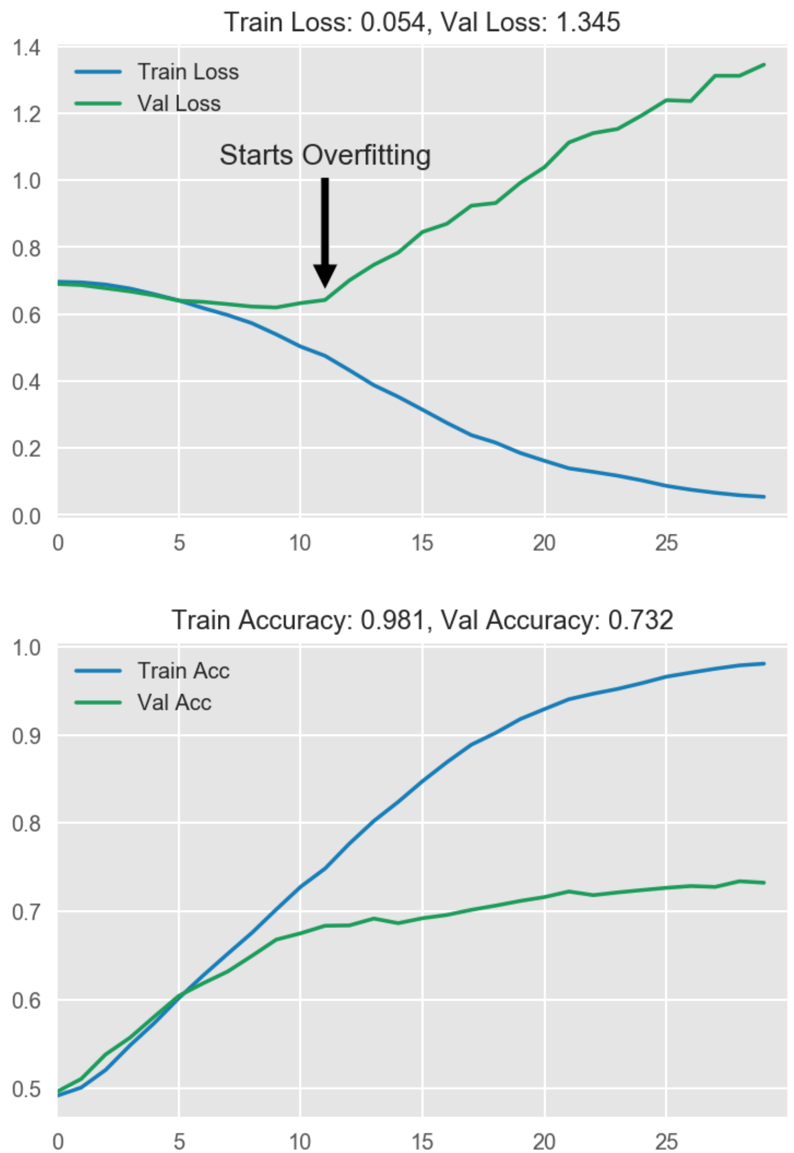
Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Dropout



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

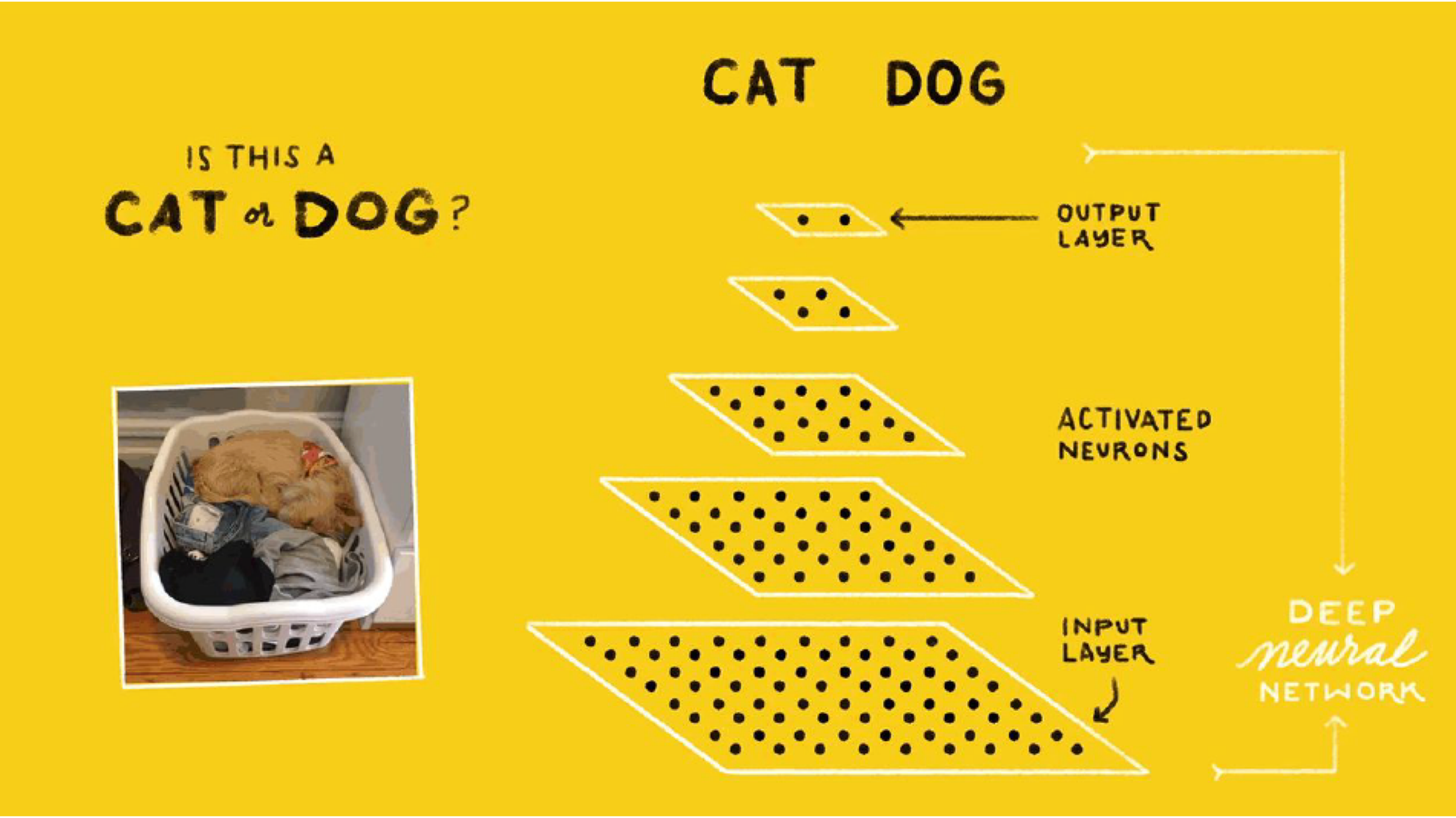
Model Performance



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Visual Recognition

Image Classification

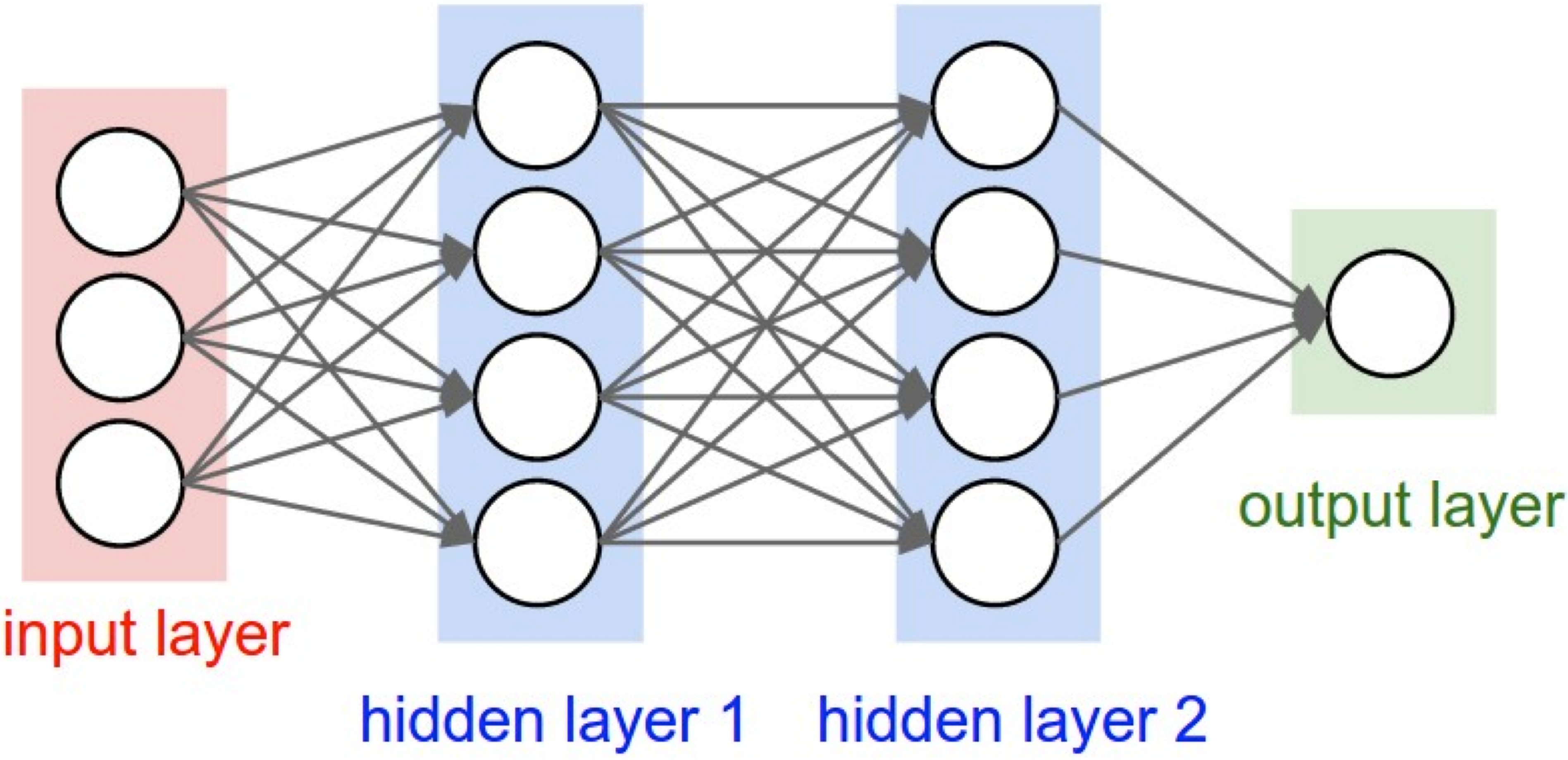


Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Convolutional Neural Networks (CNNs / ConvNets)

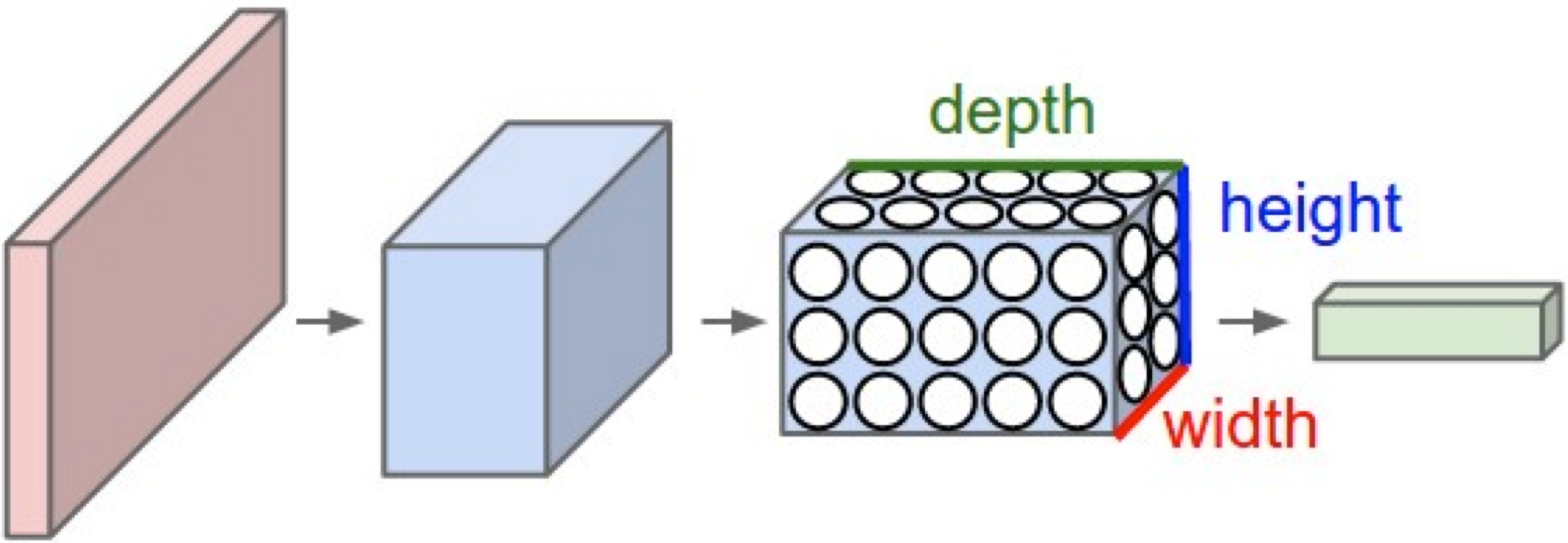
http://cs231n.github.io/convolutional-networks/

A regular 3-layer Neural Network



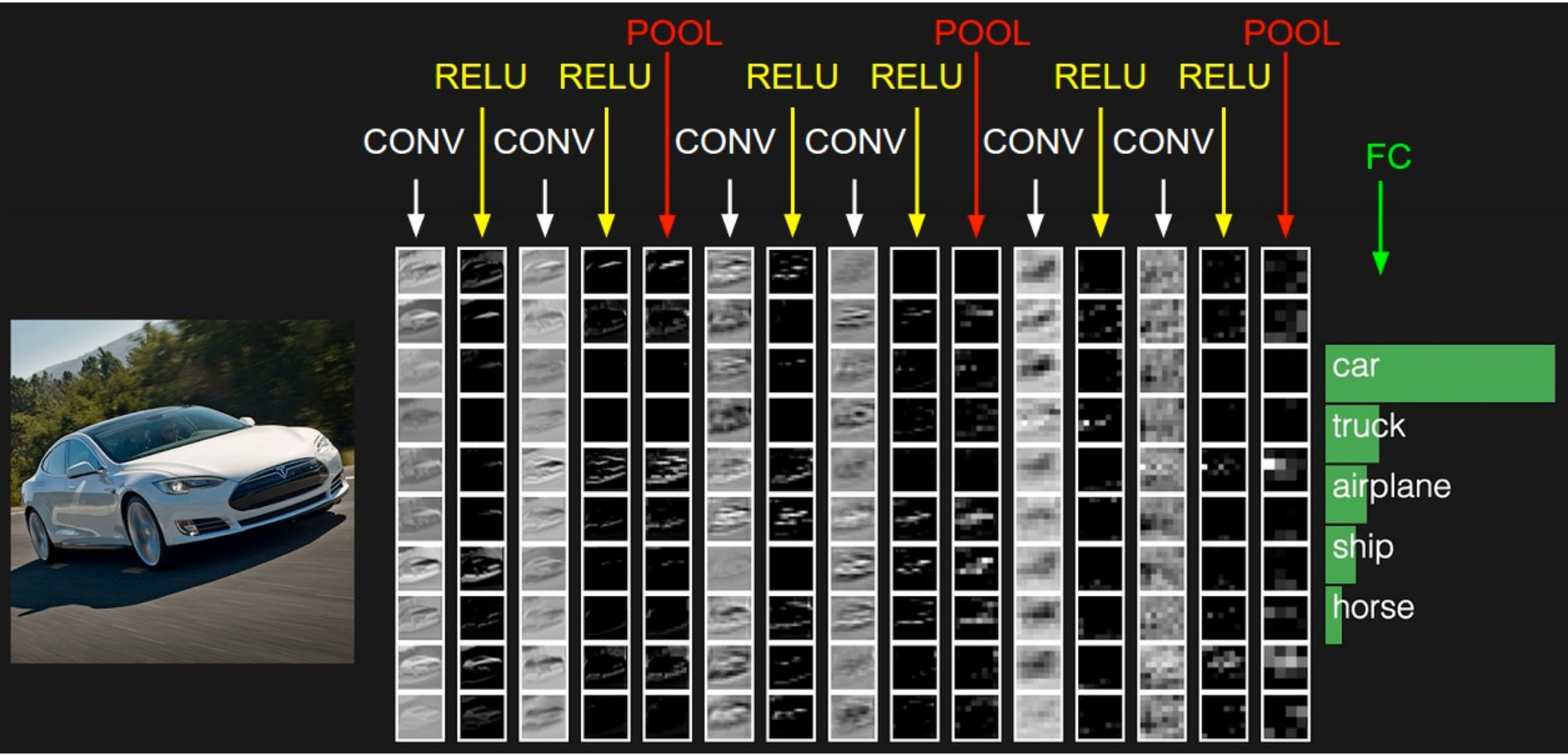
http://cs231n.github.io/convolutional-networks/

A ConvNet arranges its neurons in three dimensions (width, height, depth)



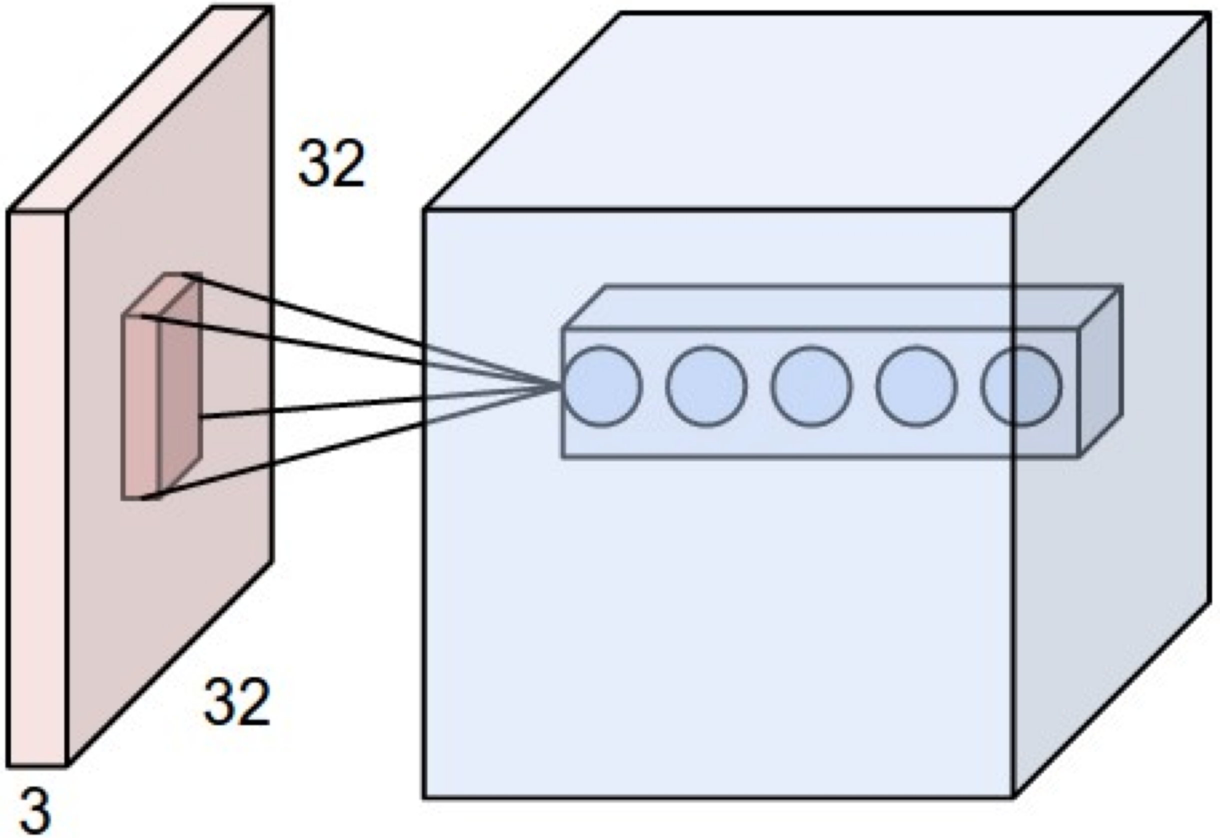
http://cs231n.github.io/convolutional-networks/

The activations of an example ConvNet architecture.



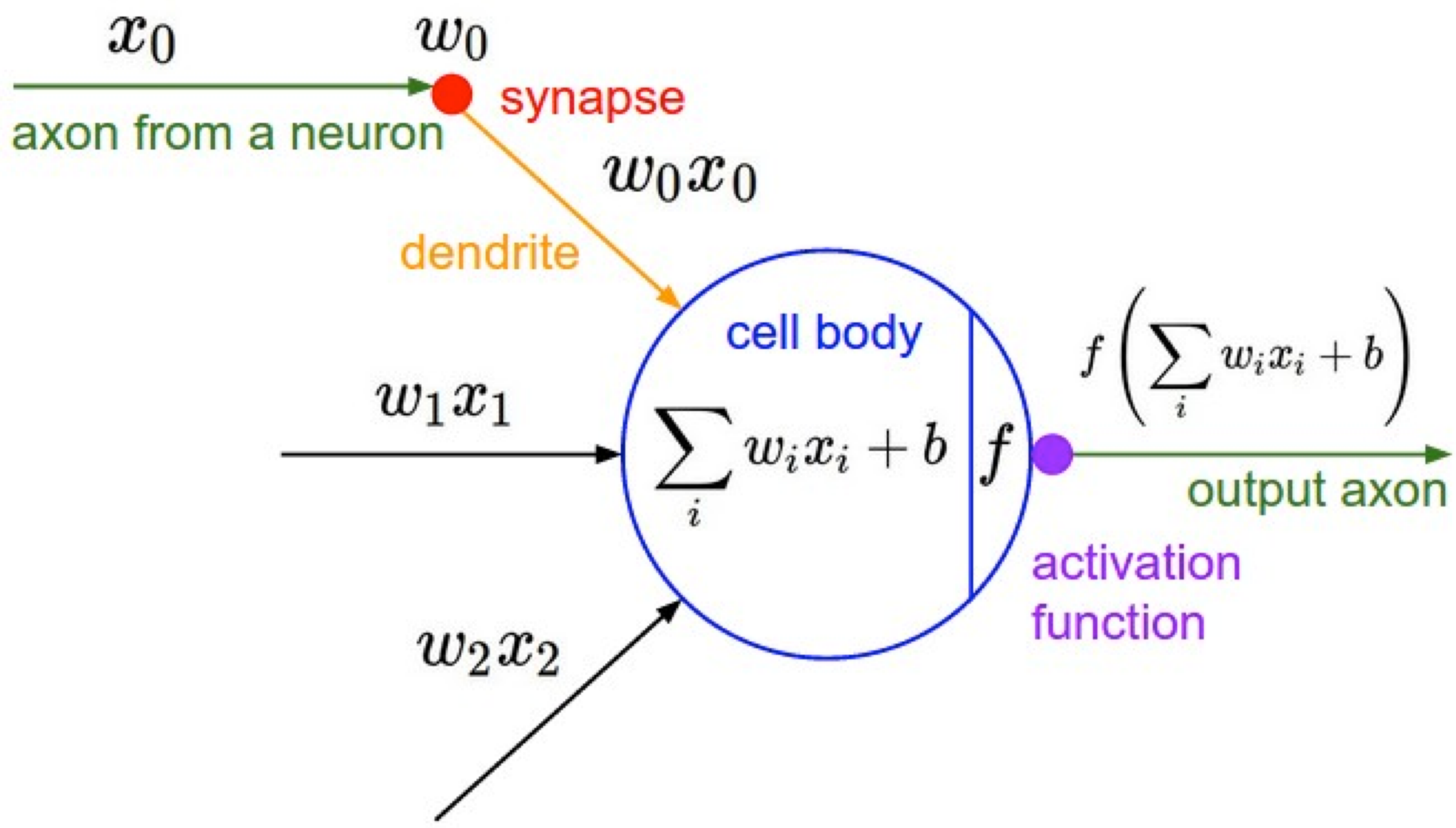
http://cs231n.github.io/convolutional-networks/

ConvNets



http://cs231n.github.io/convolutional-networks/

ConvNets



http://cs231n.github.io/convolutional-networks/

Convolution Demo

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	1	2	0	2	1	0
0	2	2	2	1	1	0
0	2	2	2	0	1	0
0	2	2	1	2	1	0
0	2	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	0	2	2	1	2	0
0	1	2	0	0	2	0
0	0	1	2	1	0	0
0	2	2	2	2	0	0
0	2	2	2	0	2	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	1	0	0	1	0	0
0	0	2	0	0	0	0
0	0	0	1	1	1	0
0	2	2	2	1	2	0
0	1	2	0	0	2	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	-1	0
1	1	1
-1	0	1

w0[:,:,1]

0	0	1
0	1	0
0	0	1

w0[:,:,2]

-1	-1	0
1	0	-1
-1	0	-1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

1	-1	0
0	1	1
0	-1	1

w1[:,:,1]

-1	1	0
-1	-1	1
0	0	0

w1[:,:,2]

1	0	-1
0	0	-1
1	0	1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

6	3	6
7	-1	-2
2	3	-2

o[:,:,1]

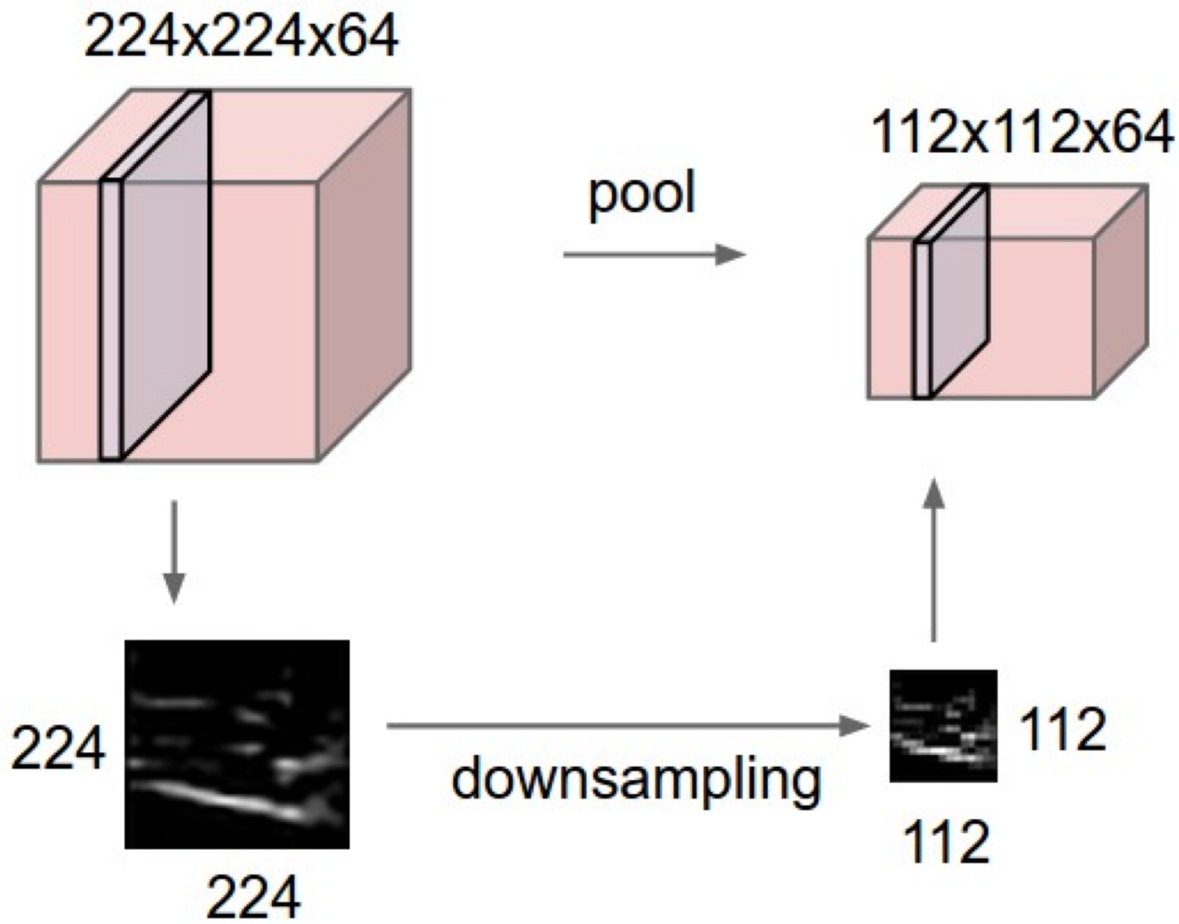
7	-1	-3
4	3	2
-1	0	-1

toggle movement

http://cs231n.github.io/convolutional-networks/

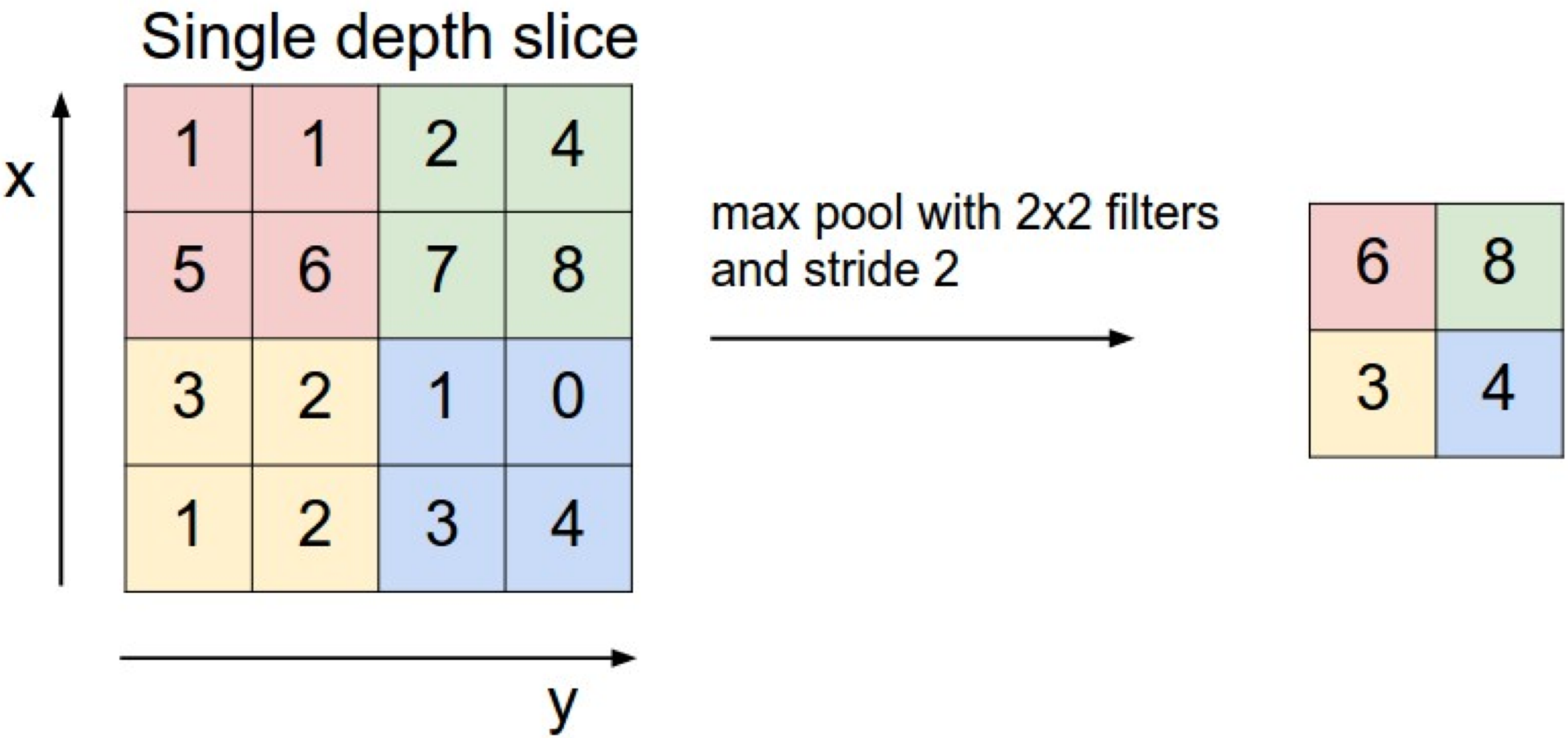
ConvNets

input volume of size [224x224x64]
is pooled with **filter** size 2, **stride** 2
into output volume of size [112x112x64]



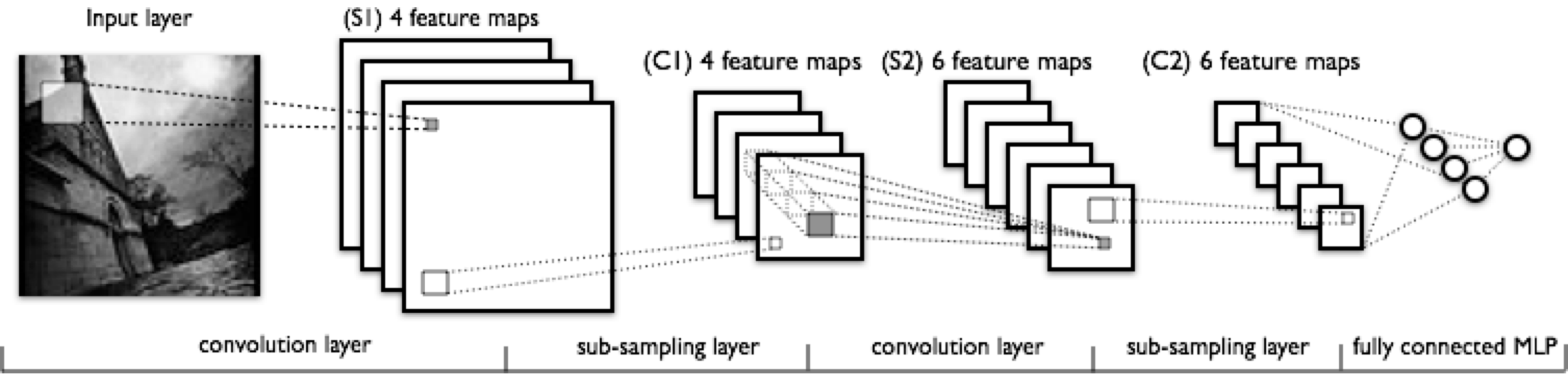
http://cs231n.github.io/convolutional-networks/

ConvNets
max pooling



http://cs231n.github.io/convolutional-networks/

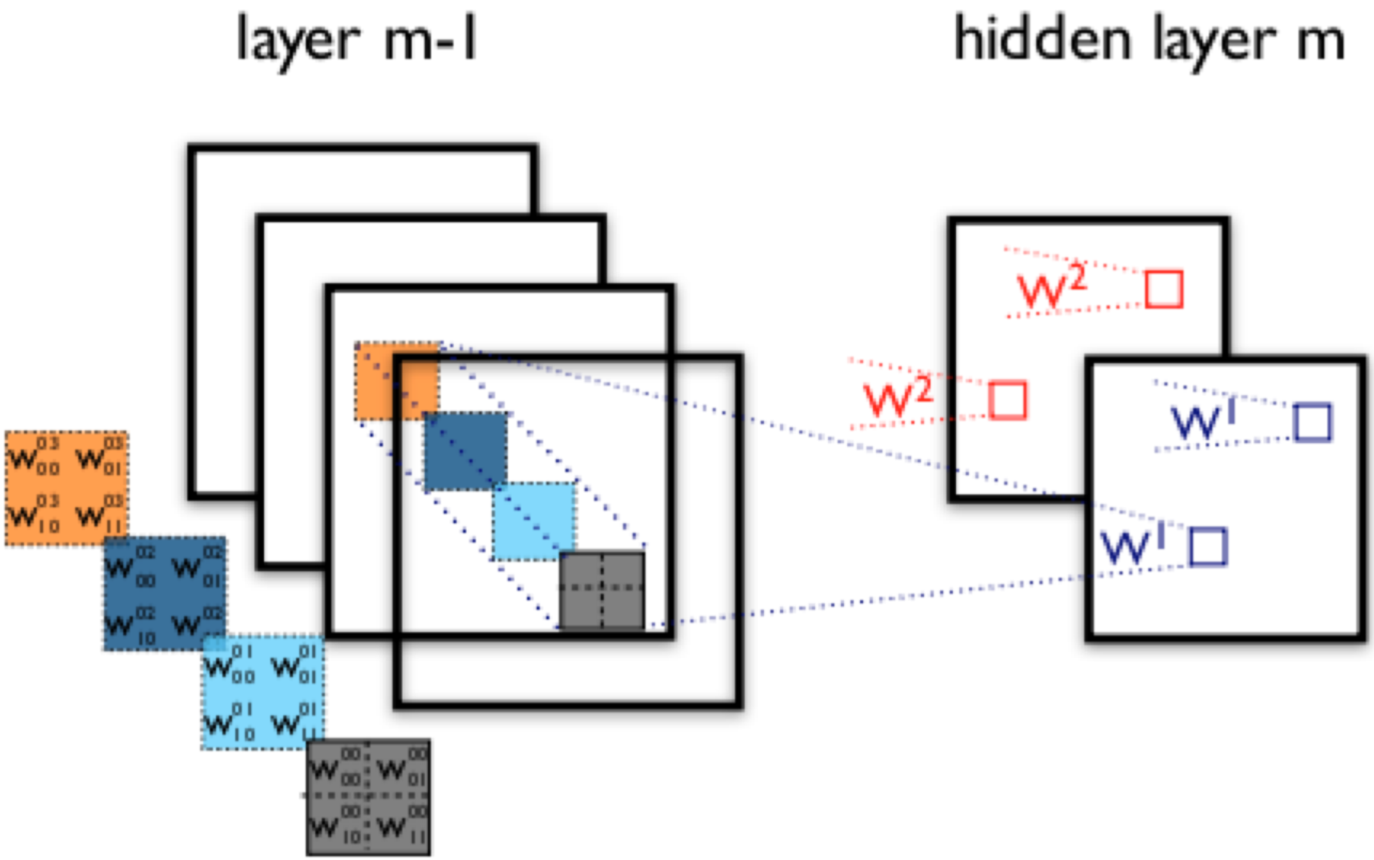
Convolutional Neural Networks (CNN) (LeNet)



Source: http://deeplearning.net/tutorial/lenet.html

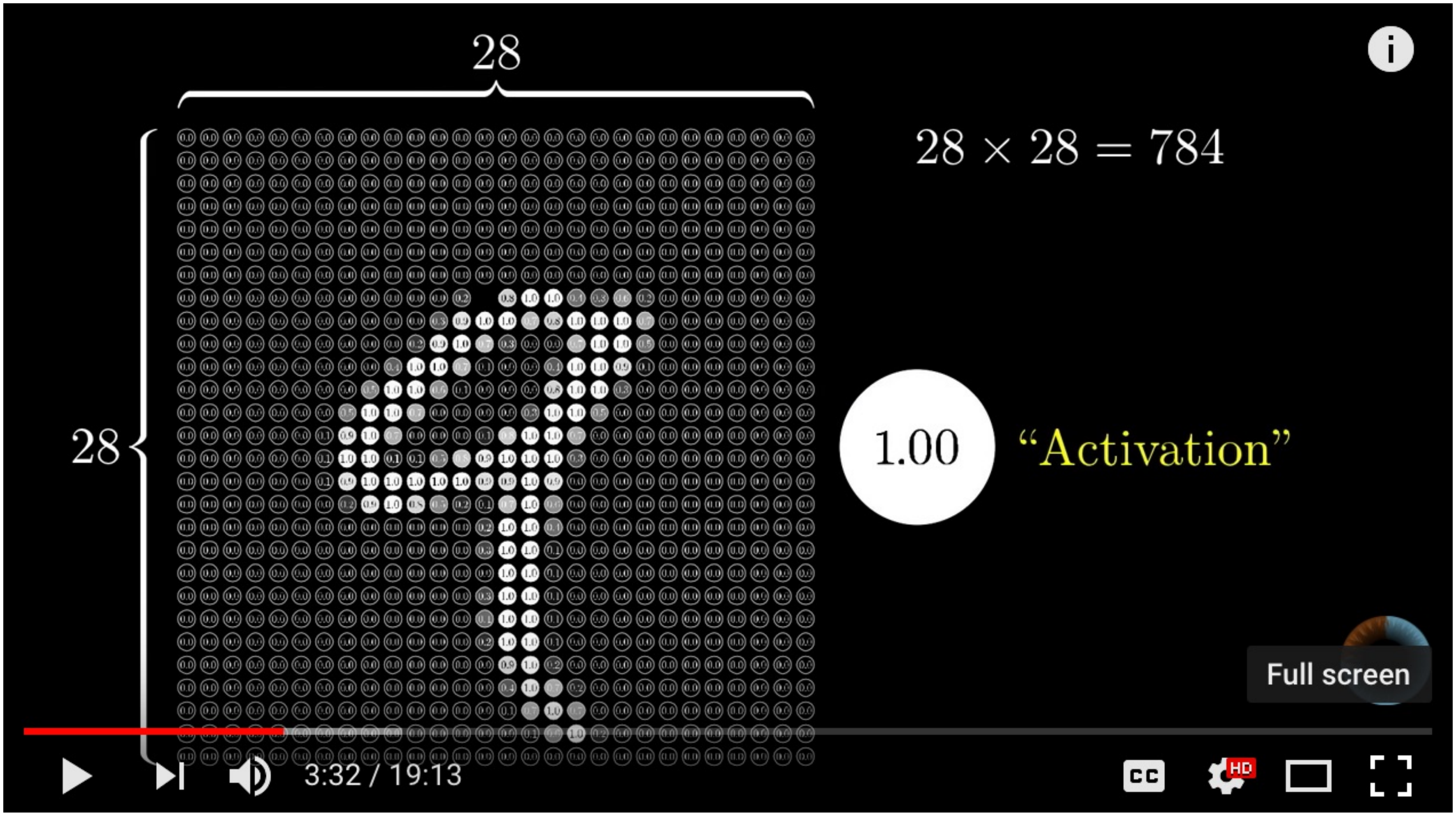
Convolutional Neural Networks (CNN) (LeNet)

example of a convolutional layer



Source: http://deeplearning.net/tutorial/lenet.html

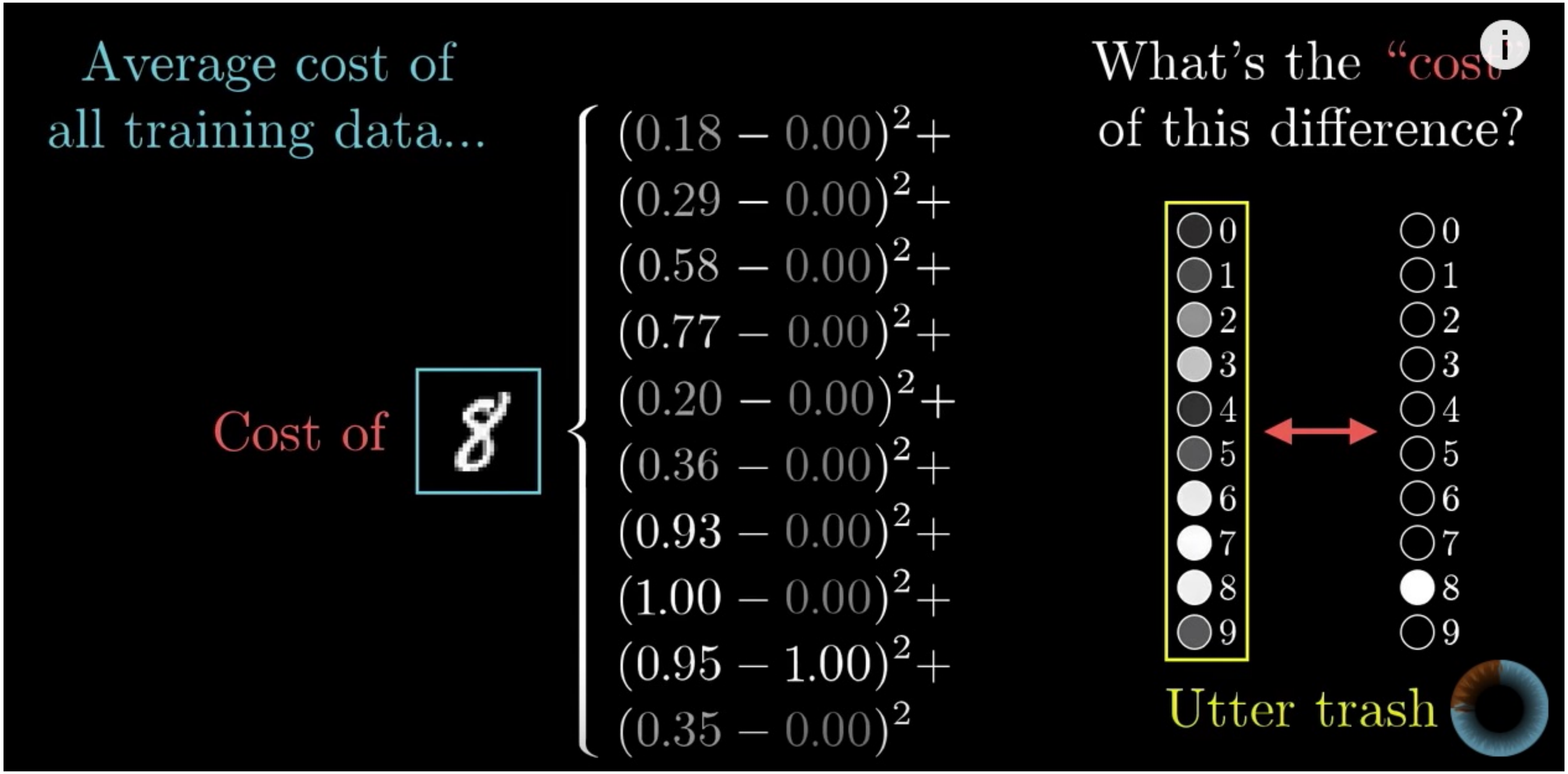
Neural Network and Deep Learning



Source: 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk

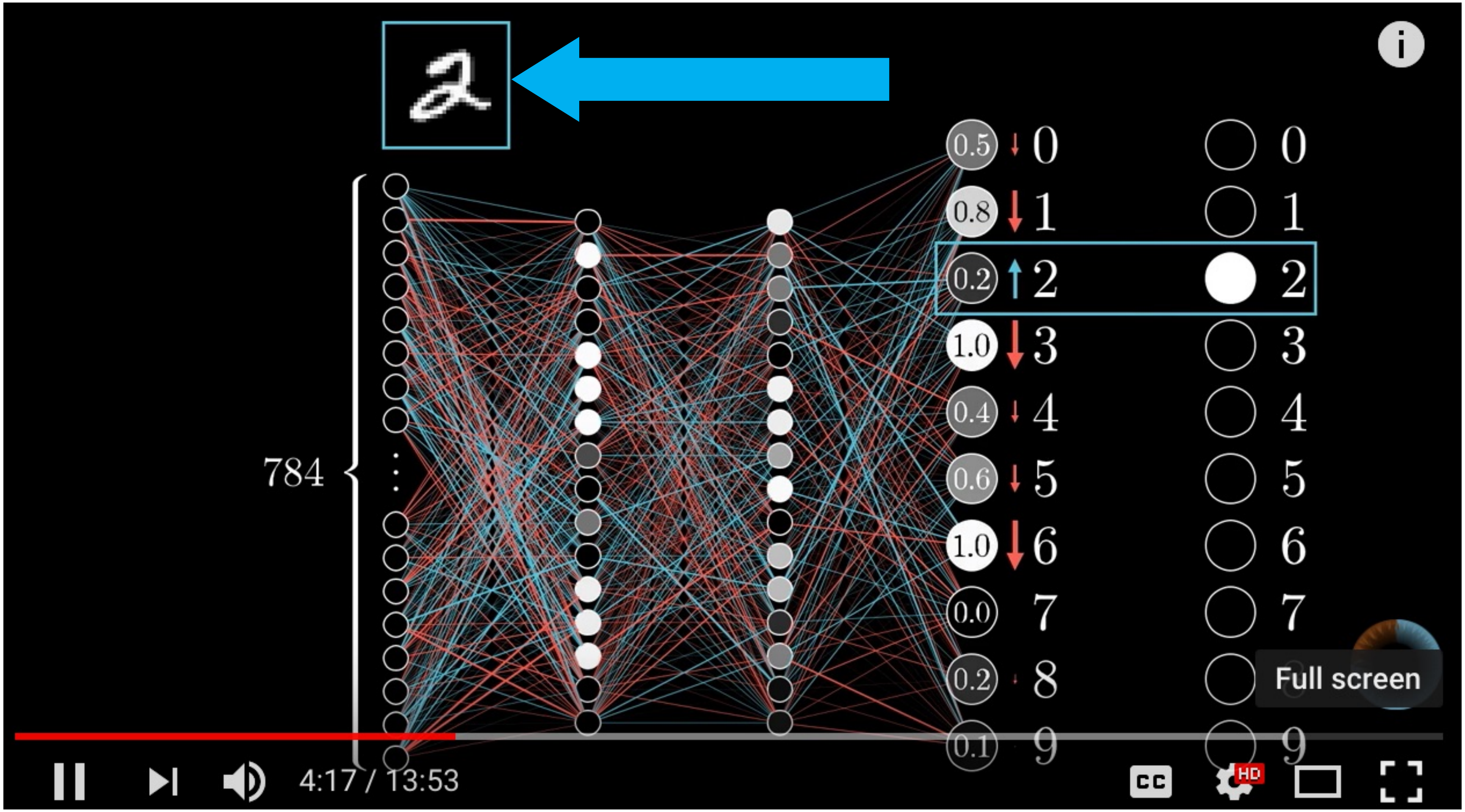
Gradient Descent

how neural networks learn



Source: 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=IHZwWFHWa-w

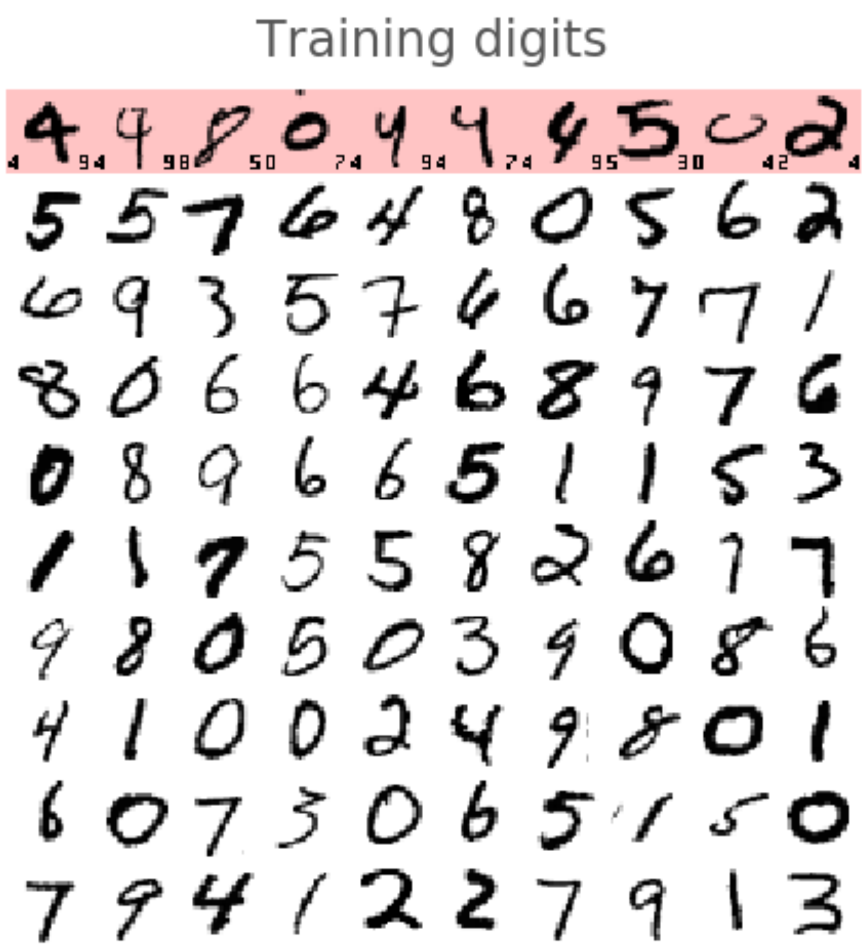
Backpropagation



Source: 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U

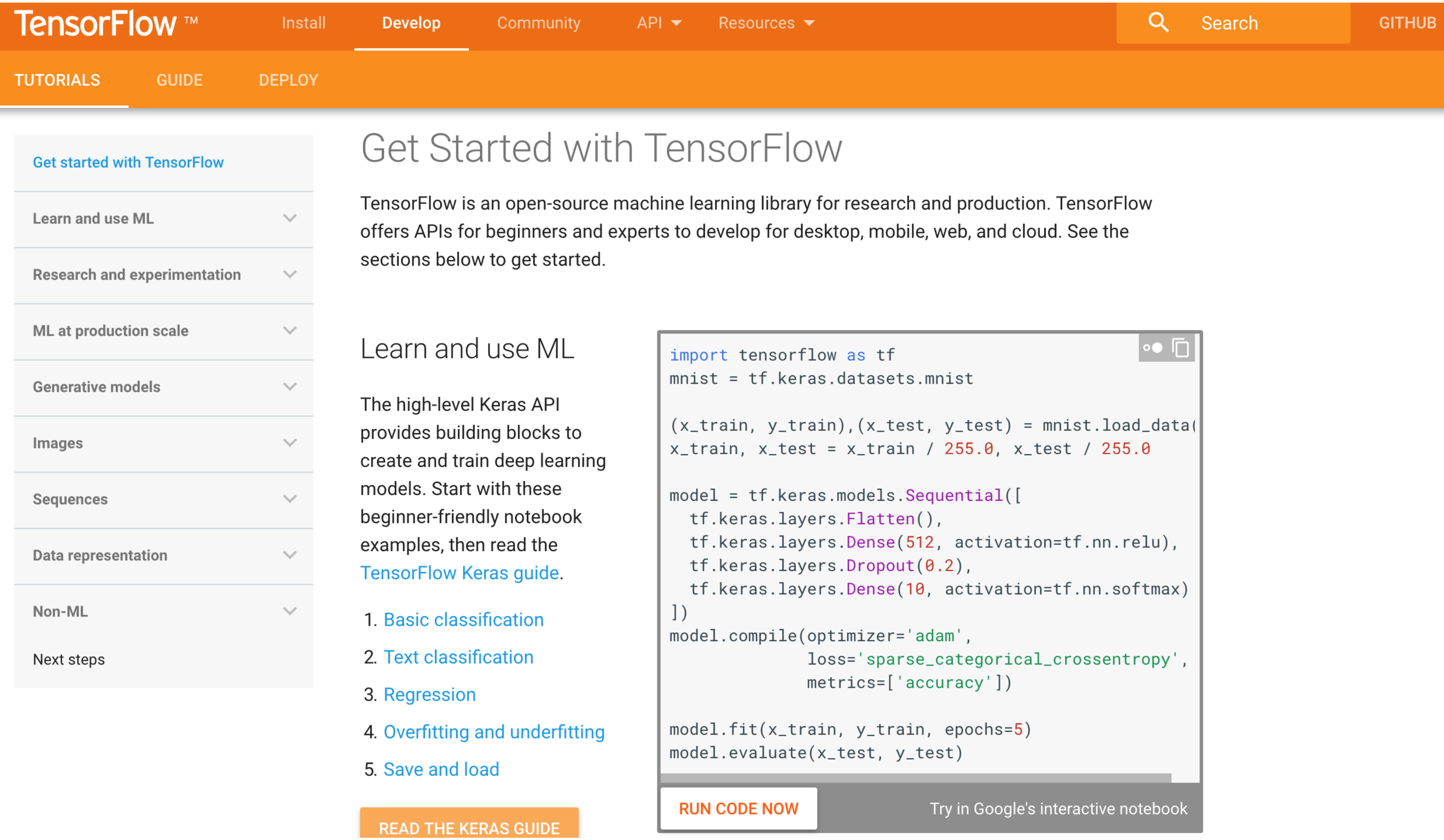
TensorFlow Image Recognition

MNIST dataset:
60,000 labeled digits



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0

Get Started with TensorFlow



TensorFlow ™ | Install | Develop | Community | API | Resources | Search | GITHUB

TUTORIALS | GUIDE | DEPLOY

- Get started with TensorFlow
- Learn and use ML
- Research and experimentation
- ML at production scale
- Generative models
- Images
- Sequences
- Data representation
- Non-ML
- Next steps

Get Started with TensorFlow

TensorFlow is an open-source machine learning library for research and production. TensorFlow offers APIs for beginners and experts to develop for desktop, mobile, web, and cloud. See the sections below to get started.

Learn and use ML

The high-level Keras API provides building blocks to create and train deep learning models. Start with these beginner-friendly notebook examples, then read the TensorFlow Keras guide.

1. Basic classification
2. Text classification
3. Regression
4. Overfitting and underfitting
5. Save and load

READ THE KERAS GUIDE

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data(
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

RUN CODE NOW | Try in Google's interactive notebook

https://www.tensorflow.org/tutorials/

Get Started with TensorFlow MNIST

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

https://www.tensorflow.org/tutorials/

Get Started with TensorFlow MNIST

CO _index.ipynb

File Edit View Insert Runtime Tools Help

SHARE A

CODE TEXT CELL CELL COPY TO DRIVE CONNECT EDITING

Get Started with TensorFlow

View on TensorFlow.org Run in Google Colab View source on GitHub

This is a Google Colaboratory notebook file. Python programs are run directly in the browser—a great way to learn and use TensorFlow. To run the Colab notebook:

1. Connect to a Python runtime: At the top-right of the menu bar, select *CONNECT*.
2. Run all the notebook code cells: Select *Runtime* > *Run all*.

For more examples and guides (including details for this program), see Get Started with TensorFlow.

Let's get started, import the TensorFlow library into your program:

```
[ ]  import tensorflow as tf
```

Load and prepare the MNIST dataset. Convert the samples from integers to floating-point numbers:

```
[ ]  mnist = tf.keras.datasets.mnist

     (x_train, y_train), (x_test, y_test) = mnist.load_data()
     x_train, x_test = x_train / 255.0, x_test / 255.0
```

Build the `tf.keras` model by stacking layers. Select an optimizer and loss function used for training:

```
[ ]  model = tf.keras.models.Sequential([
       tf.keras.layers.Flatten(),
       tf.keras.layers.Dense(512, activation=tf.nn.relu),
       tf.keras.layers.Dropout(0.2),
       tf.keras.layers.Dense(10, activation=tf.nn.softmax)
     ])
```

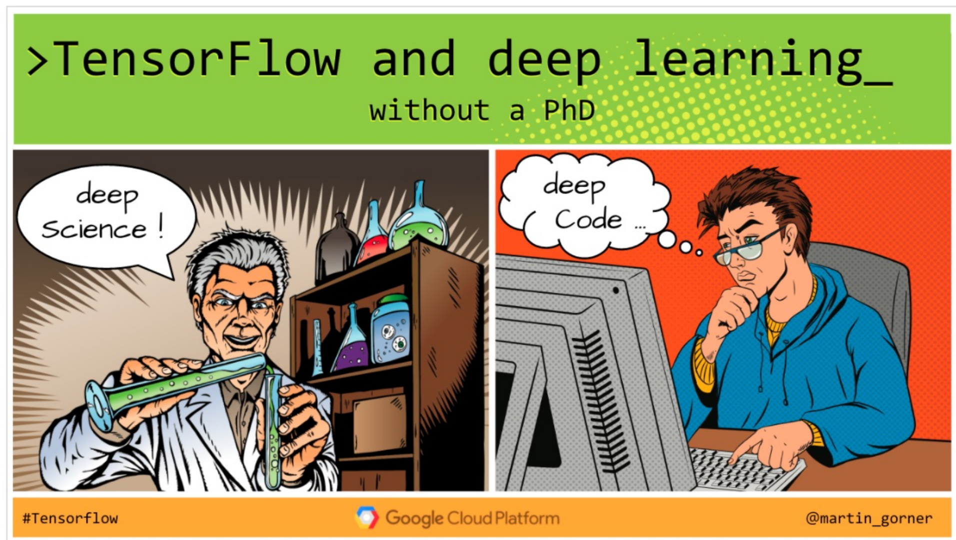
https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/_index.ipynb

TensorFlow and Deep Learning

Did you find a mistake? Please file a bug.

← TensorFlow and deep learning, without a PhD   149 min remaining

1. Overview



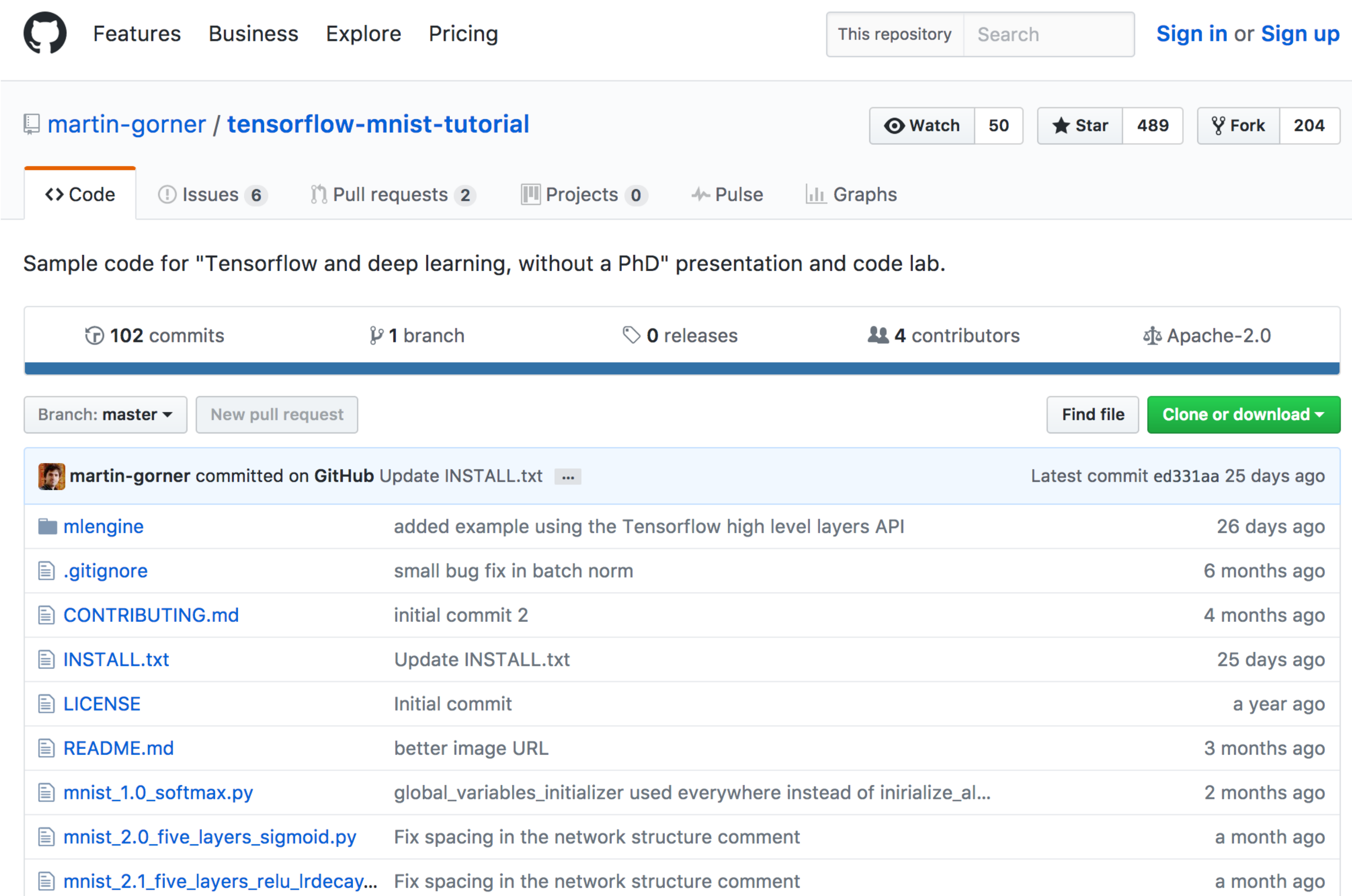
In this codelab, you will learn how to build and train a neural network that recognises handwritten digits. Along the way, as you enhance your neural network to achieve 99% accuracy, you will also discover the tools of the trade that deep learning professionals use to train their models efficiently.

This codelab uses the MNIST dataset, a collection of 60,000 labeled digits that has kept generations of PhDs busy for almost two decades. You will solve the problem with less than 100 lines of Python / TensorFlow code.

What you'll learn

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0

TensorFlow MNIST Tutorial



https://github.com/martin-gorner/tensorflow-mnist-tutorial/

TensorFlow and Deep Learning

- What is a neural network and how to train it
- How to build a basic 1-layer neural network using TensorFlow
- How to add more layers
- Training tips and tricks: overfitting, dropout, learning rate decay ...
- How to troubleshoot deep neural networks
- How to build convolutional networks

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0

TensorFlow MNIST Tutorial

```
git clone https://github.com/martin-gorner/tensorflow-mnist-tutorial.git
```

```
cd tensorflow-mnist-tutorial
```

```
python3 mnist_1.0_softmax.py
```

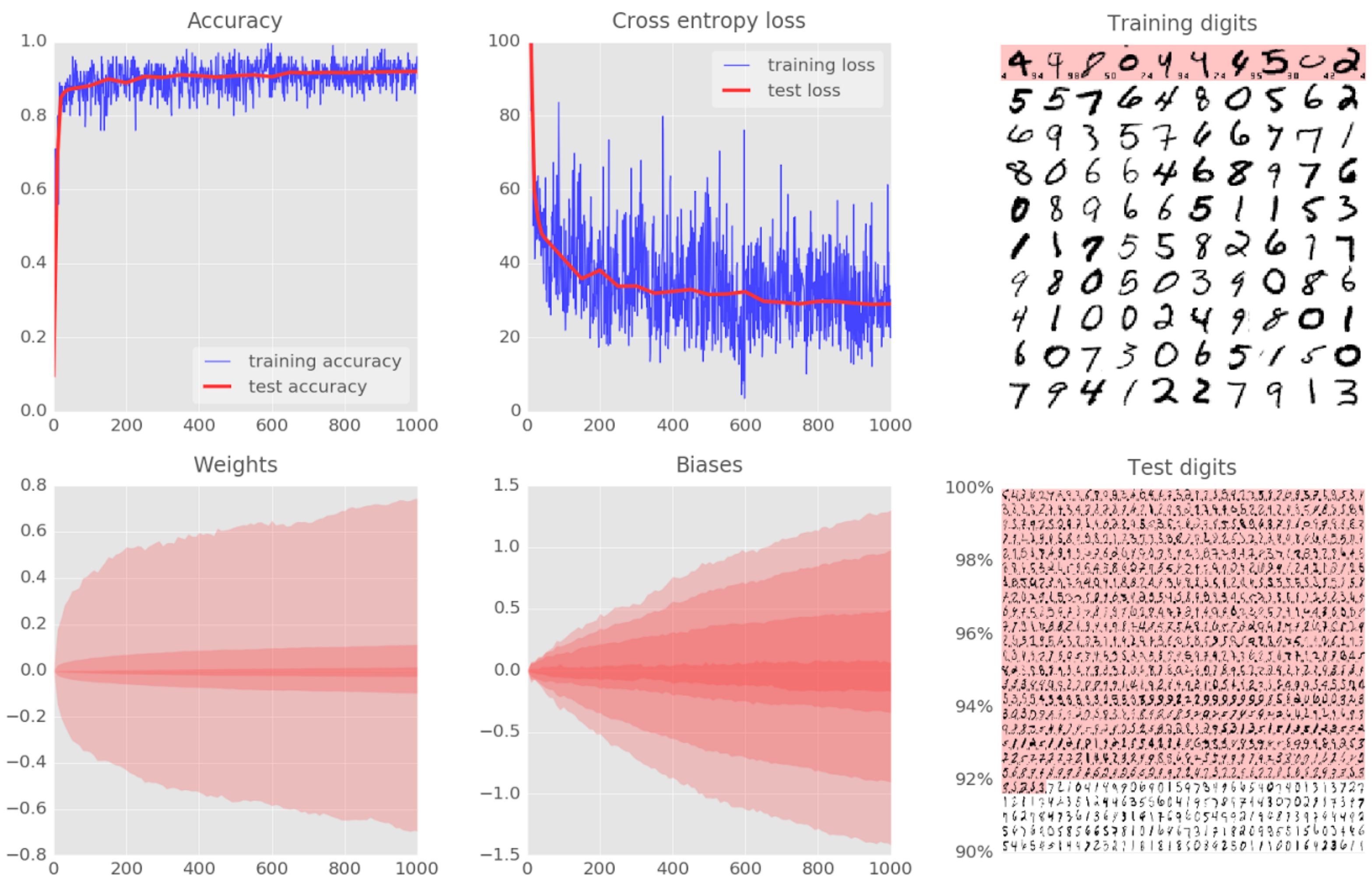
```
python mnist_1.0_softmax.py
```

```
pythonw mnist_1.0_softmax.py
```

Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/blob/master/INSTALL.txt

```
cd tensorflow-mnist-tutorial
```

```
python3 mnist_1.0_softmax.py
```

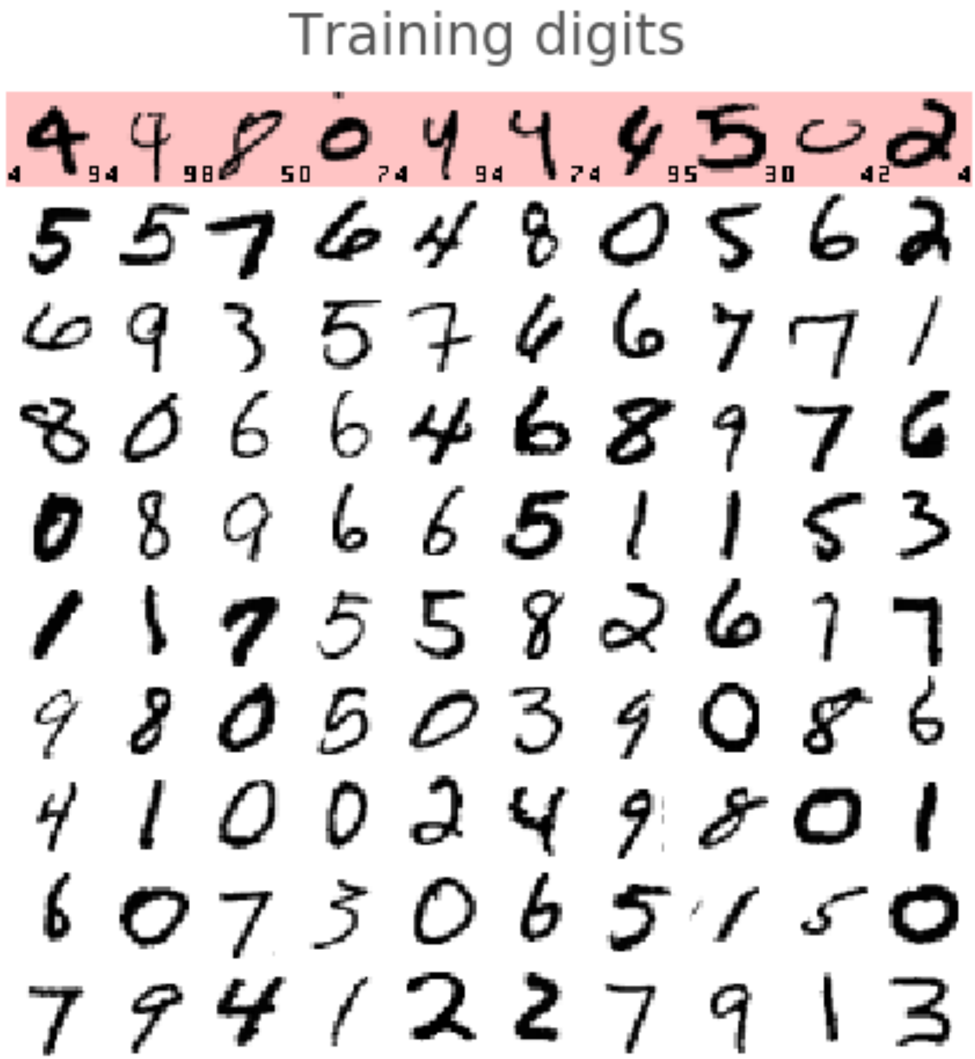


Train a Neural Network

Training digits
updates to weights and biases =>
better recognition (loop)

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

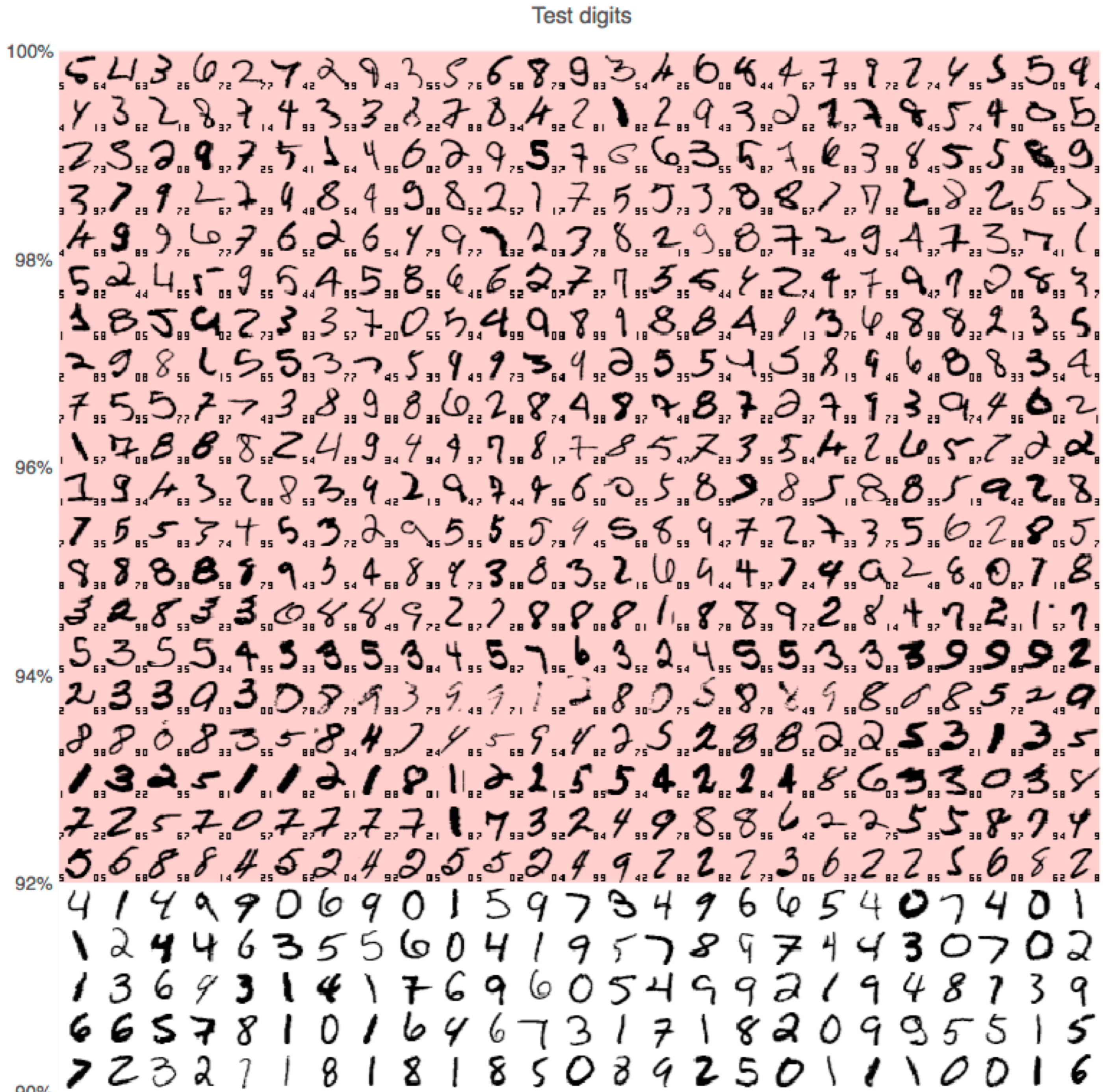
Training digits



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Training digits

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

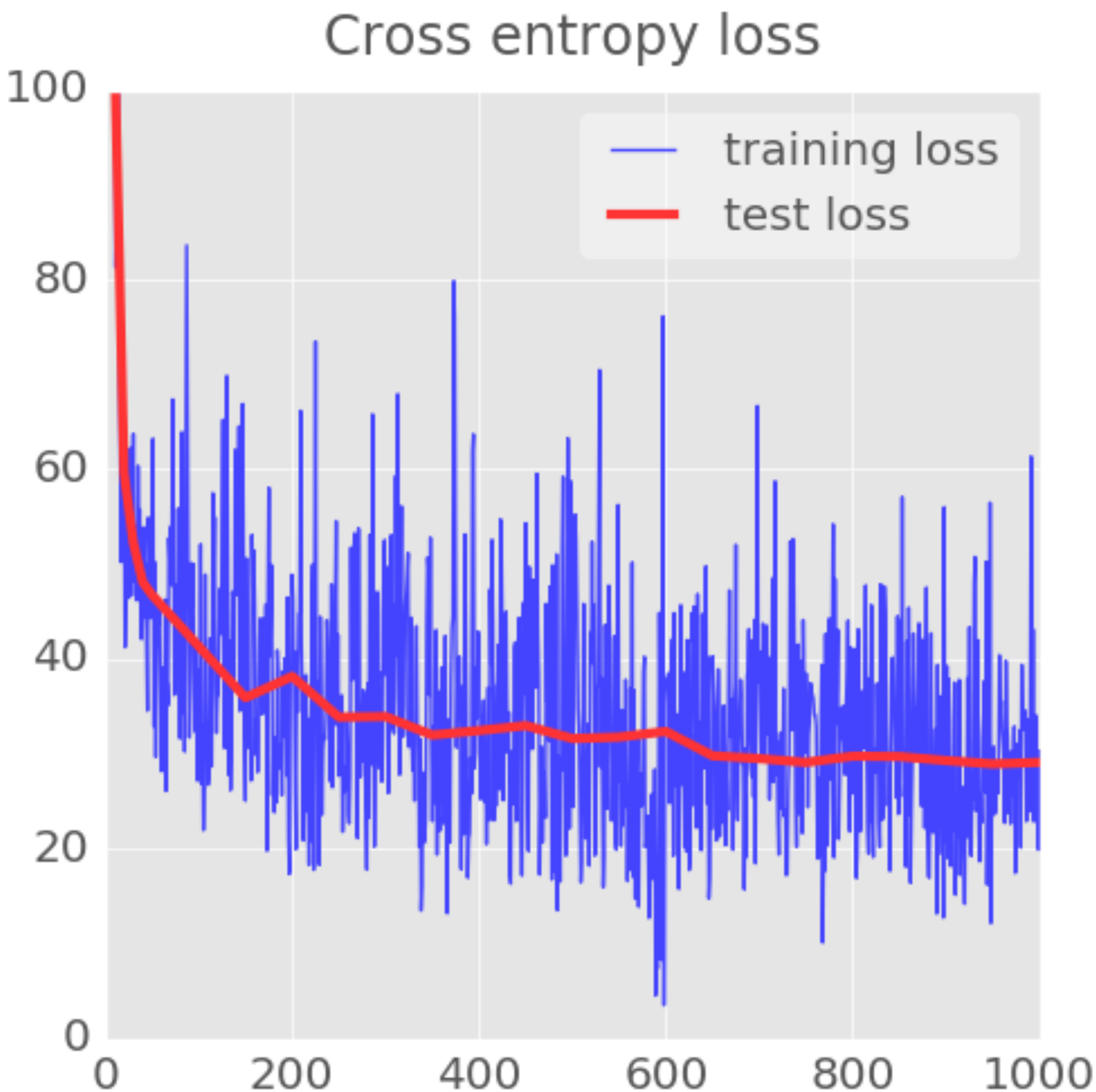


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Test digits

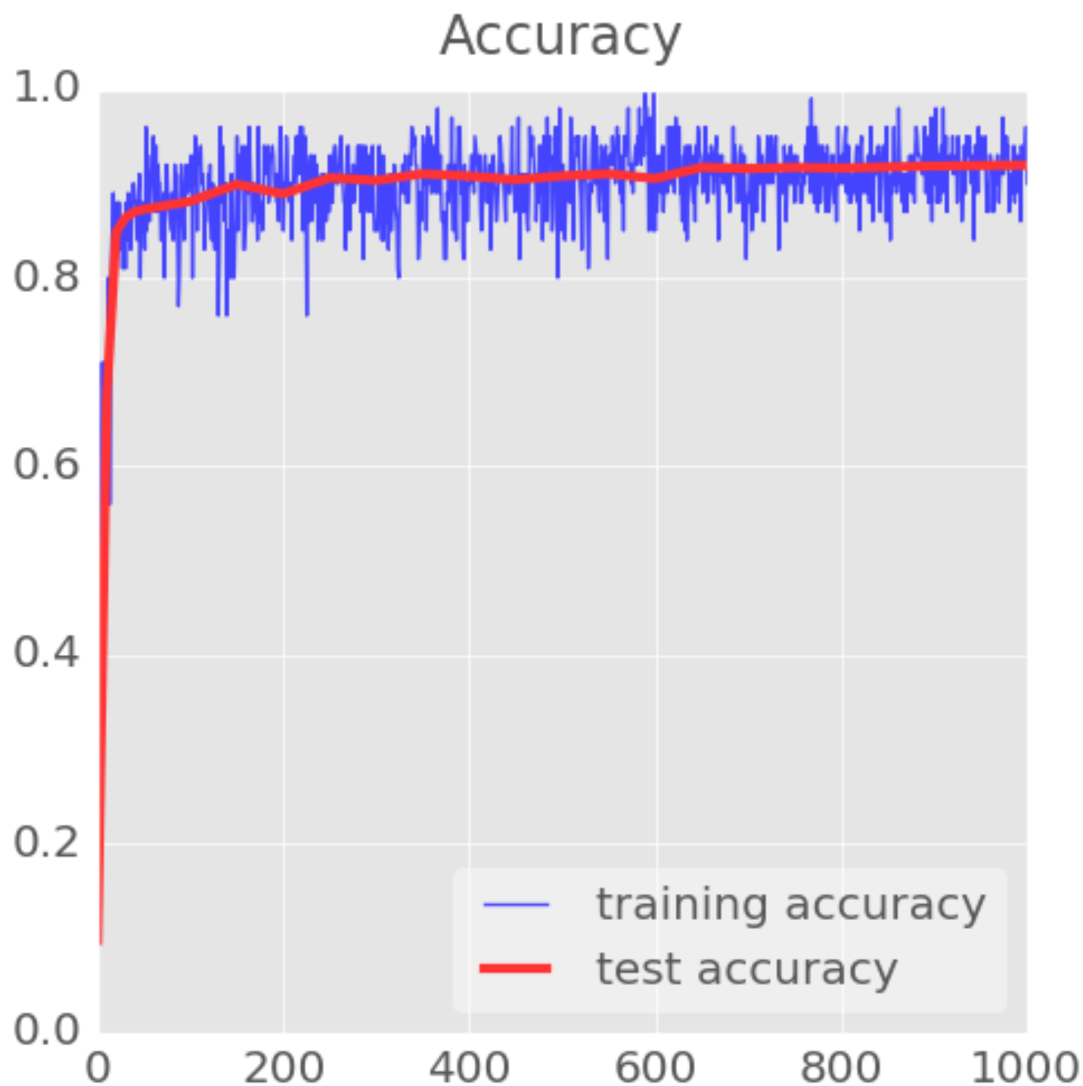
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Cross entropy loss



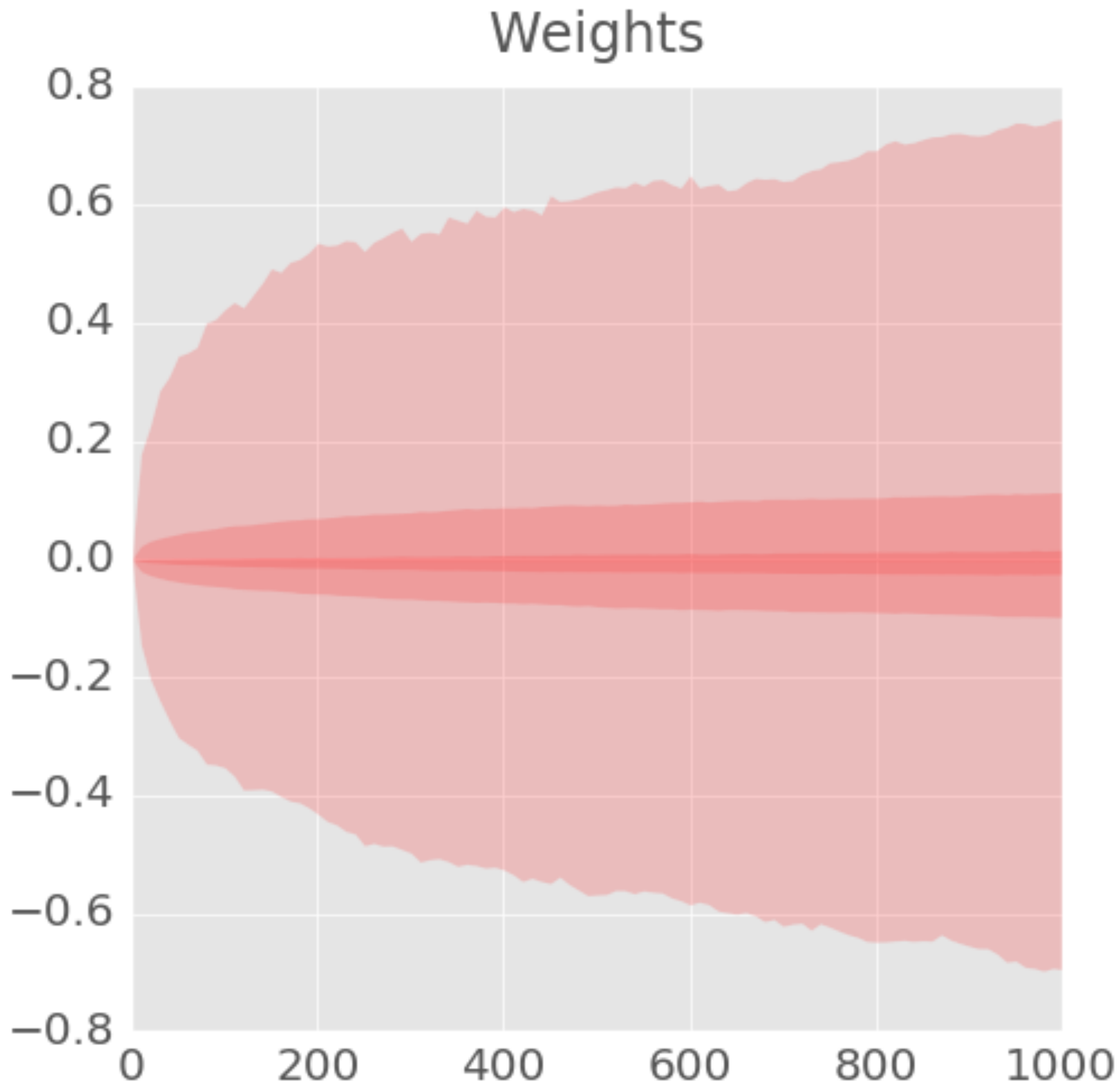
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Accuracy



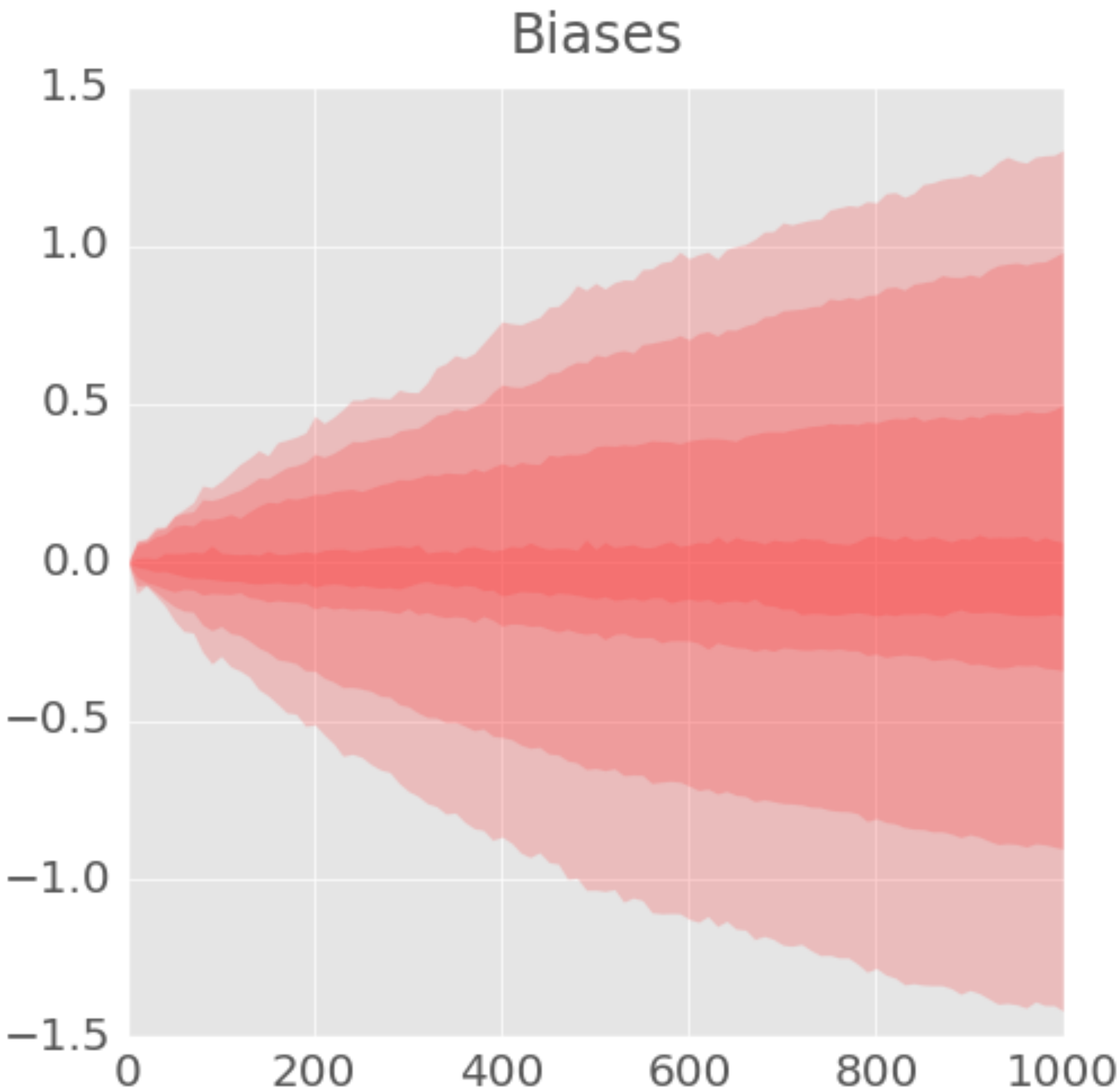
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Weights



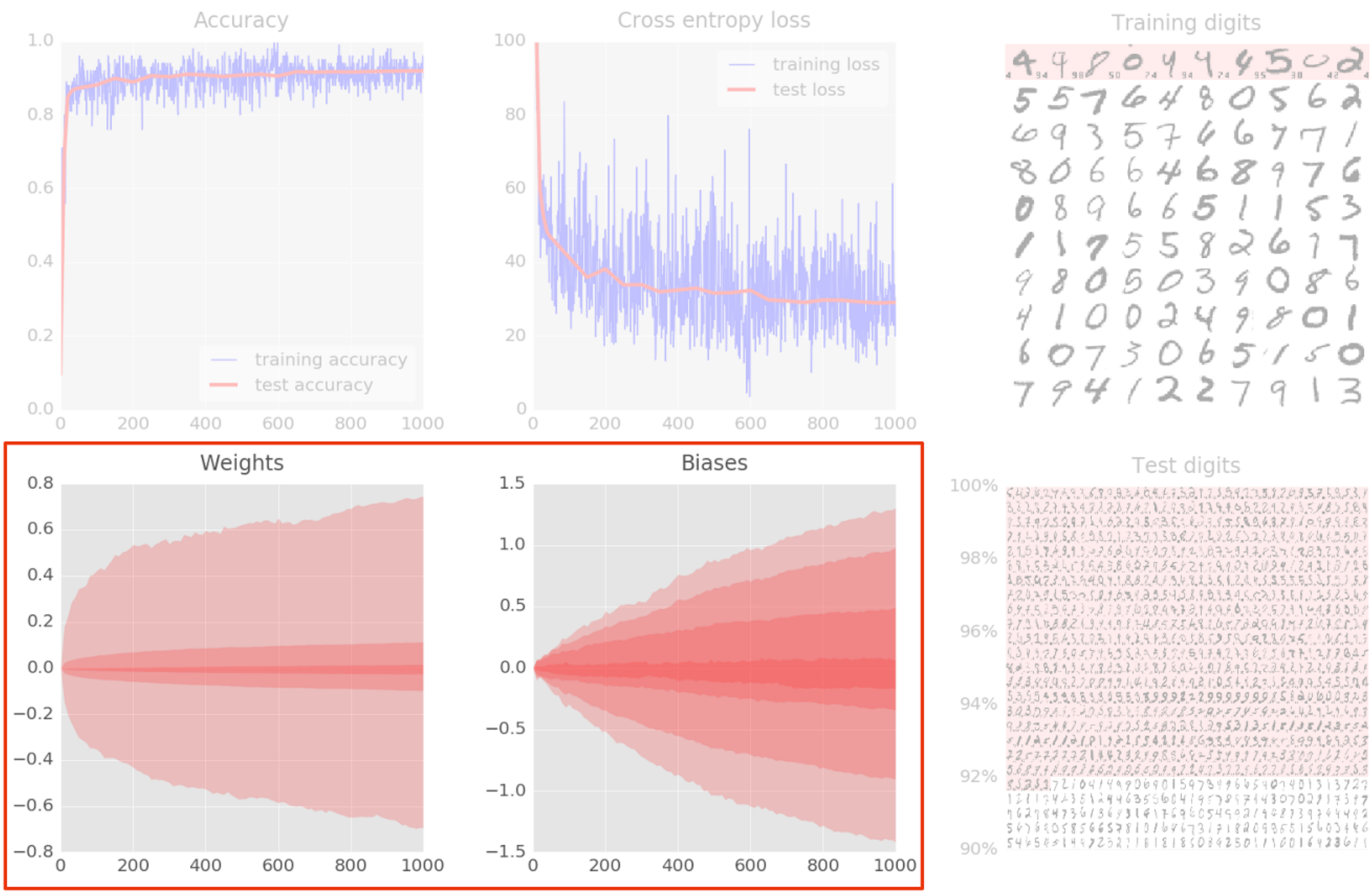
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Biases



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Weights and Biases

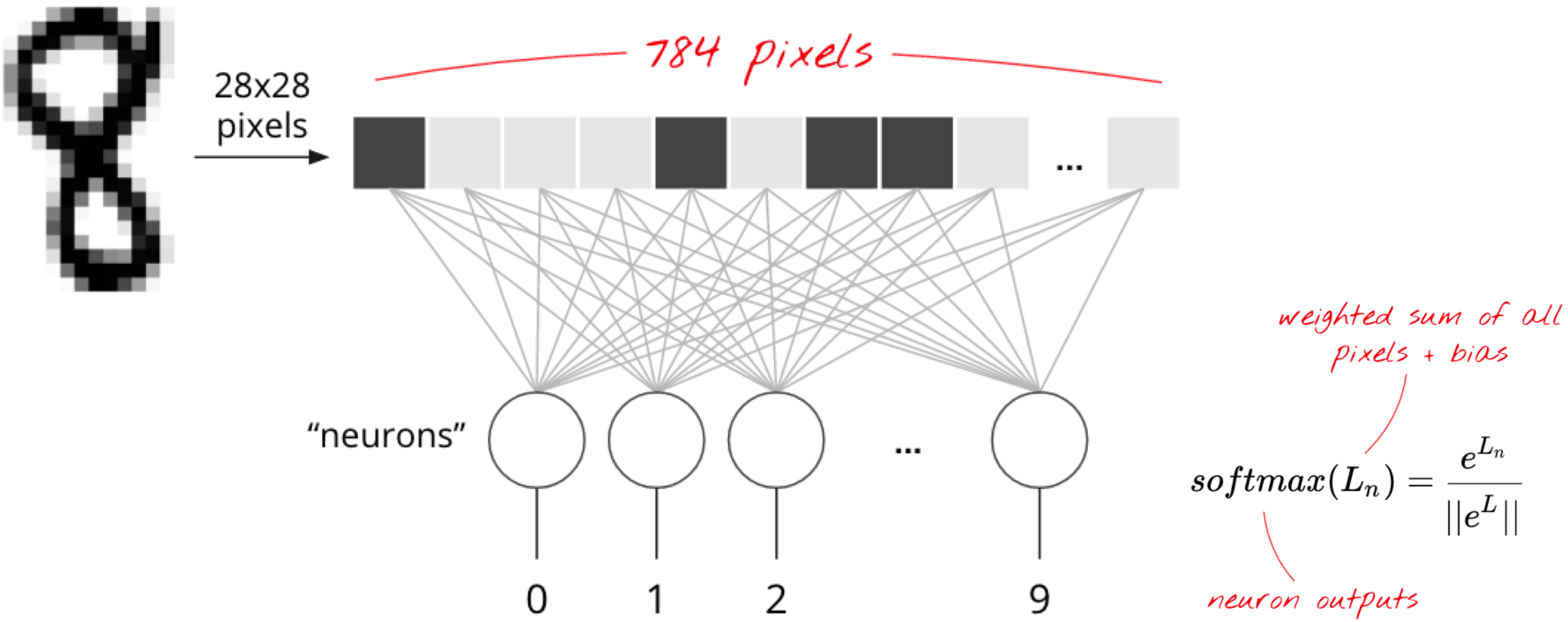


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2

Cookbook

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

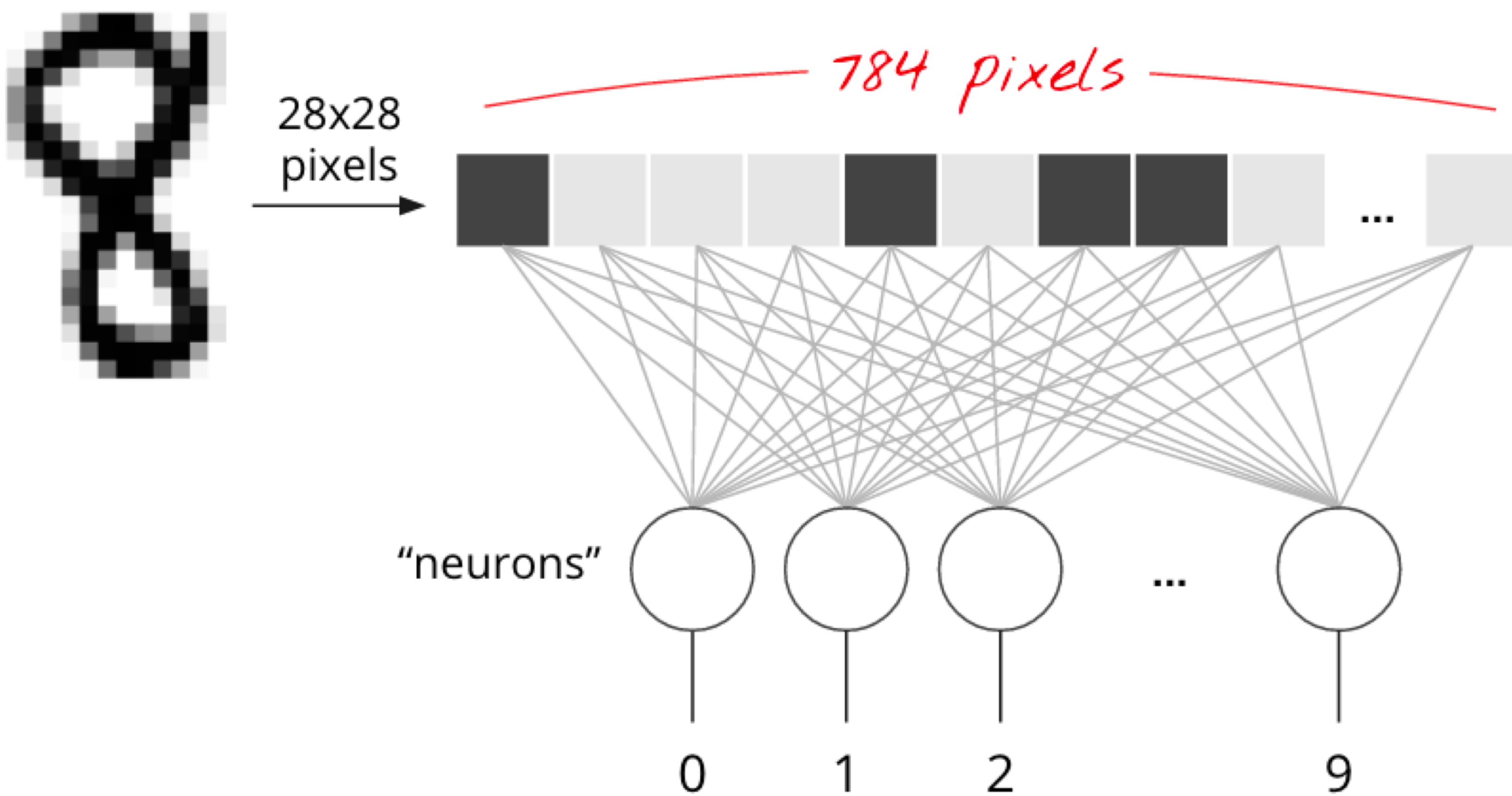
Very Simple Model: Softmax Classification



$$softmax(L_n) = \frac{e^{L_n}}{||e^L||}$$

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Very Simple Model: Softmax Classification



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Very Simple Model: Softmax Classification

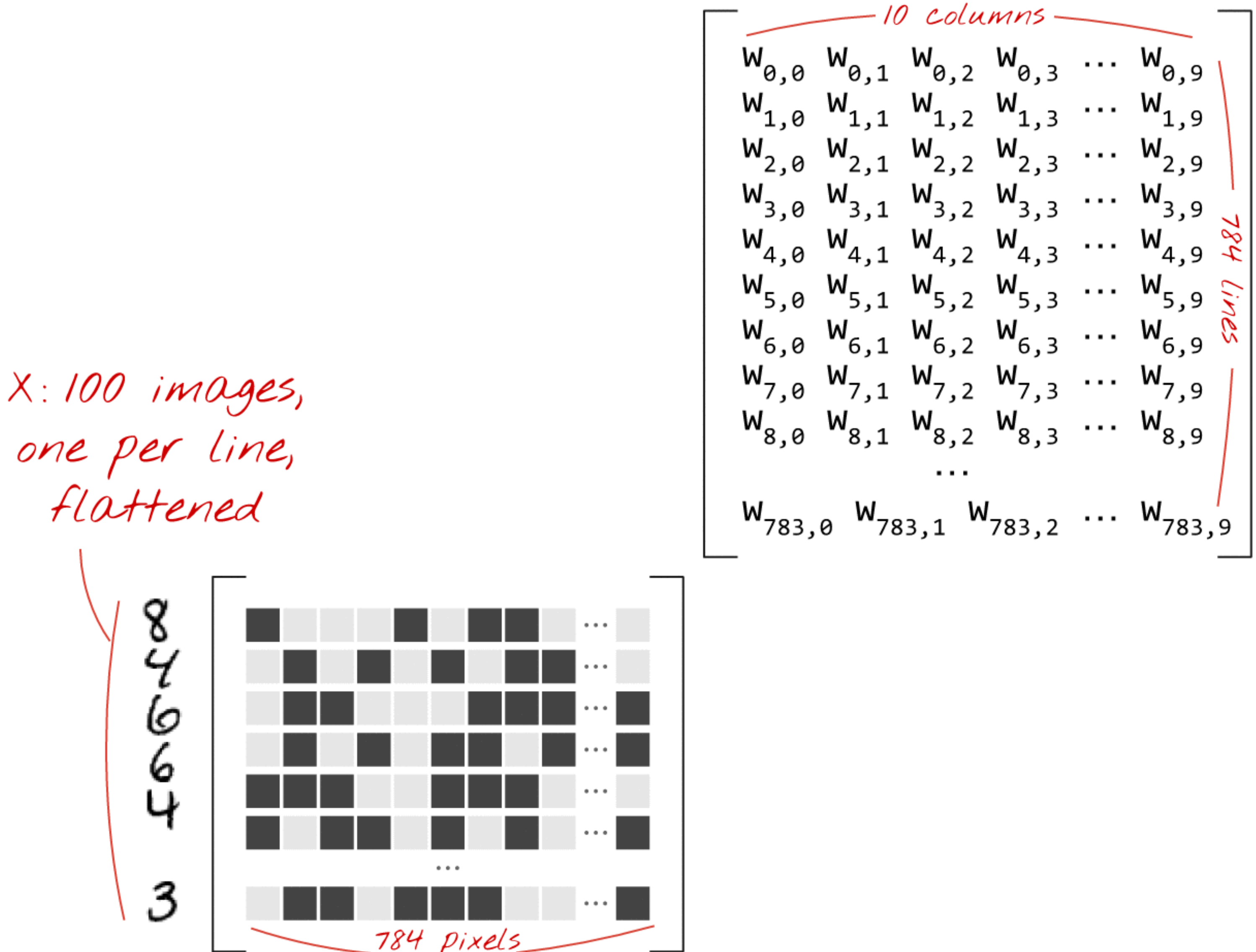
weighted sum of all pixels + bias

$$softmax(L_n) = \frac{e^{L_n}}{||e^L||}$$

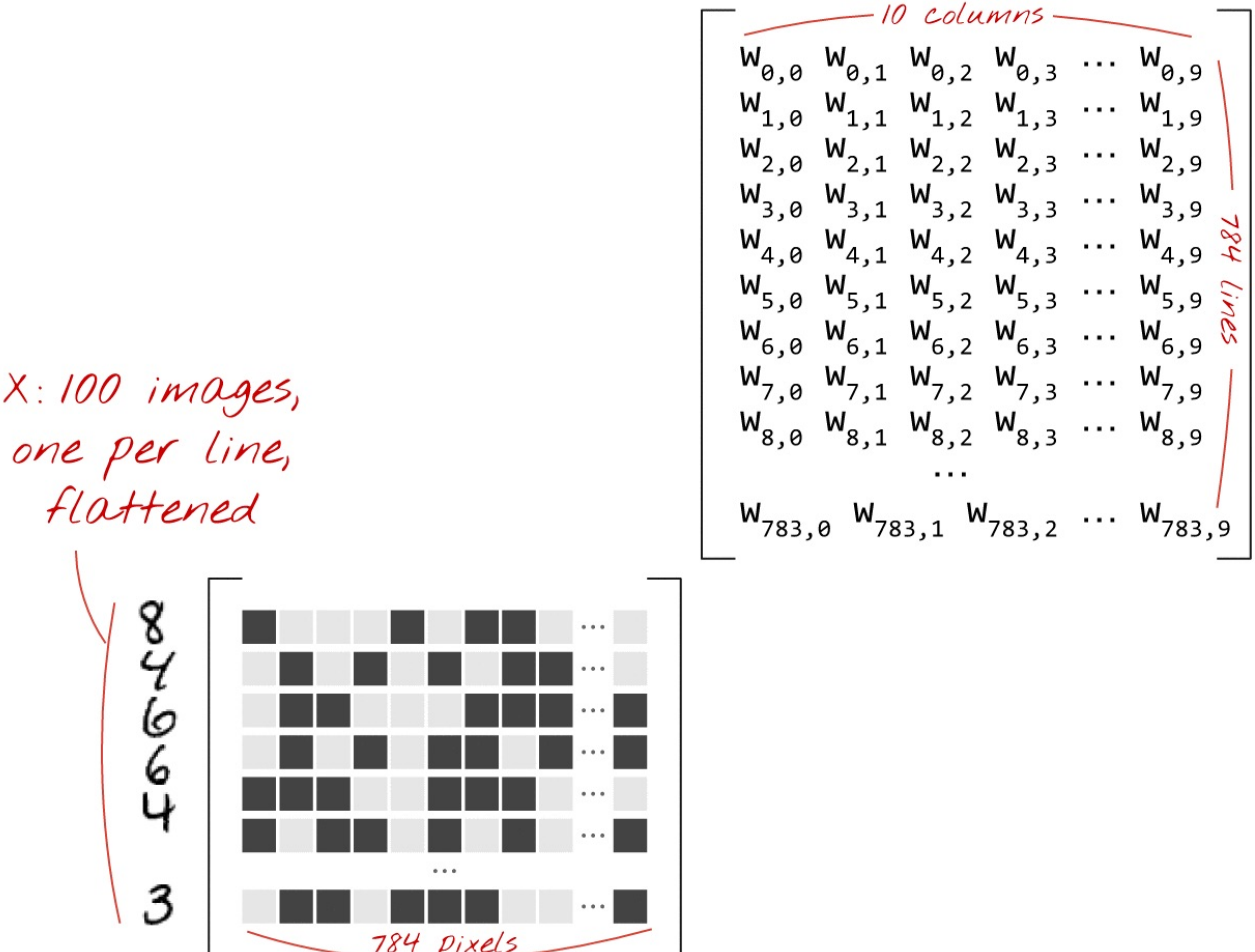
neuron outputs

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

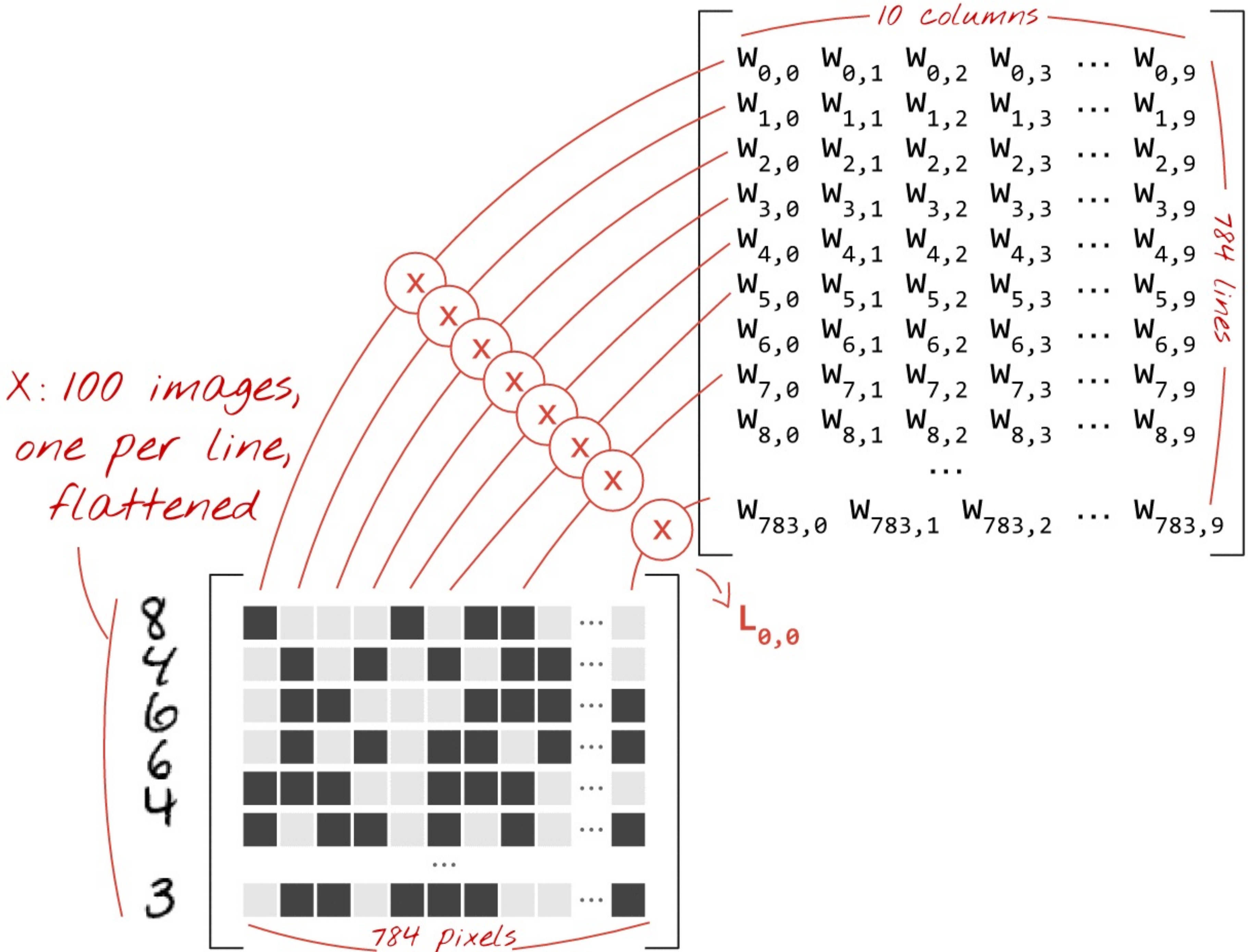
In Matrix notation, 100 images at a time



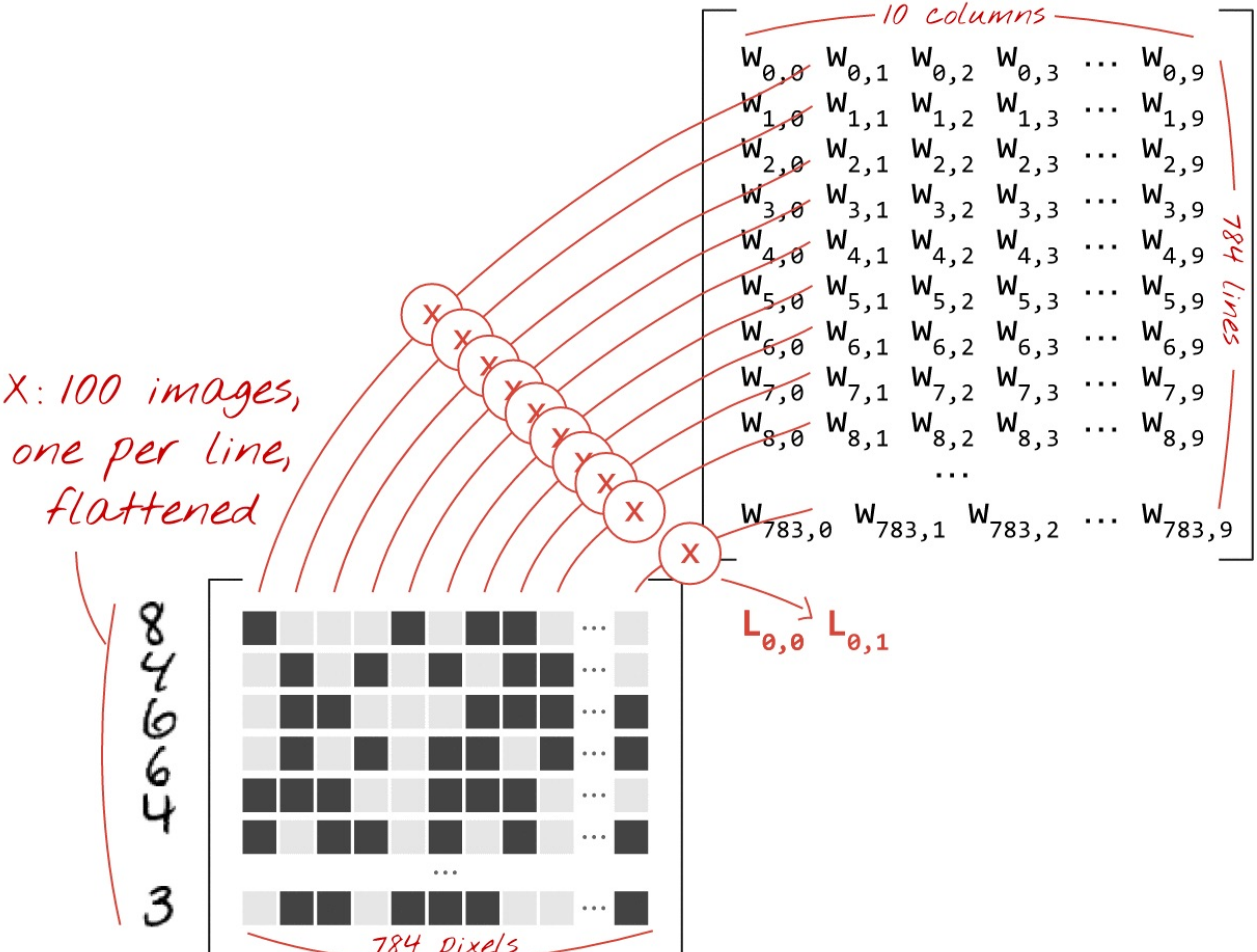
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/



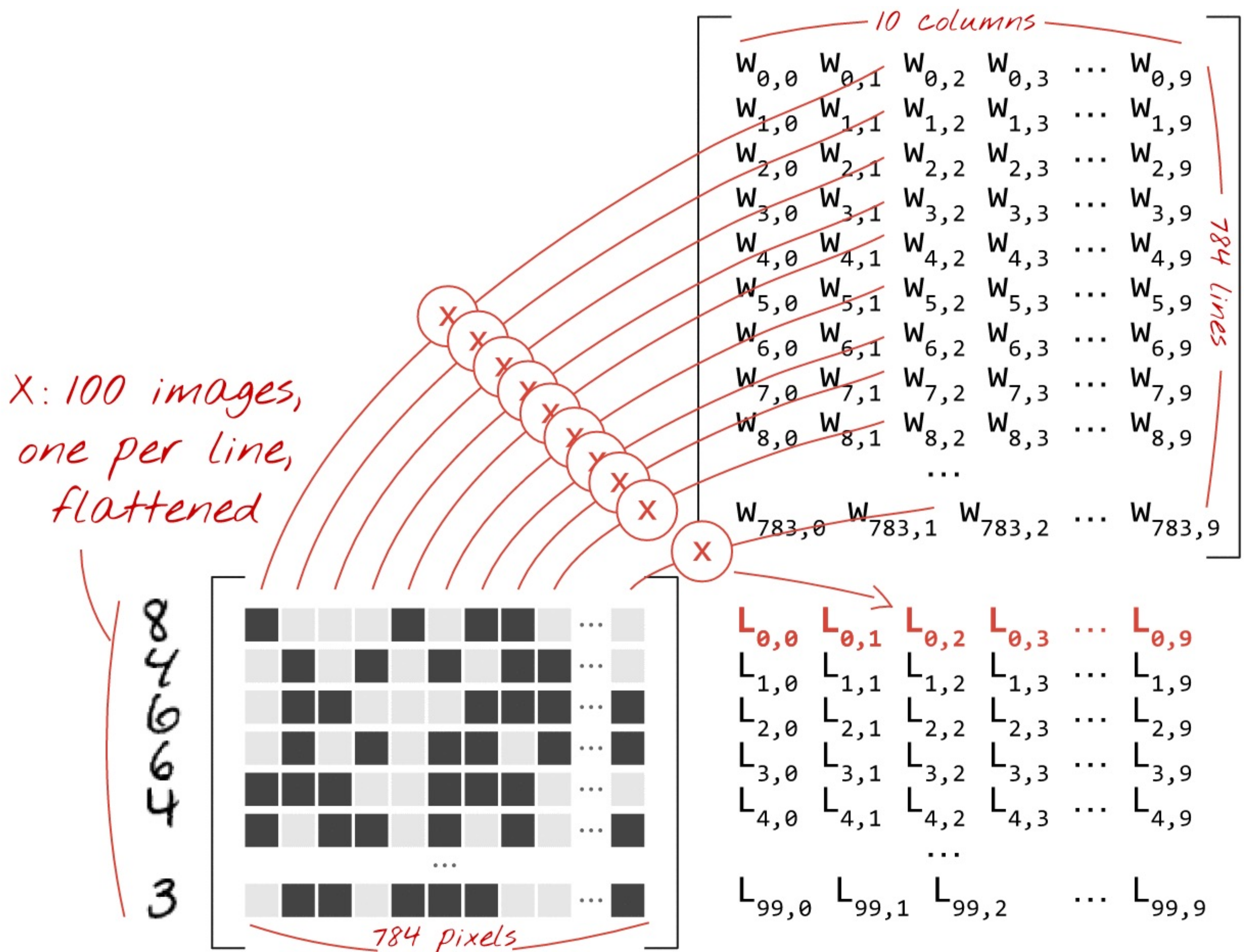
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

What are "weights" and "biases" ?

How is the "cross-entropy" computed ?

How exactly does the training algorithm work ?

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

$Y = f(X)$

Predictions — Y[100, 10]

Images — X[100, 784]

Weights — W[784,10]

Biases — b[10]

$$Y = softmax(X.W + b)$$

applied line by line

matrix multiply

broadcast on all lines

tensor shapes in [ ]

```
Y = tf.nn.softmax(tf.matmul(X, W) + b)
```

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow (Python) Softmax

Predictions:
Y[100, 10]

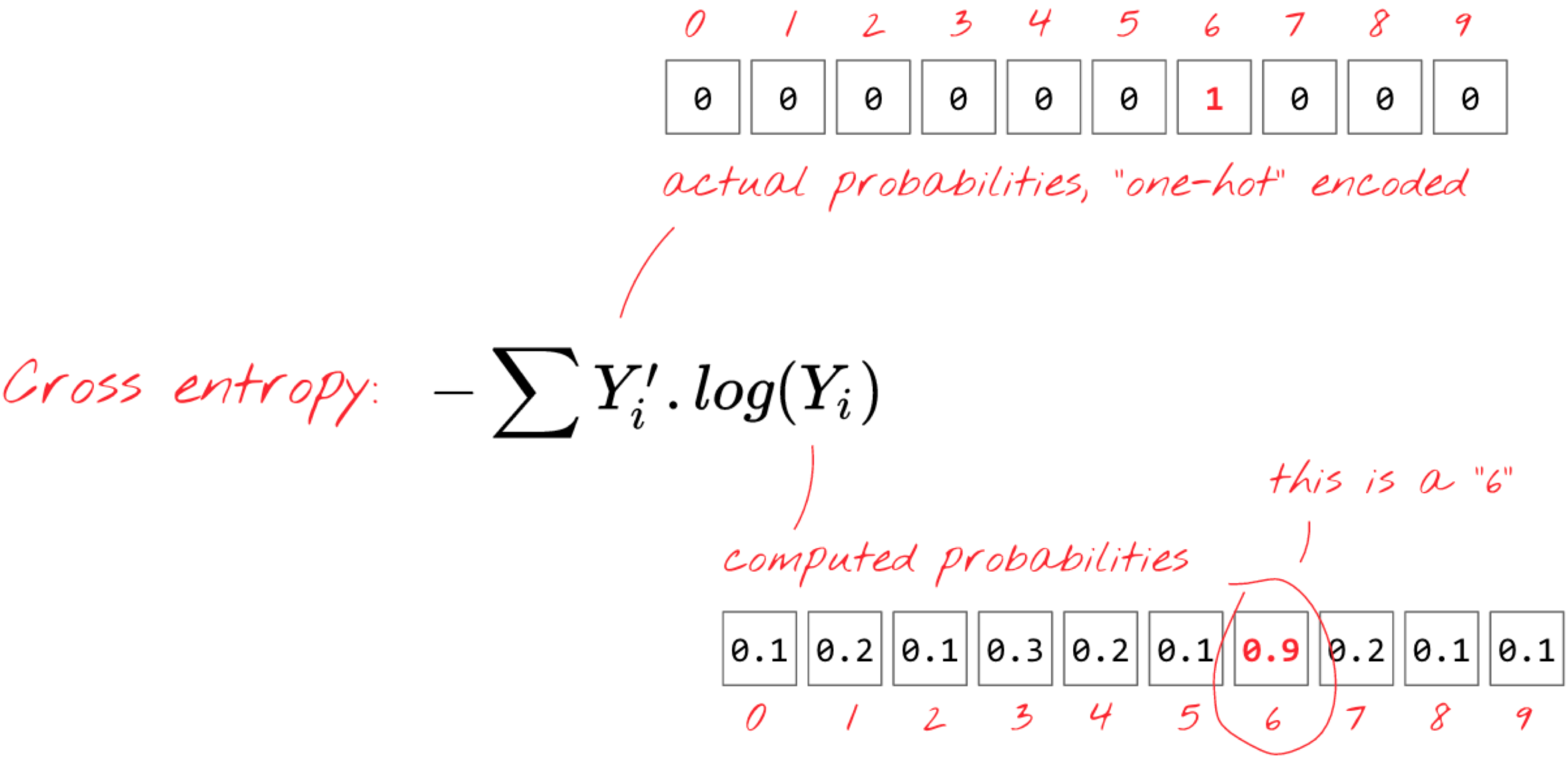
tensor shapes: X[100, 784] W[784, 10] b[10]

```
Y = tf.nn.softmax(tf.matmul(X, W) + b)
```

matrix multiply

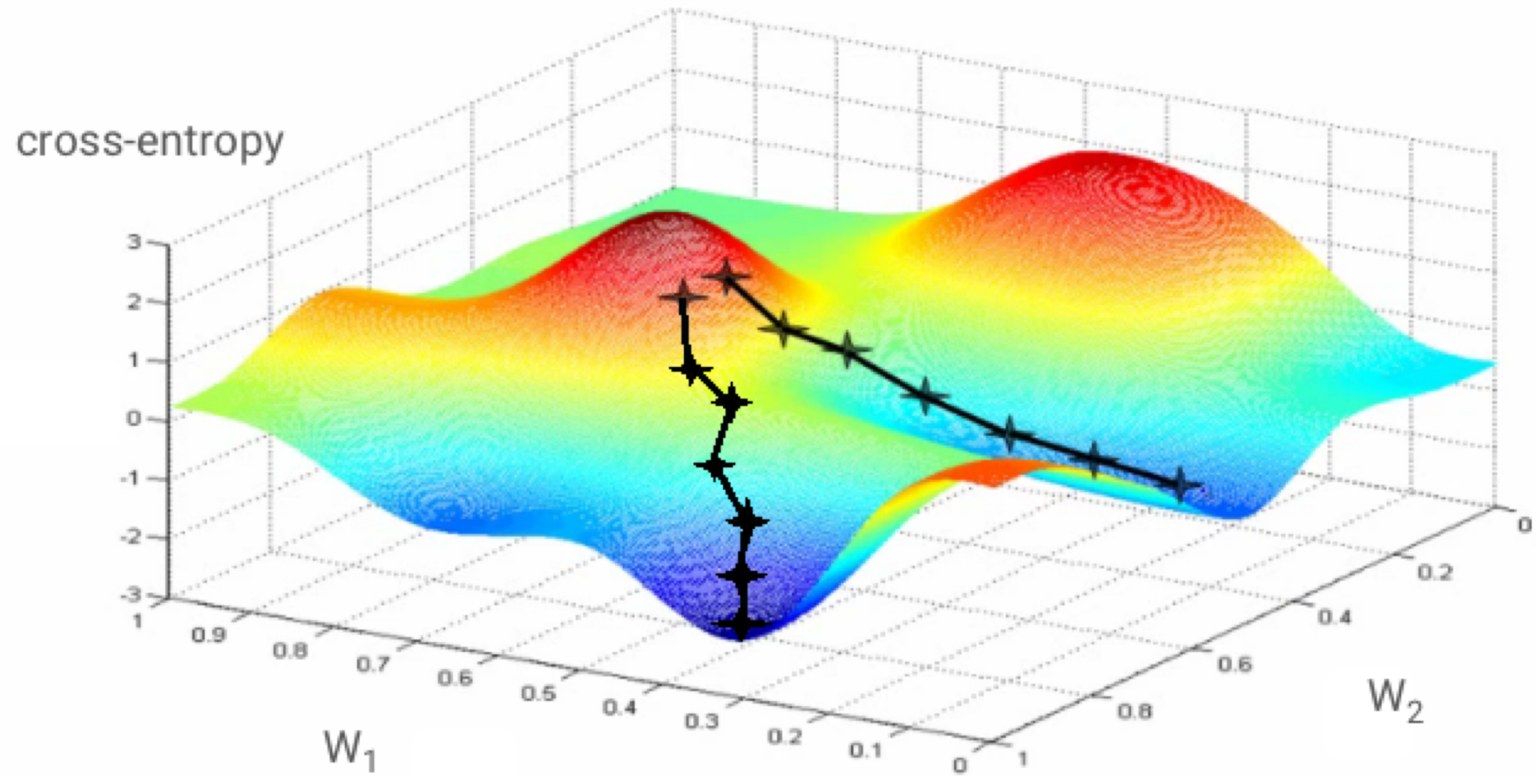
broadcast
on all lines

Cross Entropy



$$-\sum Y_i' . log(Y_i)$$

Minimizing Cross Entropy (Minimizing Loss)



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Training Loop

Training digits and labels
=> loss function
=> gradient (partial derivatives)
=> steepest descent
=> update weights and biases
=> repeat with next mini-batch of training images and labels

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

"mini-batches": 100 images and labels

```python
import tensorflow as tf
```

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5

mnist_1.0_softmax.py

```
import tensorflow as tf
X = tf.placeholder(tf.float32, [None, 28, 28, 1])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))

init = tf.initialize_all_variables()
```

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5

mnist_1.0_softmax.py

```
# model
Y = tf.nn.softmax(tf.matmul(tf.reshape(X, [-1, 784]), W) + b)
# placeholder for correct labels
Y_ = tf.placeholder(tf.float32, [None, 10])

# loss function
cross_entropy = -tf.reduce_sum(Y_ * tf.log(Y))
# % of correct answers found in batch
is_correct = tf.equal(tf.argmax(Y,1), tf.argmax(Y_,1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
```

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5

mnist_1.0_softmax.py

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

for i in range(1000):
    # load batch of images and correct answers
    batch_X, batch_Y = mnist.train.next_batch(100)
    train_data={X: batch_X, Y_: batch_Y}

    # train
    sess.run(train_step, feed_dict=train_data)
```

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5

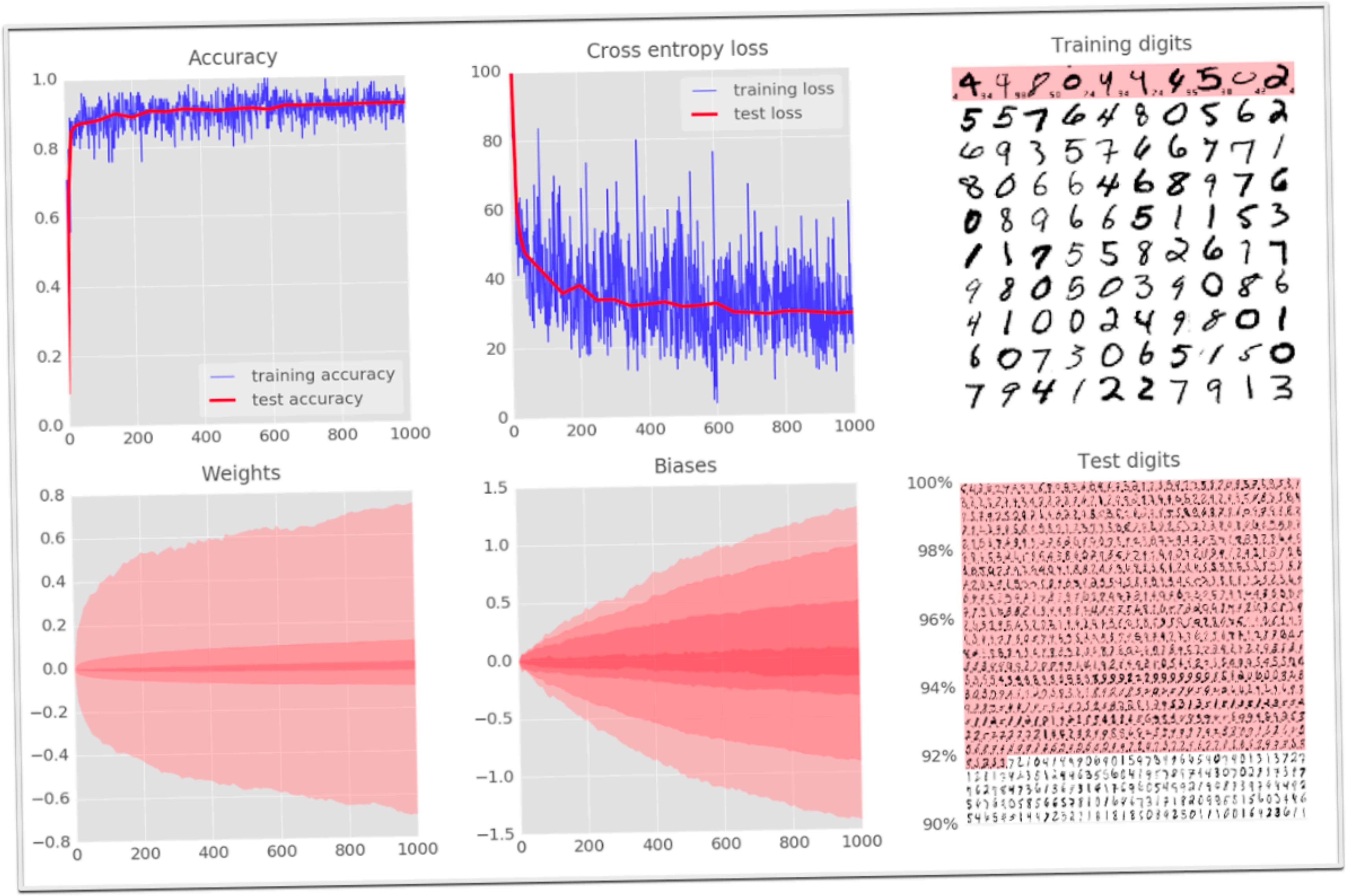
mnist_1.0_softmax.py

```
# success ?
a,c = sess.run([accuracy, cross_entropy],
feed_dict=train_data)


# success on test data ?
test_data={X: mnist.test.images, Y_: mnist.test.labels}
a,c = sess.run([accuracy, cross_entropy], feed=test_data)
```

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5

mnist_1.0_softmax.py

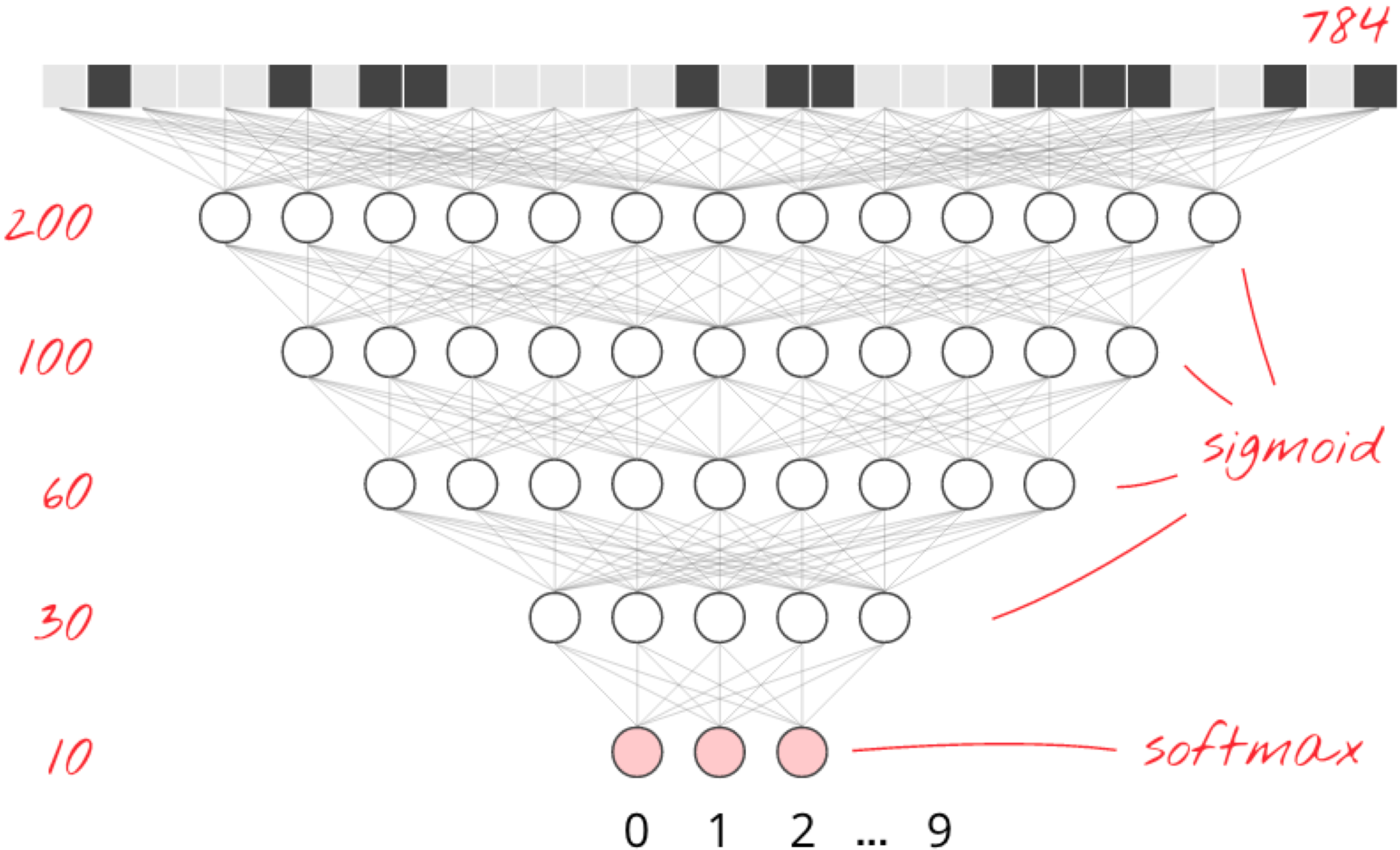


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5

Deep Learning

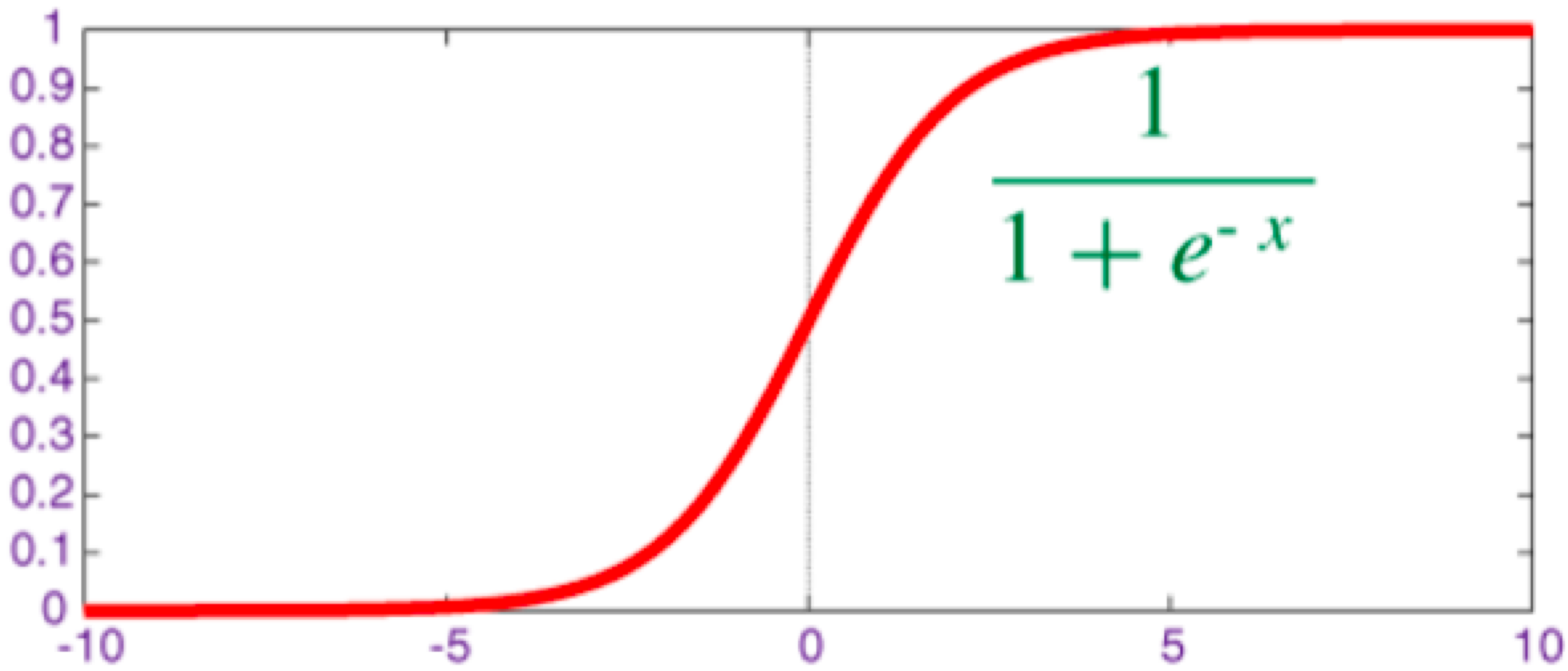
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

5 fully-connected layers



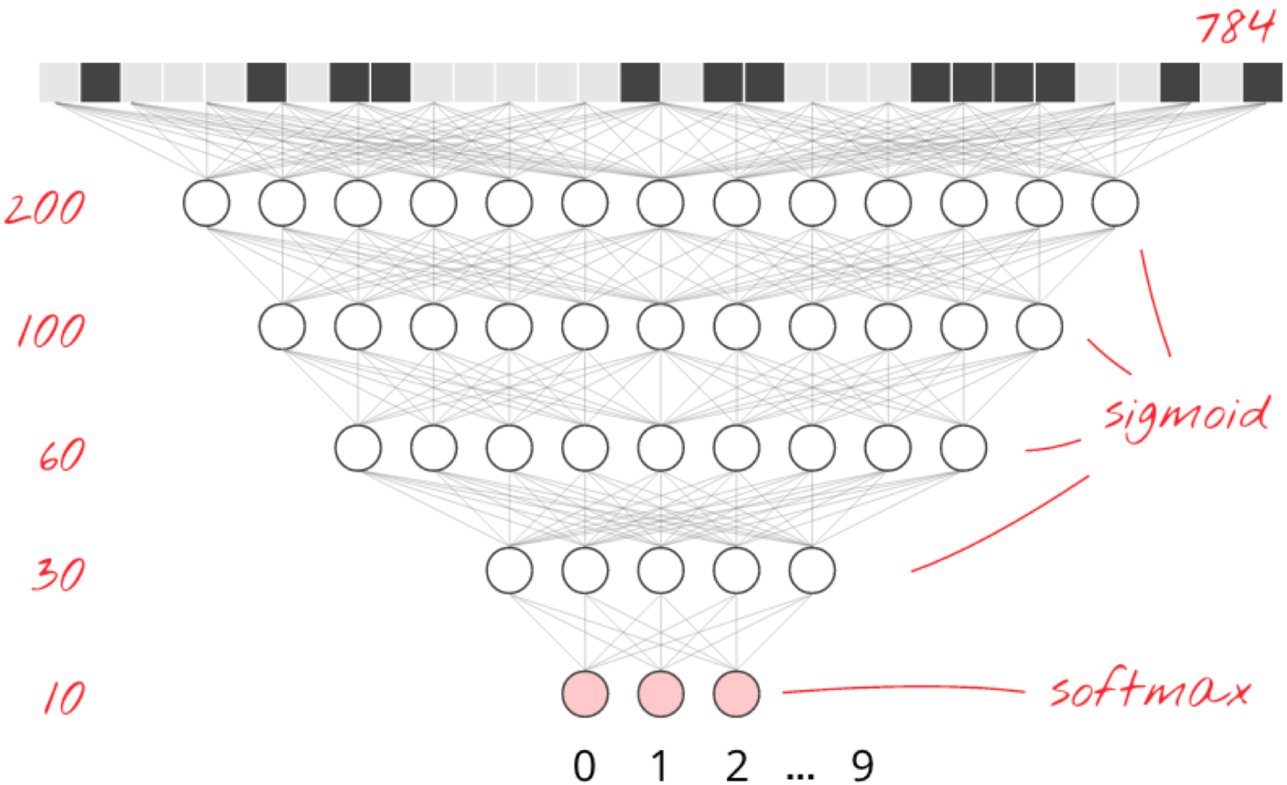
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

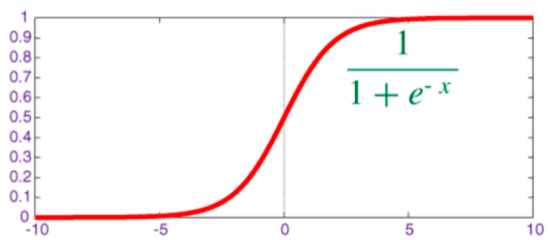
Sigmoid



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

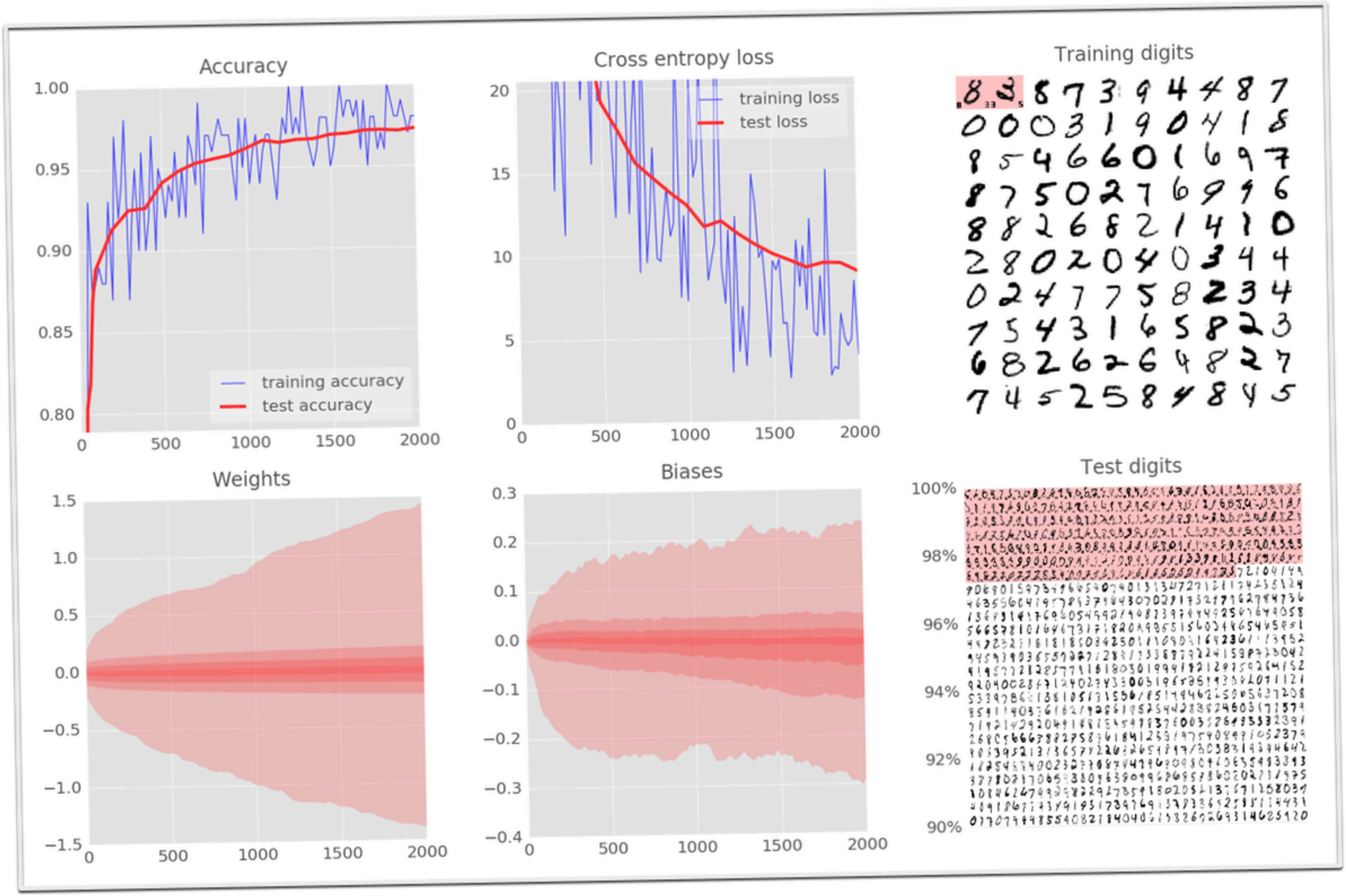
5 fully-connected layers





Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial



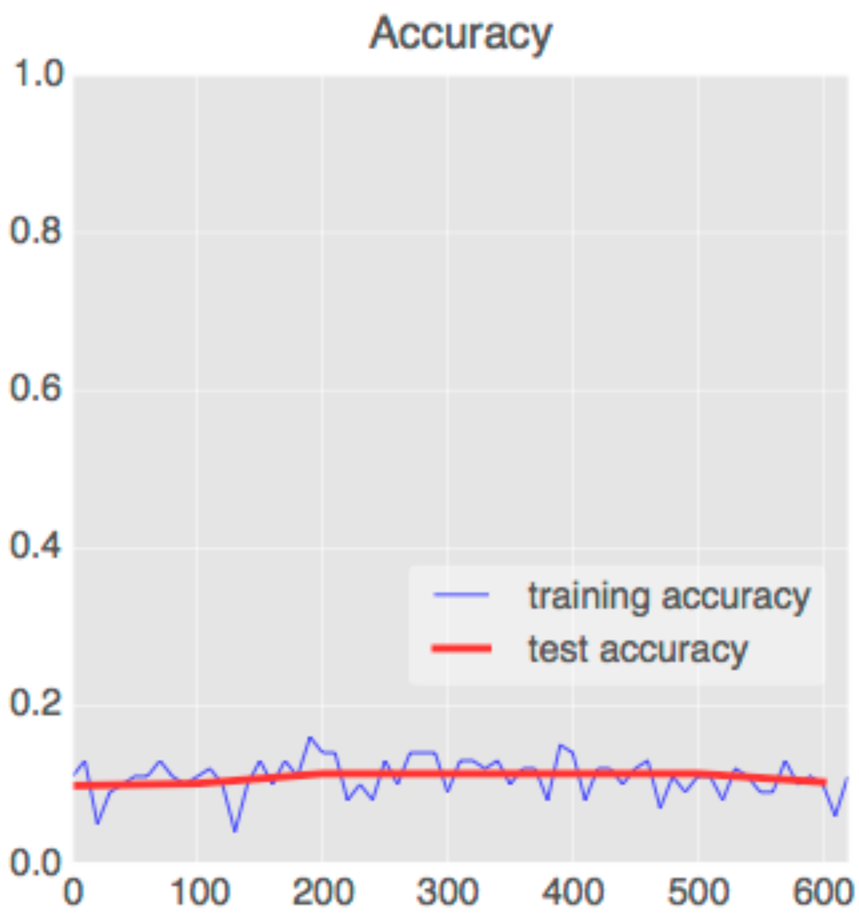
97%

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

ReLU

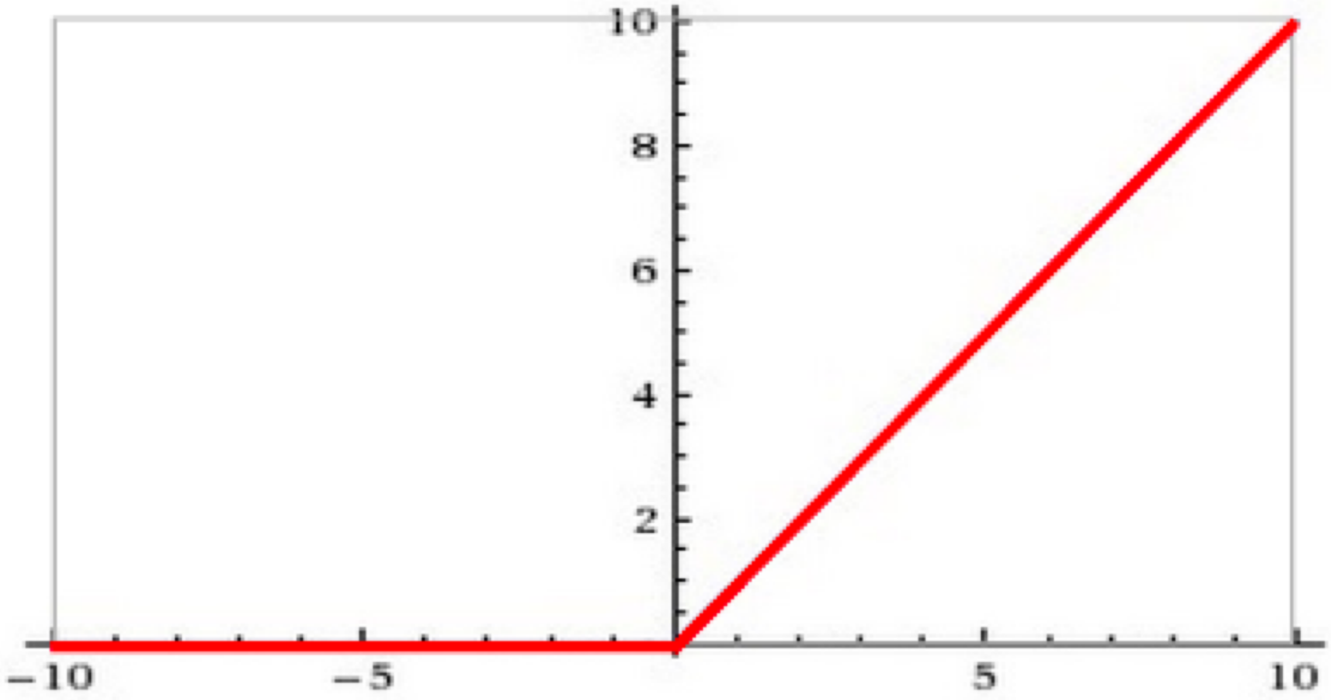
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial



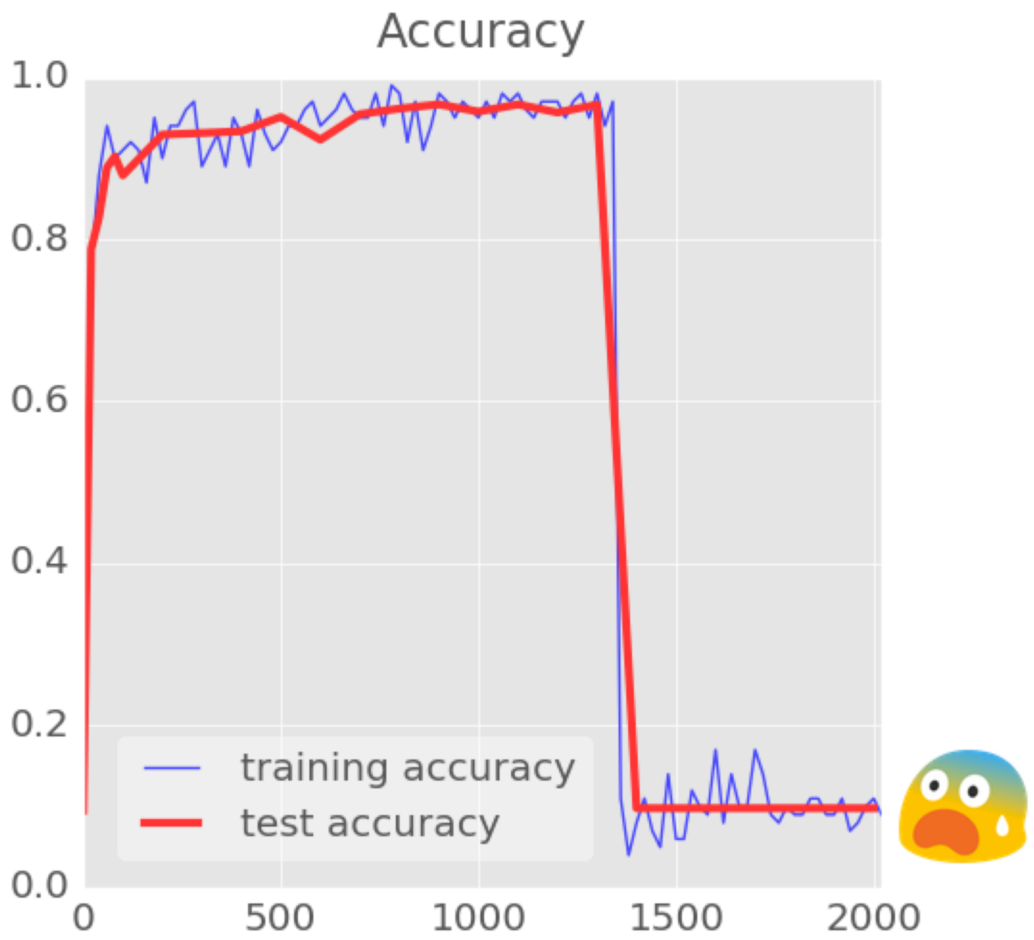
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

ReLU



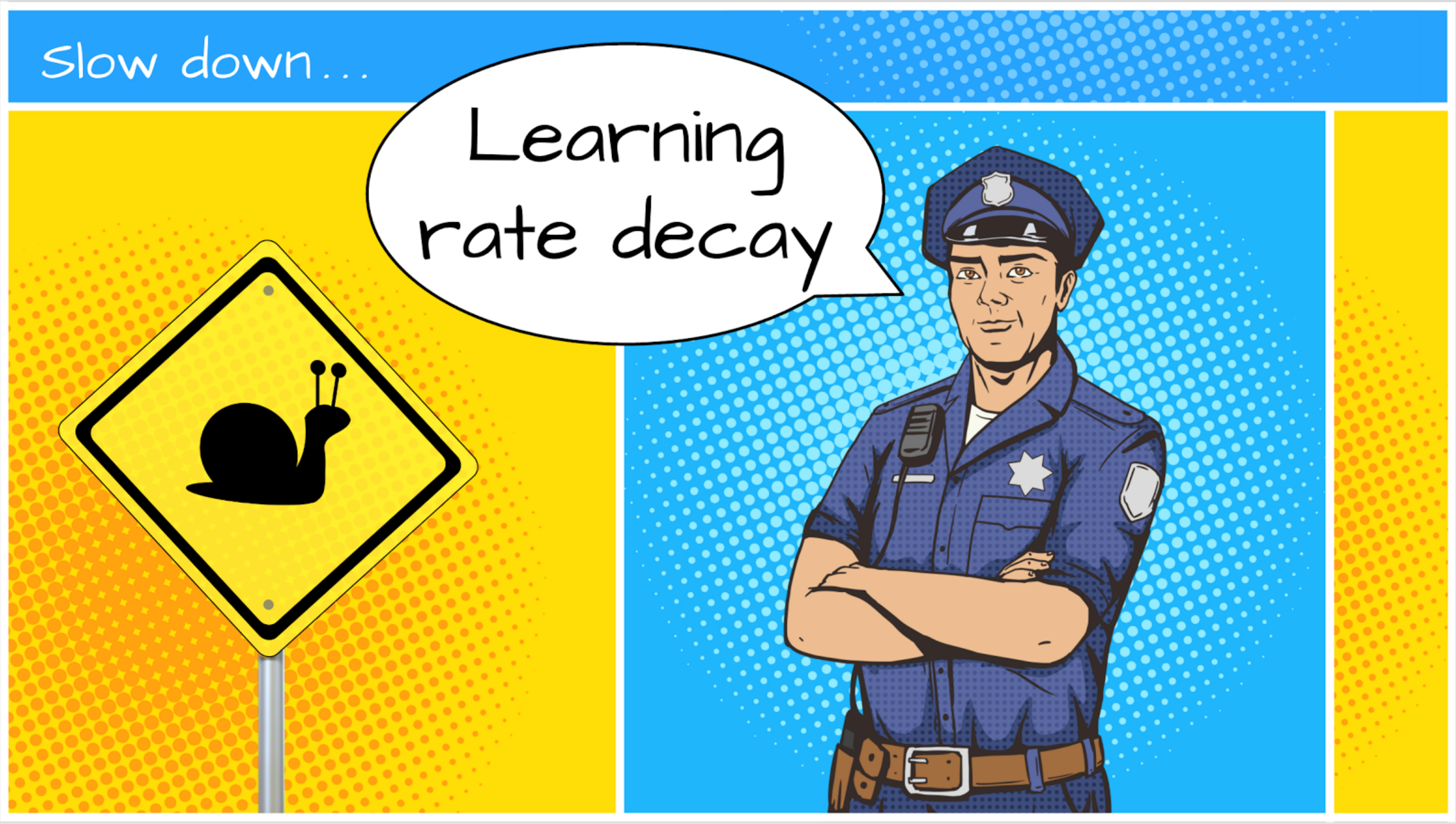
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial

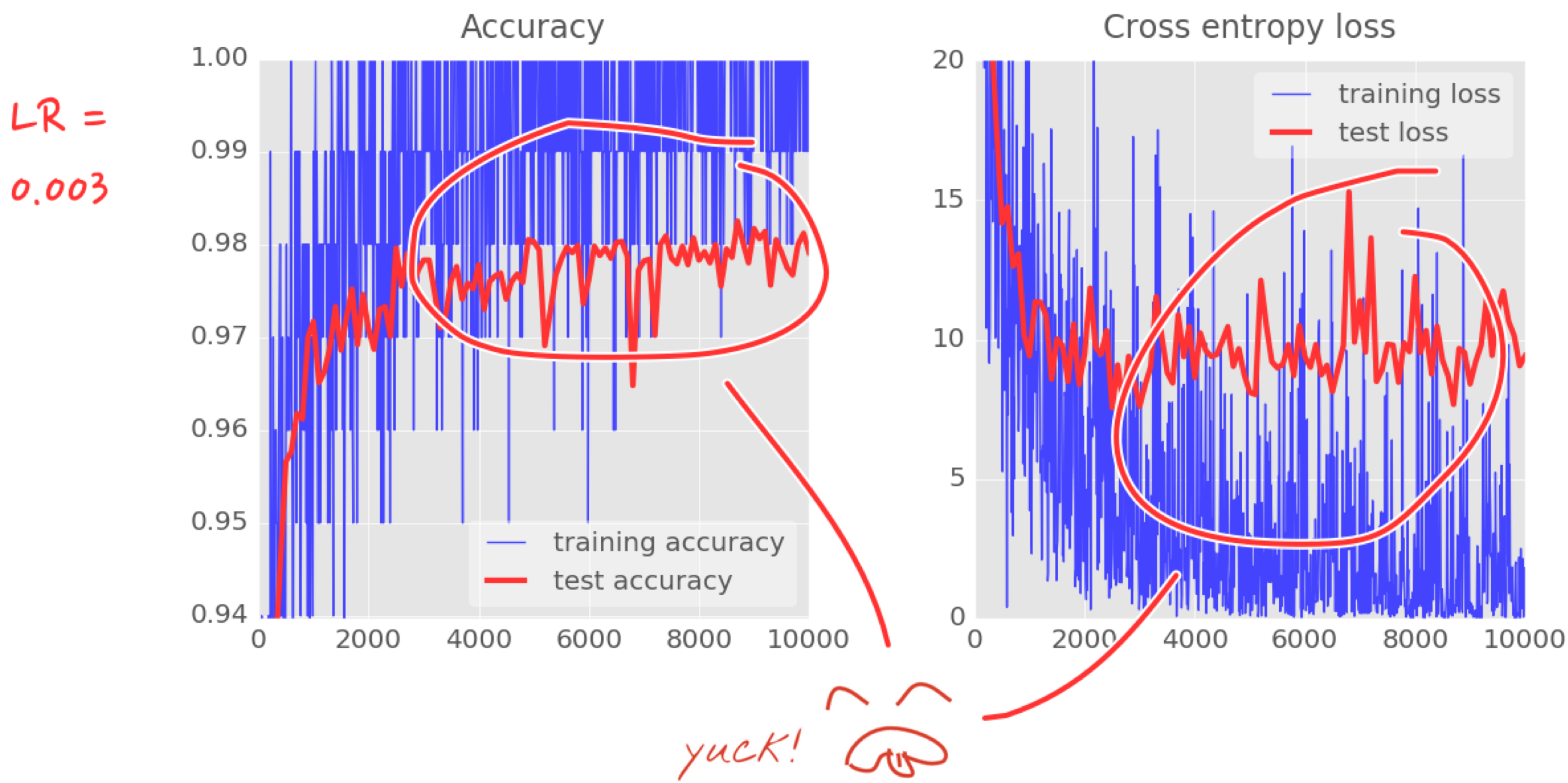


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Learning Rate

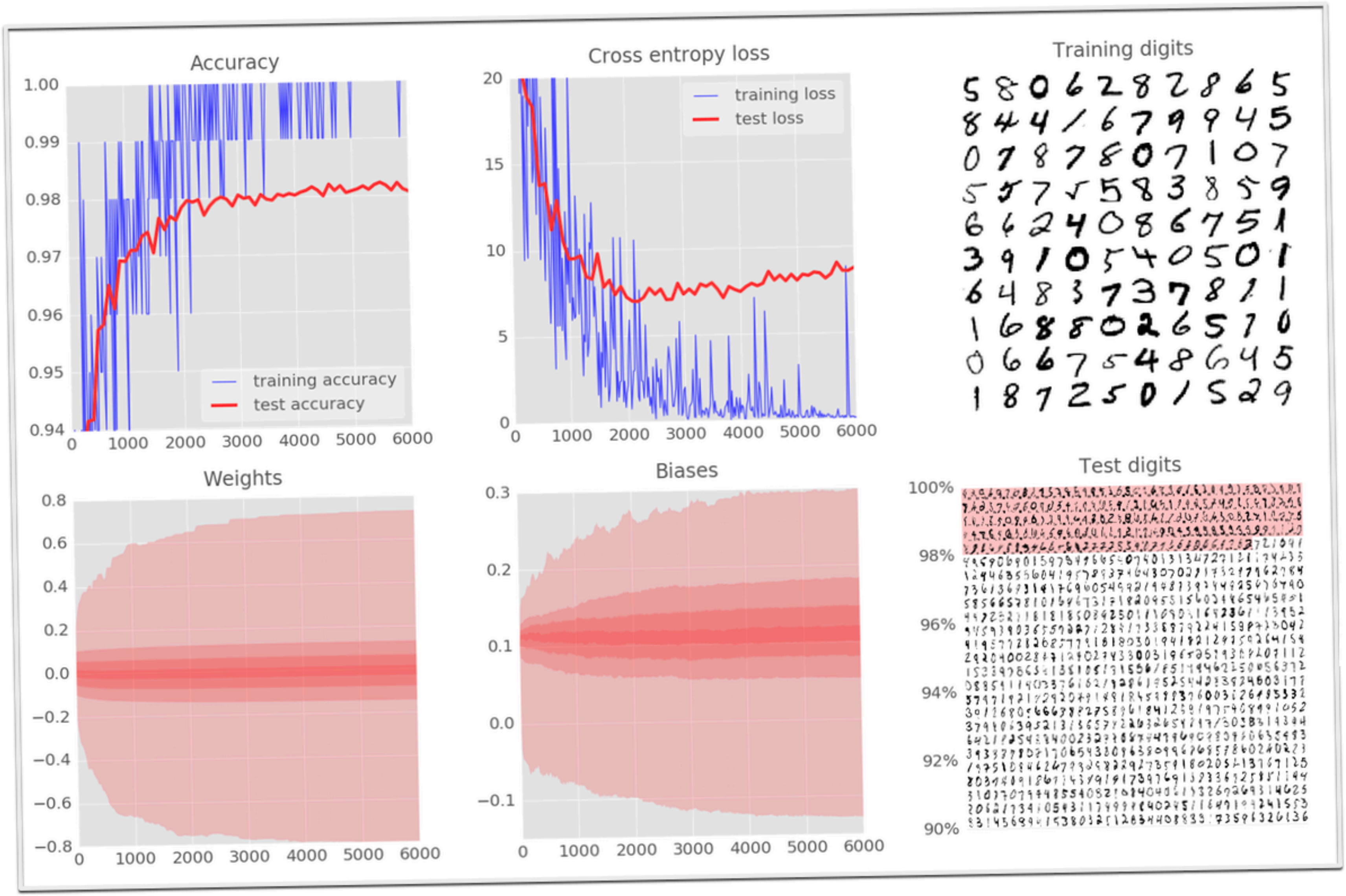


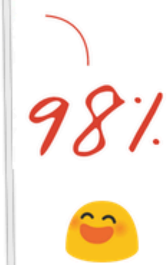
TensorFlow MNIST Tutorial



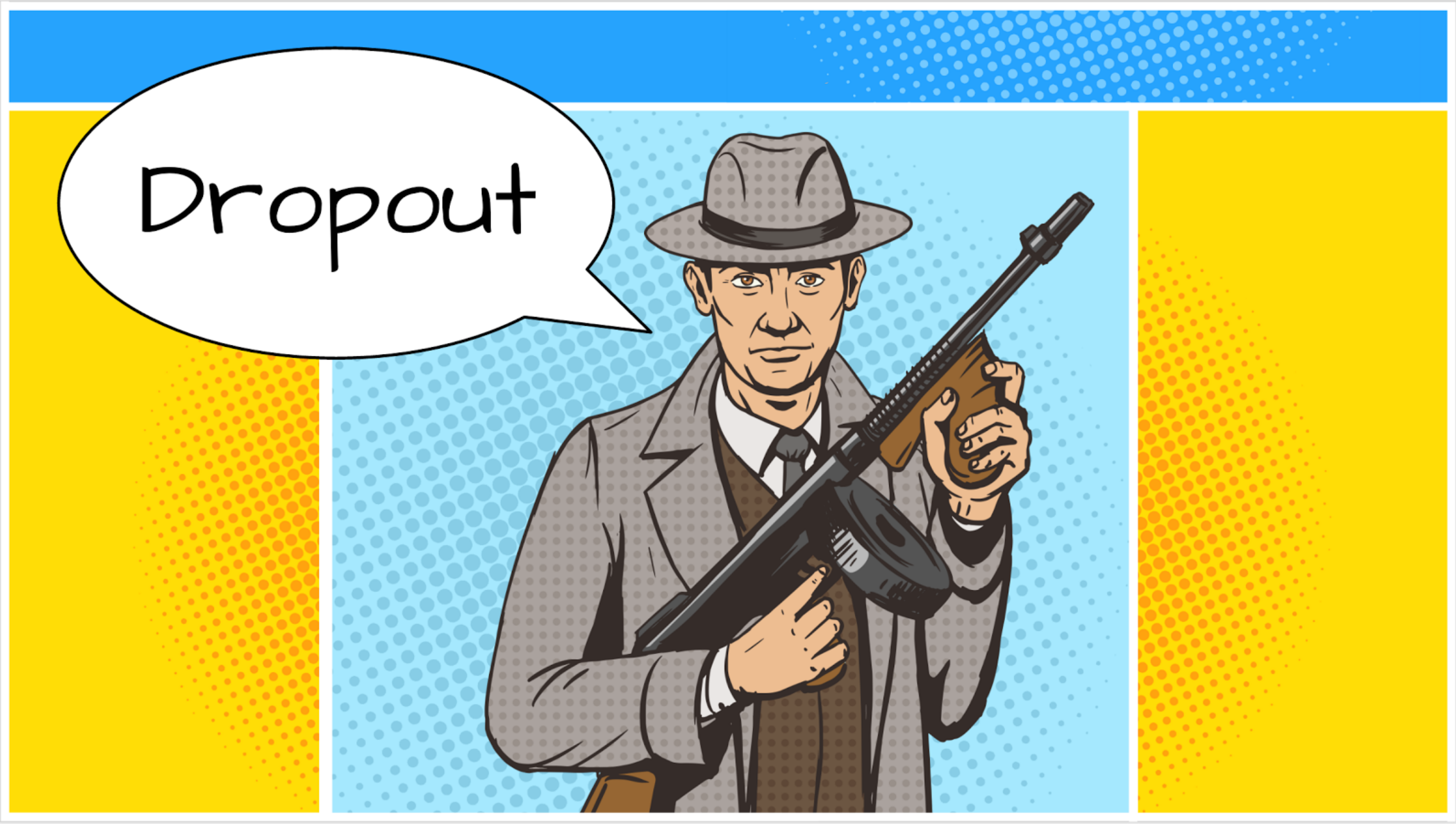
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial



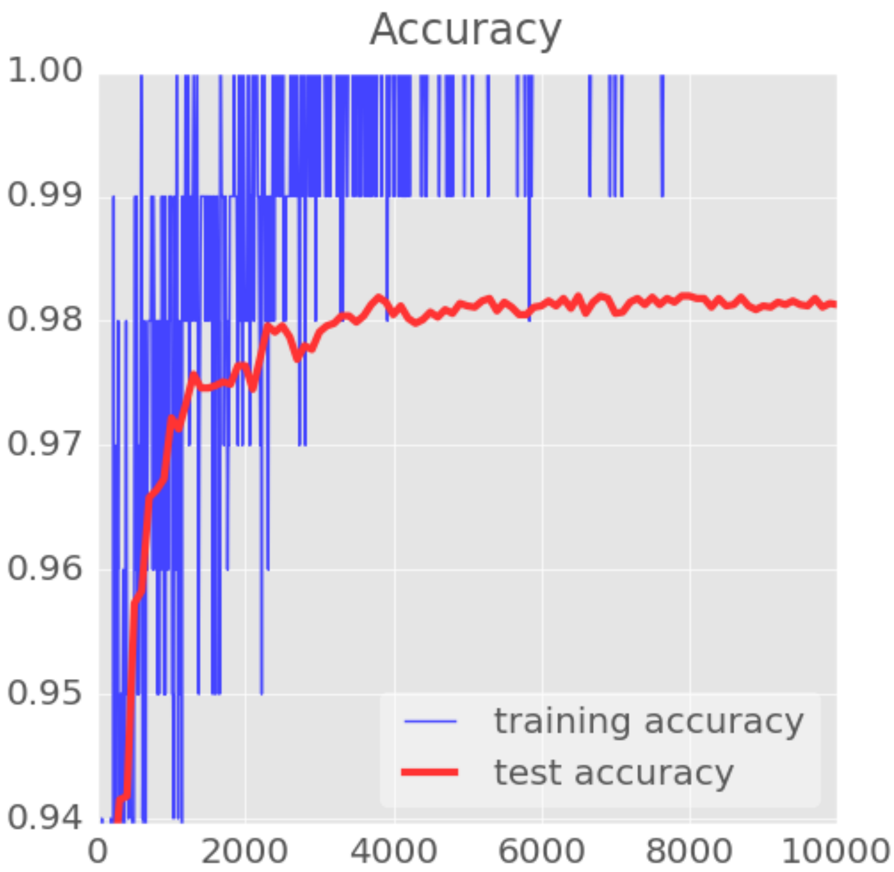


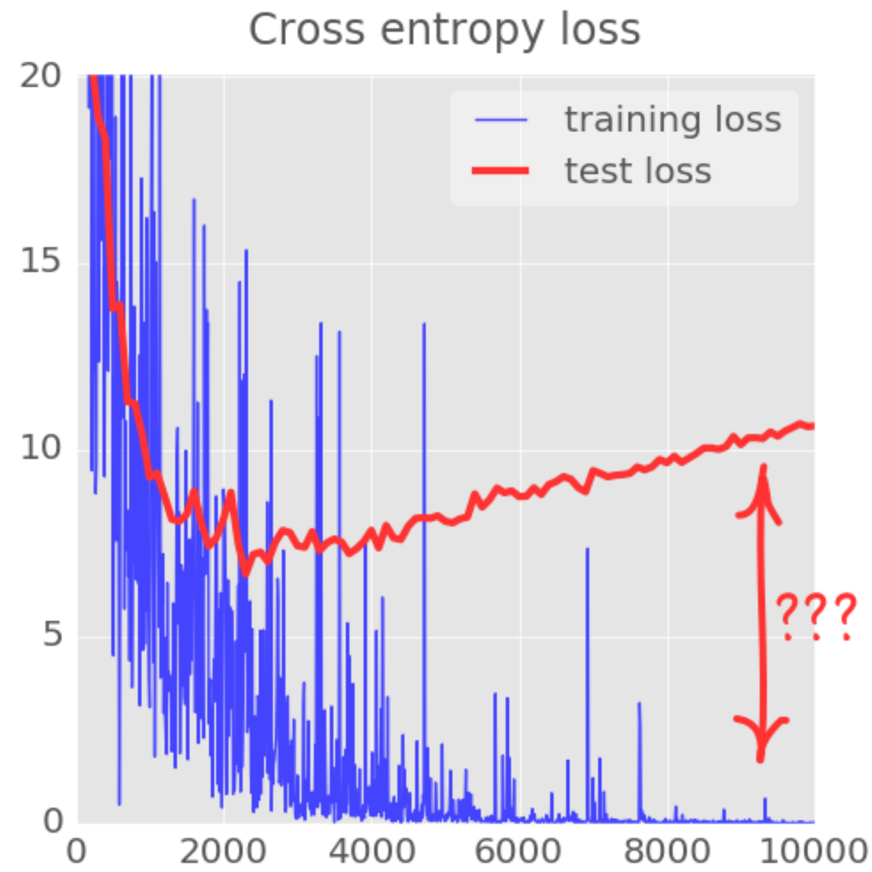
Dropout



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

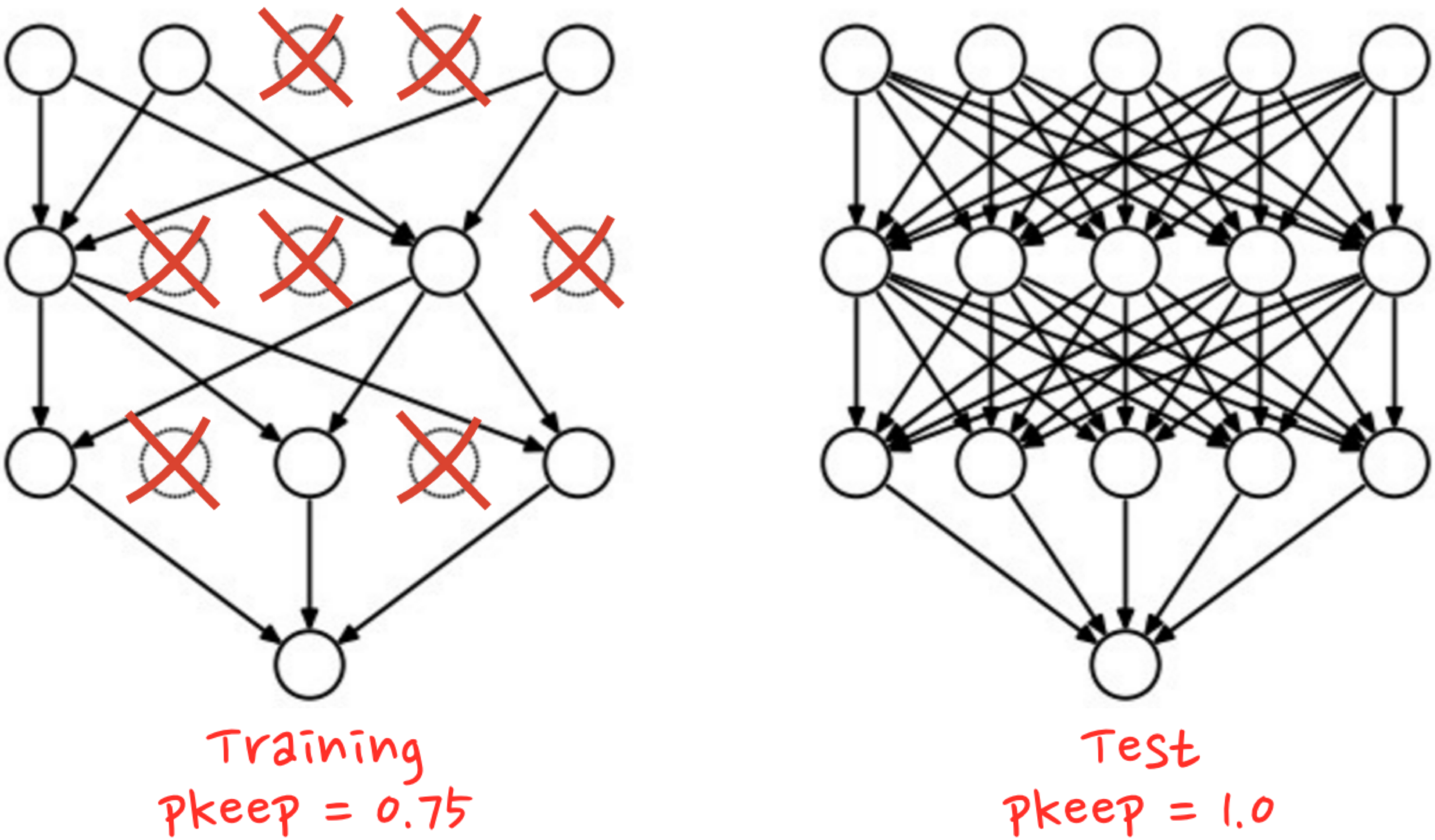
Overfitting





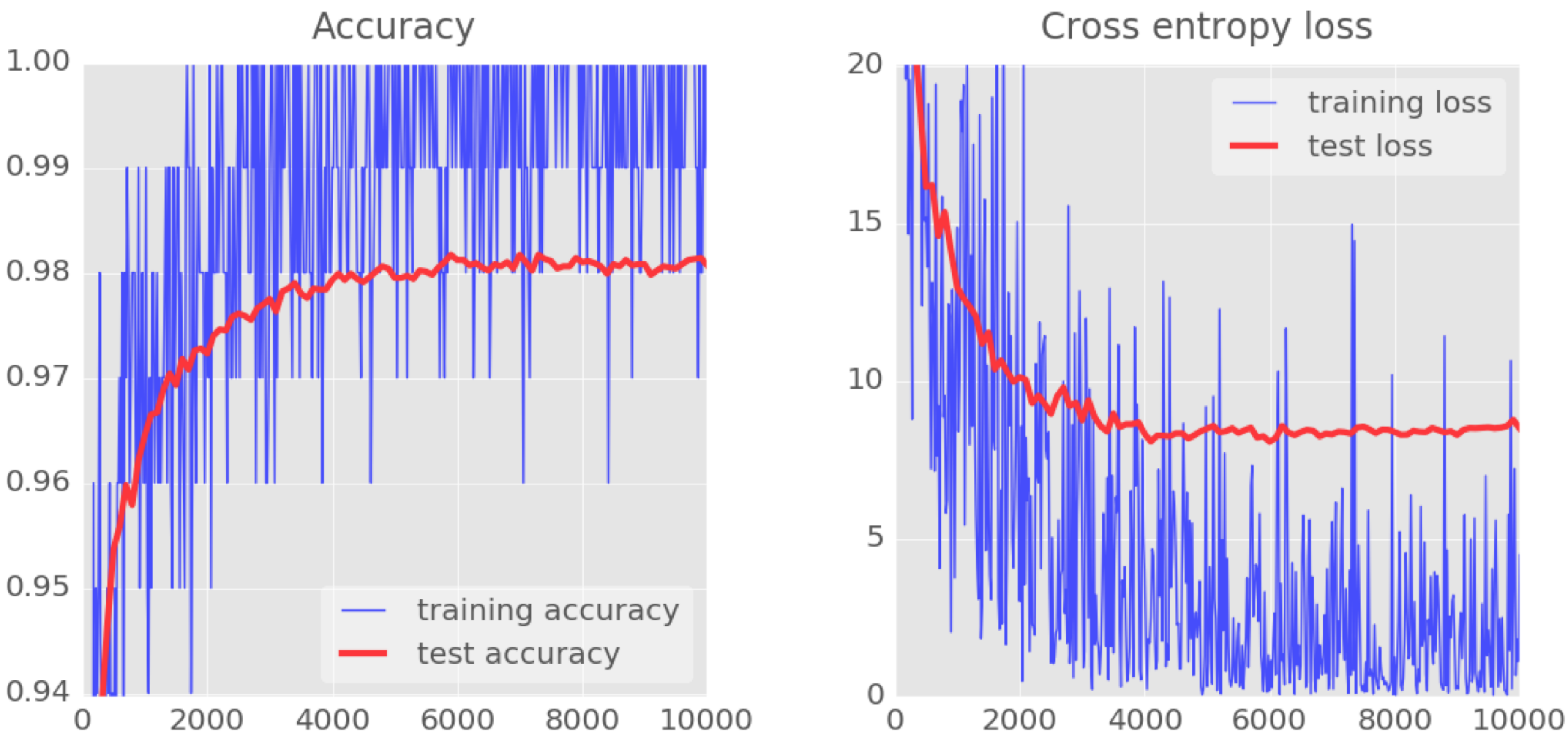
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Dropout



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

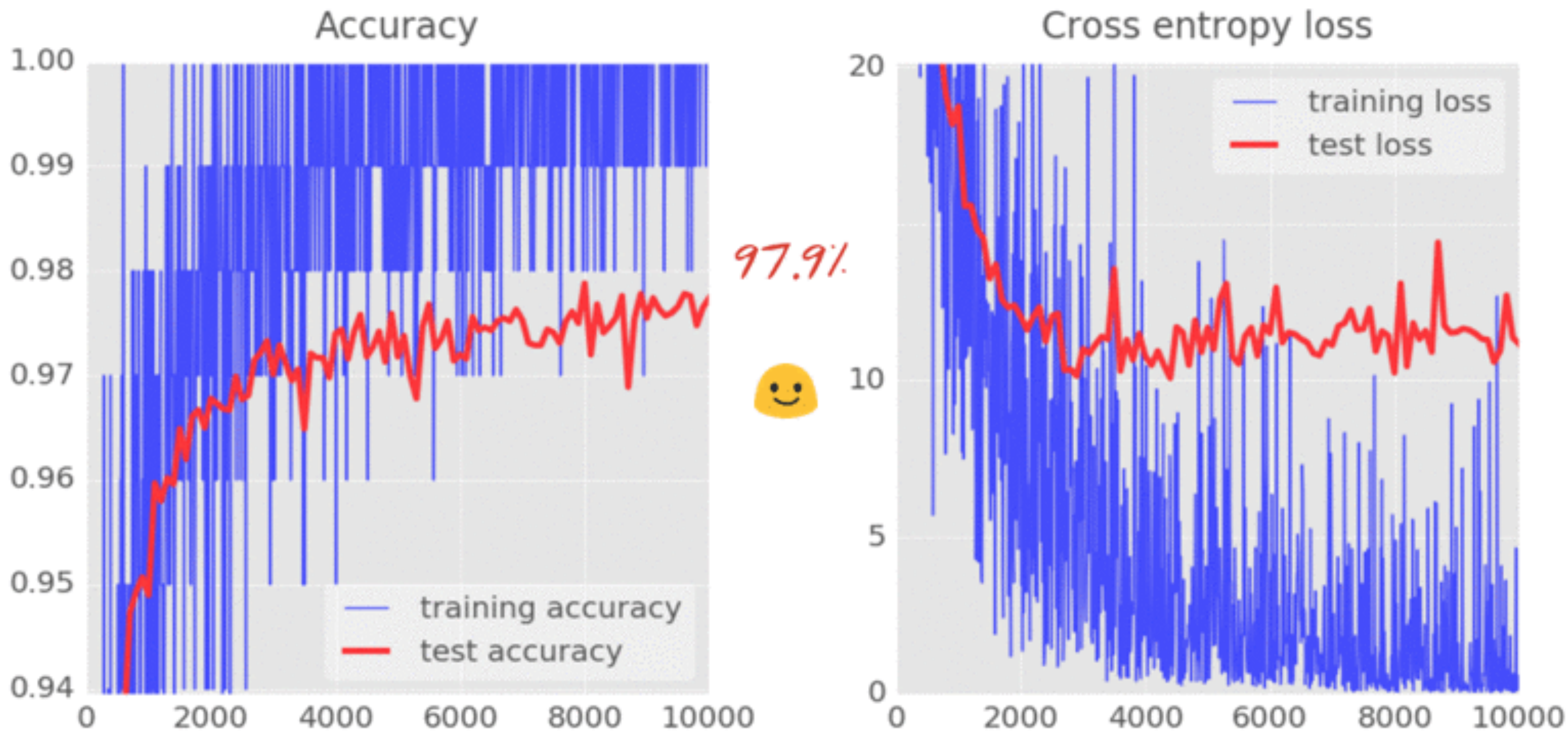
TensorFlow MNIST Tutorial



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial

5 layers
sigmoid

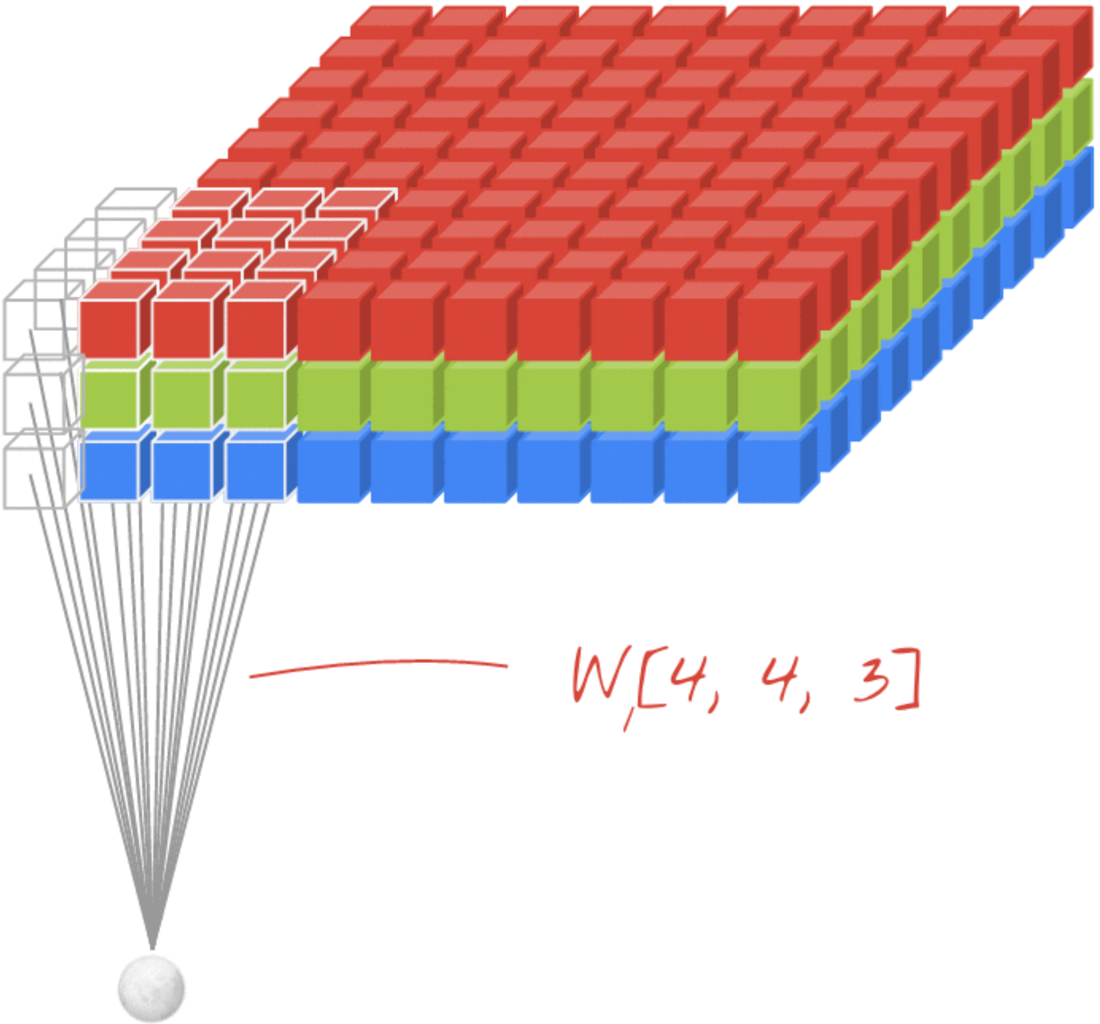


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Overfitting

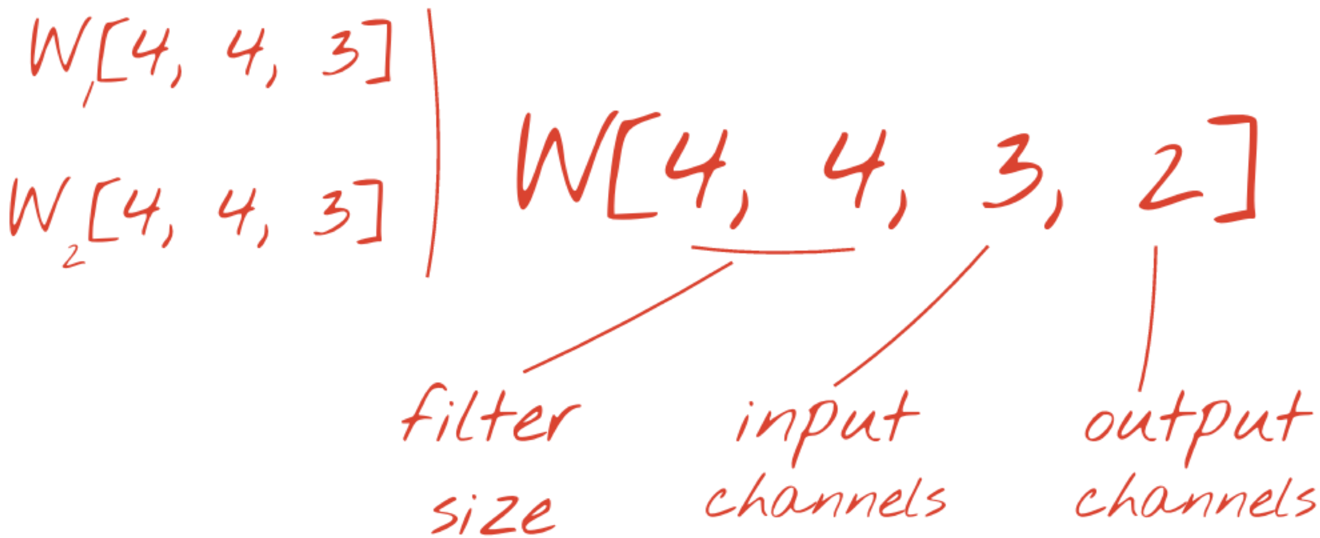
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Convolutional Layer



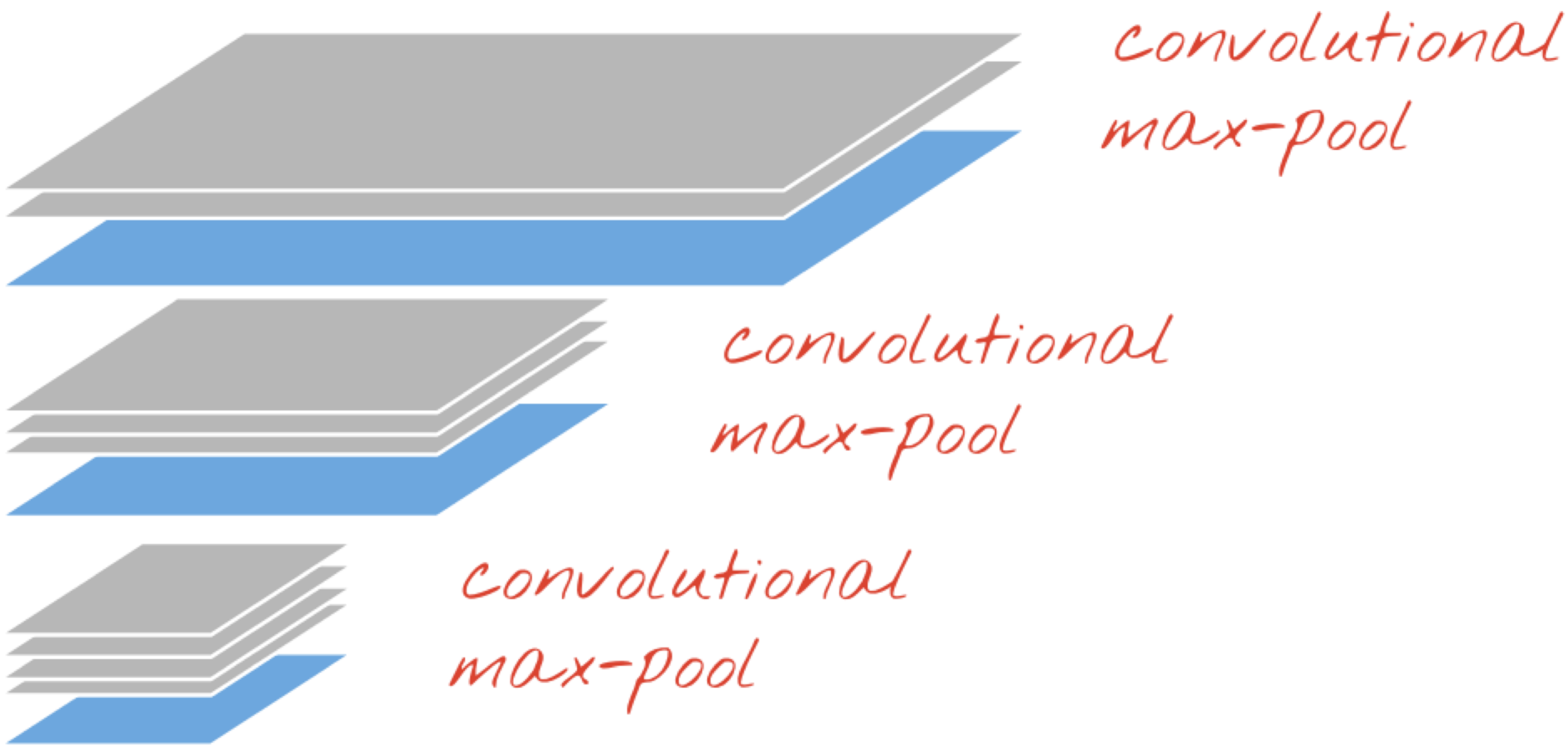
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Convolutional Layer



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Convolutional Max-Pool

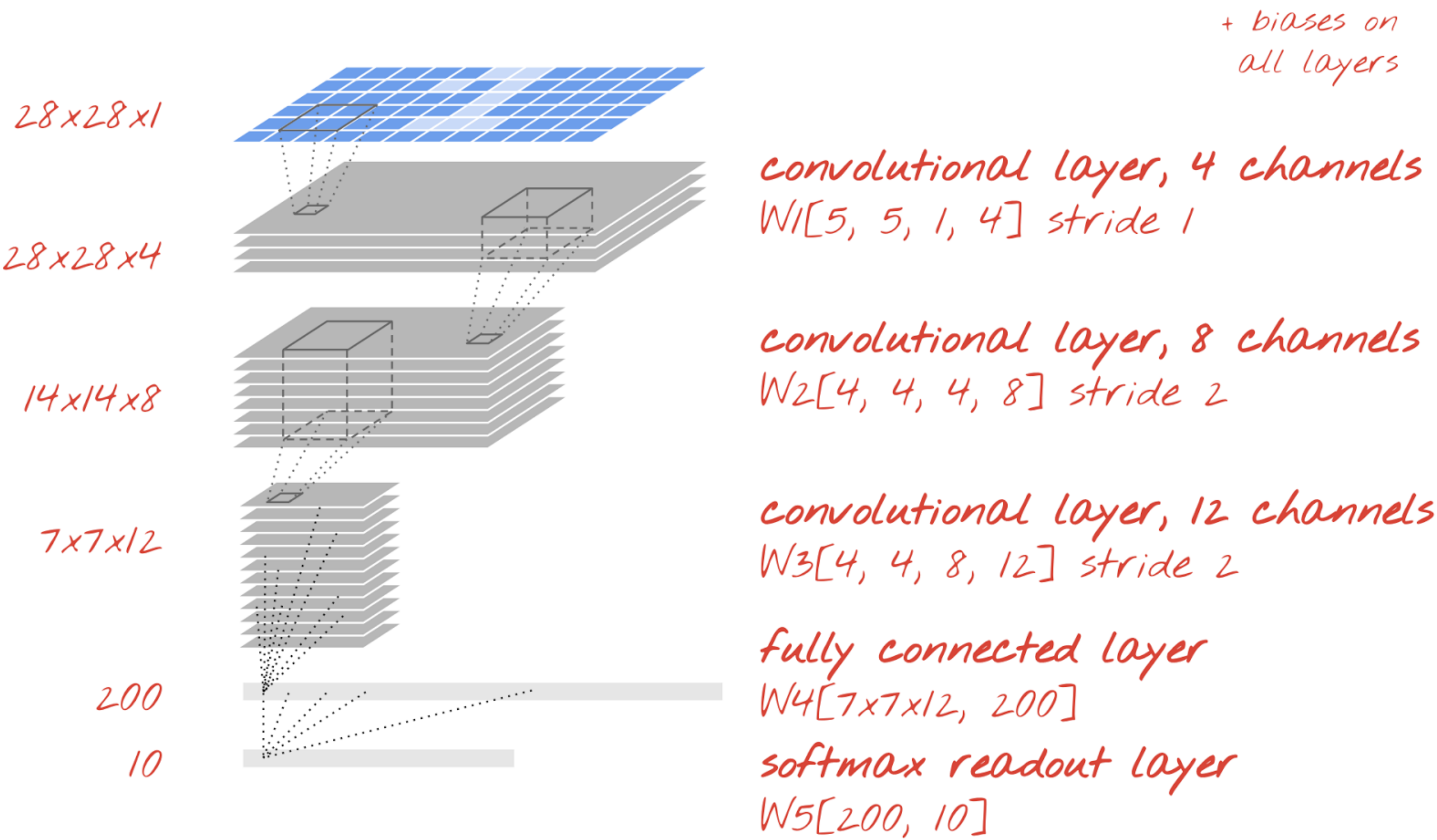


Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

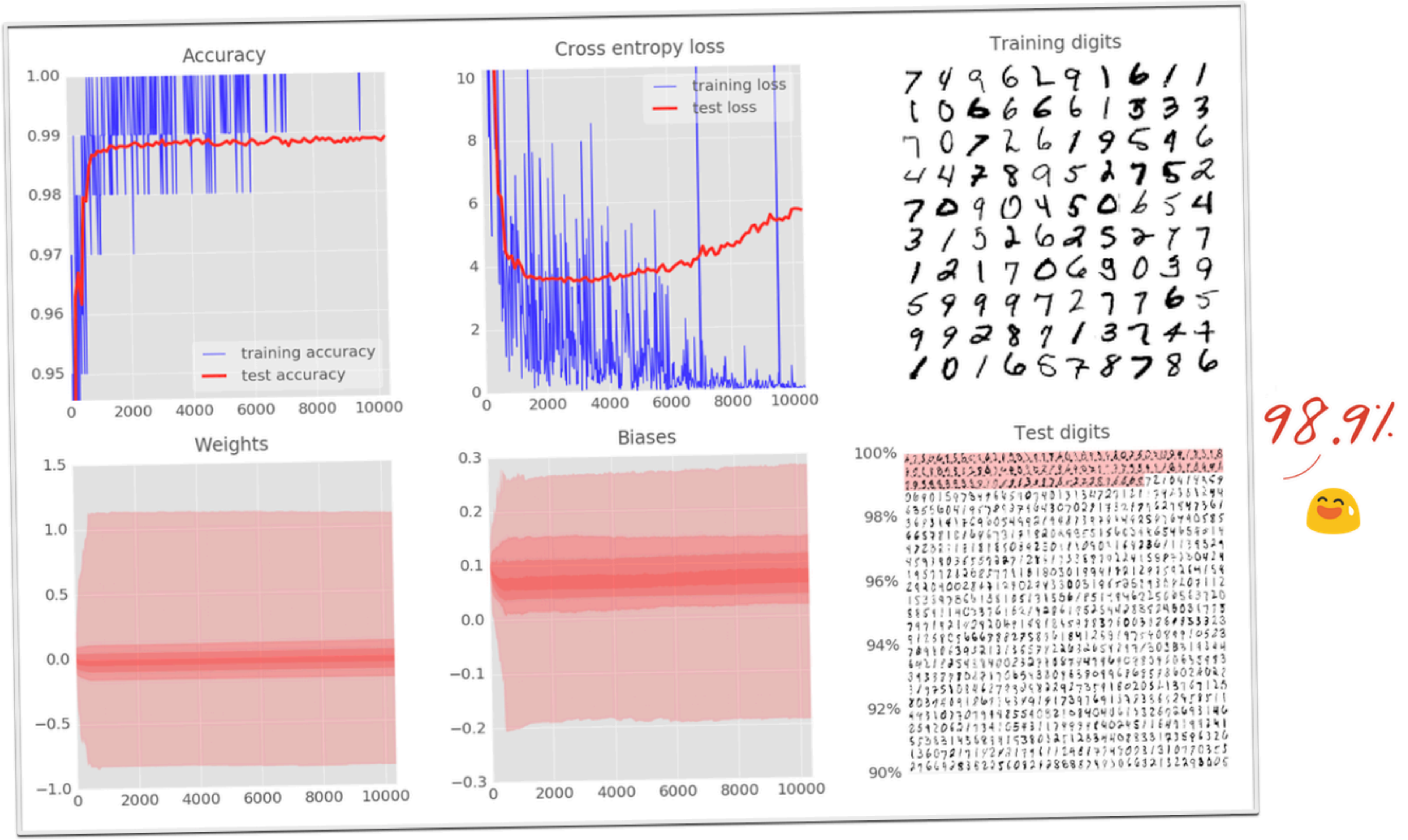
All Convolutional

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Bigger Convolutional Neural Network



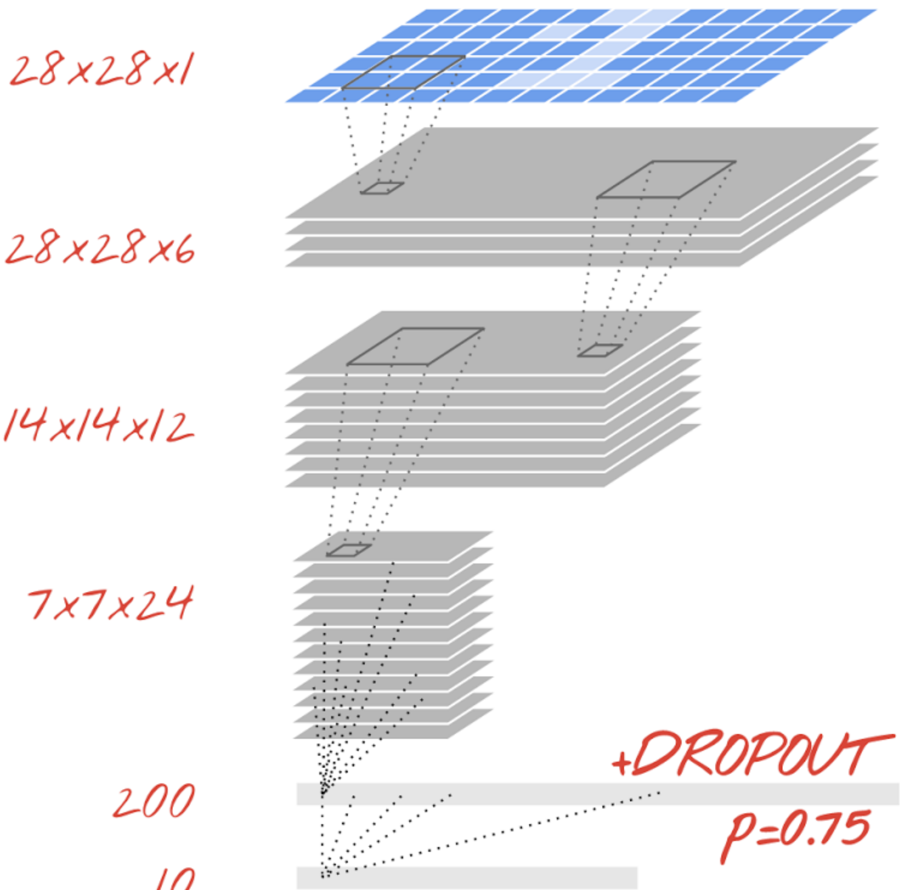
Bigger Convolutional Neural Network



Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

Bigger Convolutional Neural Network + Dropout

+ biases on all layers



convolutional layer, **6 channels**
W1[**6, 6**, 1, 6] stride 1

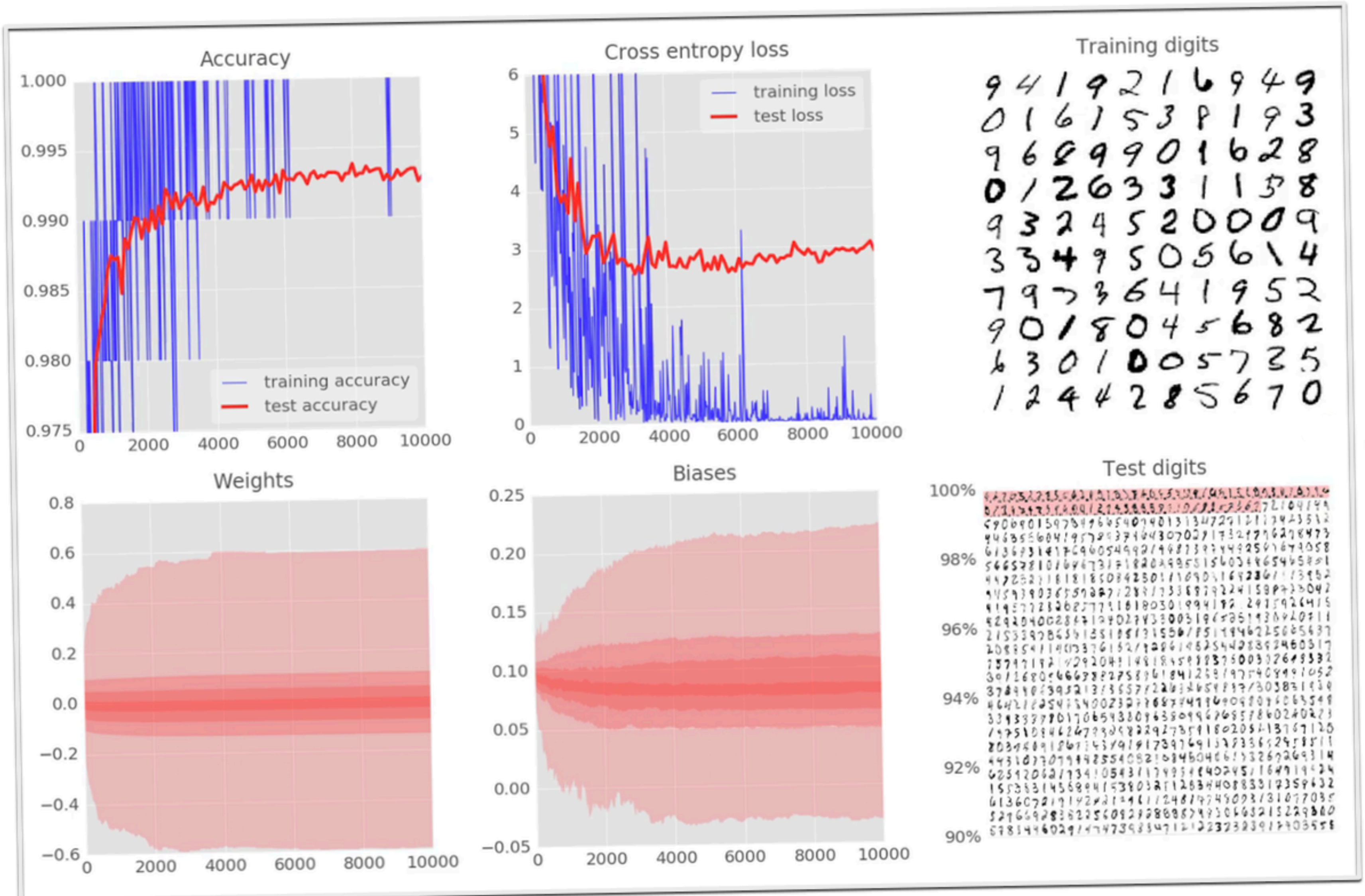
convolutional layer, **12 channels**
W2[**5, 5**, 6, 12] stride 2

convolutional layer, **24 channels**
W3[**4, 4**, 12, 24] stride 2

fully connected layer
W4[7x7x24, 200]

softmax readout layer
W5[200, 10]

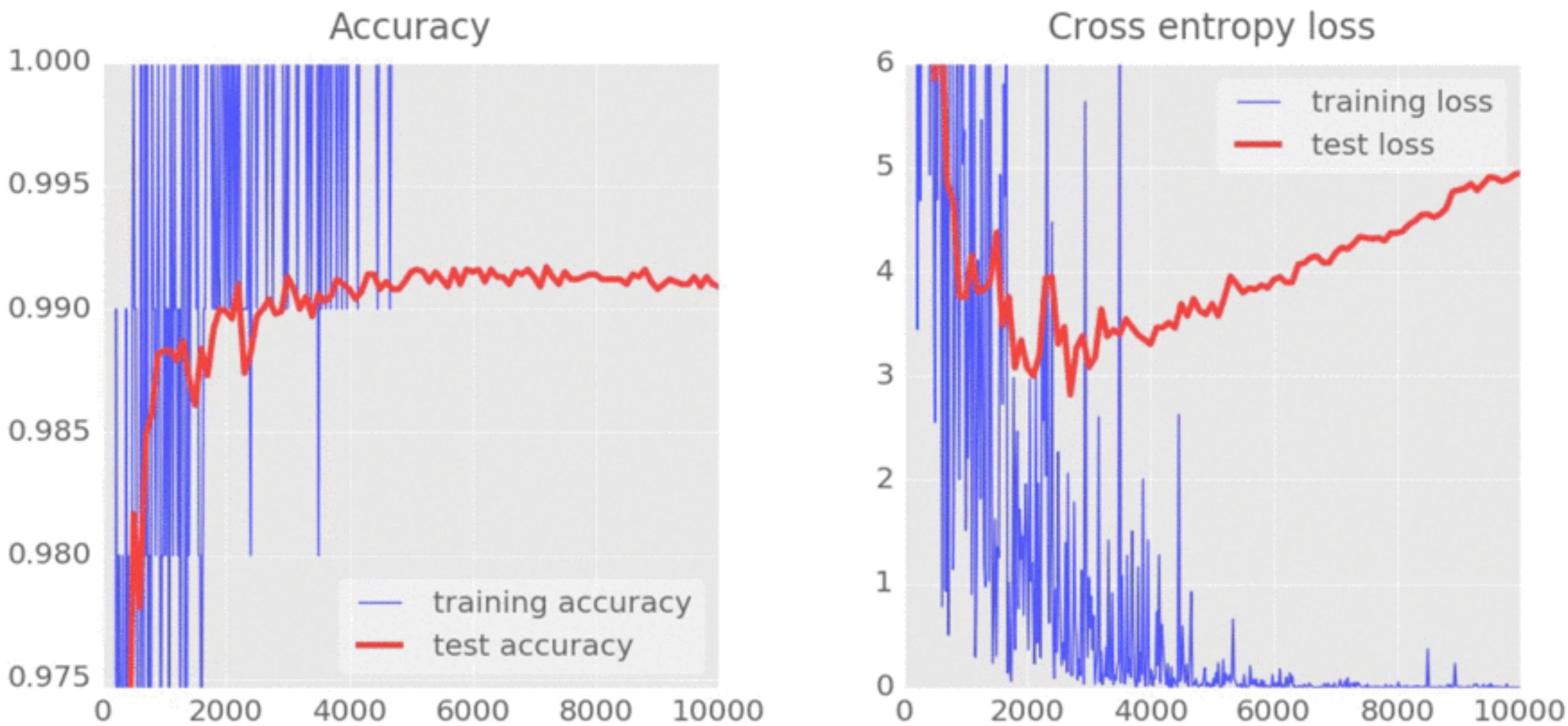
TensorFlow MNIST Tutorial



99.3%

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

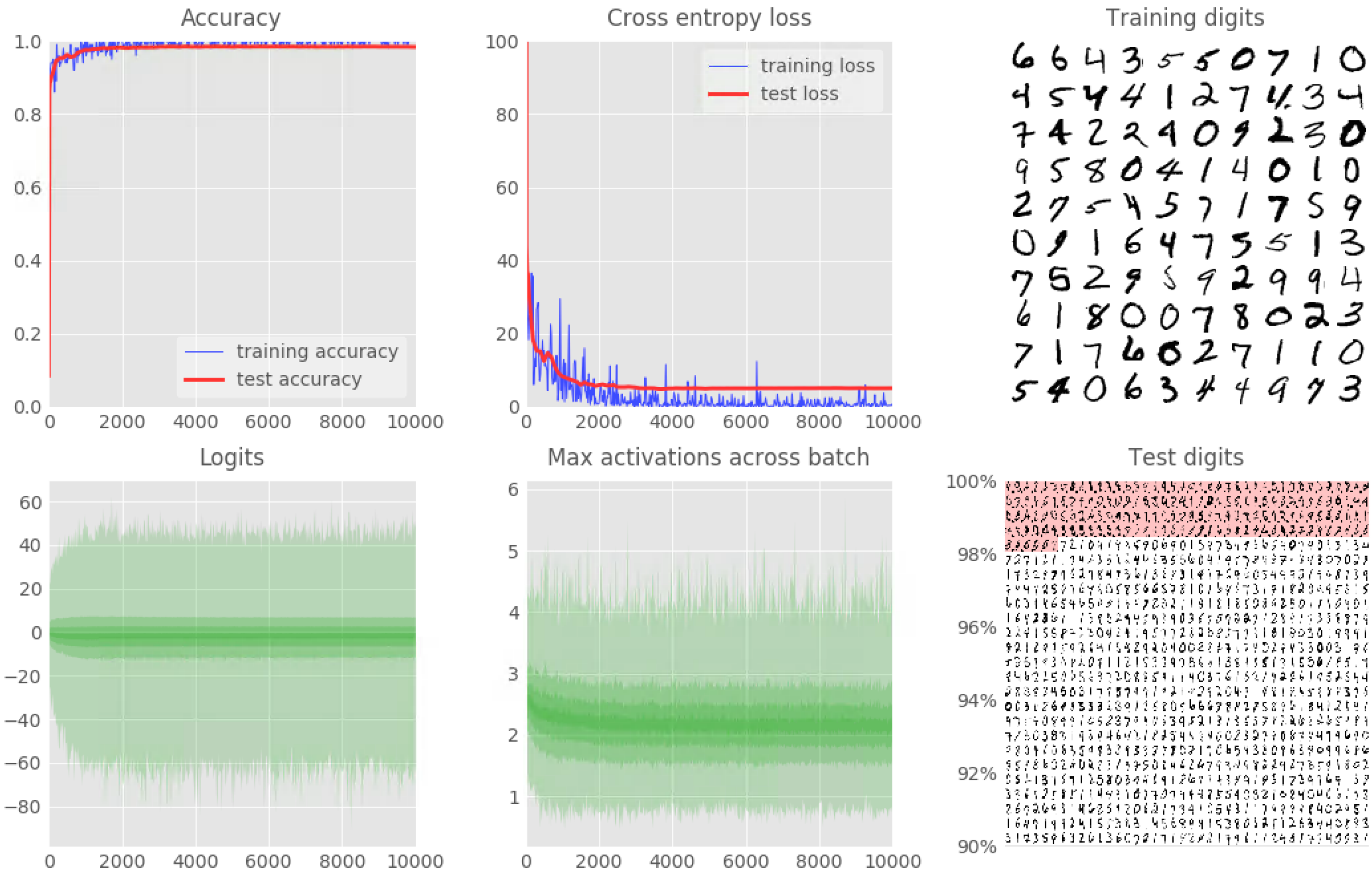
TensorFlow MNIST Tutorial



larger convolutional network

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

TensorFlow MNIST Tutorial



Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/

TensorFlow MNIST Tutorial

```
python3 mnist_1.0_softmax.py
```

```
python mnist_1.0_softmax.py
```

```
pythonw mnist_1.0_softmax.py
```

```
python3 mnist_2.0_five_layers_sigmoid.py
```

```
python3 mnist_2.2_five_layers_relu_lrdecay_dropout.py
```

```
python3 mnist_3.0_convolutional.py
```

```
python3 mnist_3.1_convolutional_bigger_dropout.py
```

```
python3 mnist_4.0_batchnorm_five_layers_sigmoid.py
```

```
python3 mnist_4.1_batchnorm_five_layers_relu.py
```

```
python3 mnist_4.2_batchnorm_convolutional.py
```

Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/

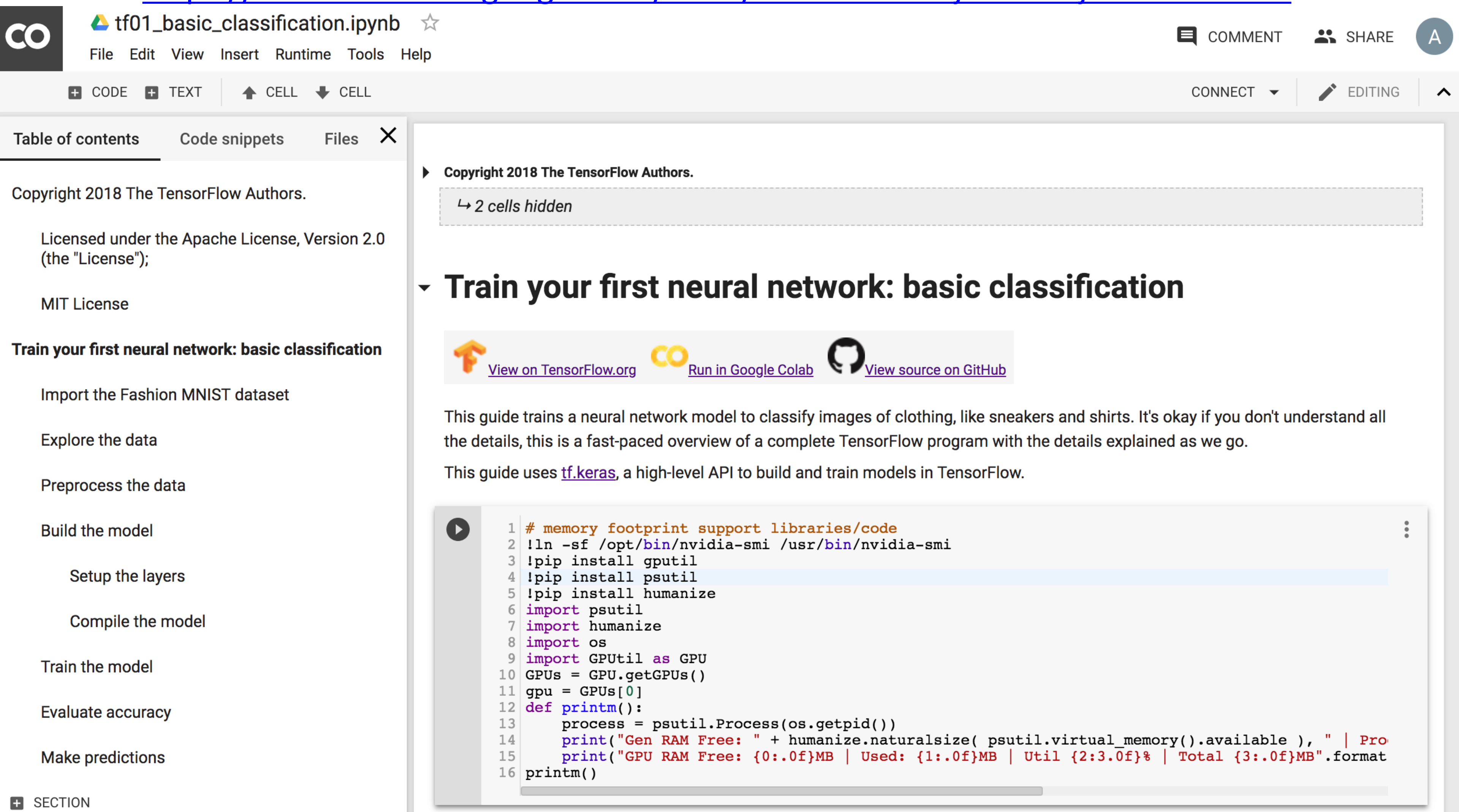
Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/

AI + VDI POS
TensorFlow Models

- M1: Basic Classification (Image Classification) (65 Seconds)
  - https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_classification.ipynb
- M2: Basic Text Classification (Text Classification) (46 Seconds)
  - https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb
- M3: Basic Regression (Predict House Prices) (43 Seconds)
  - https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_regression.ipynb
- M4: Pix2Pix Eager (Option) (7-8 Hours)
  - https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/pix2pix/pix2pix_eager.ipynb
- M5. NMT with Attention (Option) (20-30 minutes)
  - https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/nmt_with_attention/nmt_with_attention.ipynb

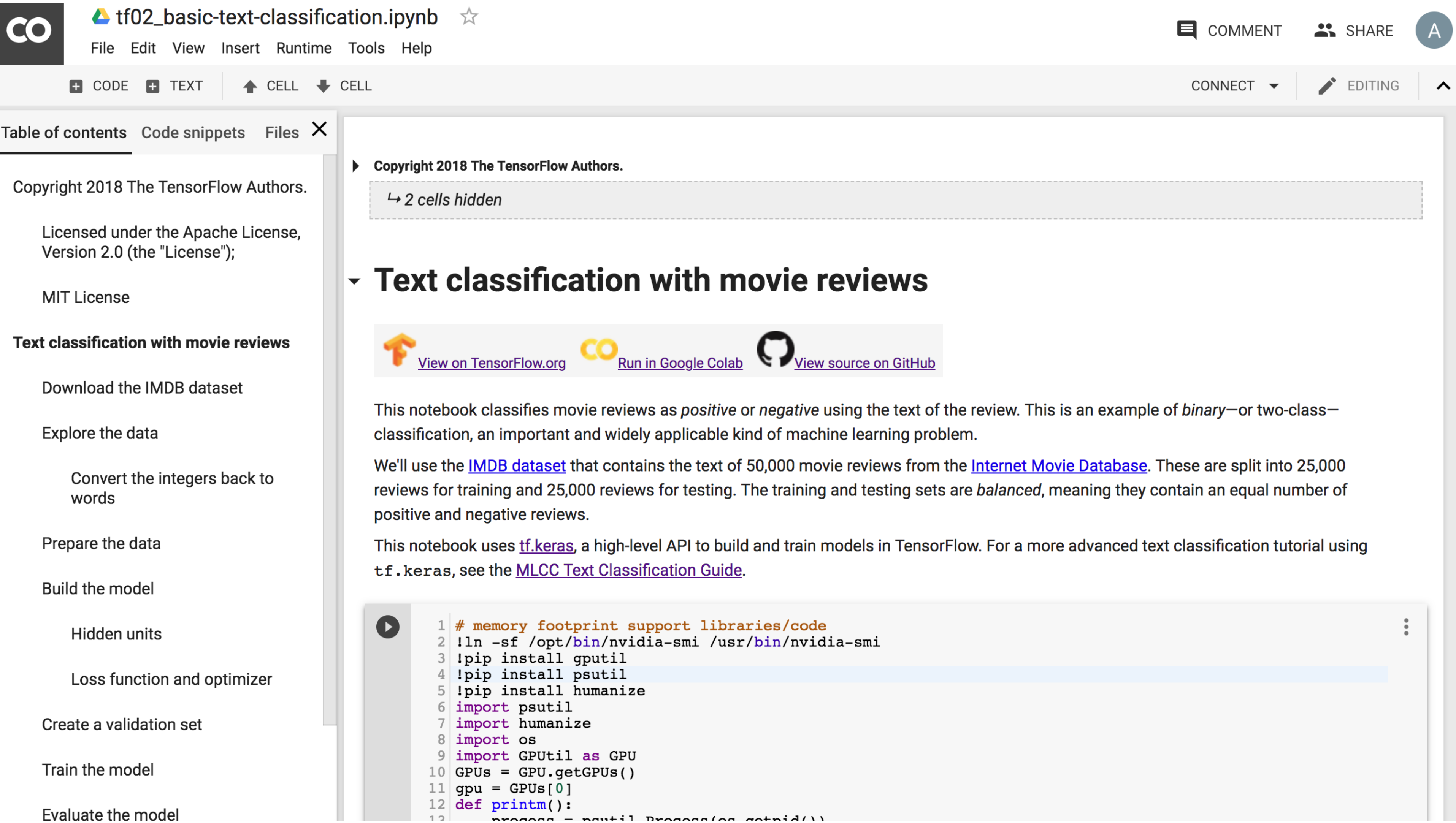
Basic Classification
Fashion MNIST Image Classification

https://colab.research.google.com/drive/19PJOJi1vn1kjcutlzNHjRSLbeVl4kd5z



Text Classification
IMDB Movie Reviews

https://colab.research.google.com/drive/1x16h1GhHsLIrLYtPCvCHaoO1W-i_gror



Source: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb

Basic Regression
Predict House Prices

https://colab.research.google.com/drive/1v4c8ZHTnRtgd2_25K_AURjR6SCVBRdlj

CO tf03_basic-regression.ipynb ☆

File Edit View Insert Runtime Tools Help

COMMENT SHARE A

CODE TEXT CELL CELL CONNECT EDITING

Table of contents Code snippets Files

SECTION

Copyright 2018 The TensorFlow Authors.

↳ *2 cells hidden*

Predict house prices: regression

View on TensorFlow.org Run in Google Colab View source on GitHub

In a *regression* problem, we aim to predict the output of a continuous value, like a price or a probability. Contrast this with a *classification* problem, where we aim to predict a discrete label (for example, where a picture contains an apple or an orange).

This notebook builds a model to predict the median price of homes in a Boston suburb during the mid-1970s. To do this, we'll provide the model with some data points about the suburb, such as the crime rate and the local property tax rate.

This example uses the `tf.keras` API, see this guide for details.

```
# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printm():
    process = psutil.Process(os.getpid())
    print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Proc size: "
    print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | Total {3:.0f}MB".format(gpu.memo
```

AI+VDI POC
ISAC+TKU Test

- AI+VDI POC Folder (3+1 ipynb) (v3.0.20181120)
  - https://drive.google.com/open?id=1qHOemktbEmUz-ot8eFxIKbGwJvXlrjtc
- run3models.ipynb
  - https://colab.research.google.com/drive/1HQ1GrIqQUUPCct7_AVgoMwMrh0UqMm0f

Summary

- Convolutional Neural Networks (CNN)
- TensorFlow Image Recognition

References

- Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
- Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 1 (Google Cloud Next '17), https://www.youtube.com/watch?v=u4alGiomYP4
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 2 (Google Cloud Next '17), https://www.youtube.com/watch?v=fTUwdXUFfI8
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, https://goo.gl/pHeXe7, https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist
- Deep Learning Basics: Neural Networks Demystified, https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
- Deep Learning SIMPLIFIED, https://www.youtube.com/playlist?list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu
- 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk
- 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=IHZwWFHWa-w
- 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U
- TensorFlow: https://www.tensorflow.org/
- Keras: http://keras.io/
- Udacity, Deep Learning, https://www.youtube.com/playlist?list=PLAwxTw4SYaPn_OWPFT9ulXLuQrImzHfOV
- http://p.migdal.pl/2017/04/30/teaching-deep-learning.html
- https://github.com/leriomaggio/deep-learning-keras-tensorflow