



Big Data Mining

Convolutional Neural Networks (CNN)

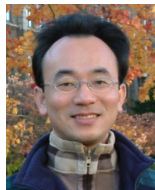
1071BDM11

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)



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)



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

Artificial Intelligence (AI)

Machine Learning (ML)

Supervised
Learning

Unsupervised
Learning

Deep Learning (DL)

CNN

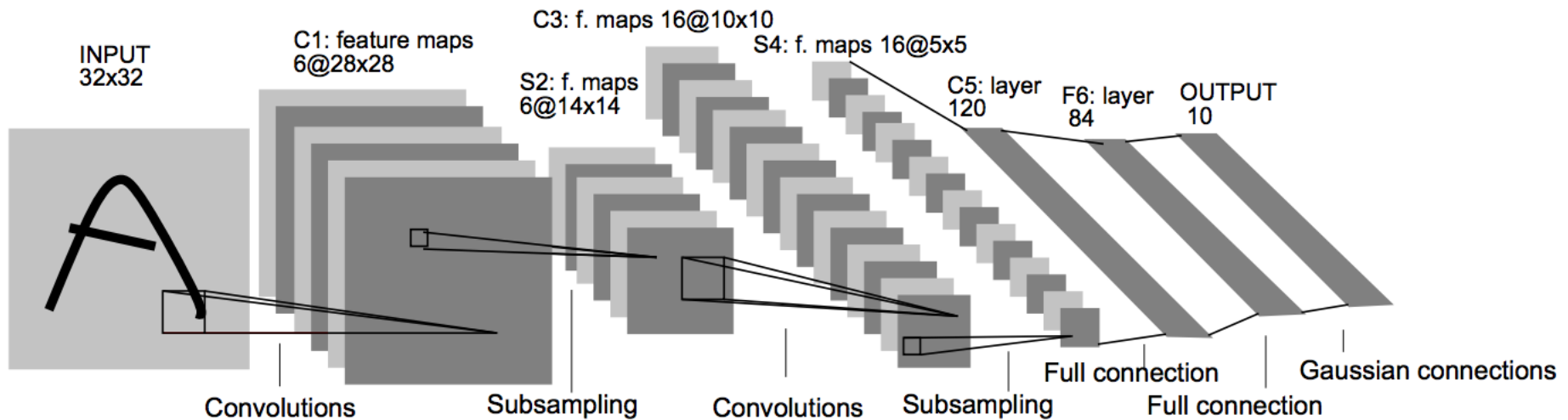
RNN LSTM GRU

GAN

Semi-supervised
Learning

Reinforcement
Learning

Convolutional Neural Networks (CNN)



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

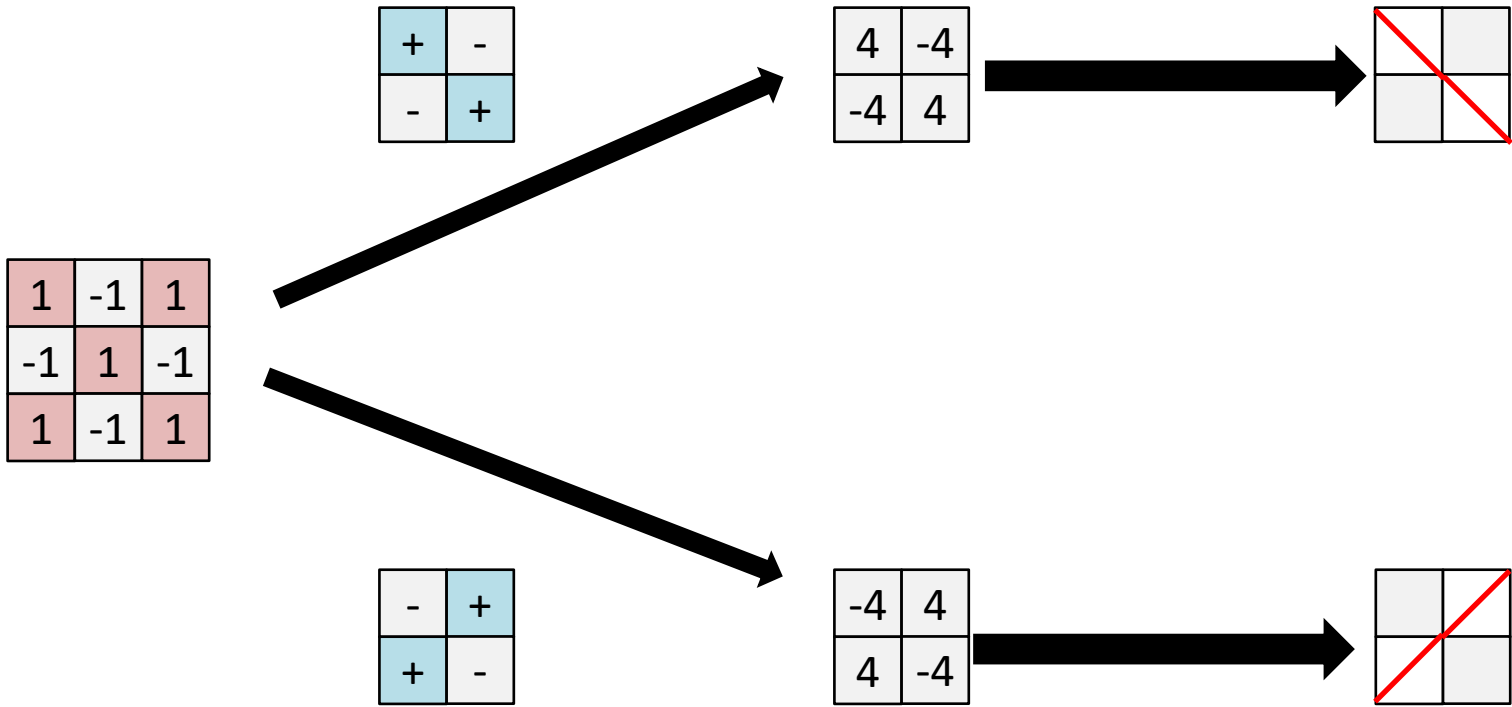
Source: <http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>

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

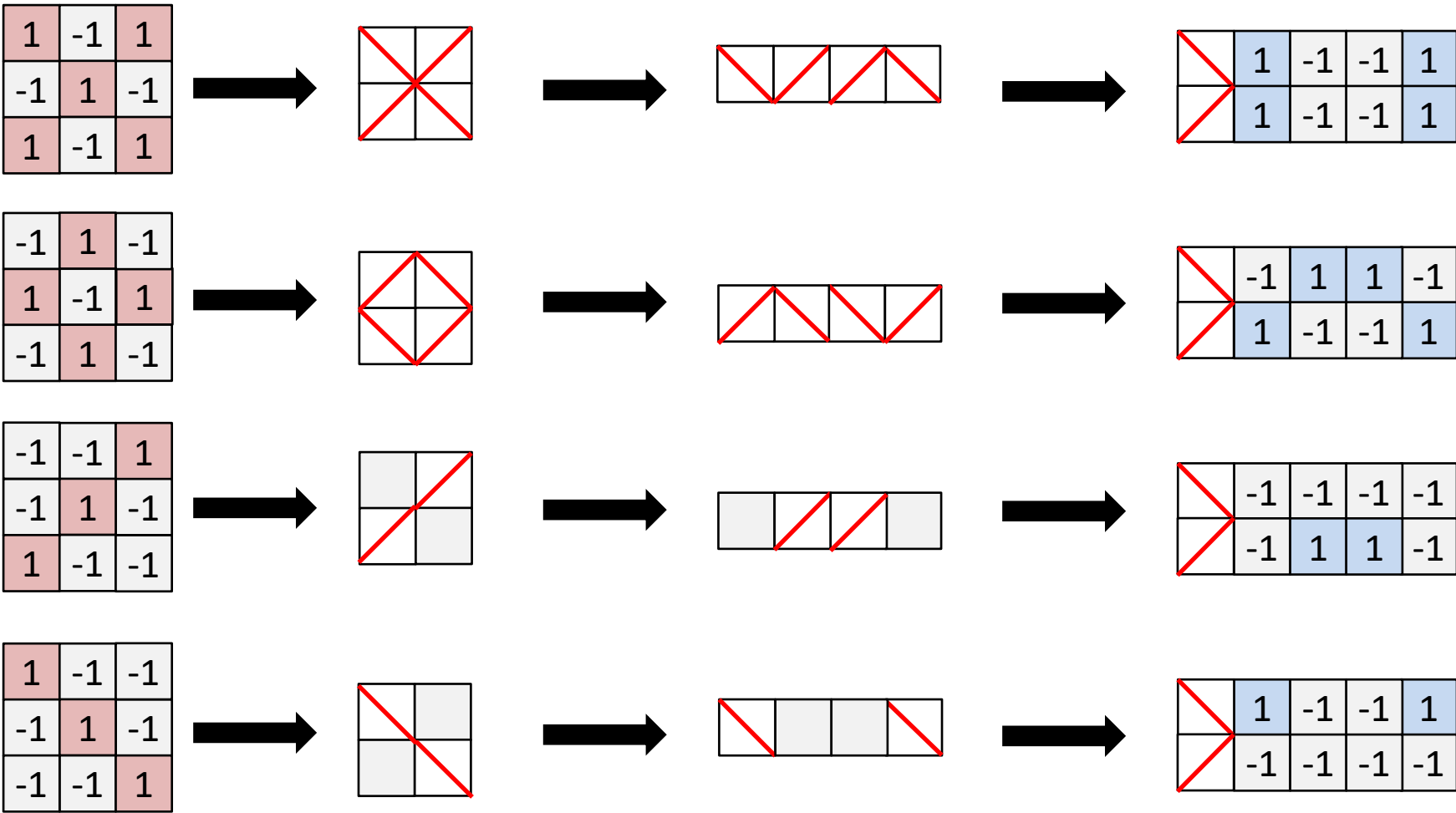


Convolution Layer

Pooling Layer

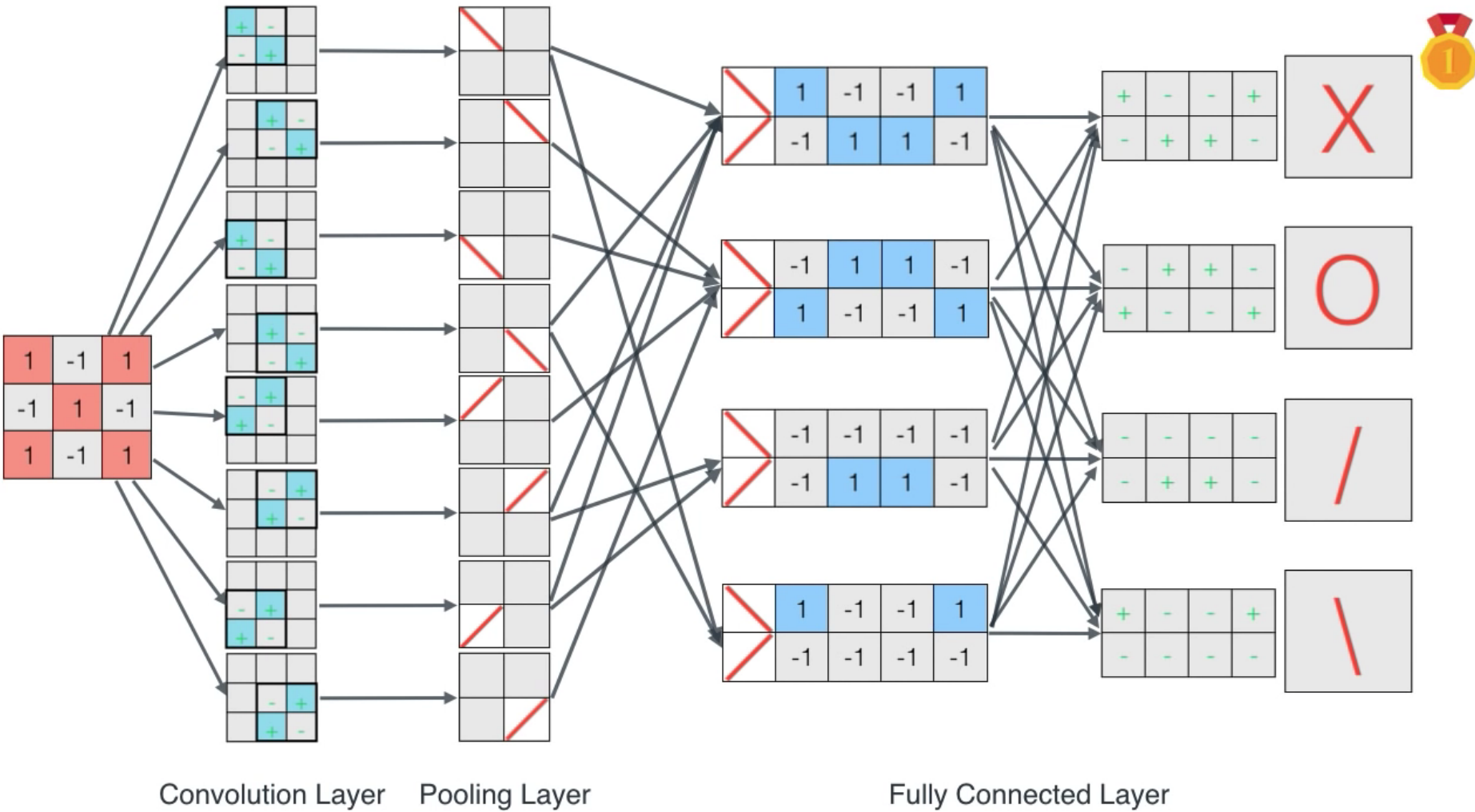
Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, <https://www.youtube.com/watch?v=2-OI7ZB0MmU>

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-OI7ZB0MmU>

A friendly introduction to Convolutional Neural Networks and Image Recognition

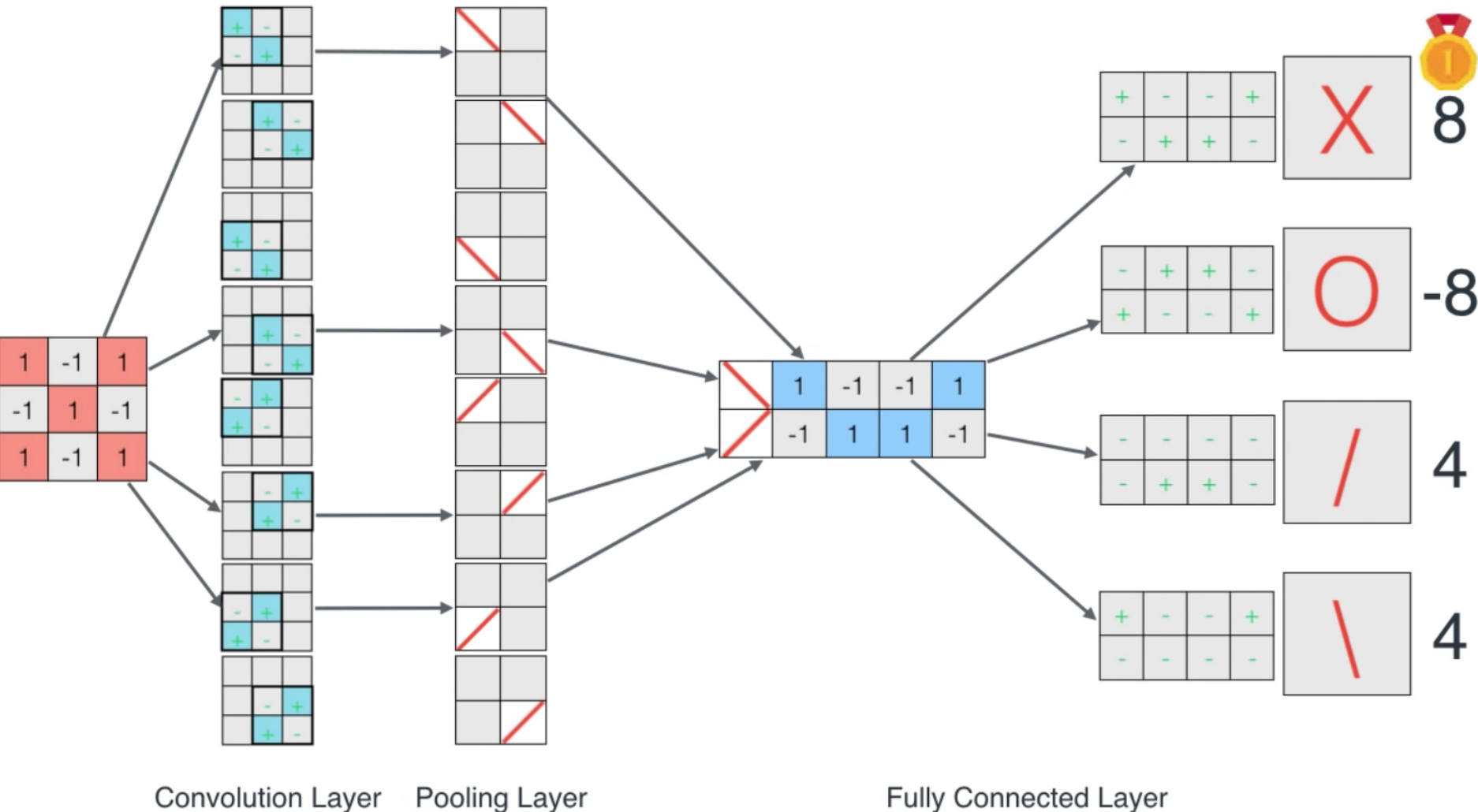


Convolution Layer Pooling Layer

Fully Connected Layer

Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, <https://www.youtube.com/watch?v=2-OI7ZB0MmU>

A friendly introduction to Convolutional Neural Networks and Image Recognition



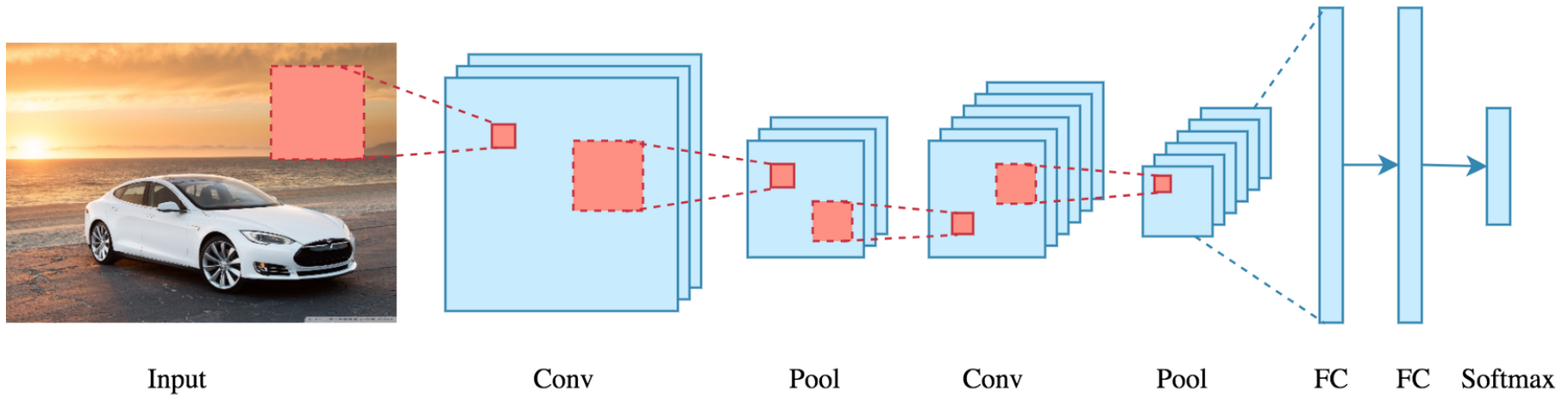
Convolution Layer

Pooling Layer

Fully Connected Layer

Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, <https://www.youtube.com/watch?v=2-OI7ZB0MmU>

CNN Architecture



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

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

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

CNN Convolution Layer

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

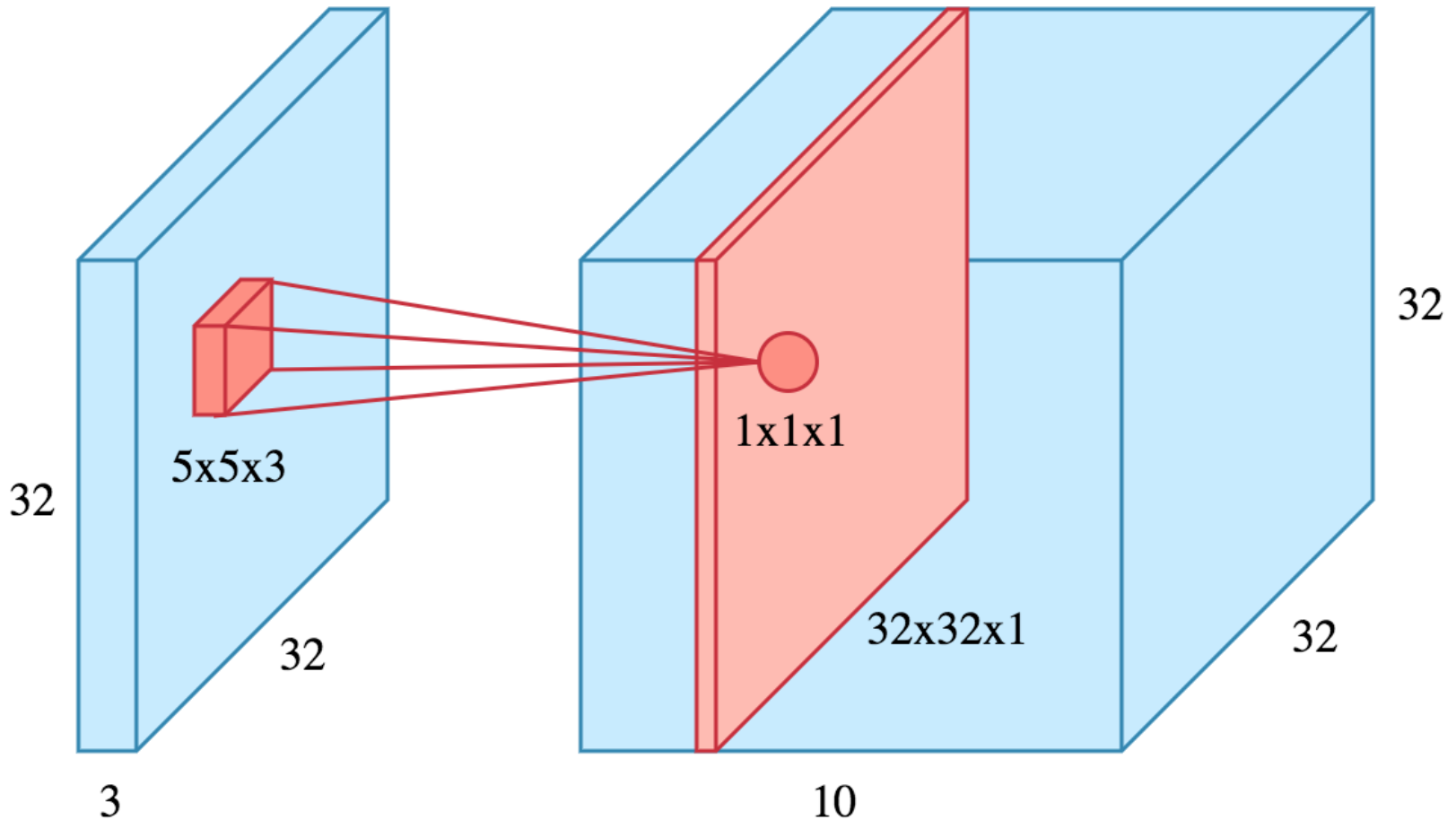
1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

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

CNN Convolution Layer

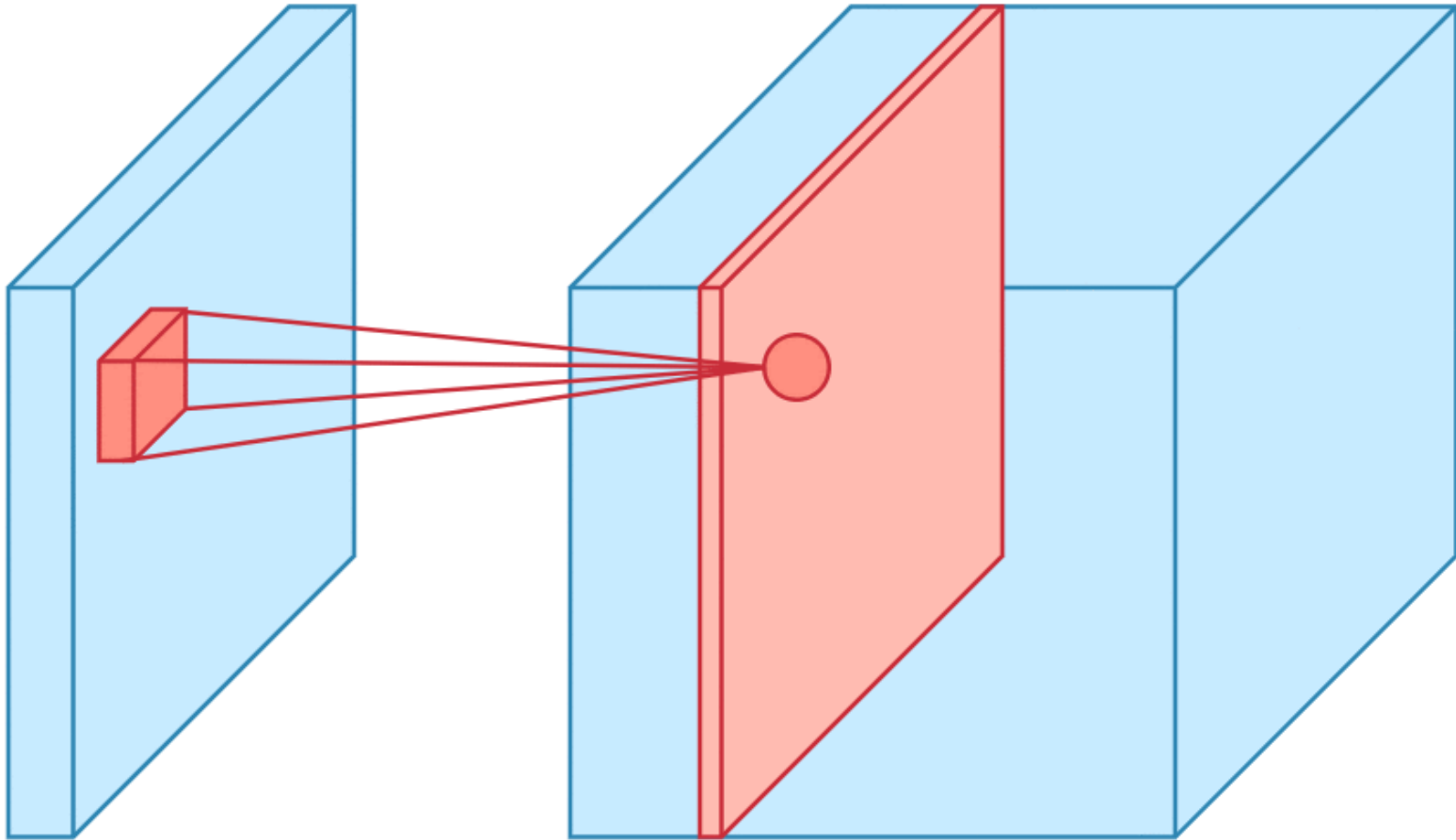
10 different filters 10 feature maps of size 32x32x1



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

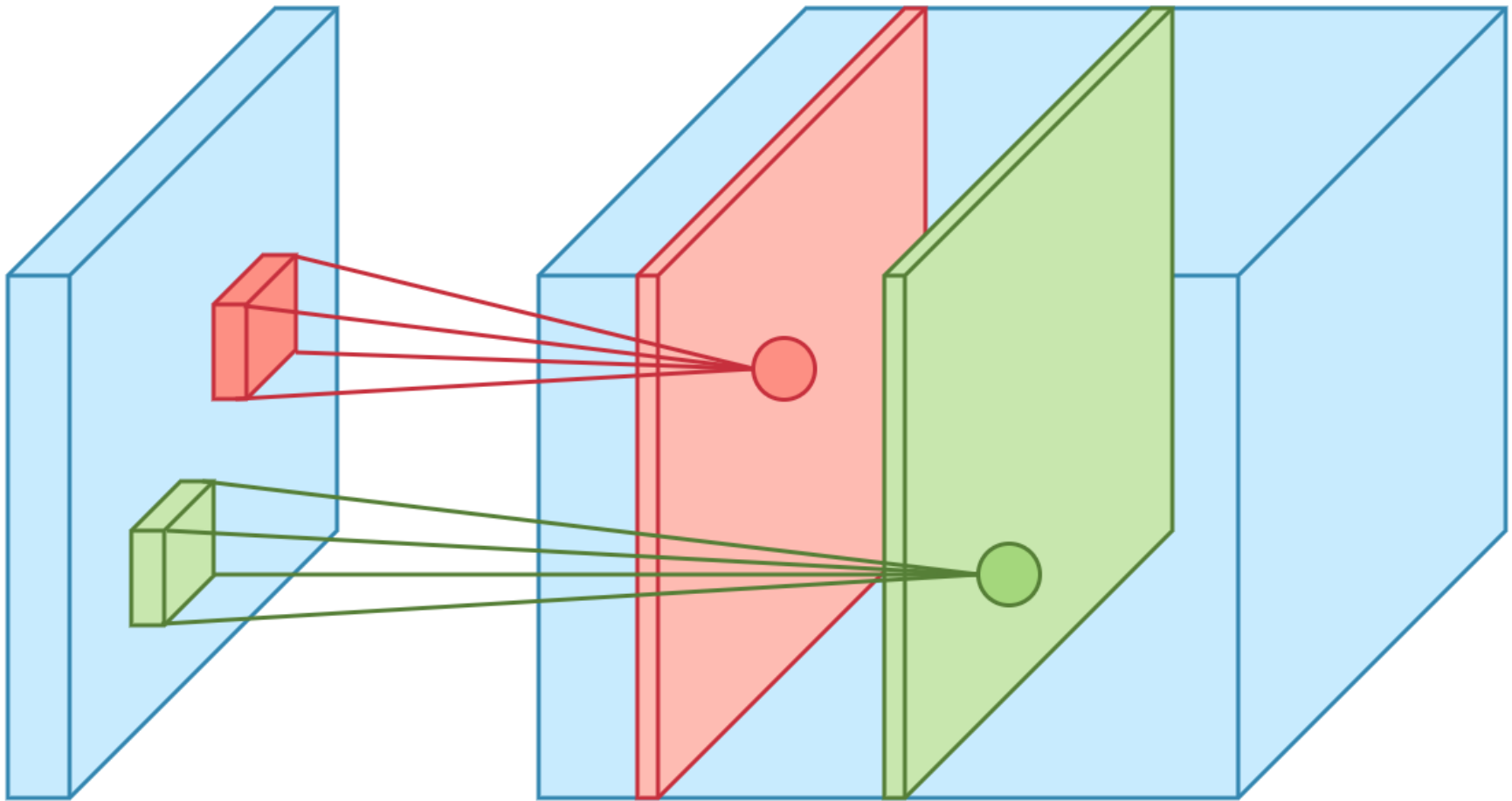
CNN Convolution Layer

Sliding operation at 4 locations



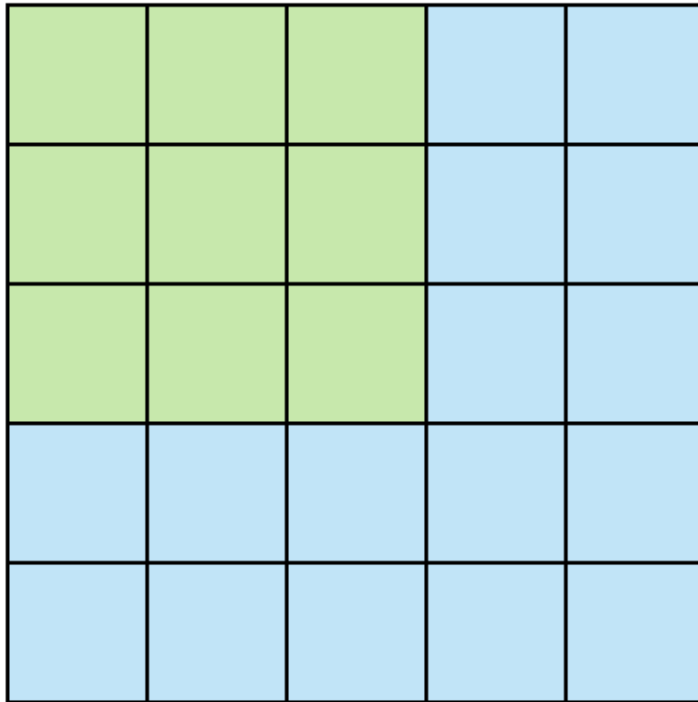
CNN Convolution Layer

two feature maps

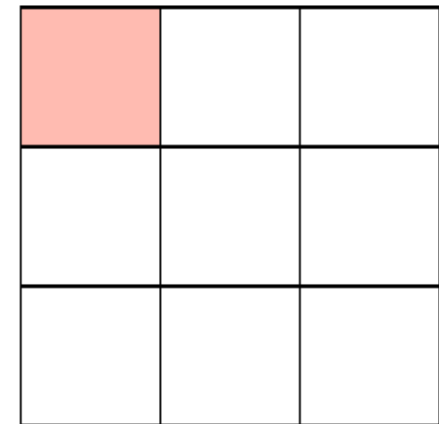


CNN Convolution Layer

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



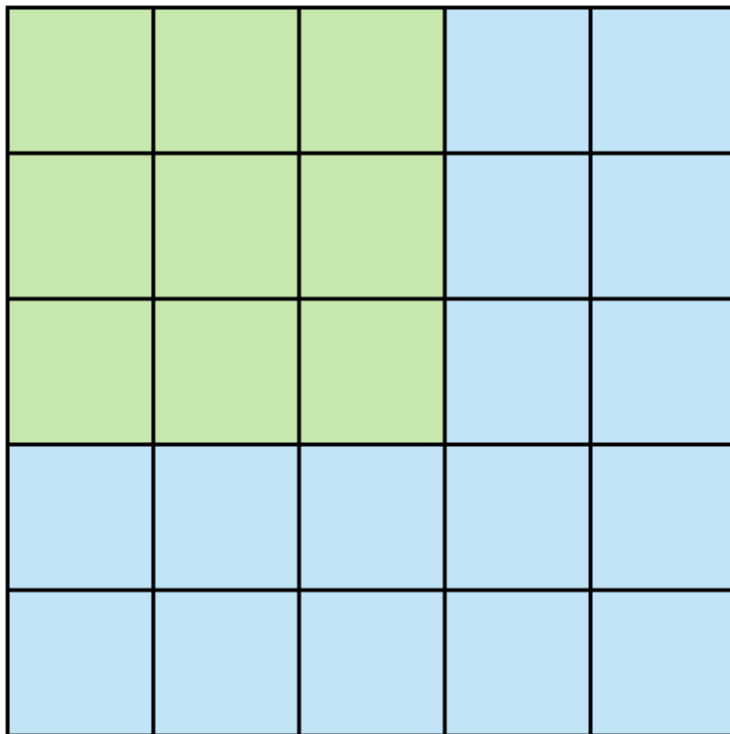
Stride 1



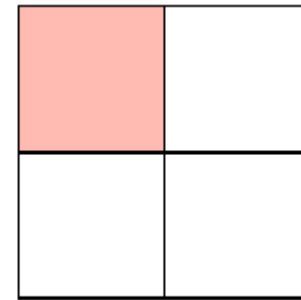
Feature Map

CNN Convolution Layer

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



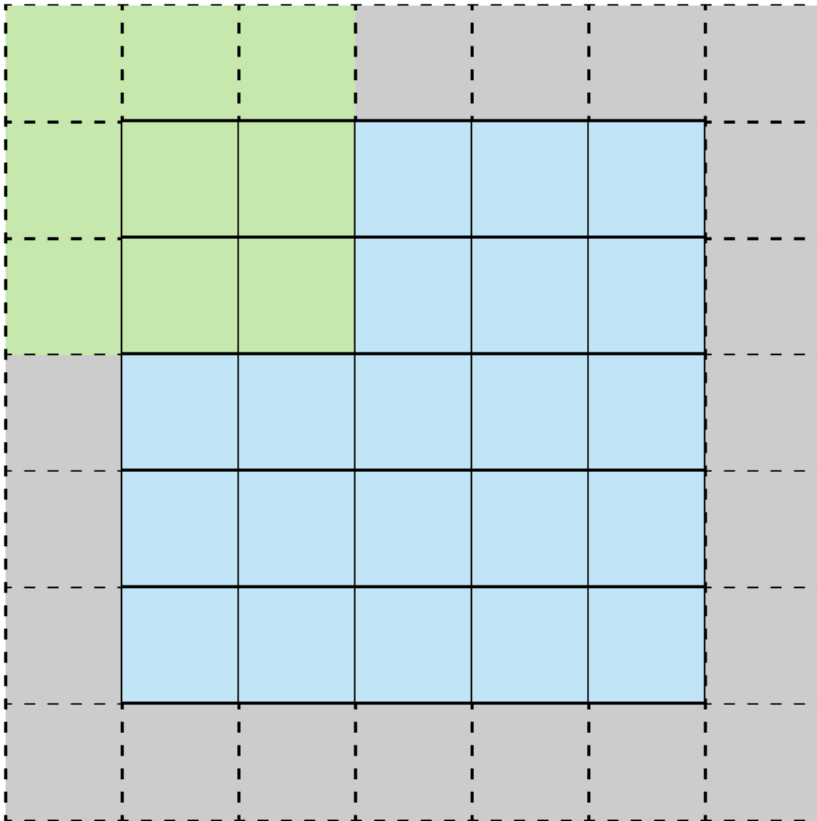
Stride 2



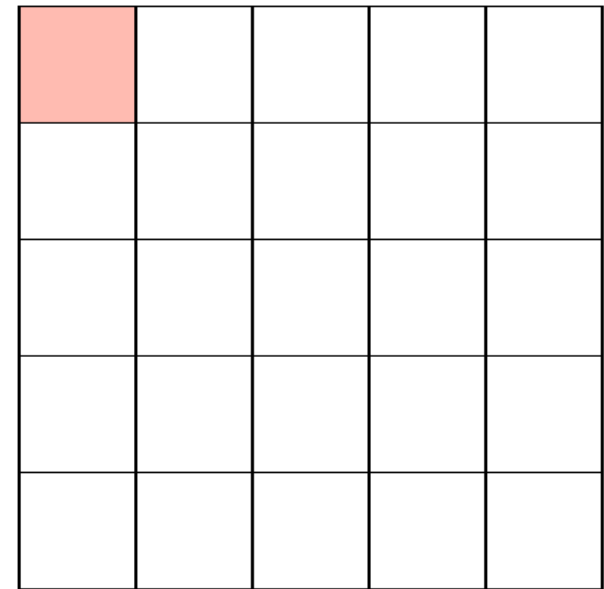
Feature Map

CNN Convolution Layer

Stride 1 with Padding



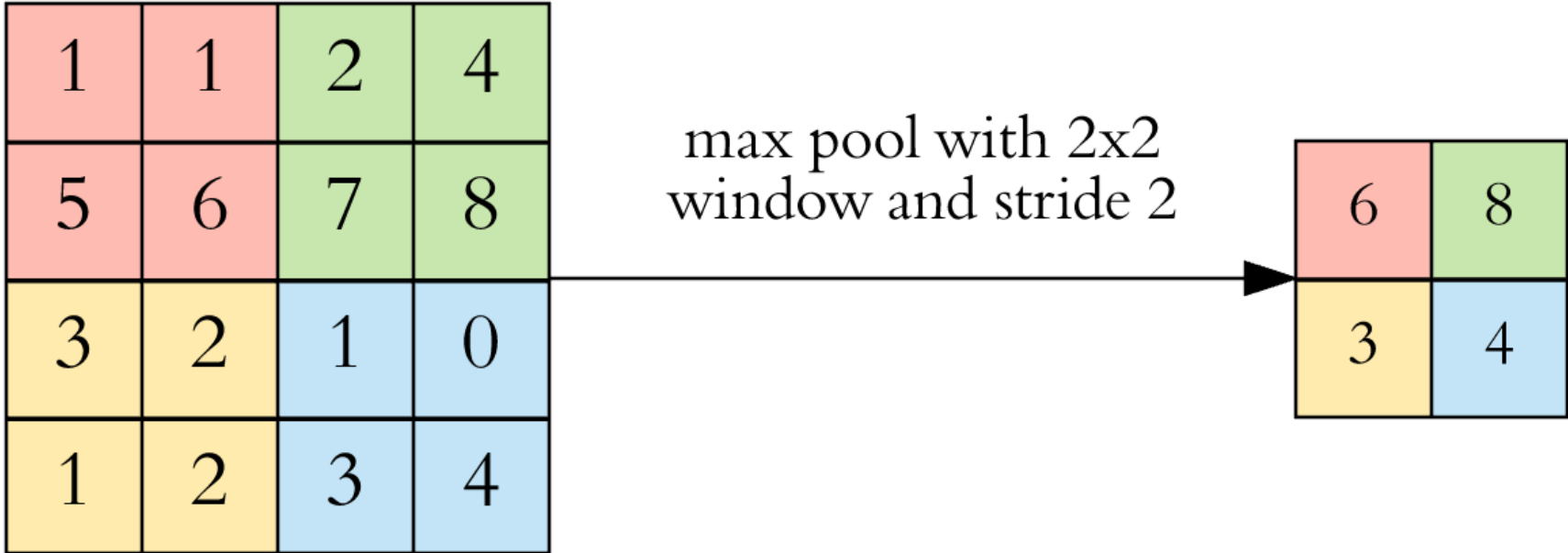
Stride 1 with Padding



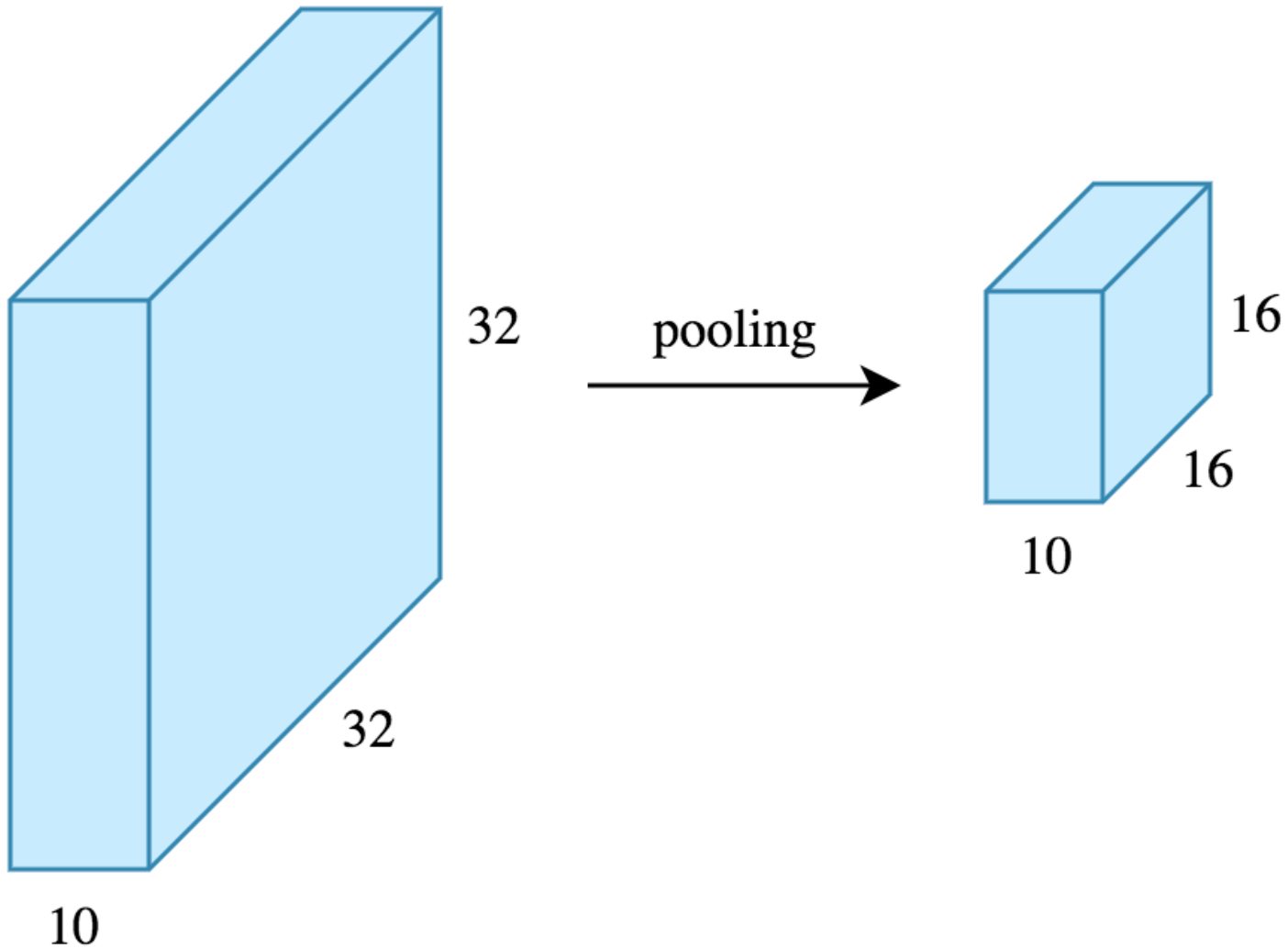
Feature Map

CNN Pooling Layer

Max Pooling

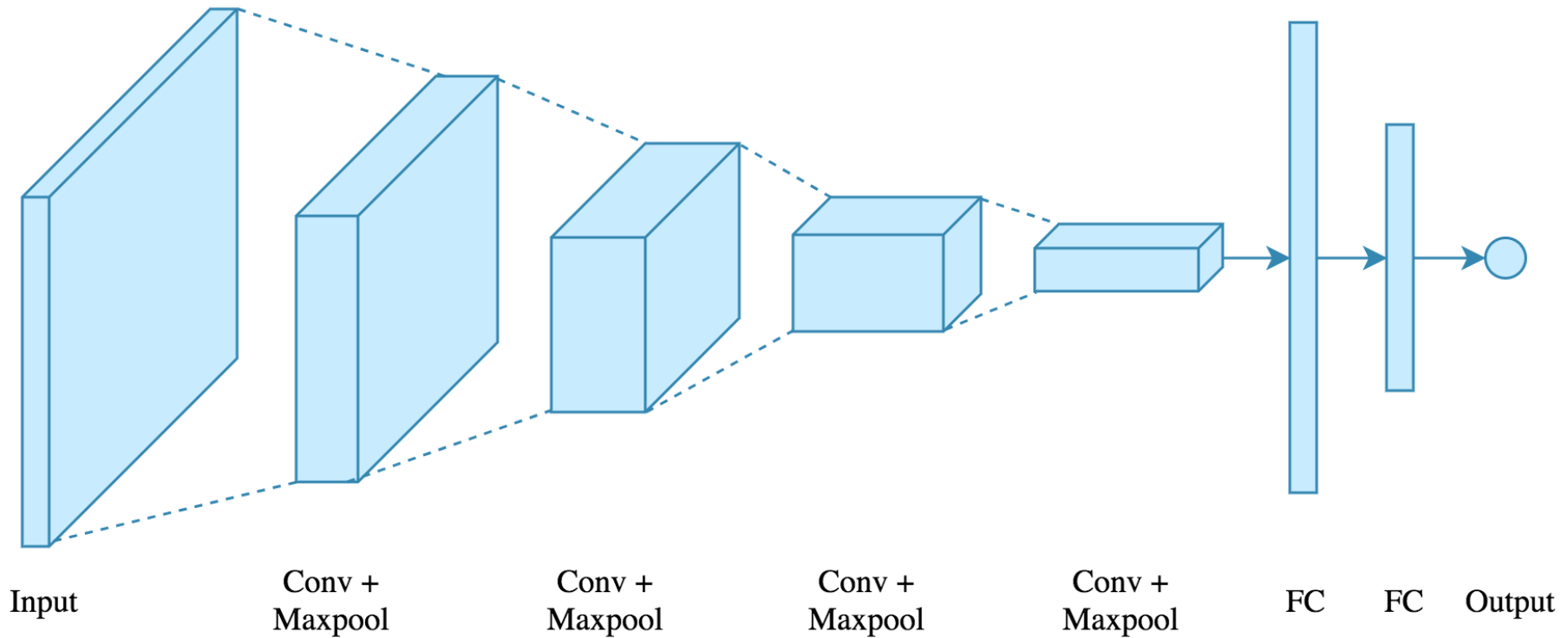


CNN Pooling Layer



CNN Architecture

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



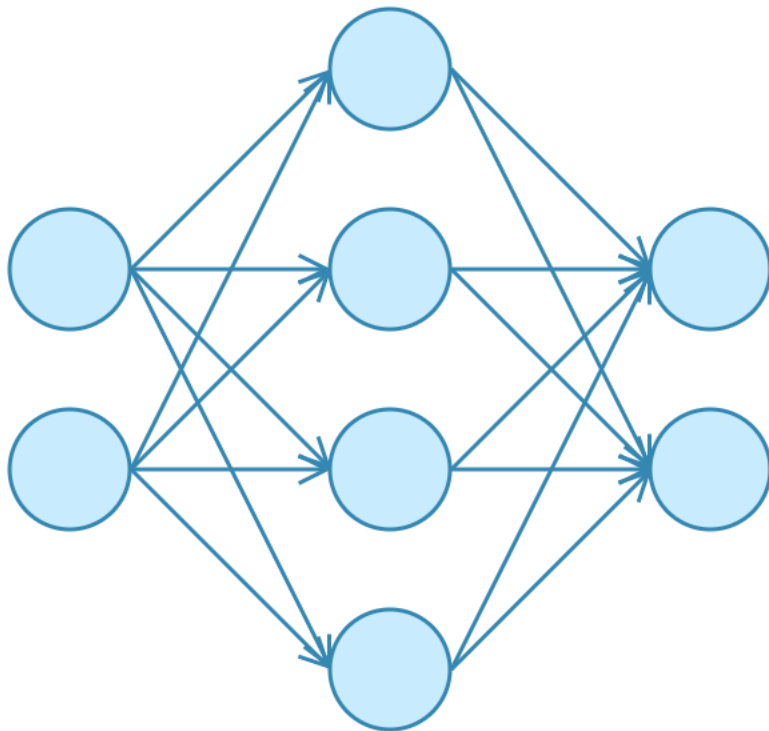
CNN Architecture

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

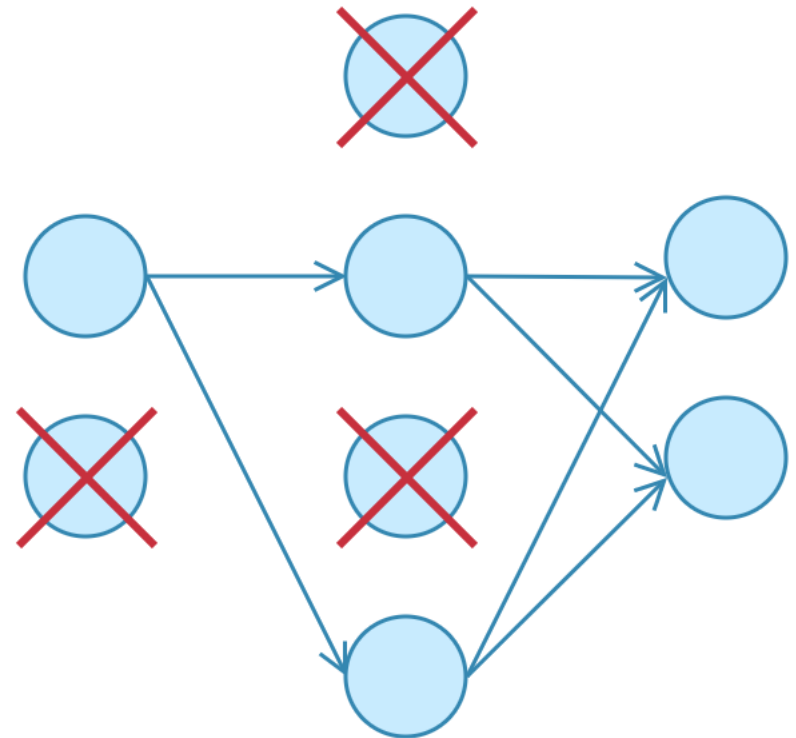
<https://gist.github.com/ardendertat/0fc5515057c47e7386fe04e9334504e3>

```
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'))
```

Dropout

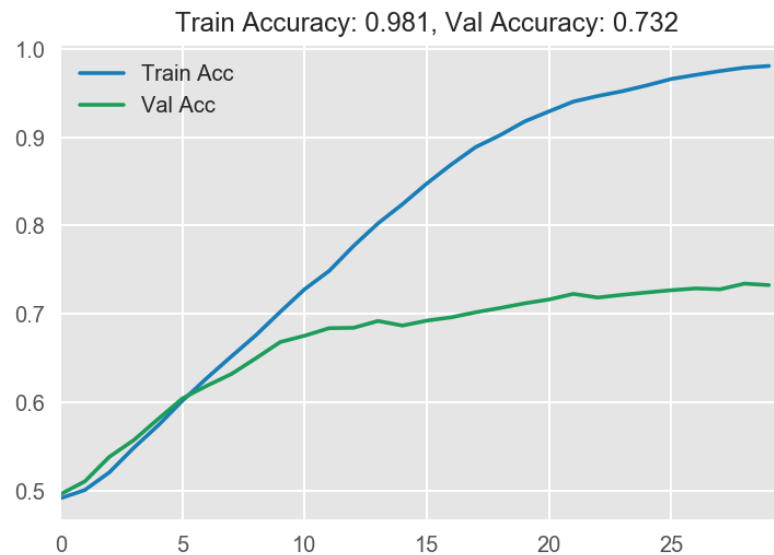
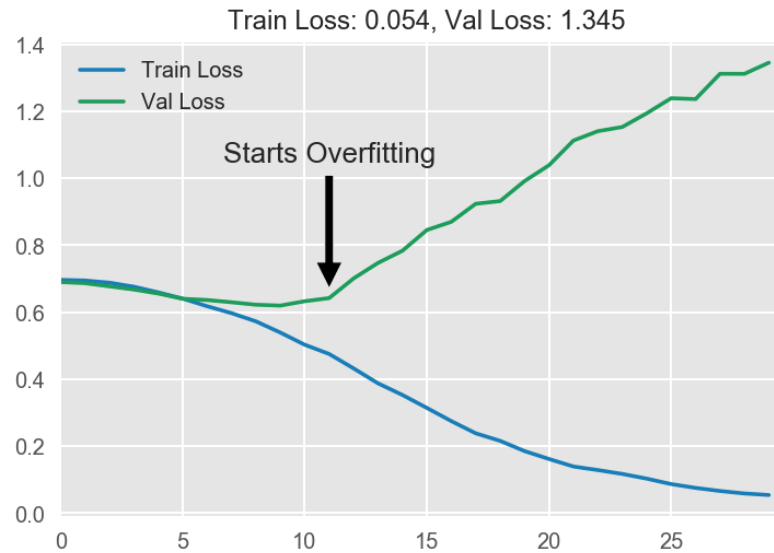


No Dropout



With Dropout

Model Performance



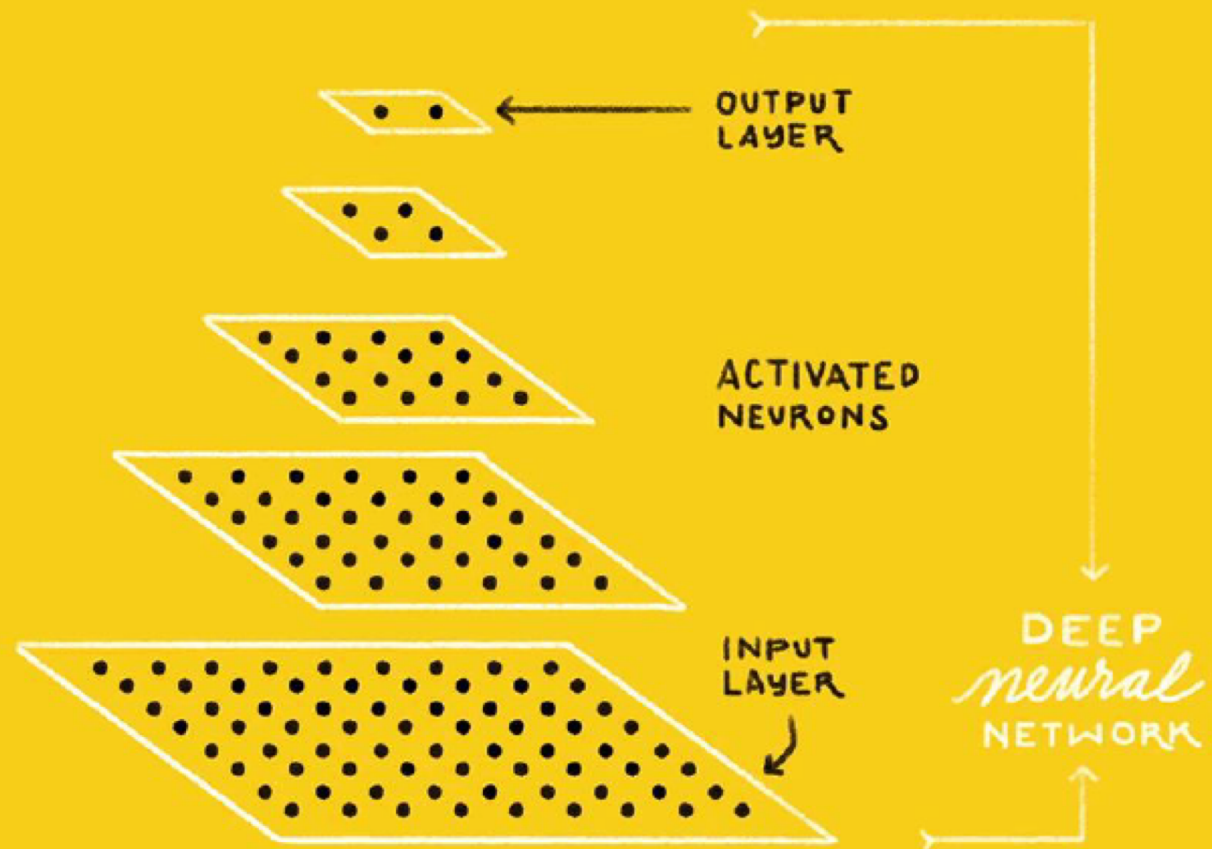
Visual Recognition

Image Classification

IS THIS A
CAT or **DOG**?

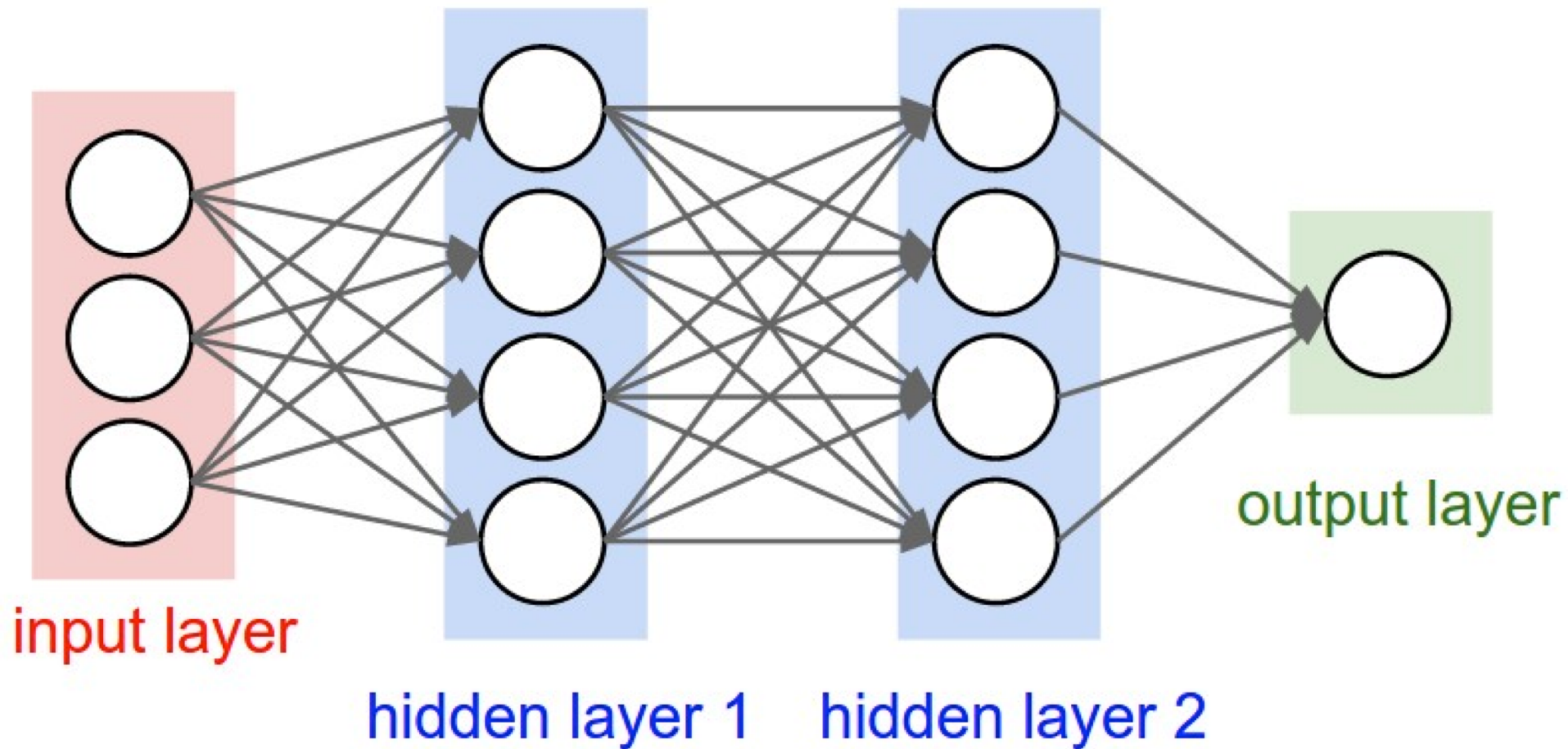


CAT **DOG**

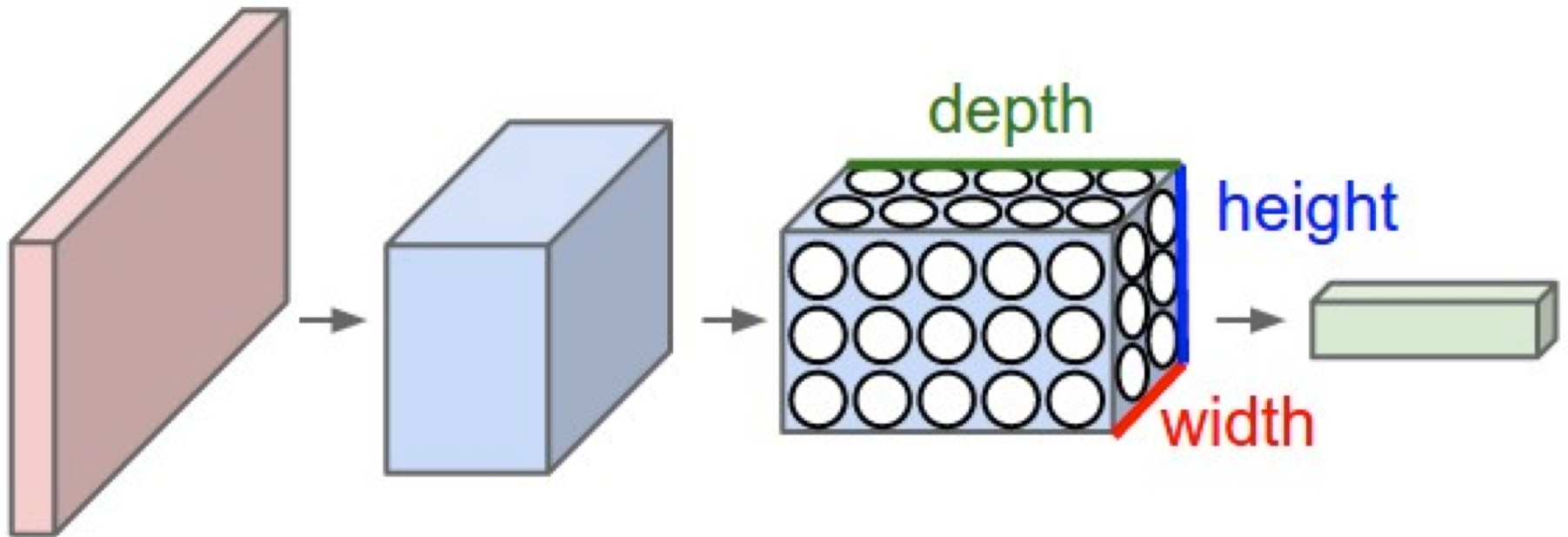


Convolutional Neural Networks (CNNs / ConvNets)

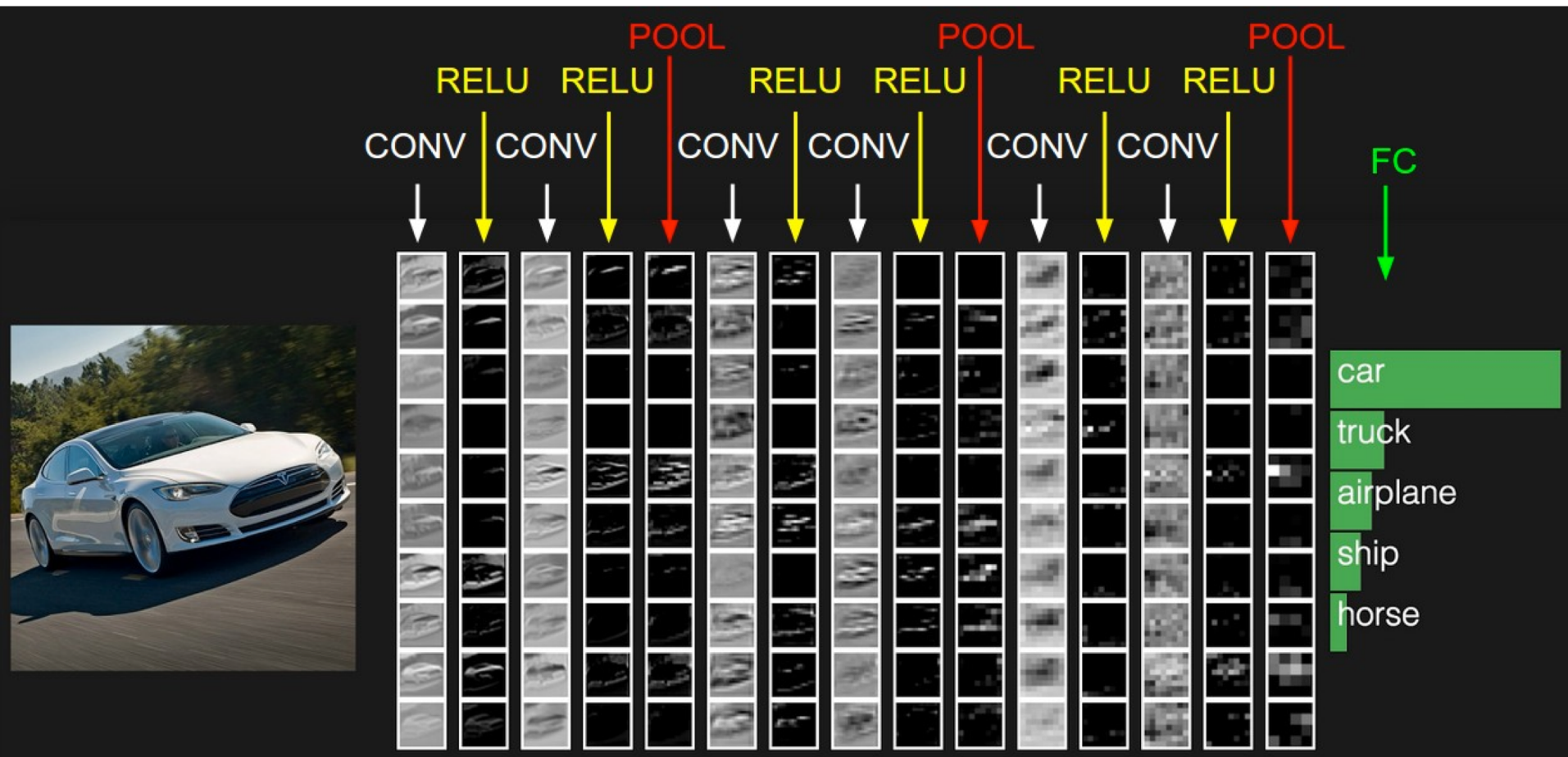
A regular 3-layer Neural Network



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



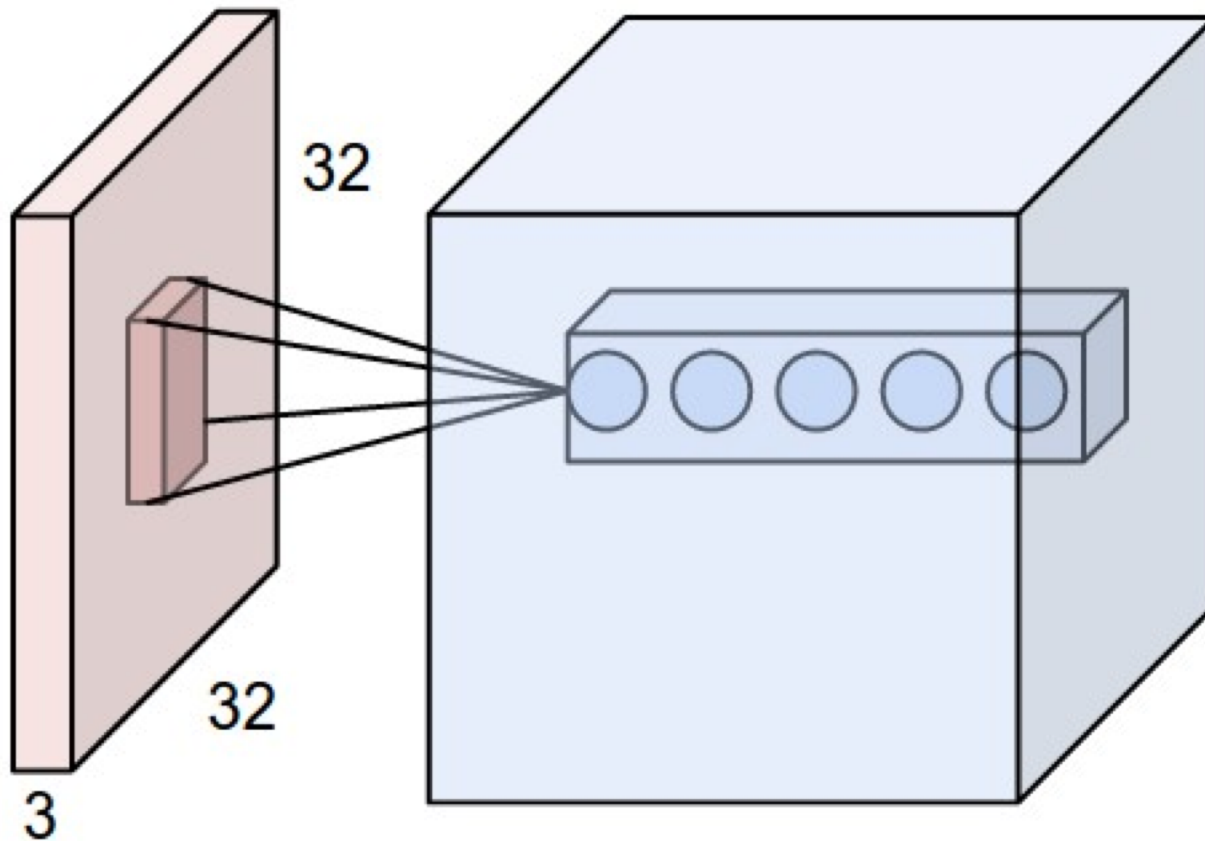
The activations of an example ConvNet architecture.



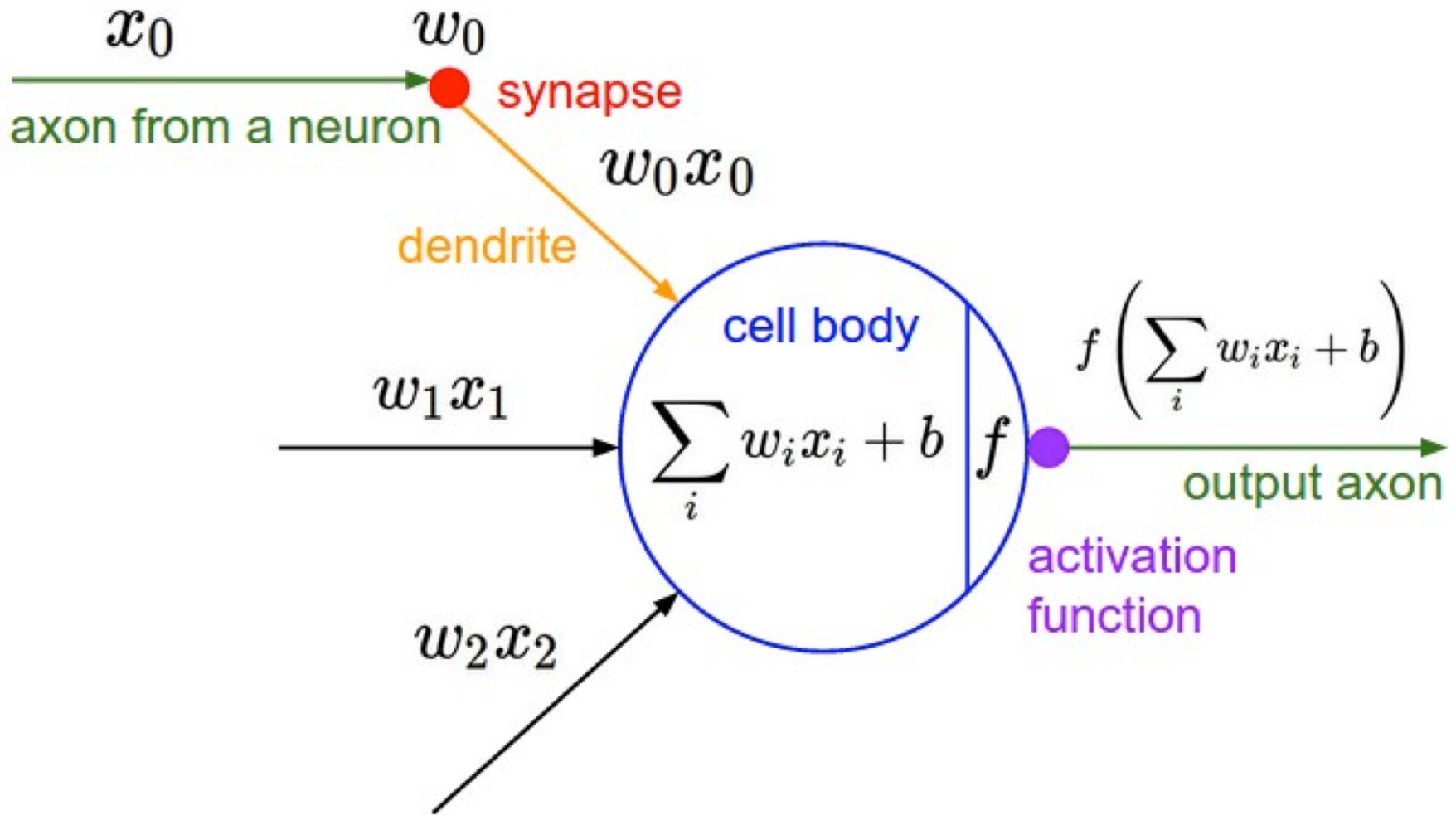
ConvNets

32x32x3 CIFAR-10 image

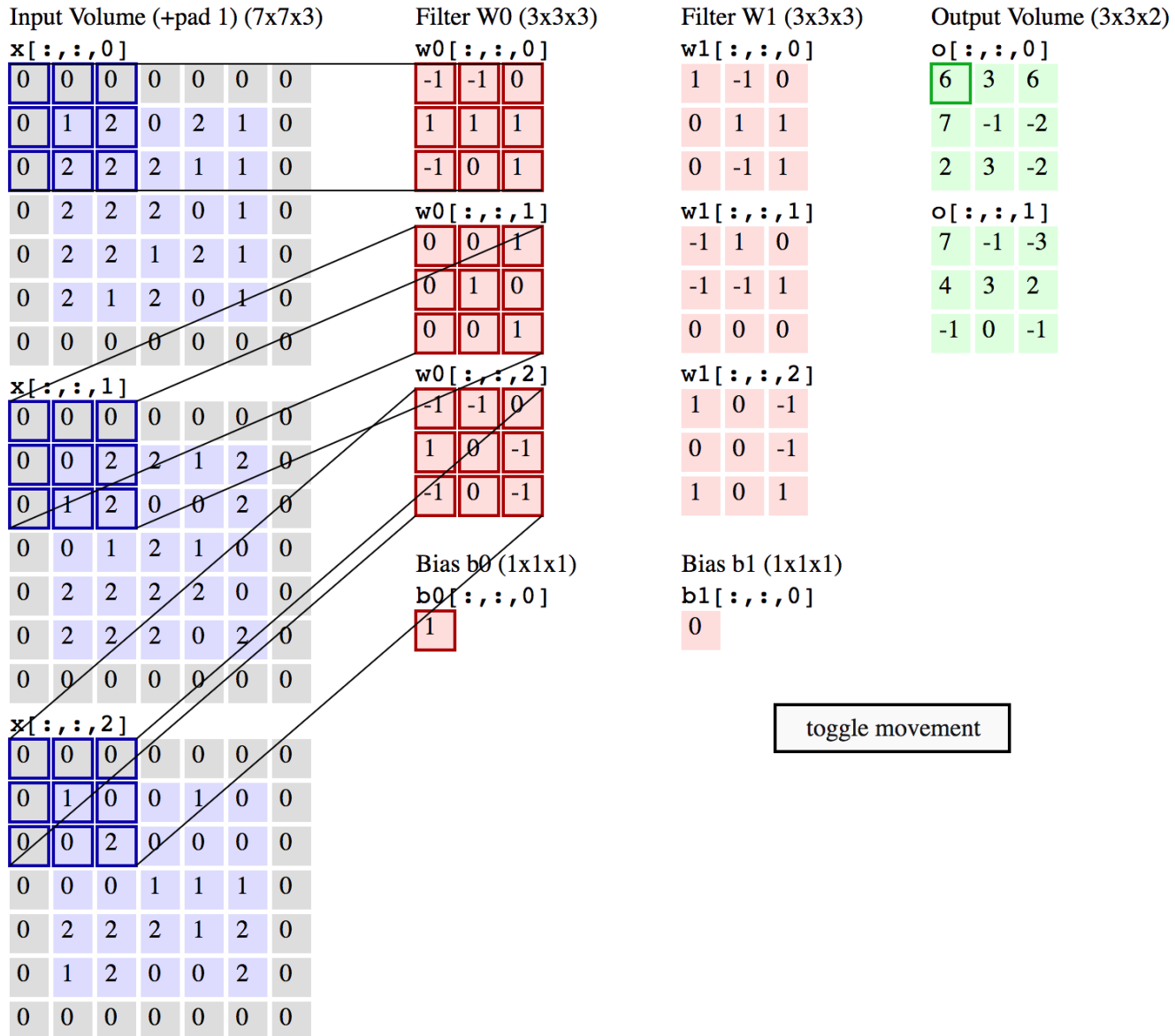
first Convolutional layer



ConvNets

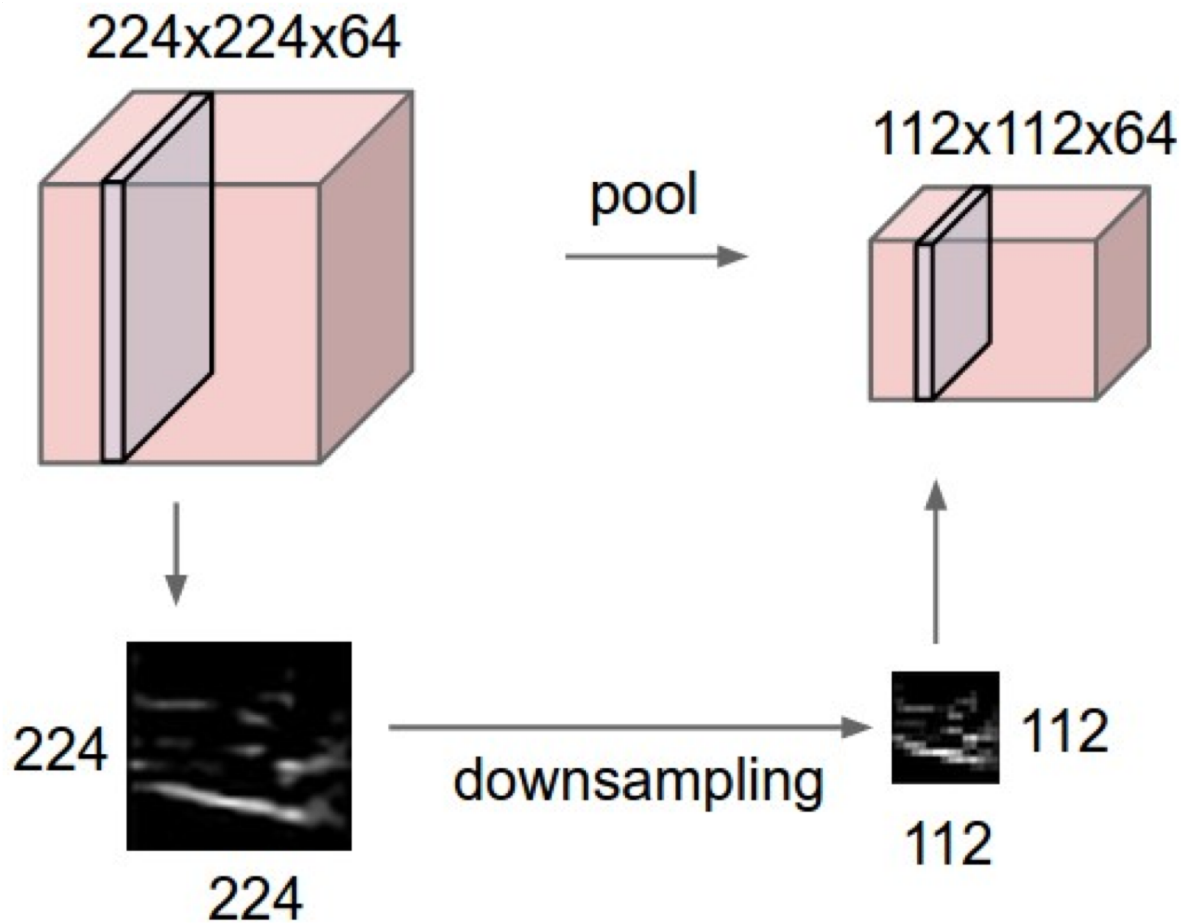


Convolution Demo



ConvNets

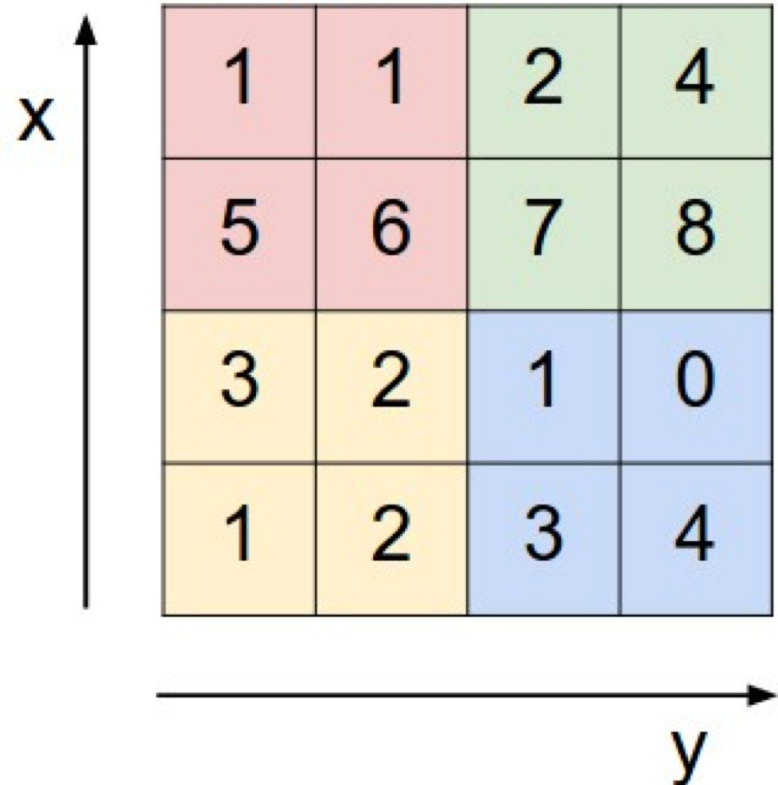
input volume of size [224x224x64]
is pooled with **filter** size 2, **stride** 2
into output volume of size [112x112x64]



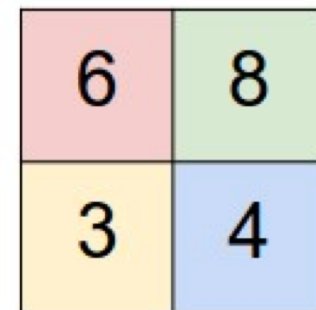
ConvNets

max pooling

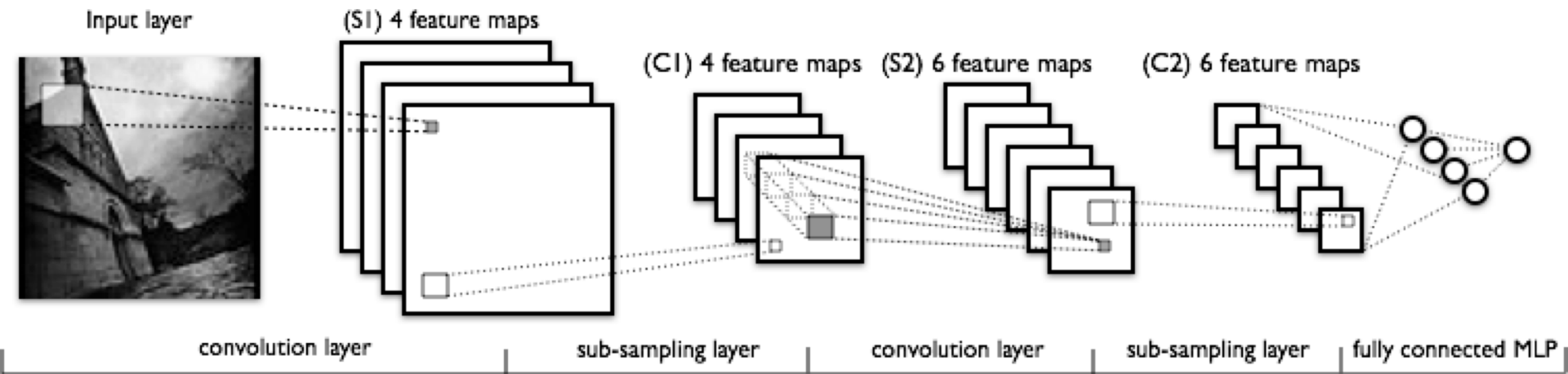
Single depth slice



max pool with 2x2 filters
and stride 2

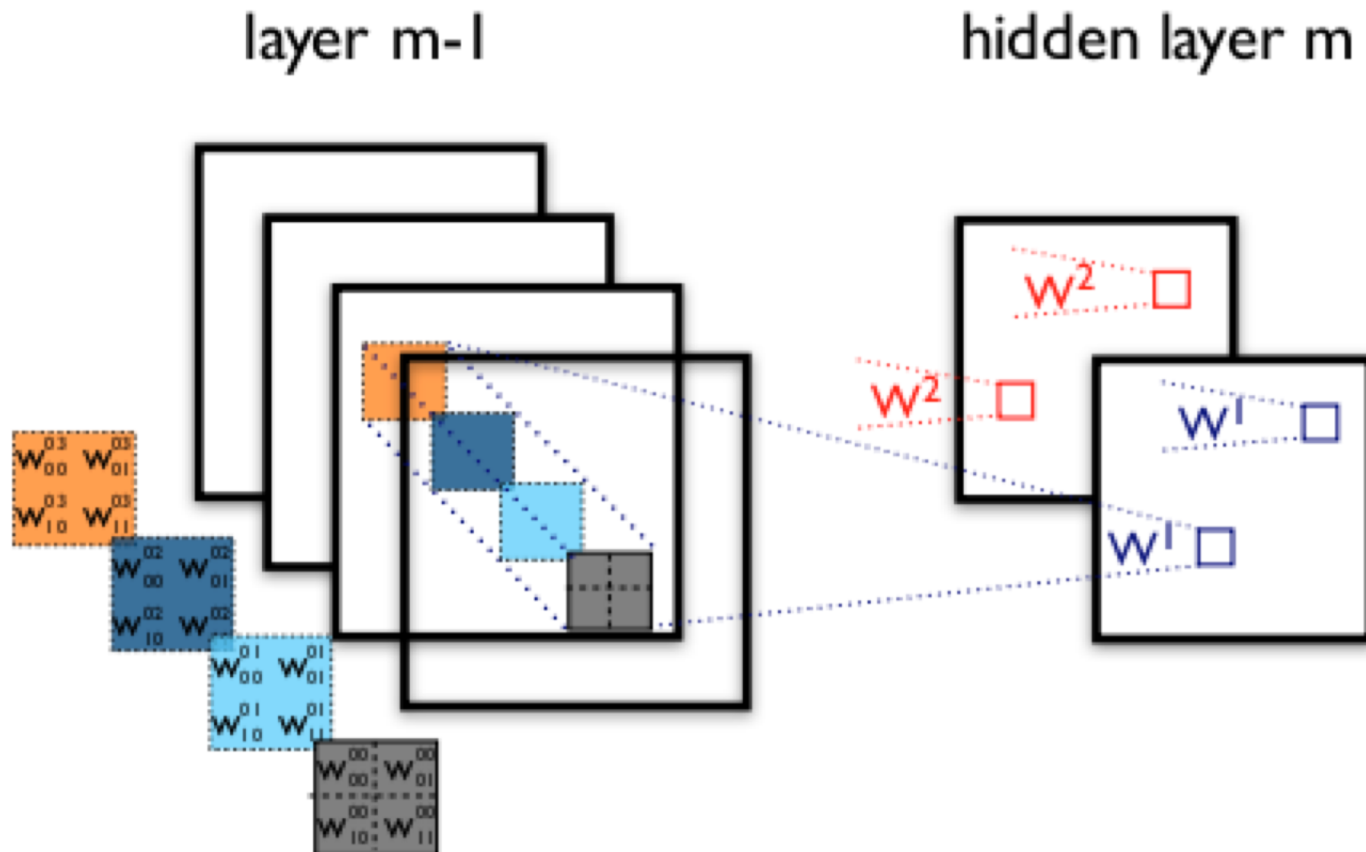


Convolutional Neural Networks (CNN) (LeNet)



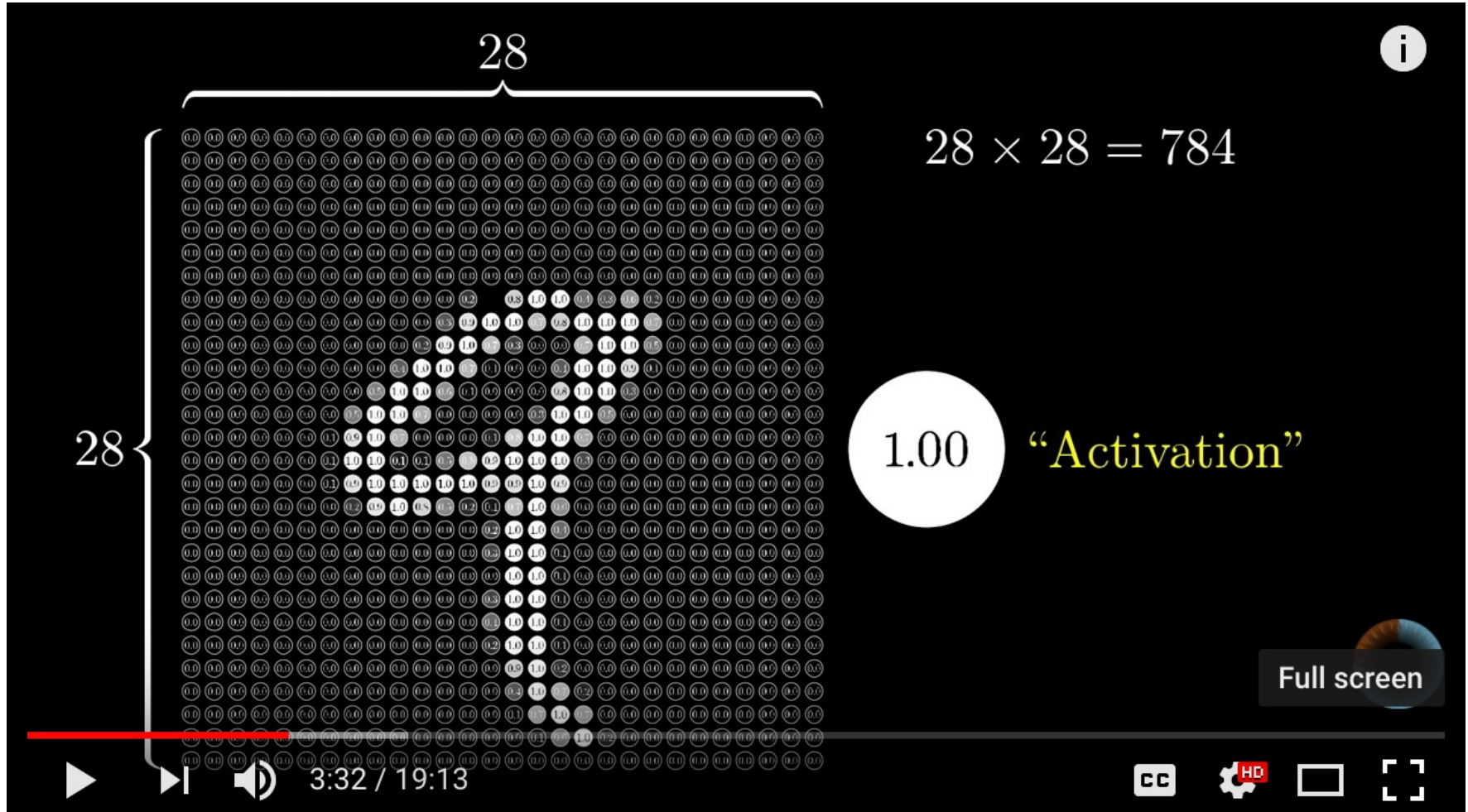
Convolutional Neural Networks (CNN) (LeNet)

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

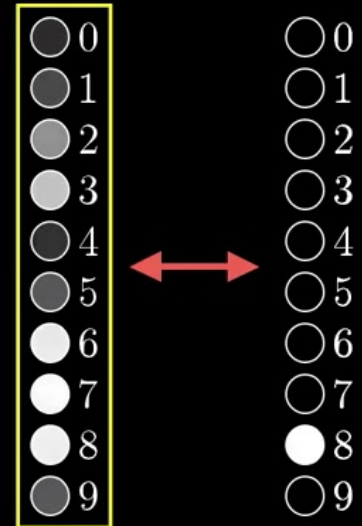
Average cost of
all training data...

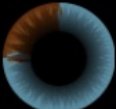
Cost of



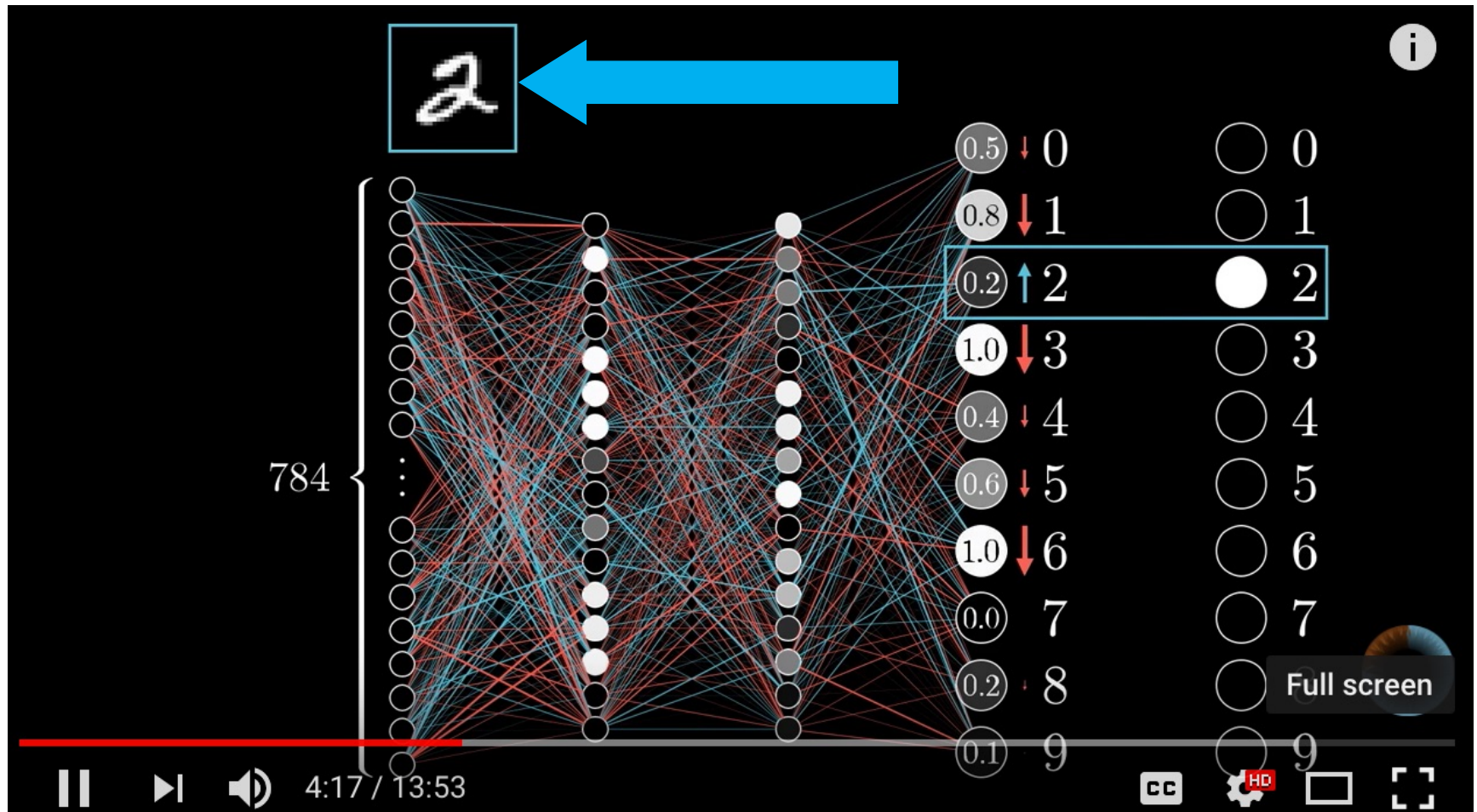
$$\left\{ \begin{array}{l} (0.18 - 0.00)^2 + \\ (0.29 - 0.00)^2 + \\ (0.58 - 0.00)^2 + \\ (0.77 - 0.00)^2 + \\ (0.20 - 0.00)^2 + \\ (0.36 - 0.00)^2 + \\ (0.93 - 0.00)^2 + \\ (1.00 - 0.00)^2 + \\ (0.95 - 1.00)^2 + \\ (0.35 - 0.00)^2 \end{array} \right.$$

What's the "cost" ⁱ
of this difference?



Utter trash 

Backpropagation

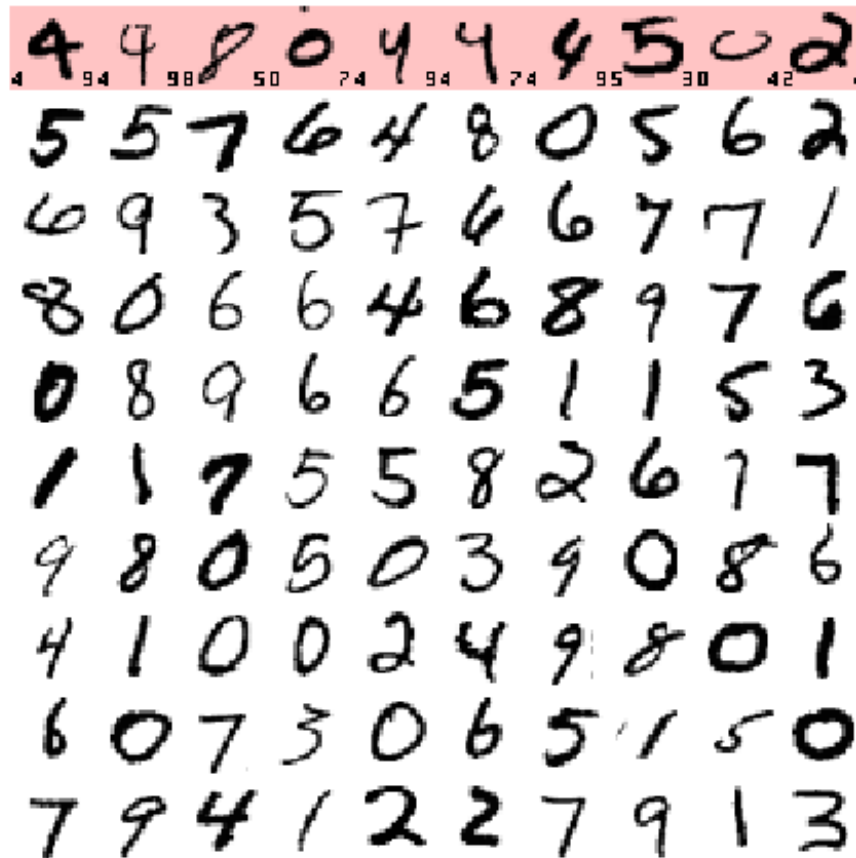


Source: 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, <https://www.youtube.com/watch?v=llg3gGewQ5U>

TensorFlow Image Recognition

MNIST dataset: 60,000 labeled digits

Training digits



Get Started with TensorFlow

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

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)
```


Get Started with TensorFlow MNIST

 `_index.ipynb` 

File Edit View Insert Runtime Tools Help

+ CODE + TEXT ↑ CELL ↓ CELL COPY TO DRIVE

CONNECT ▾ EDITING

Get Started with TensorFlow



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:

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

Load and prepare the [MNIST](#) dataset. Convert the samples from integers to floating-point numbers:

```
[ ] 1 mnist = tf.keras.datasets.mnist
2
3 (x_train, y_train), (x_test, y_test) = mnist.load_data()
4 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:

```
[ ] 1 model = tf.keras.models.Sequential([
2     tf.keras.layers.Flatten(),
3     tf.keras.layers.Dense(512, activation=tf.nn.relu),
4     tf.keras.layers.Dropout(0.2),
5     tf.keras.layers.Dense(10, activation=tf.nn.softmax)
6 ])
7
```


TensorFlow and Deep Learning

1 Overview

Preparation: Install

2 TensorFlow, get the sample code

3 Theory: train a neural network

4 Theory: a 1-layer neural network

5 Theory: gradient descent

6 Lab: let's jump into the code

7 Lab: adding layers

8 Lab: special care for deep networks

9 Lab: learning rate decay

10 Lab: dropout, overfitting

11 Theory: convolutional networks

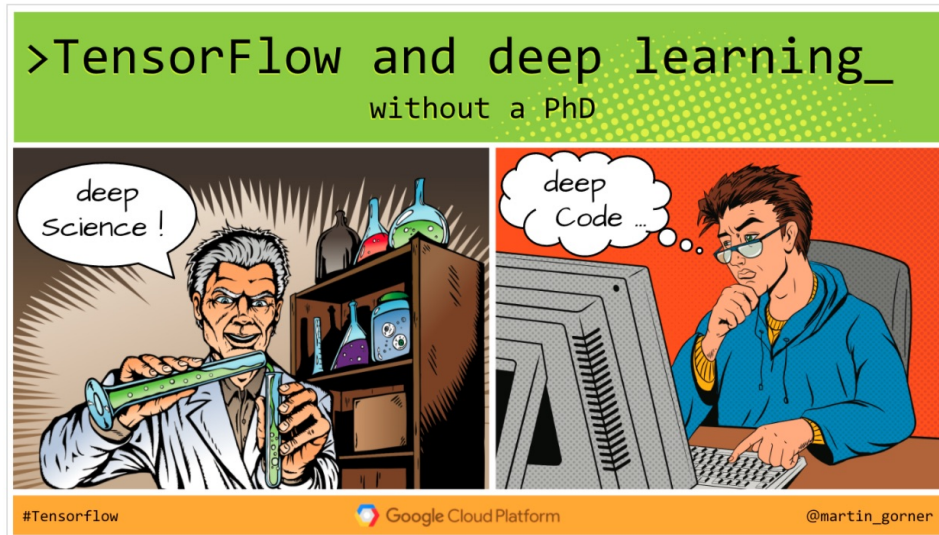
Did you find a mistake? [Please file a bug.](#)

Lab: a convolutional

← 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

TensorFlow MNIST Tutorial



Features Business Explore Pricing

This repository

Search

[Sign in](#) or [Sign up](#)

martin-gorner / tensorflow-mnist-tutorial

Watch

50

Star

489

Fork

204

Code

Issues **6**

Pull requests **2**

Projects **0**

Pulse

Graphs

Sample code for "Tensorflow and deep learning, without a PhD" presentation and code lab.

102 commits

1 branch

0 releases

4 contributors

Apache-2.0

Branch: **master** ▾

[New pull request](#)

[Find file](#)

[Clone or download ▾](#)

martin-gorner committed on GitHub Update INSTALL.txt ...	Latest commit ed331aa 25 days ago
mlengine	added example using the Tensorflow high level layers API 26 days ago
.gitignore	small bug fix in batch norm 6 months ago
CONTRIBUTING.md	initial commit 2 4 months ago
INSTALL.txt	Update INSTALL.txt 25 days ago
LICENSE	Initial commit a year ago
README.md	better image URL 3 months ago
mnist_1.0_softmax.py	global_variables_initializer used everywhere instead of initalize_al... 2 months ago
mnist_2.0_five_layers_sigmoid.py	Fix spacing in the network structure comment a month ago
mnist_2.1_five_layers_relu_lrdecay...	Fix spacing in the network structure comment a month ago

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

TensorFlow MNIST Tutorial

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

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

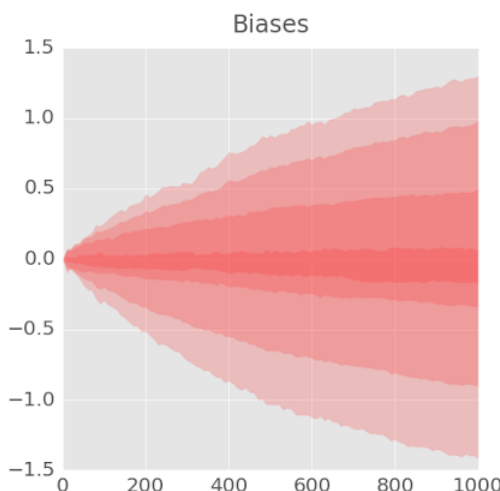
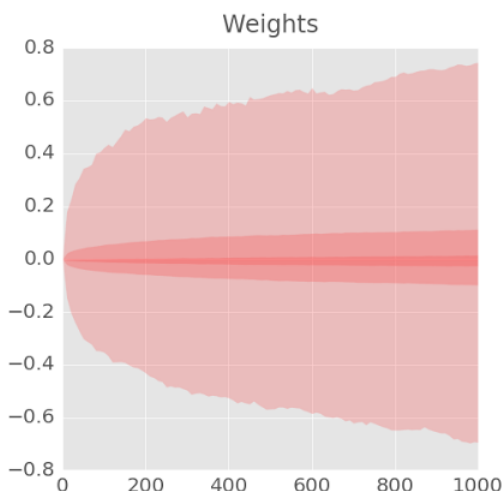
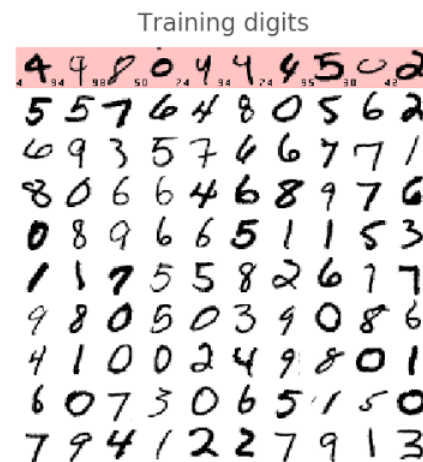
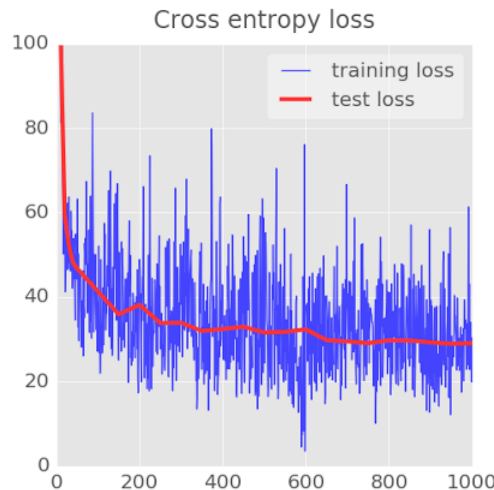
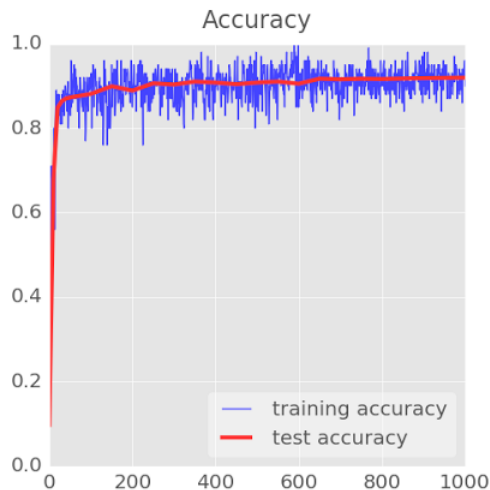
```
python3 mnist_1.0_softmax.py
```

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

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

cd tensorflow-mnist-tutorial

python3 mnist_1.0_softmax.py



Train a Neural Network

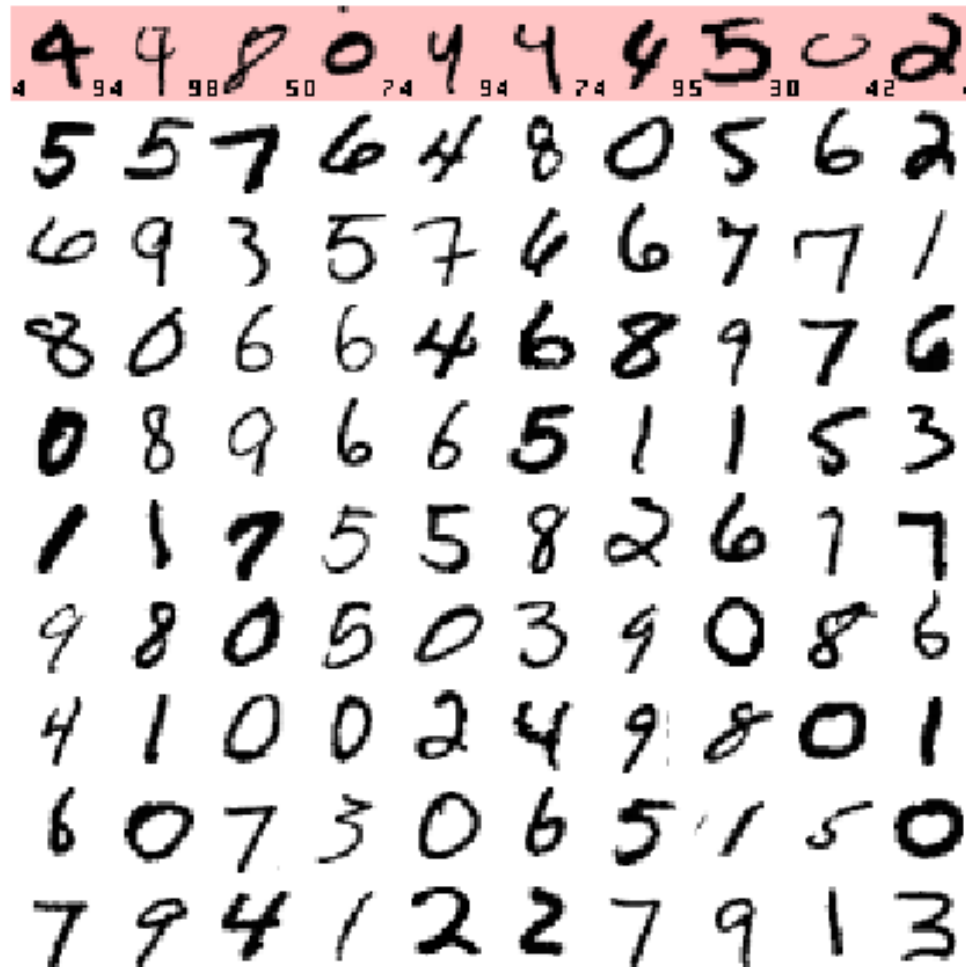
Training digits

updates to **weights** and **biases** =>

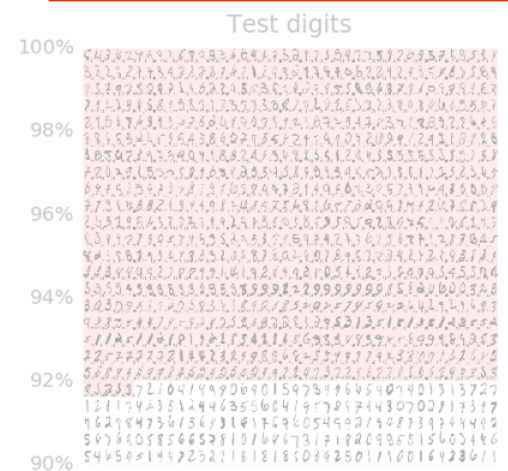
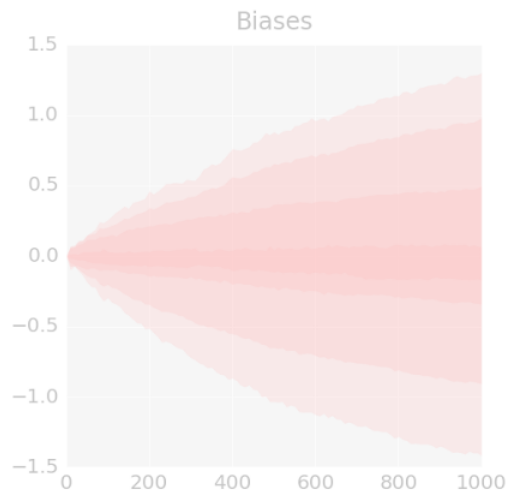
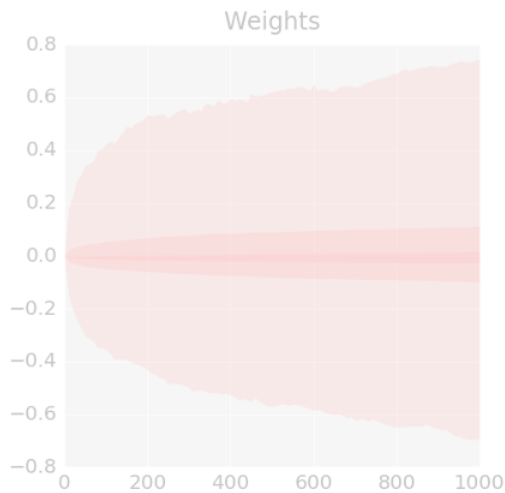
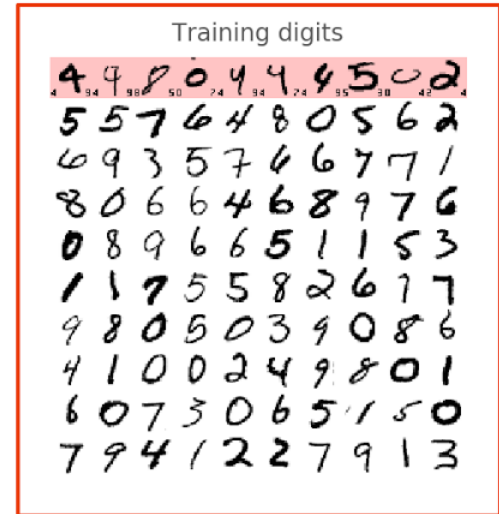
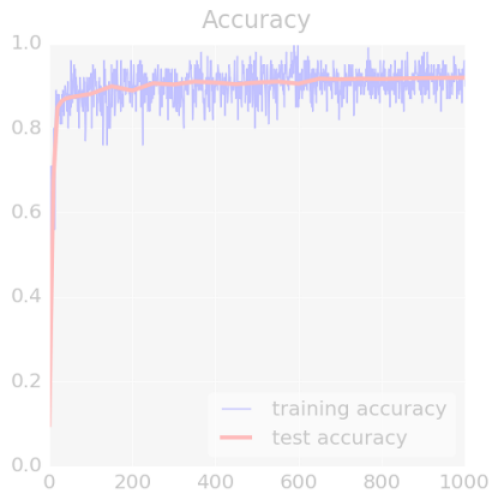
better recognition (loop)

Training digits

Training digits



Training digits

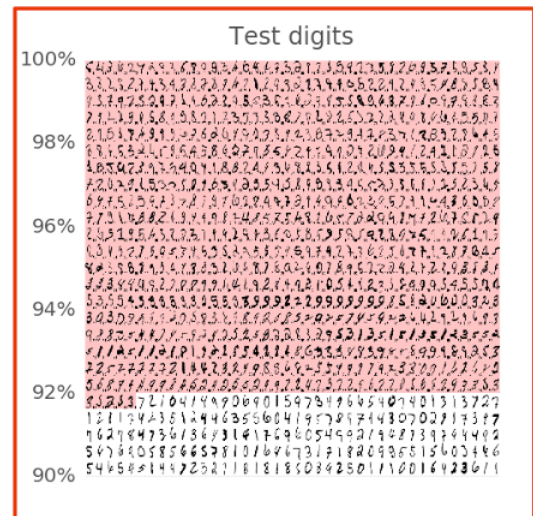
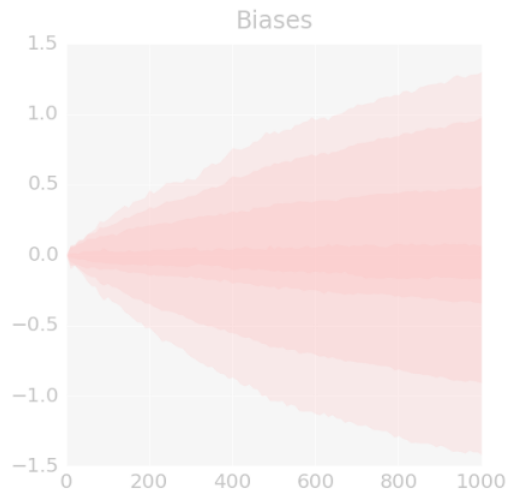
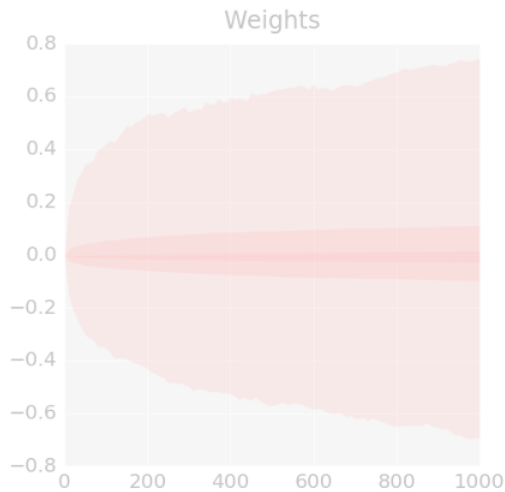
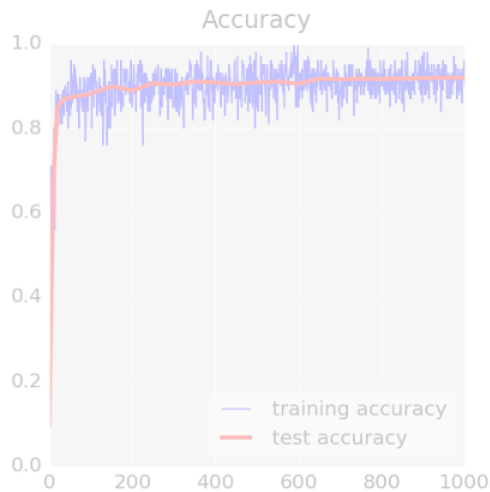


Test digits

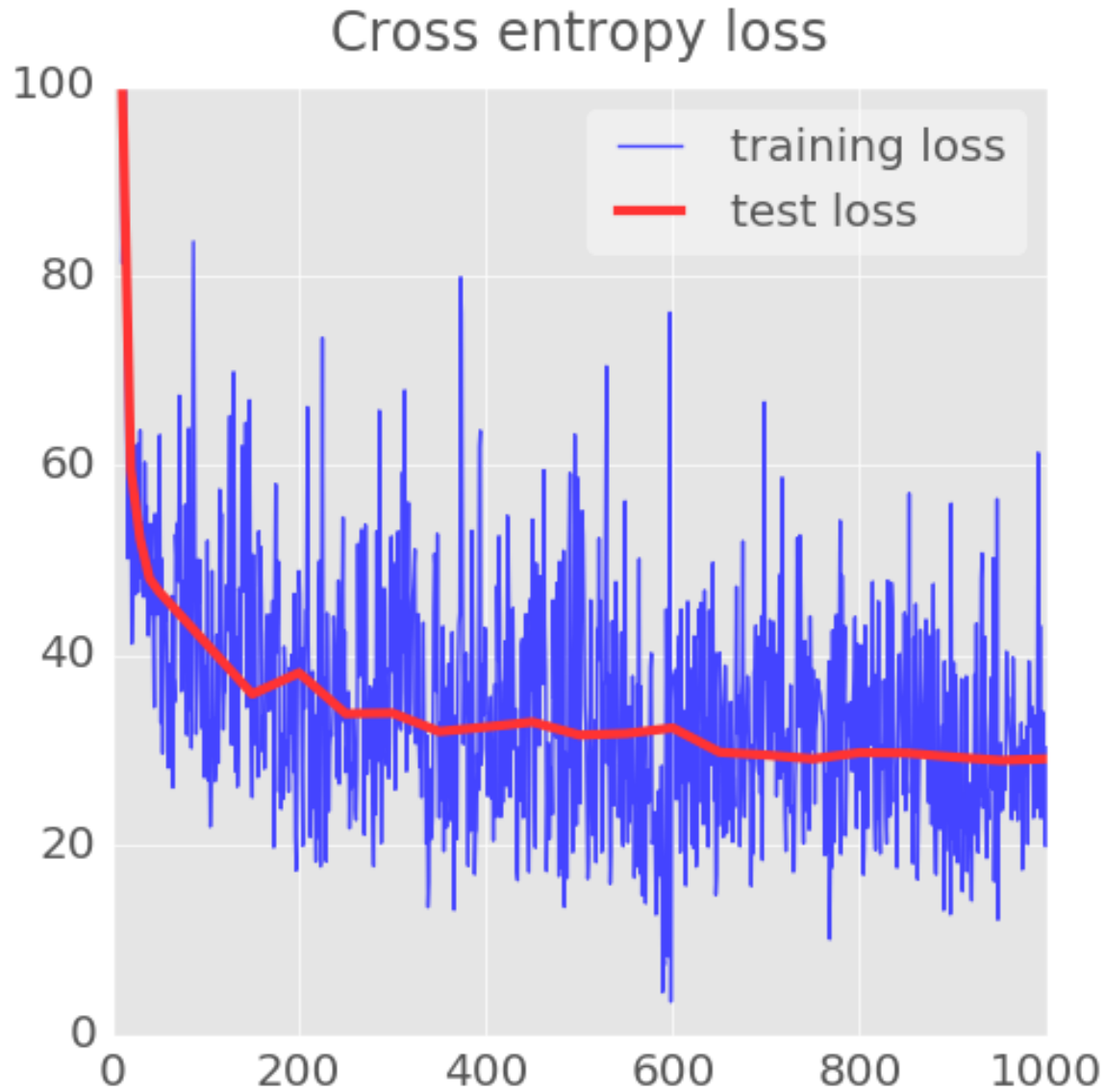


Source: <https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2>

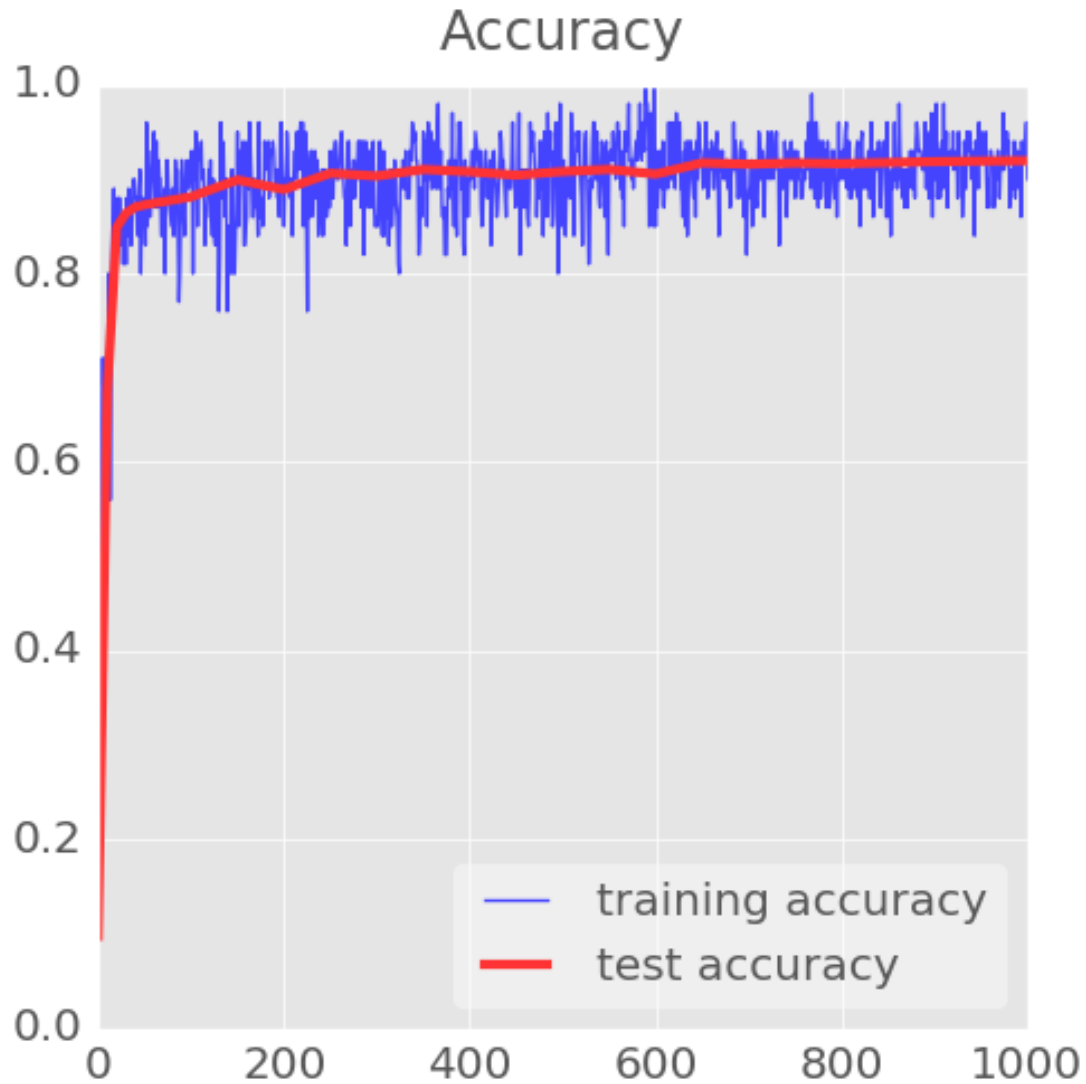
Test digits



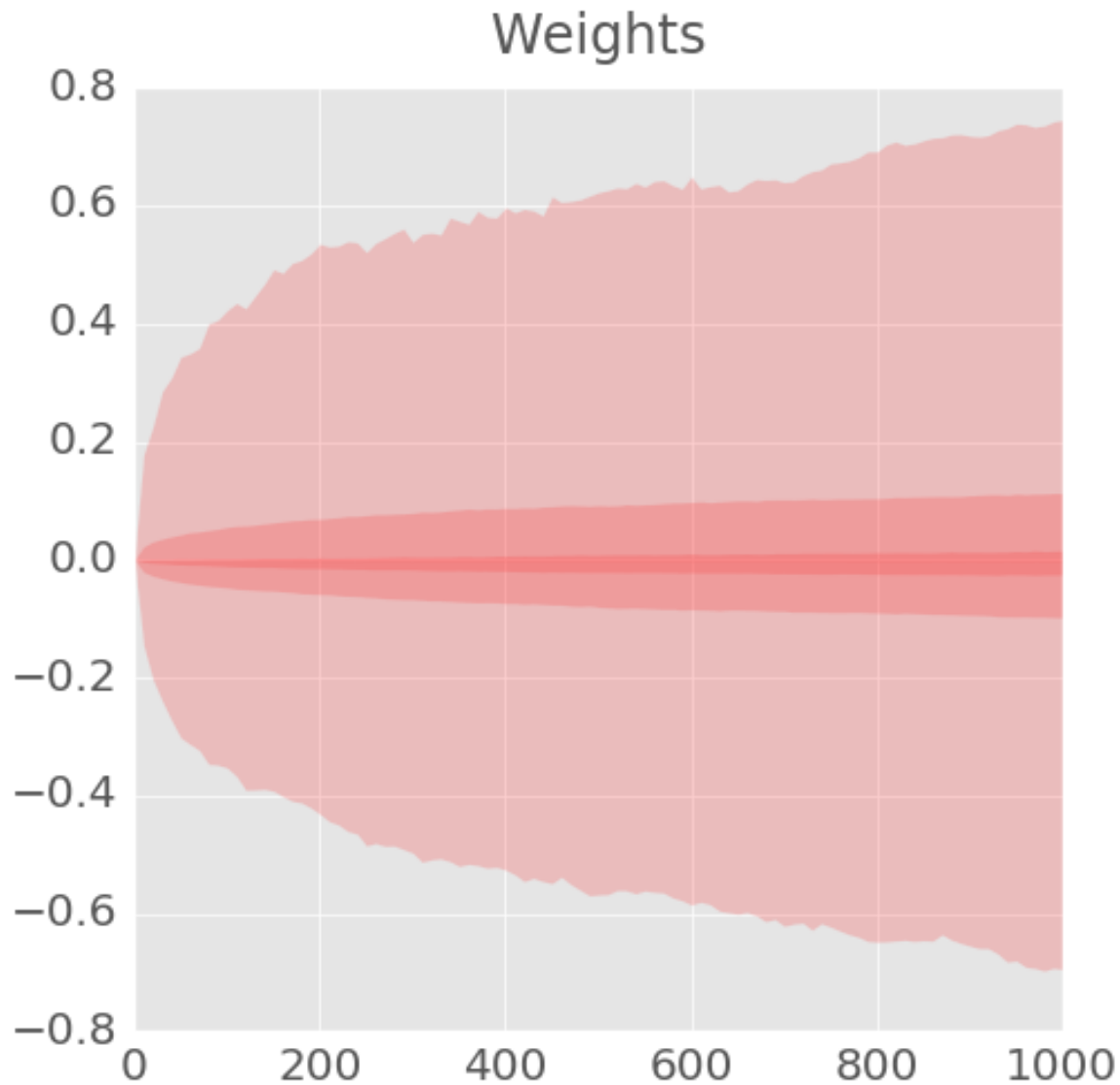
Cross entropy loss



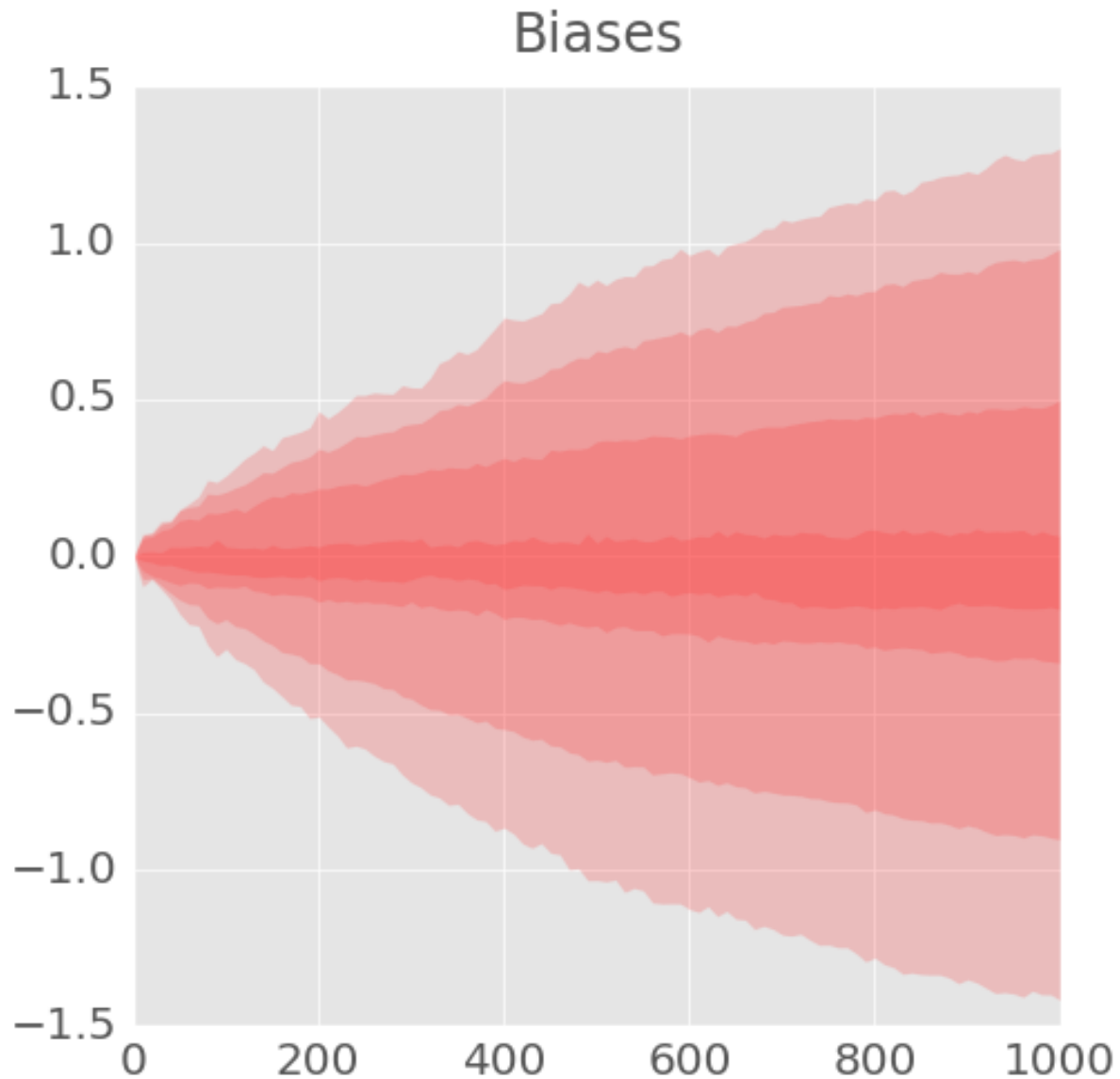
Accuracy



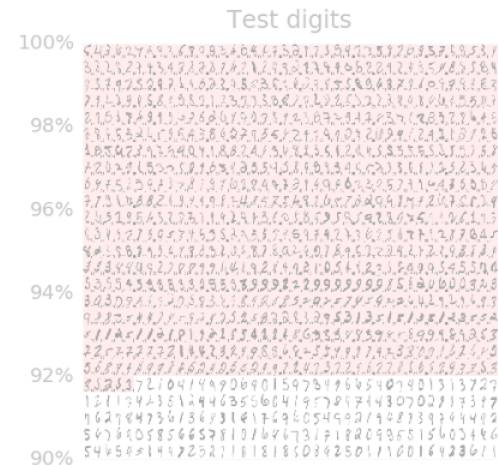
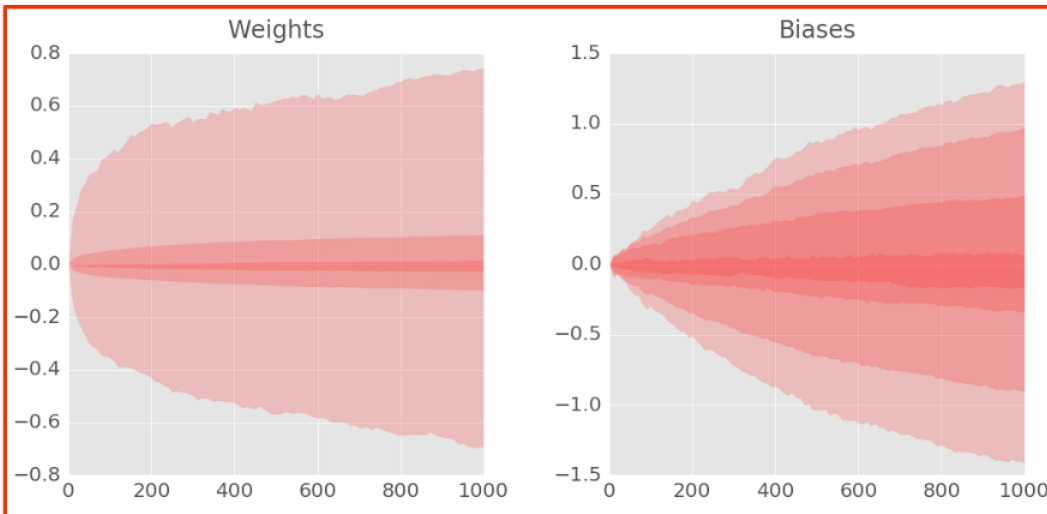
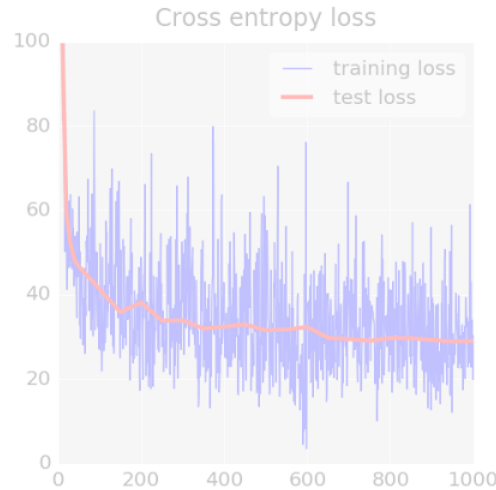
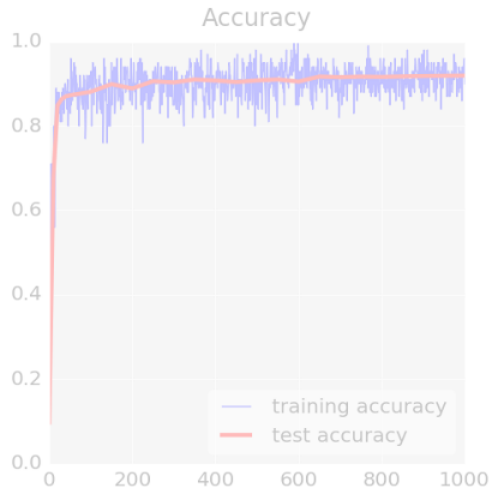
Weights



Biases



Weights and Biases

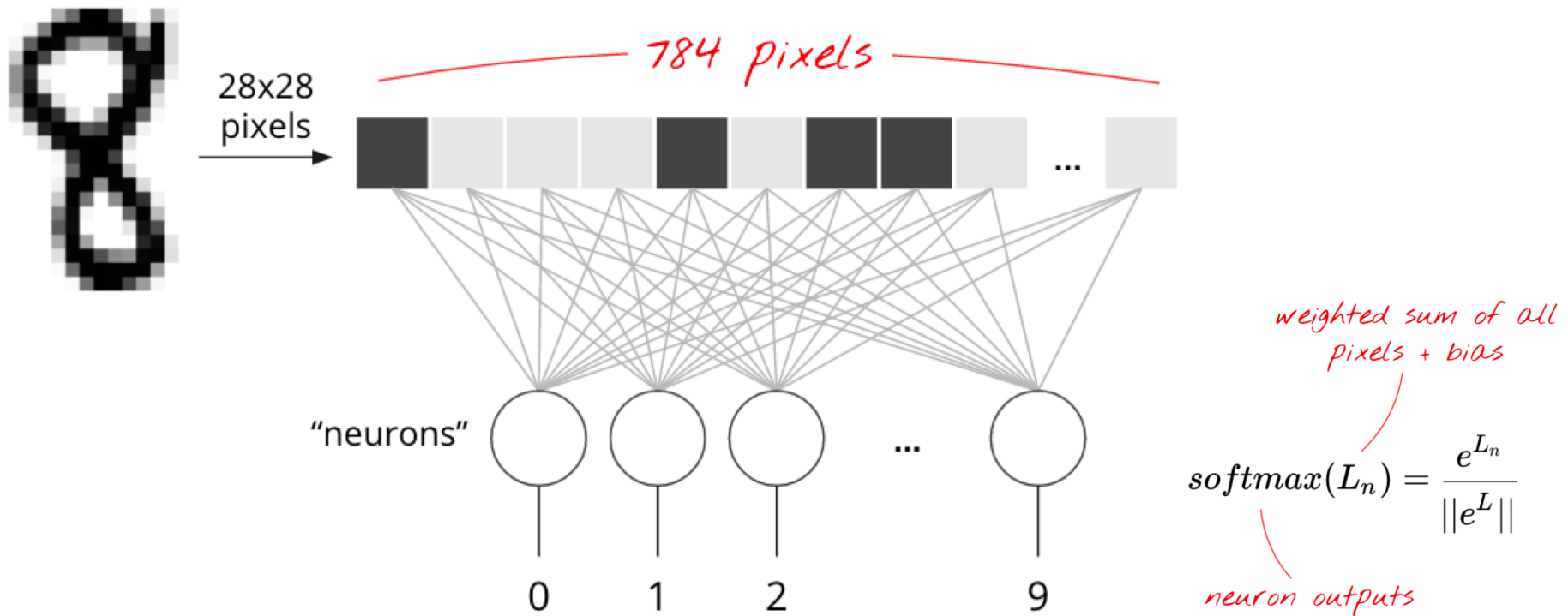


Cookbook

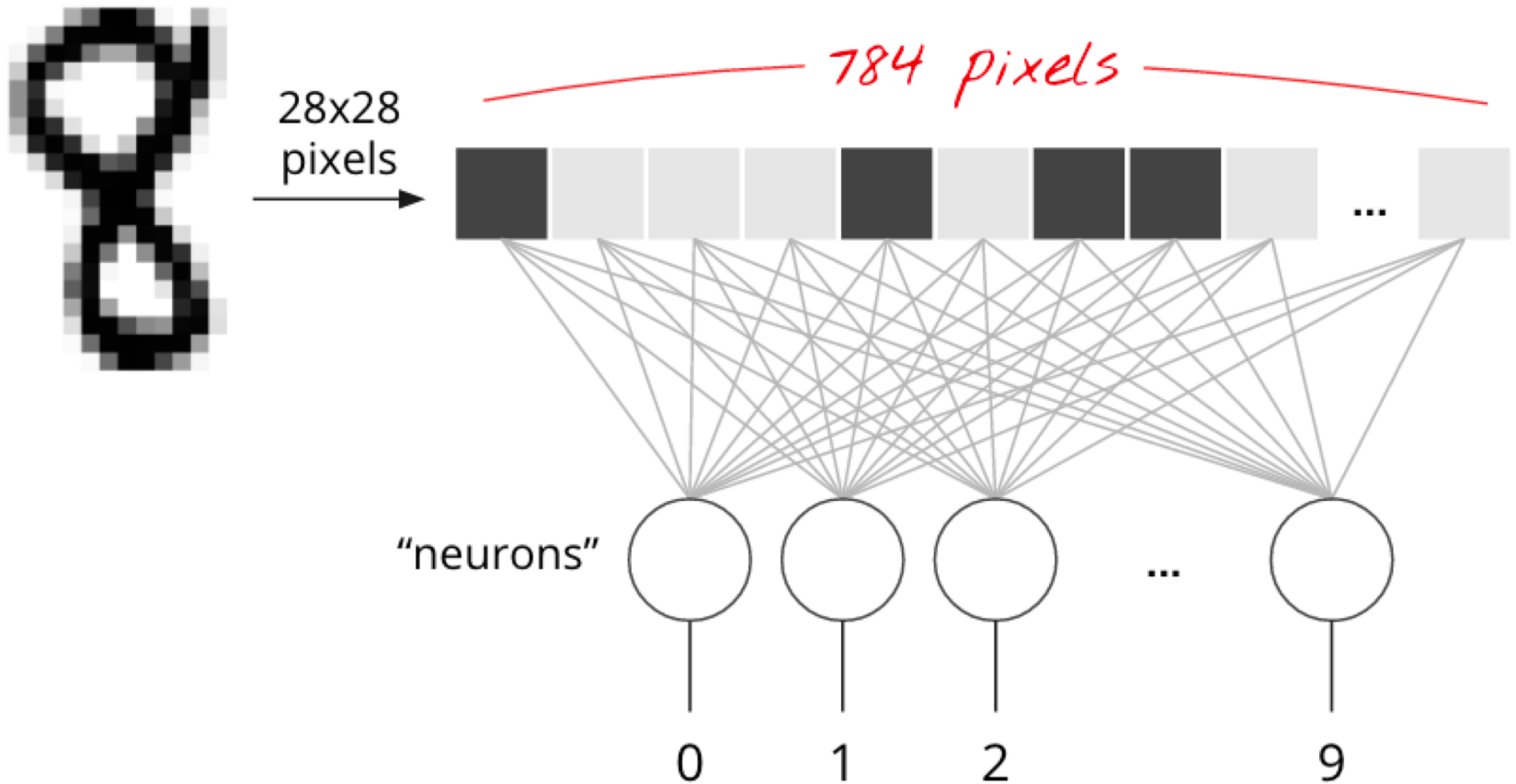
Softmax
Cross-entropy
Mini-batch



Very Simple Model: Softmax Classification



Very Simple Model: Softmax Classification



Very Simple Model: Softmax Classification

*weighted sum of all
pixels + bias*

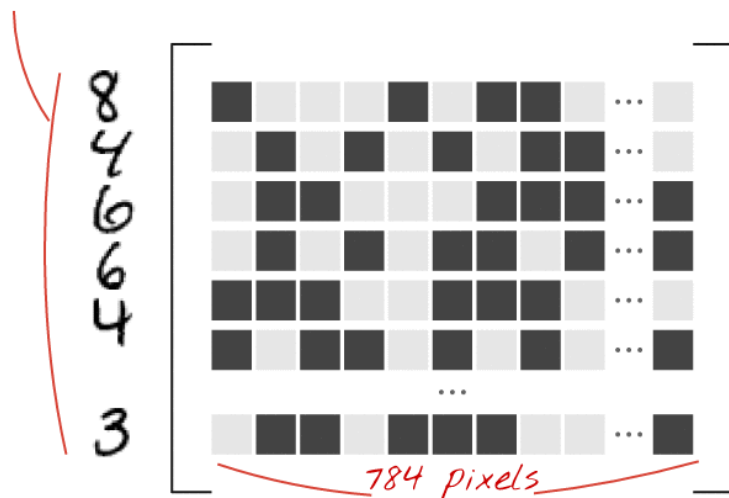
$$\text{softmax}(L_n) = \frac{e^{L_n}}{\|e^L\|}$$

neuron outputs

In Matrix notation, 100 images at a time

*X: 100 images,
one per line,
flattened*

<i>10 columns</i>									
$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$...	$W_{0,9}$	<i>784 lines</i>			
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$...	$W_{1,9}$				
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$...	$W_{2,9}$				
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$...	$W_{3,9}$				
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$...	$W_{4,9}$				
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$...	$W_{5,9}$				
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$...	$W_{6,9}$				
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$...	$W_{7,9}$				
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$...	$W_{8,9}$				
...									
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$...	$W_{783,9}$					

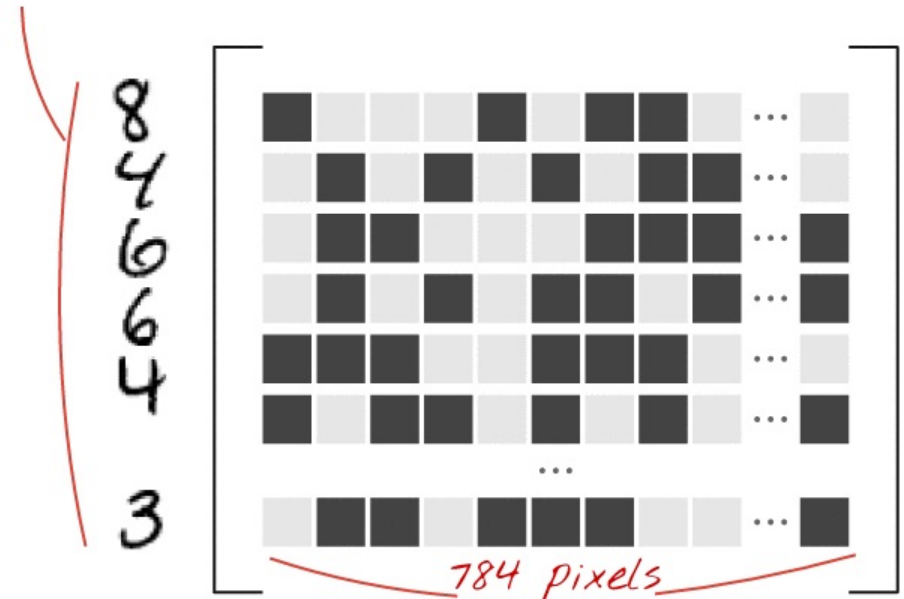


*X: 100 images,
one per line,
flattened*

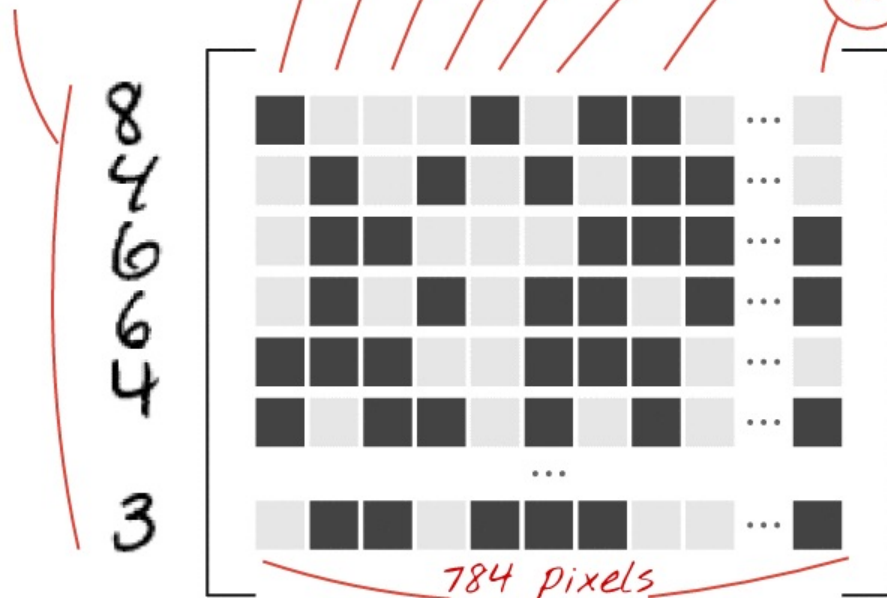
10 columns

$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$...	$W_{0,9}$
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$...	$W_{1,9}$
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$...	$W_{2,9}$
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$...	$W_{3,9}$
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$...	$W_{4,9}$
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$...	$W_{5,9}$
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$...	$W_{6,9}$
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$...	$W_{7,9}$
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$...	$W_{8,9}$
...					
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$...		$W_{783,9}$

784 lines



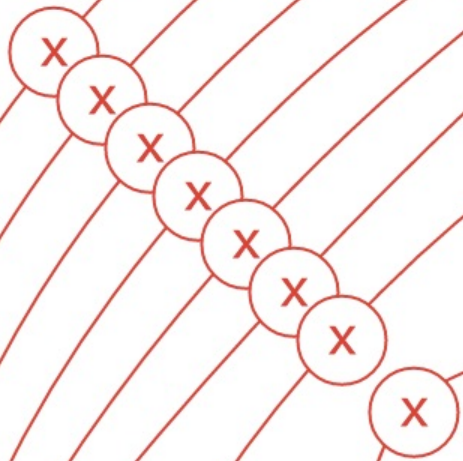
X: 100 images,
one per line,
flattened



10 columns

$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$...	$W_{0,9}$
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$...	$W_{1,9}$
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$...	$W_{2,9}$
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$...	$W_{3,9}$
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$...	$W_{4,9}$
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$...	$W_{5,9}$
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$...	$W_{6,9}$
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$...	$W_{7,9}$
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$...	$W_{8,9}$
...					
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$...	$W_{783,9}$	

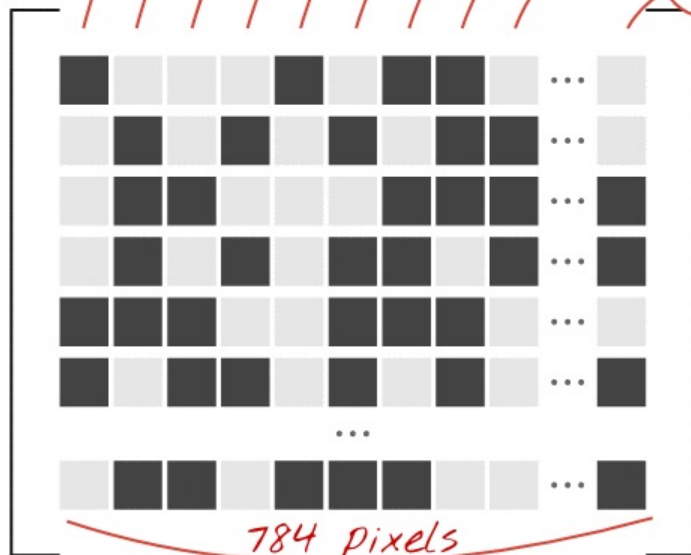
784 lines



$L_{0,0}$

X: 100 images,
one per line,
flattened

8
4
6
6
4
3



10 columns

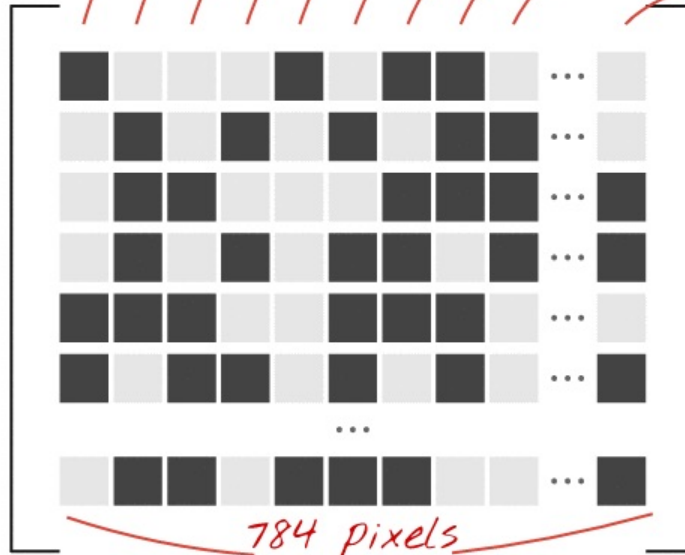
$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$...	$W_{0,9}$
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$...	$W_{1,9}$
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$...	$W_{2,9}$
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$...	$W_{3,9}$
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$...	$W_{4,9}$
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$...	$W_{5,9}$
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$...	$W_{6,9}$
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$...	$W_{7,9}$
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$...	$W_{8,9}$
...
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$	$W_{783,9}$

784 lines

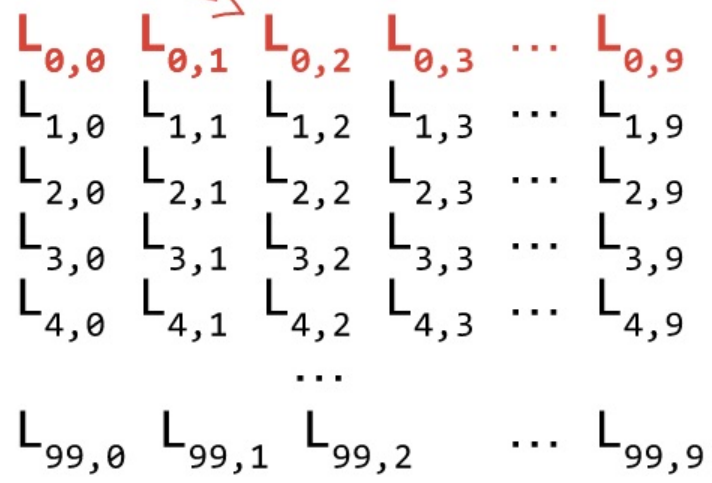
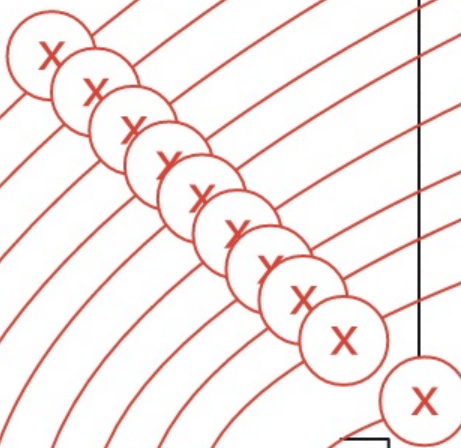
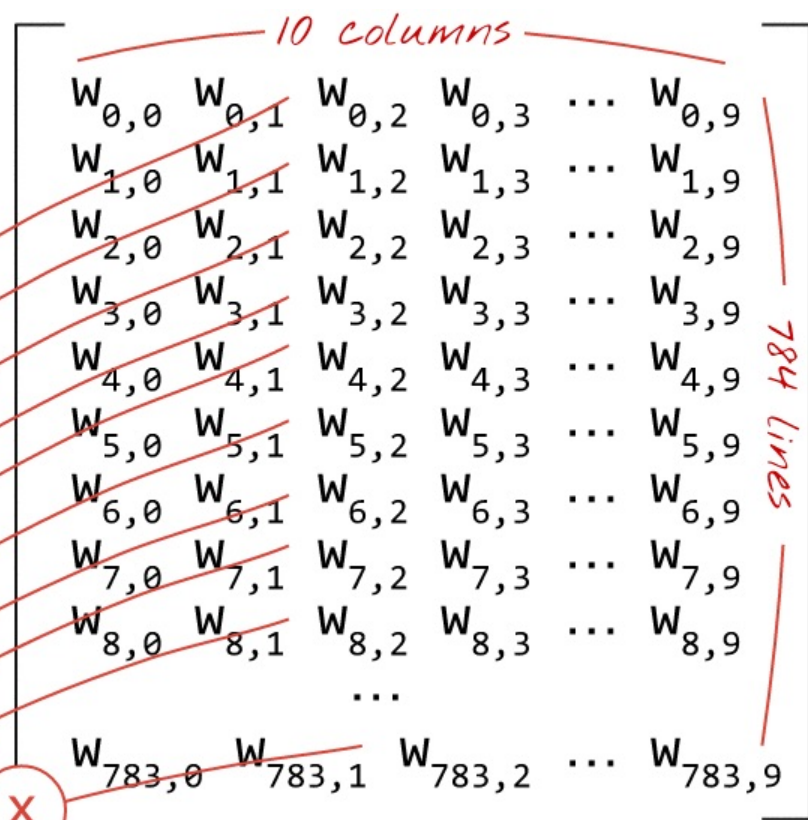
$L_{0,0}$ $L_{0,1}$

X: 100 images,
one per line,
flattened

8
4
6
6
4
3



784 pixels



What are "**weights**" and "**biases**" ?

How is the "**cross-entropy**"
computed ?

How exactly does the
training algorithm work ?

$$Y = f(X)$$

Predictions

$Y[100, 10]$

Images

$X[100, 784]$

Weights

$W[784, 10]$

Biases

$b[10]$

$$Y = \text{softmax}(X \cdot W + b)$$

applied line
by line

matrix multiply

broadcast
on all lines

tensor shapes in []

$$Y = \text{tf.nn.softmax}(\text{tf.matmul}(X, W) + b)$$

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

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	0	0	0

actual probabilities, "one-hot" encoded

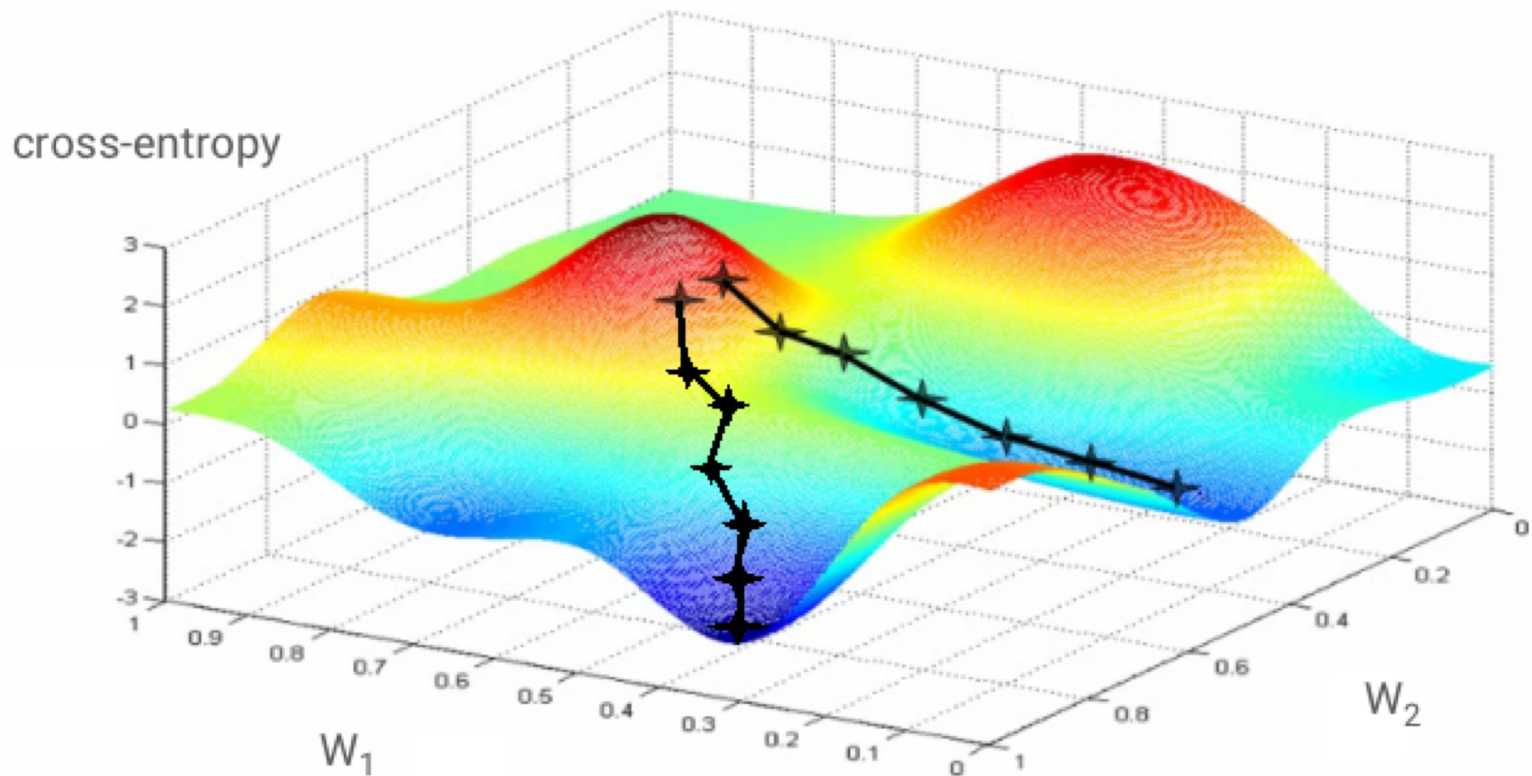
Cross entropy: $-\sum Y_i' \cdot \log(Y_i)$

computed probabilities

0.1	0.2	0.1	0.3	0.2	0.1	0.9	0.2	0.1	0.1
0	1	2	3	4	5	6	7	8	9

this is a "6"

Minimizing Cross Entropy (Minimizing Loss)



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

"mini-batches":
100 images and labels

```
import tensorflow as tf
```


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()
```

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))
```

mnist_1.0_softmax.py

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

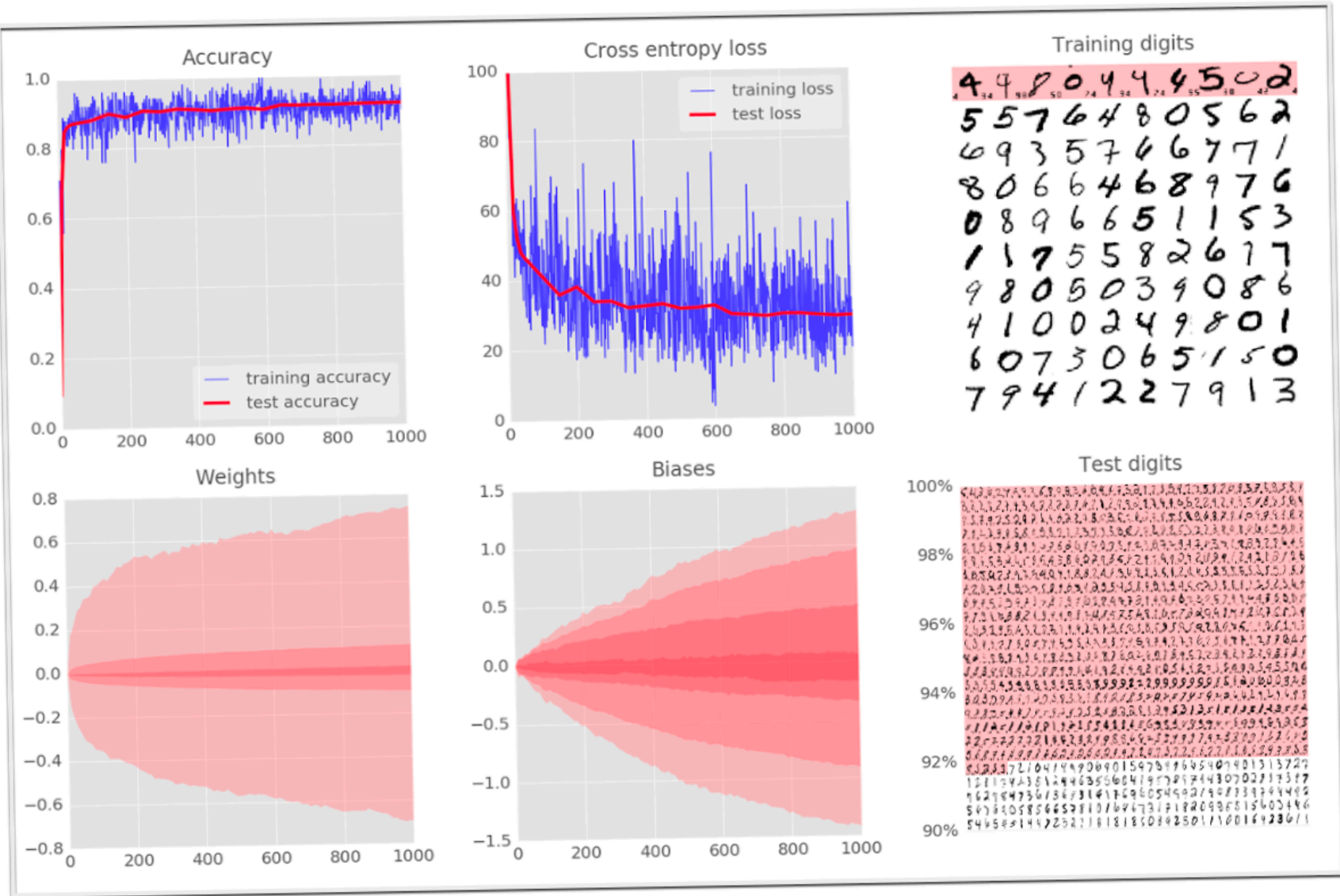
for i in range(1000):
    # load batch of images and correct answers
    batch_X, batch_Y = mnist.train.next_batch(100)
    train_data={X: batch_X, Y_: batch_Y}

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

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)
```

mnist_1.0_softmax.py

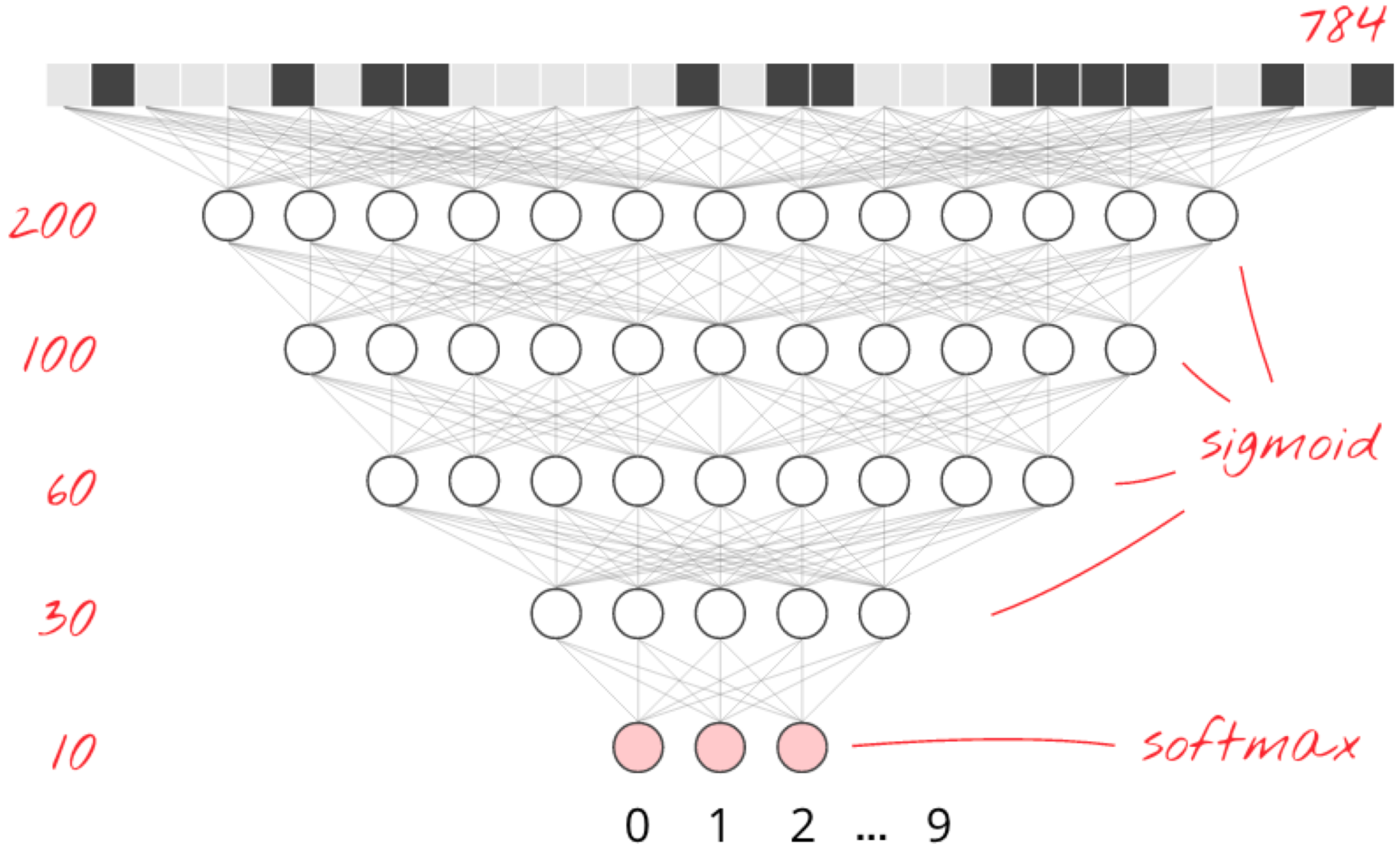


92!
😊

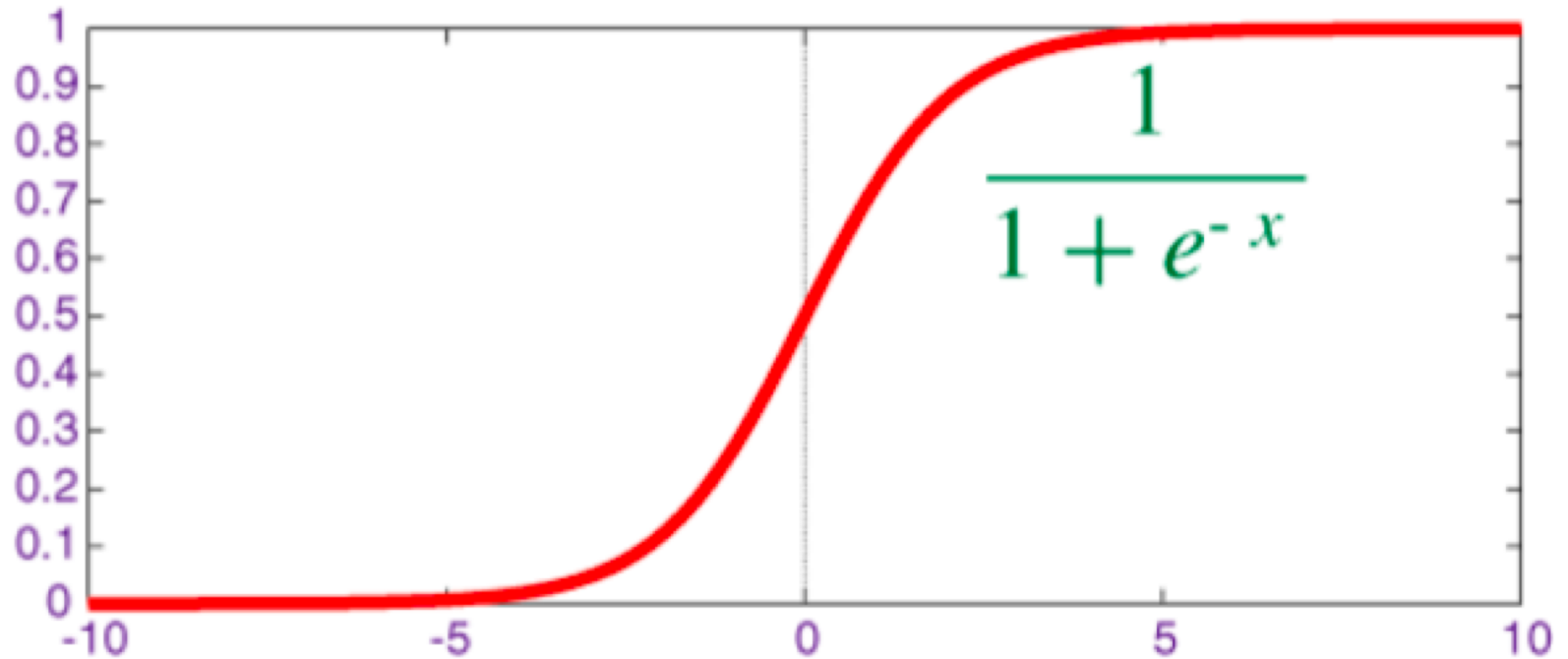
Deep Learning



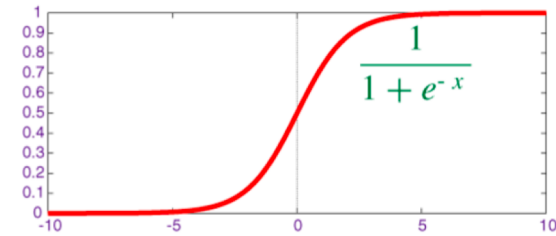
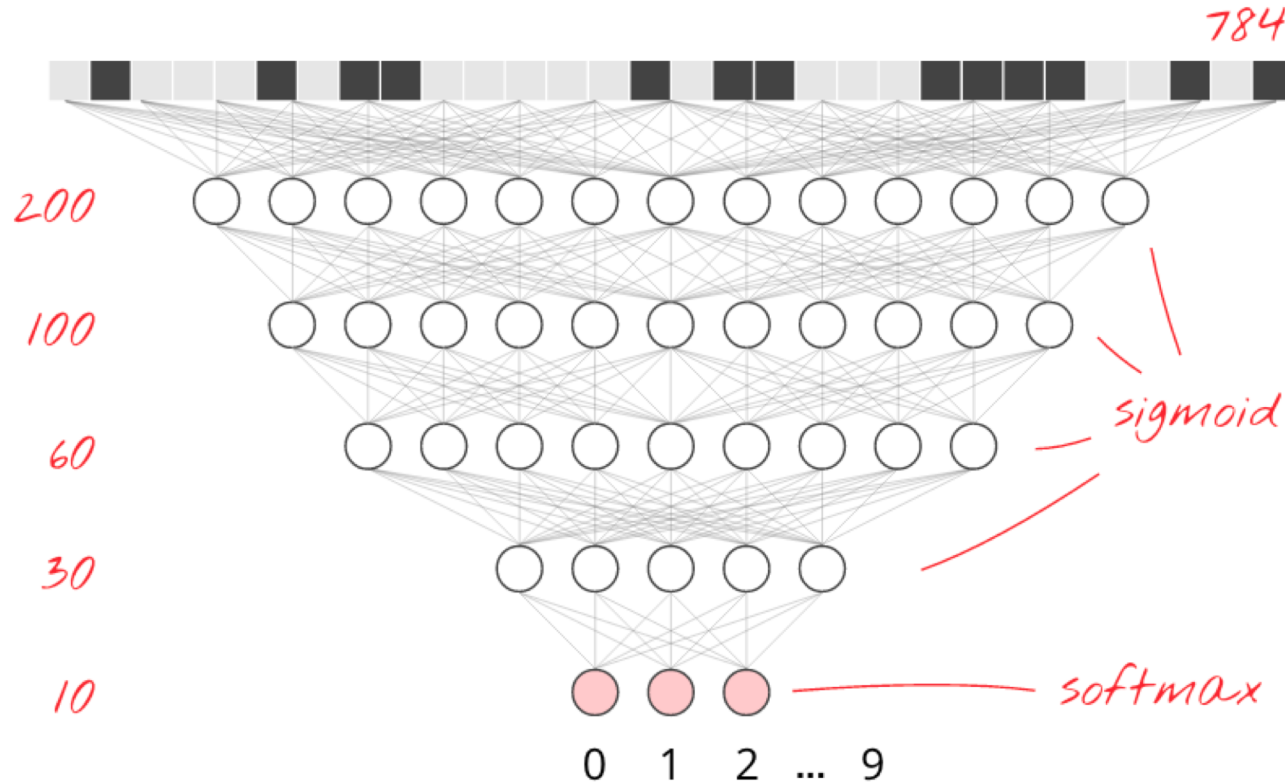
5 fully-connected layers



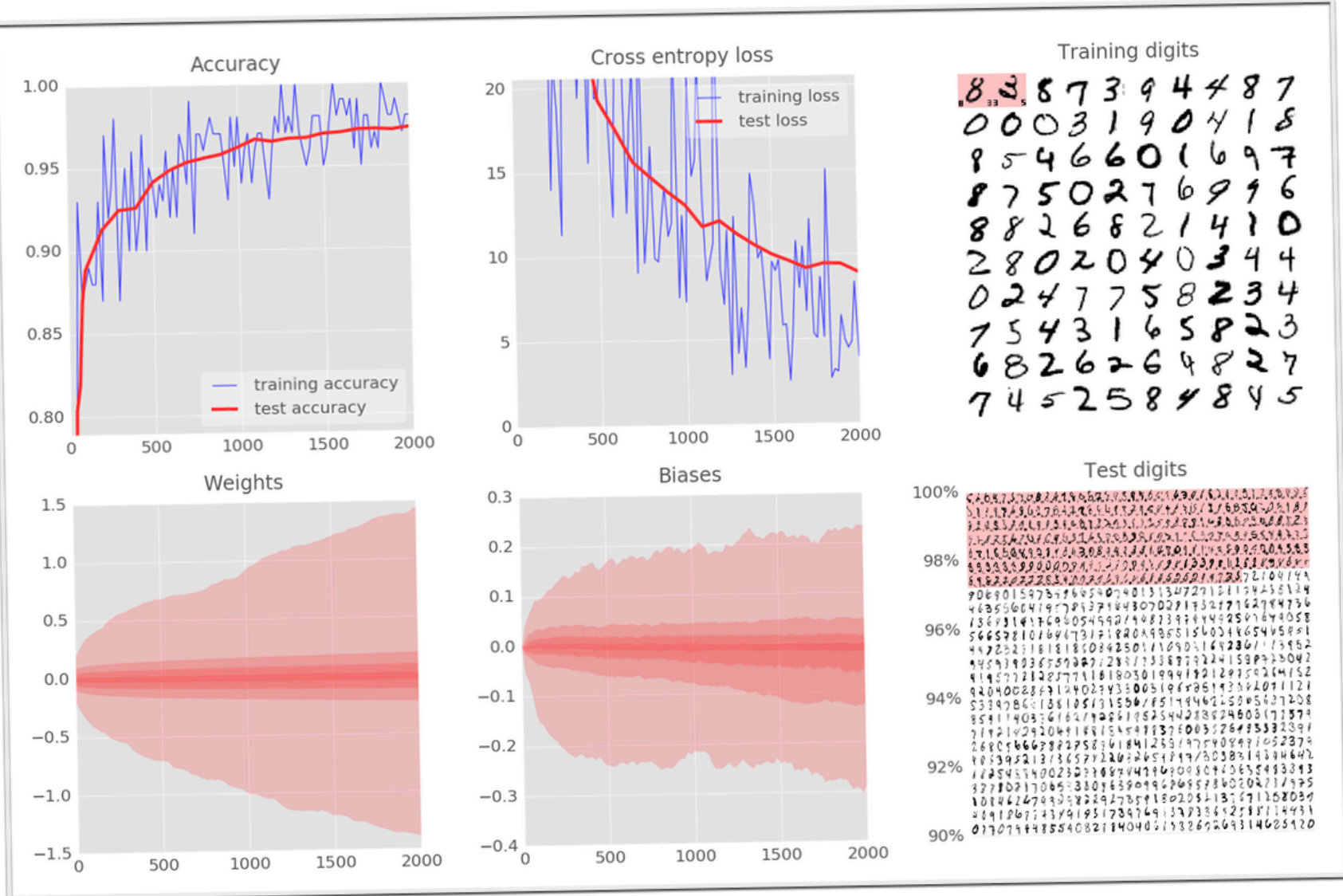
Sigmoid



5 fully-connected layers



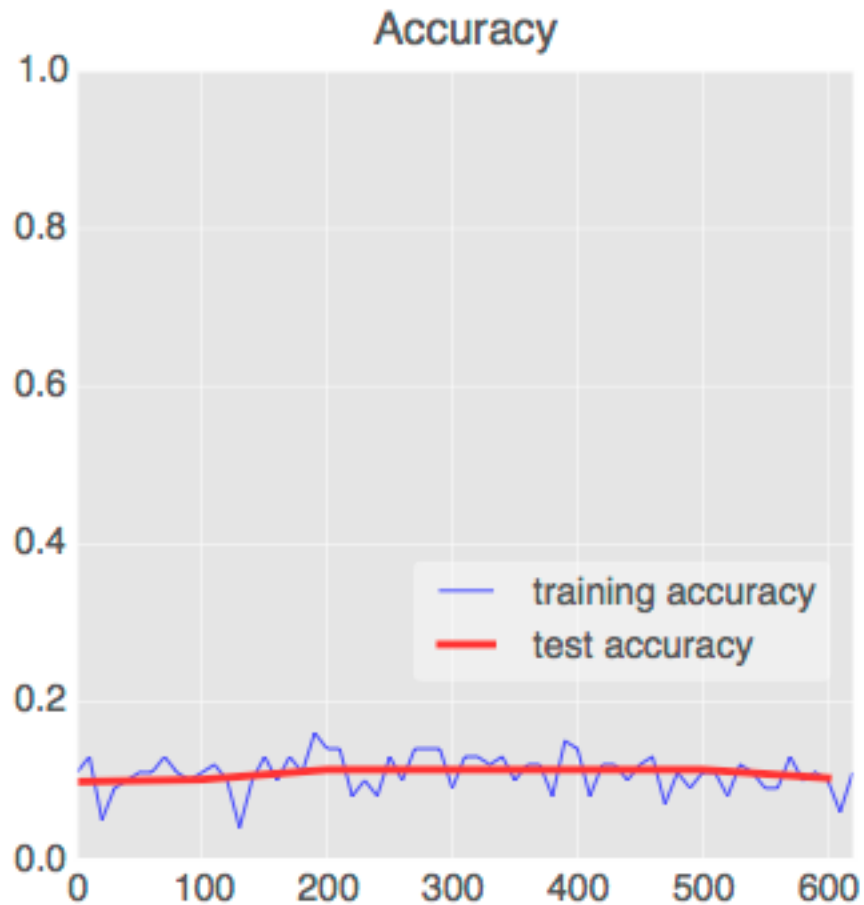
TensorFlow MNIST Tutorial



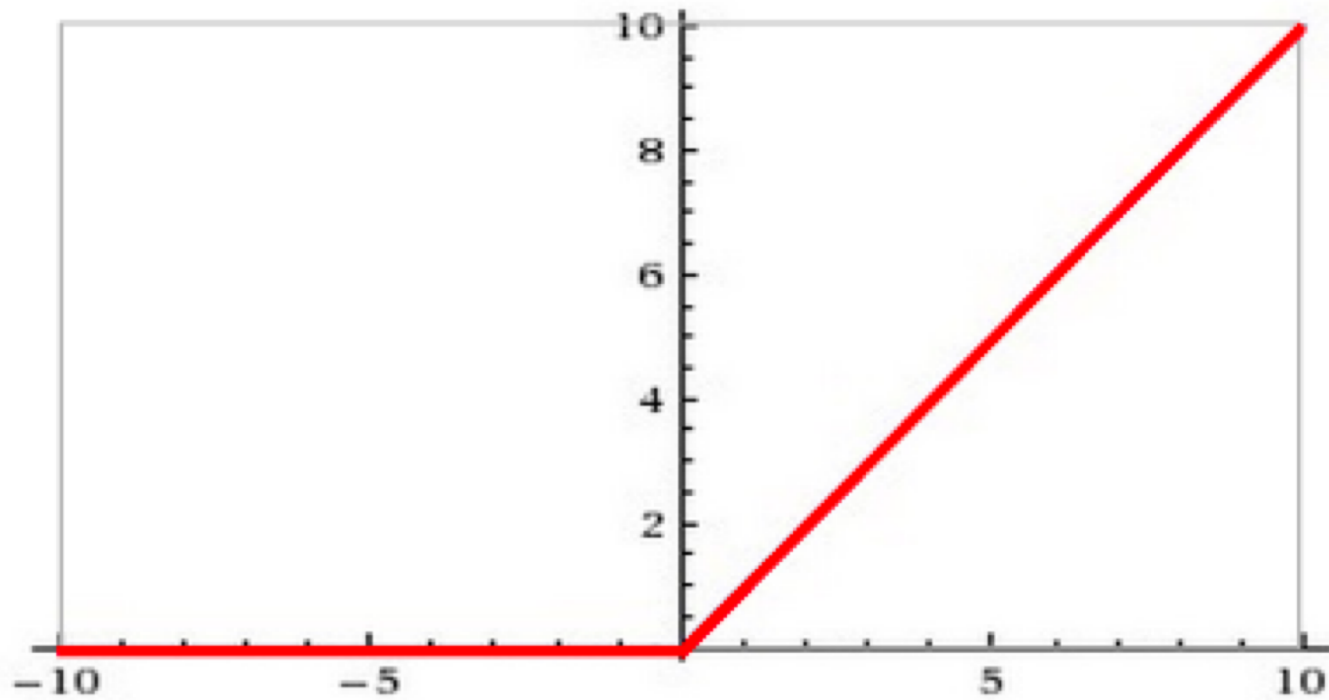
ReLU



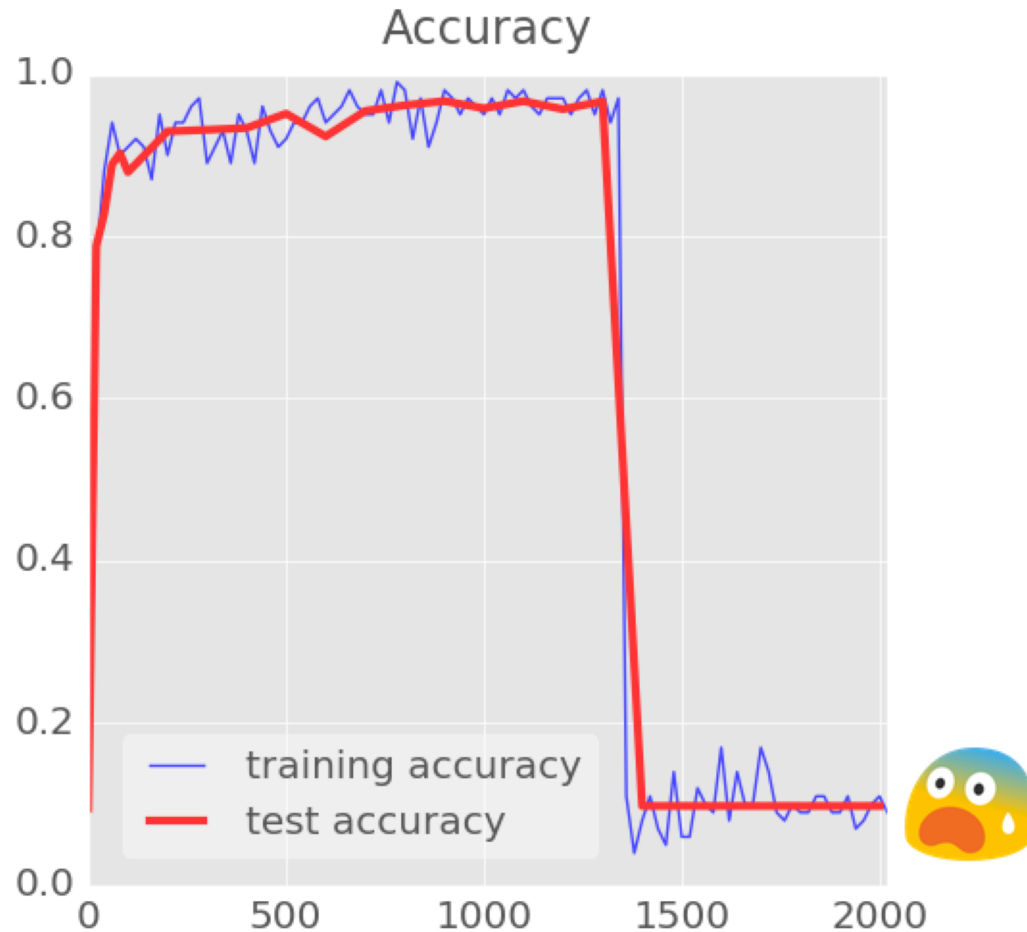
TensorFlow MNIST Tutorial



ReLU



TensorFlow MNIST Tutorial



Learning Rate

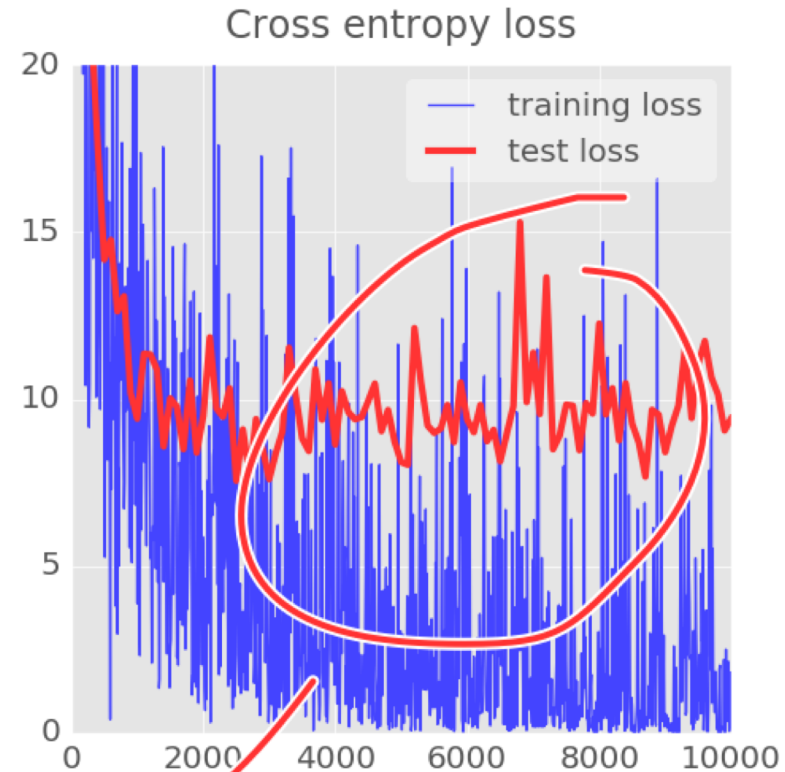
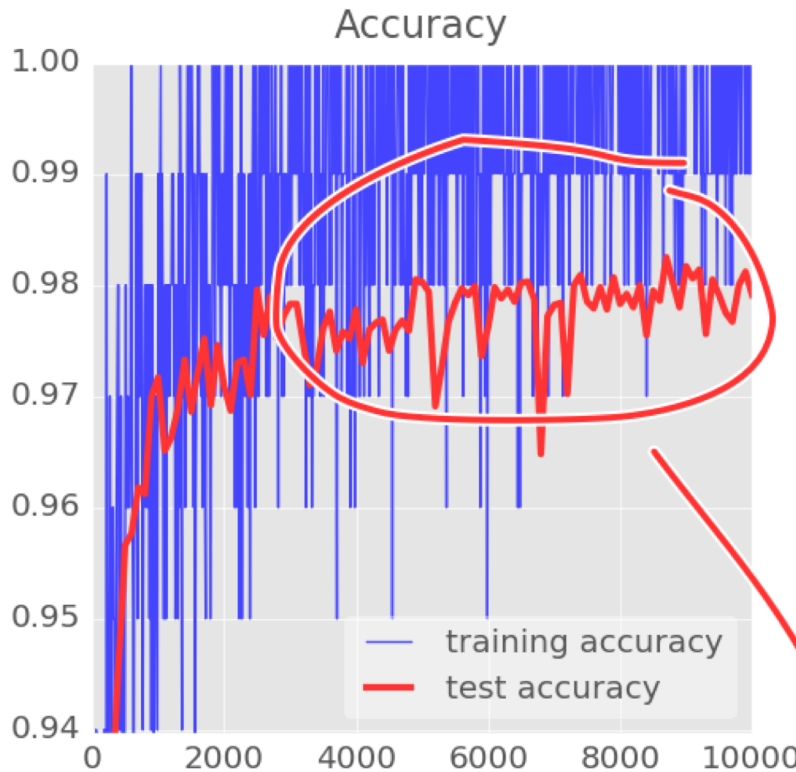
Slow down...

Learning rate decay



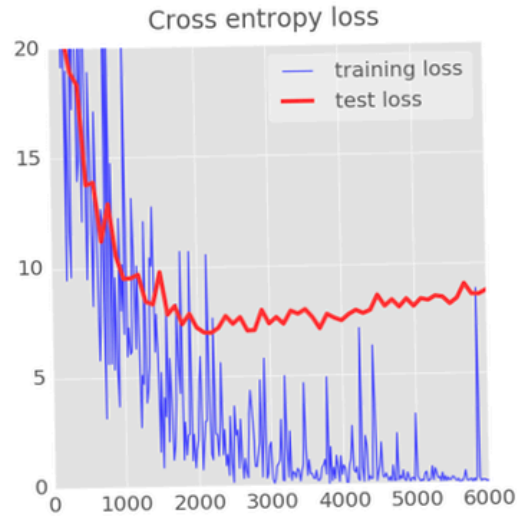
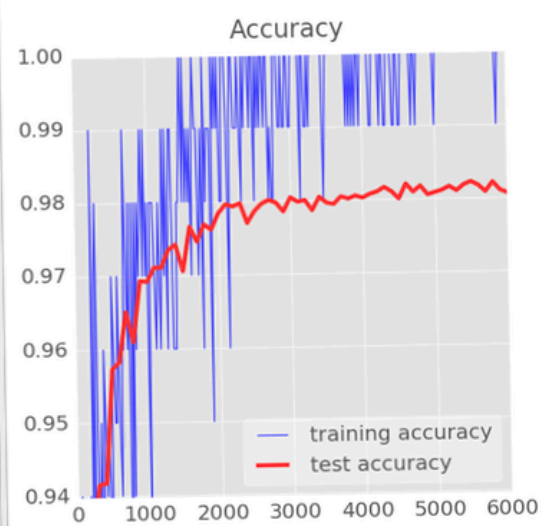
TensorFlow MNIST Tutorial

LR =
0.003



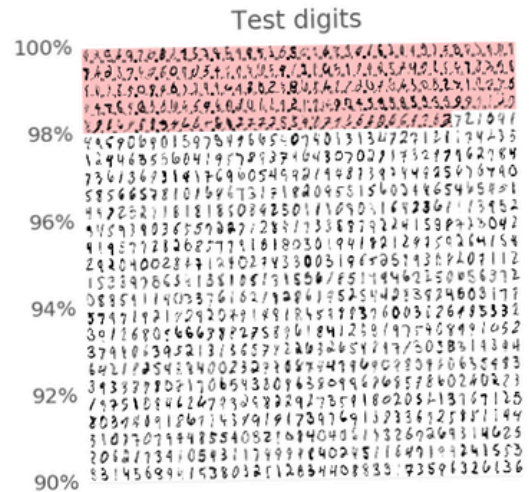
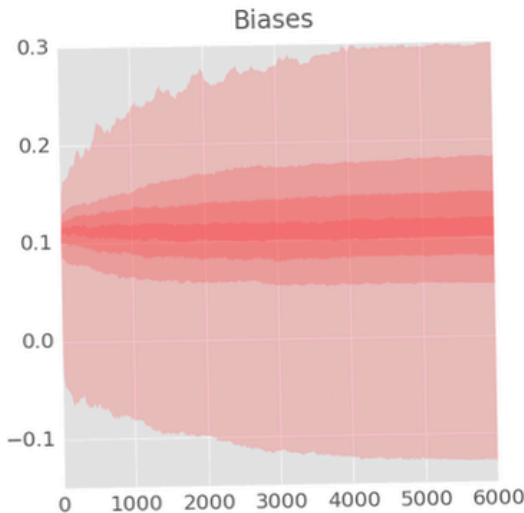
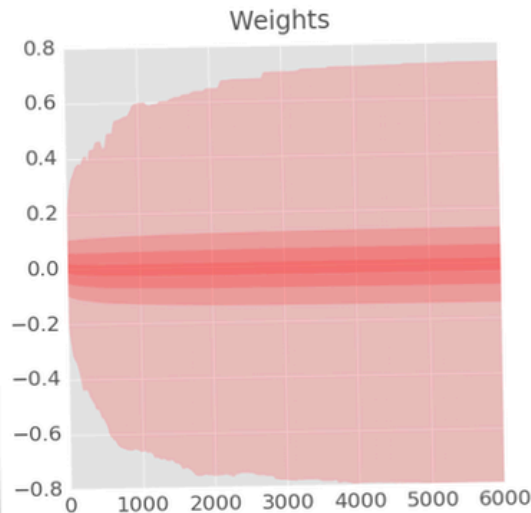
yuck!

TensorFlow MNIST Tutorial



Training digits

```
5 8 0 6 2 8 2 8 6 5
8 4 4 / 6 7 9 9 4 5
0 7 8 7 8 0 7 1 0 7
5 5 7 5 5 8 3 8 5 9
6 6 2 4 0 8 6 7 5 1
3 9 1 0 5 4 0 5 0 1
6 4 8 3 7 3 7 8 1 1
1 6 8 8 0 2 6 5 7 0
0 6 6 7 5 4 8 6 4 5
1 8 7 2 5 0 / 5 2 9
```



98%



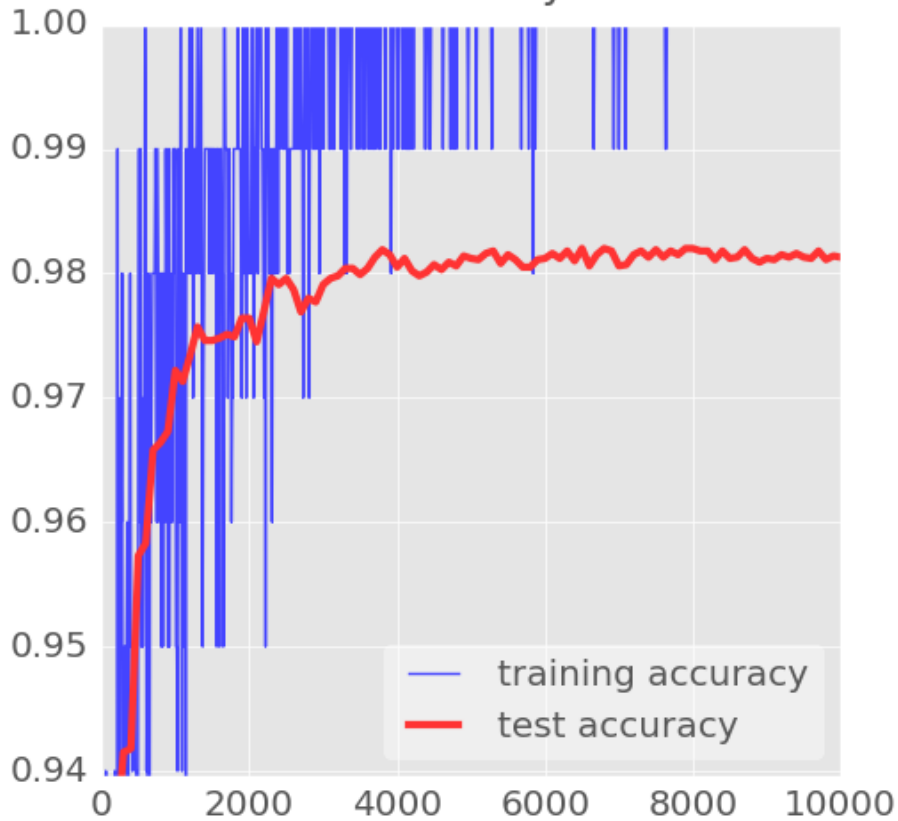
Dropout

Dropout

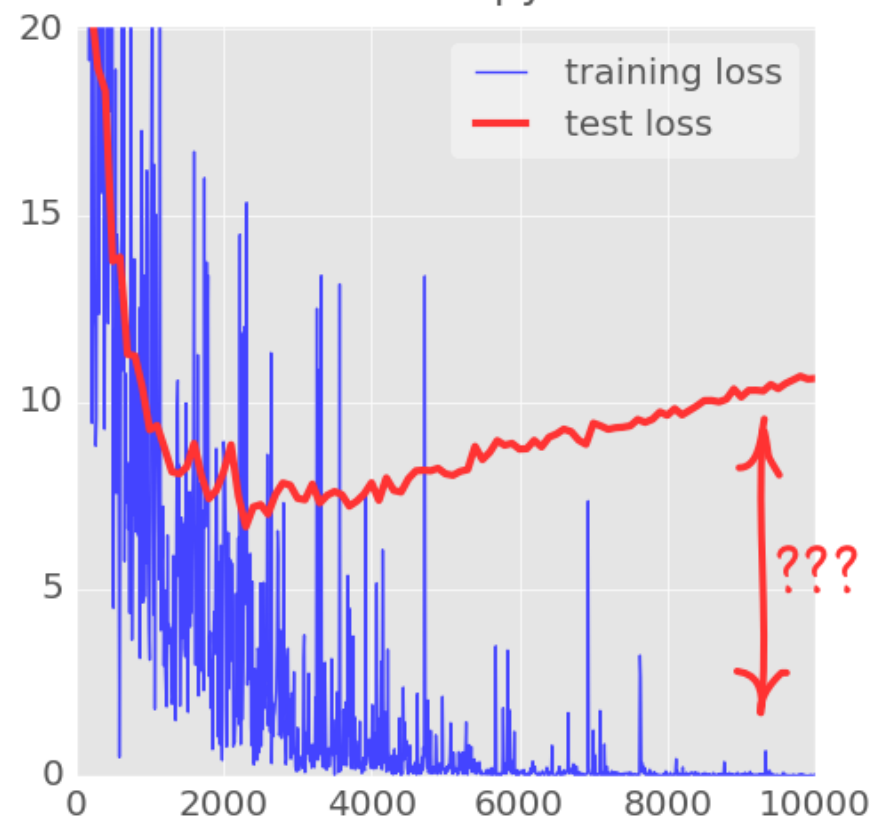


Overfitting

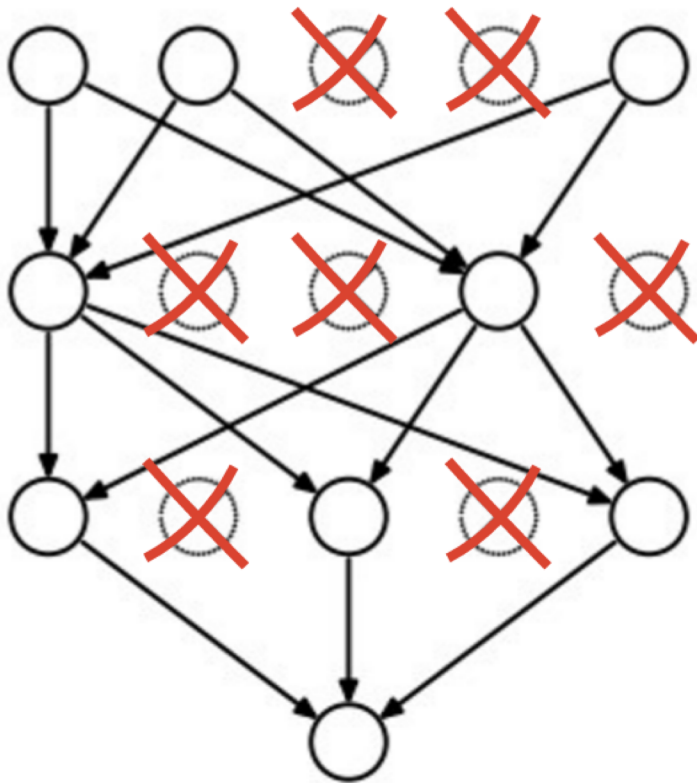
Accuracy



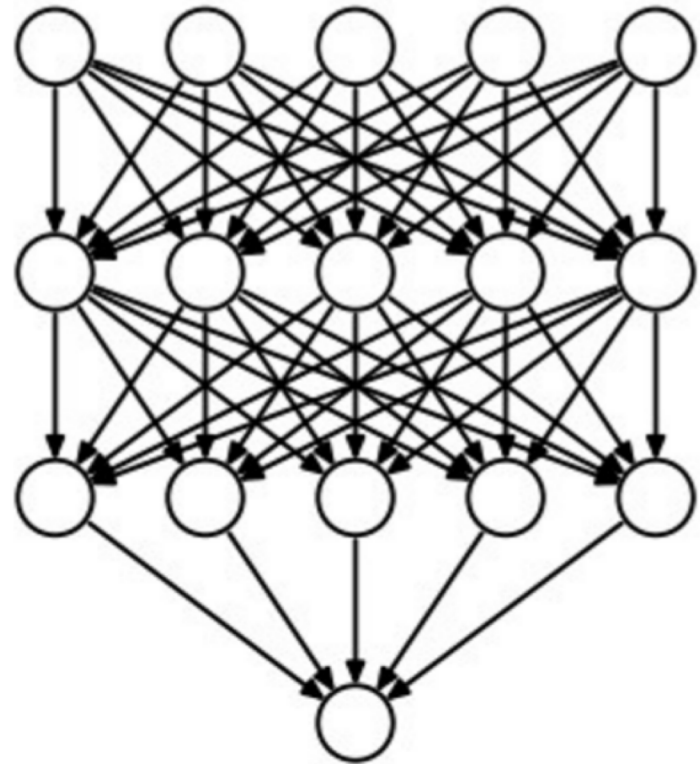
Cross entropy loss



Dropout



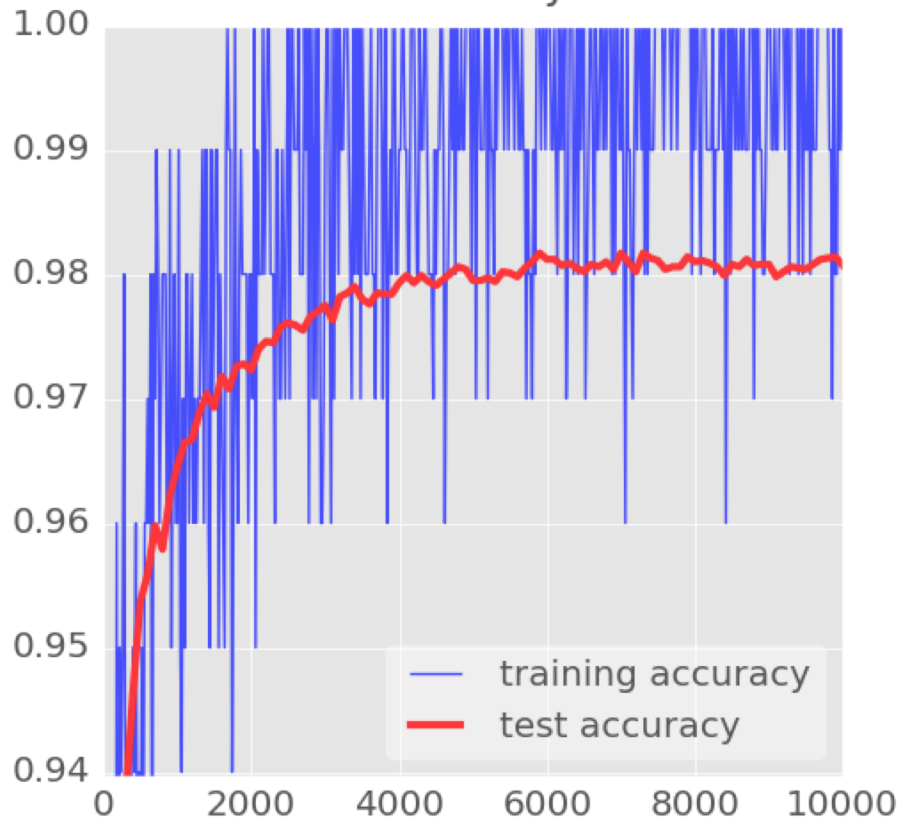
Training
 $p_{\text{keep}} = 0.75$



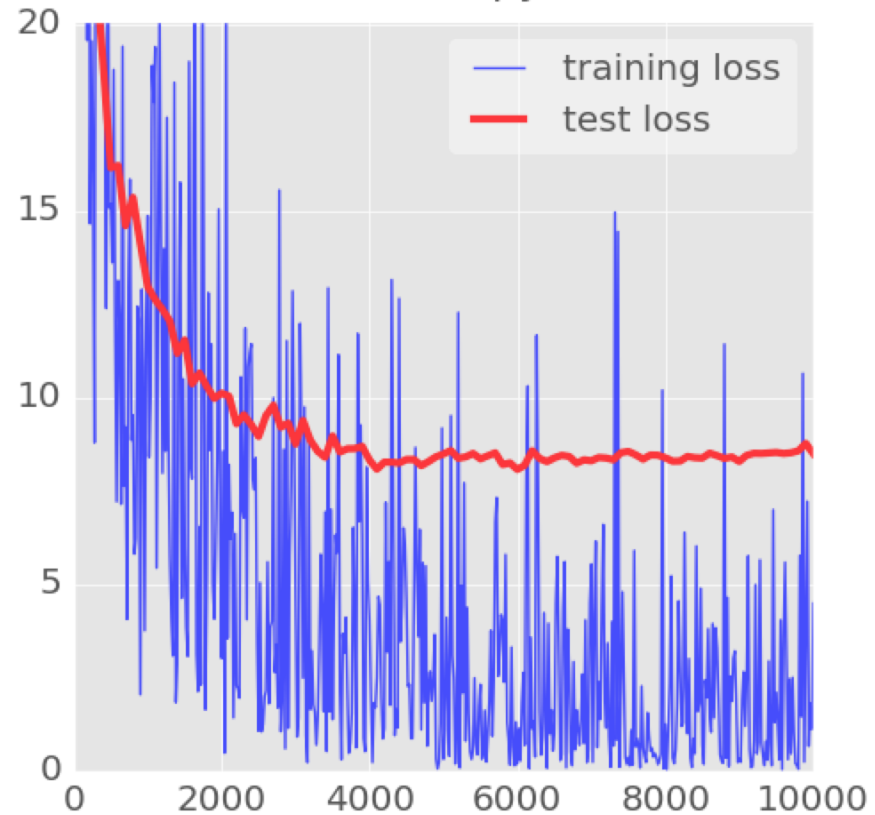
Test
 $p_{\text{keep}} = 1.0$

TensorFlow MNIST Tutorial

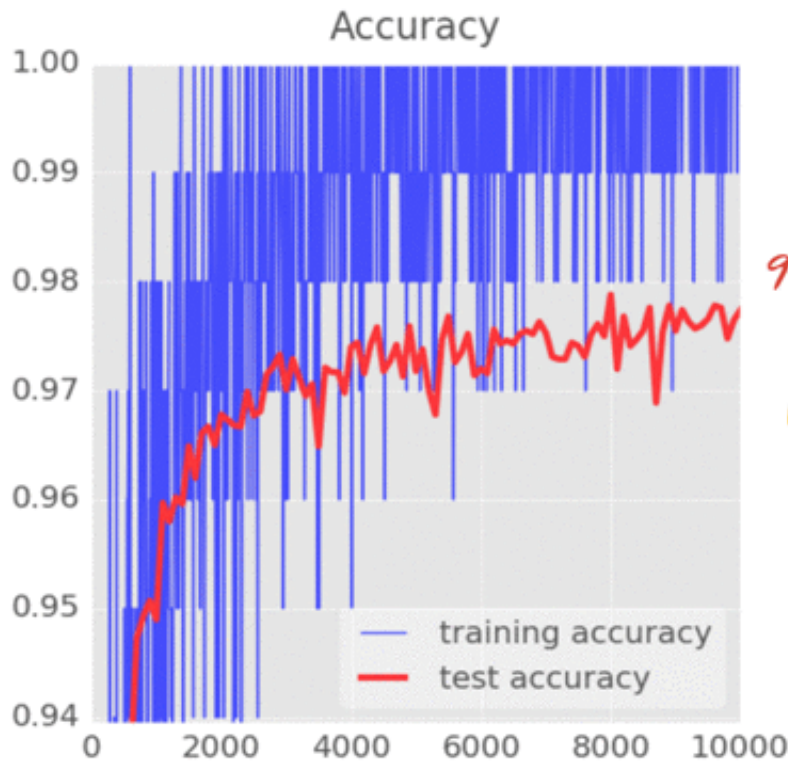
Accuracy



Cross entropy loss

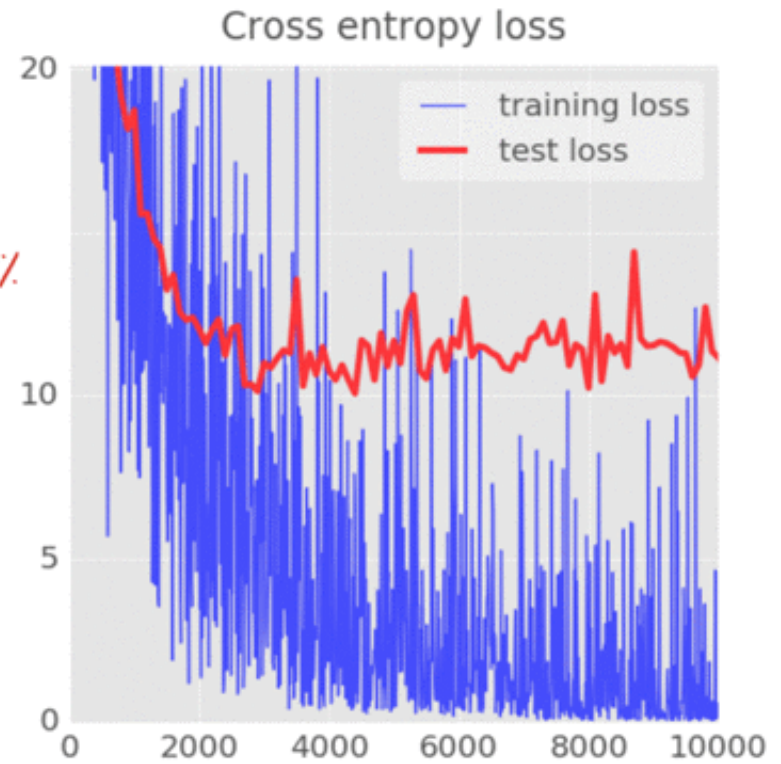


TensorFlow MNIST Tutorial



5 layers
sigmoid

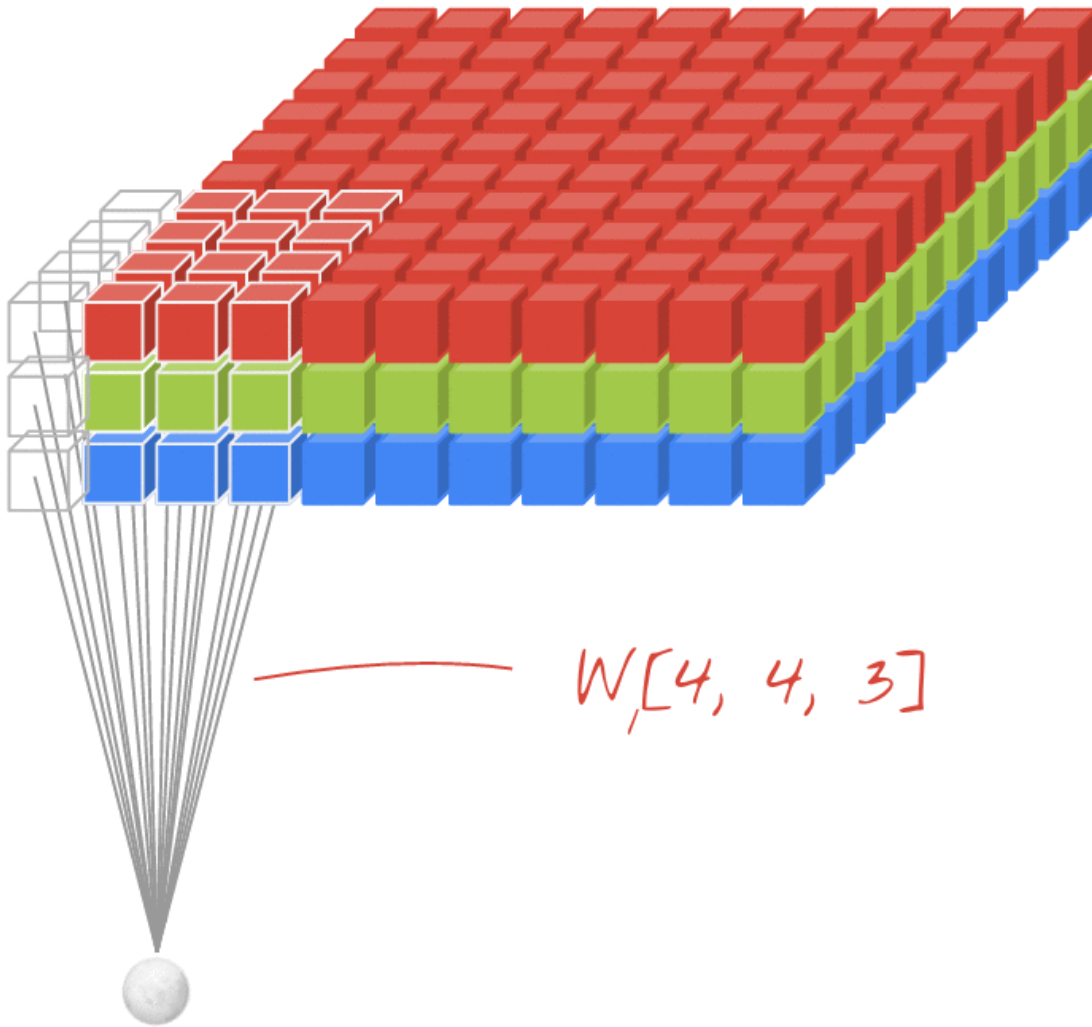
97.9%



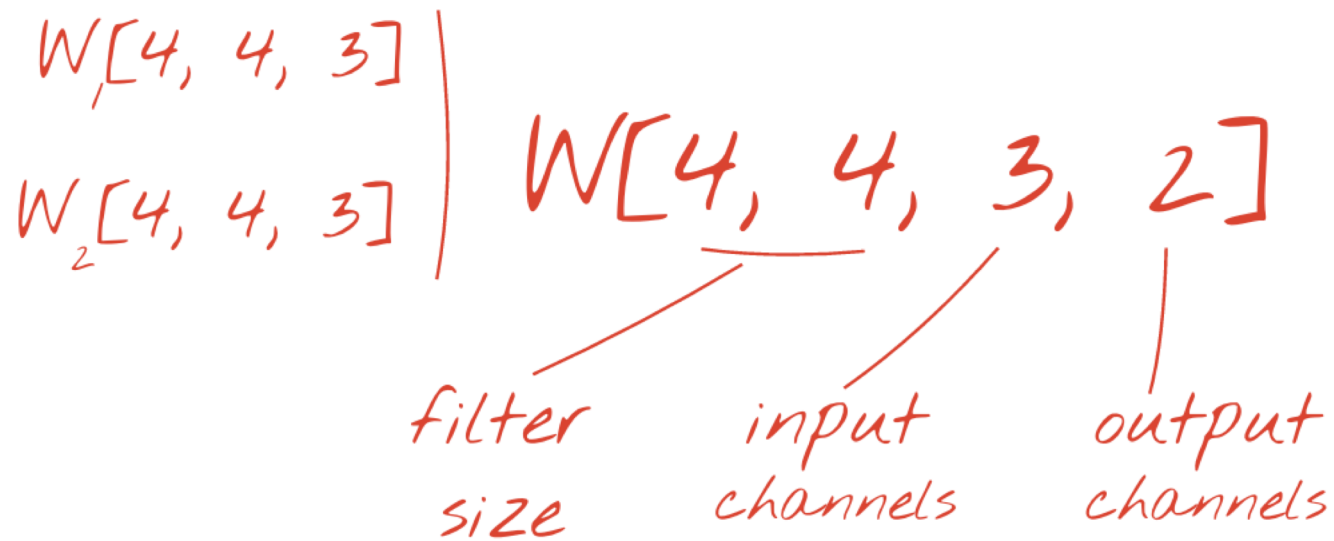
Overfitting



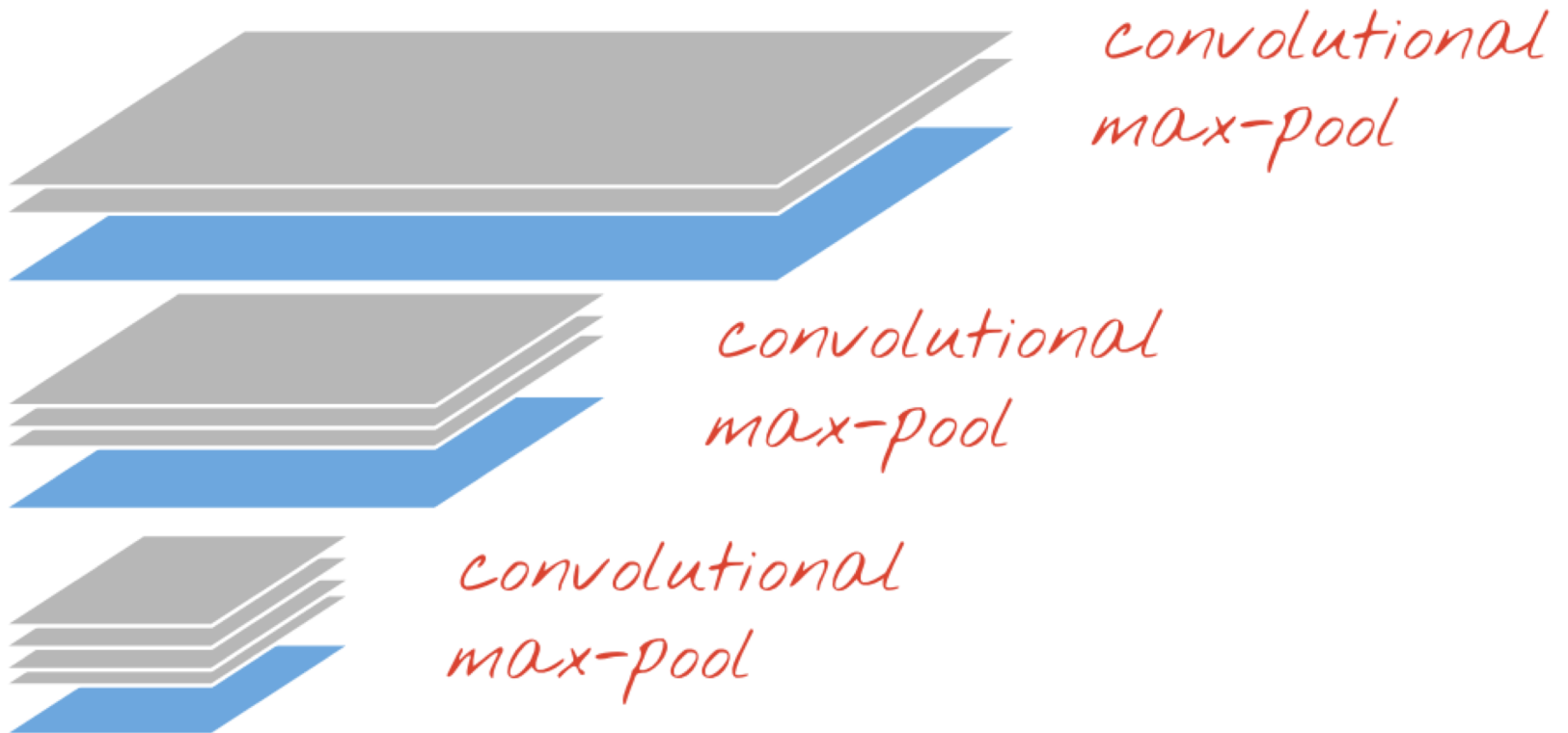
Convolutional Layer



Convolutional Layer



Convolutional Max-Pool



All Convolutional

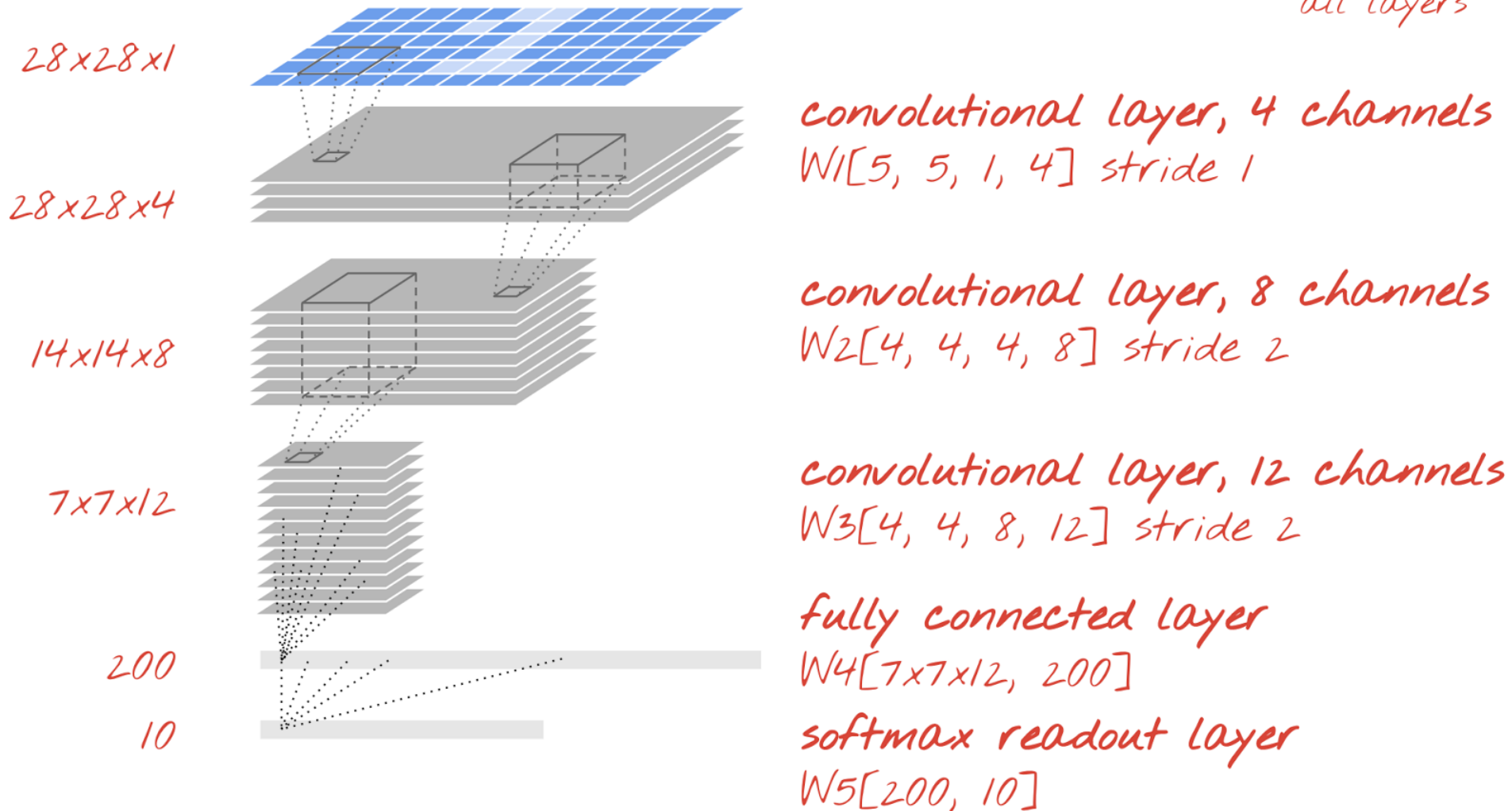
Hacker's tip



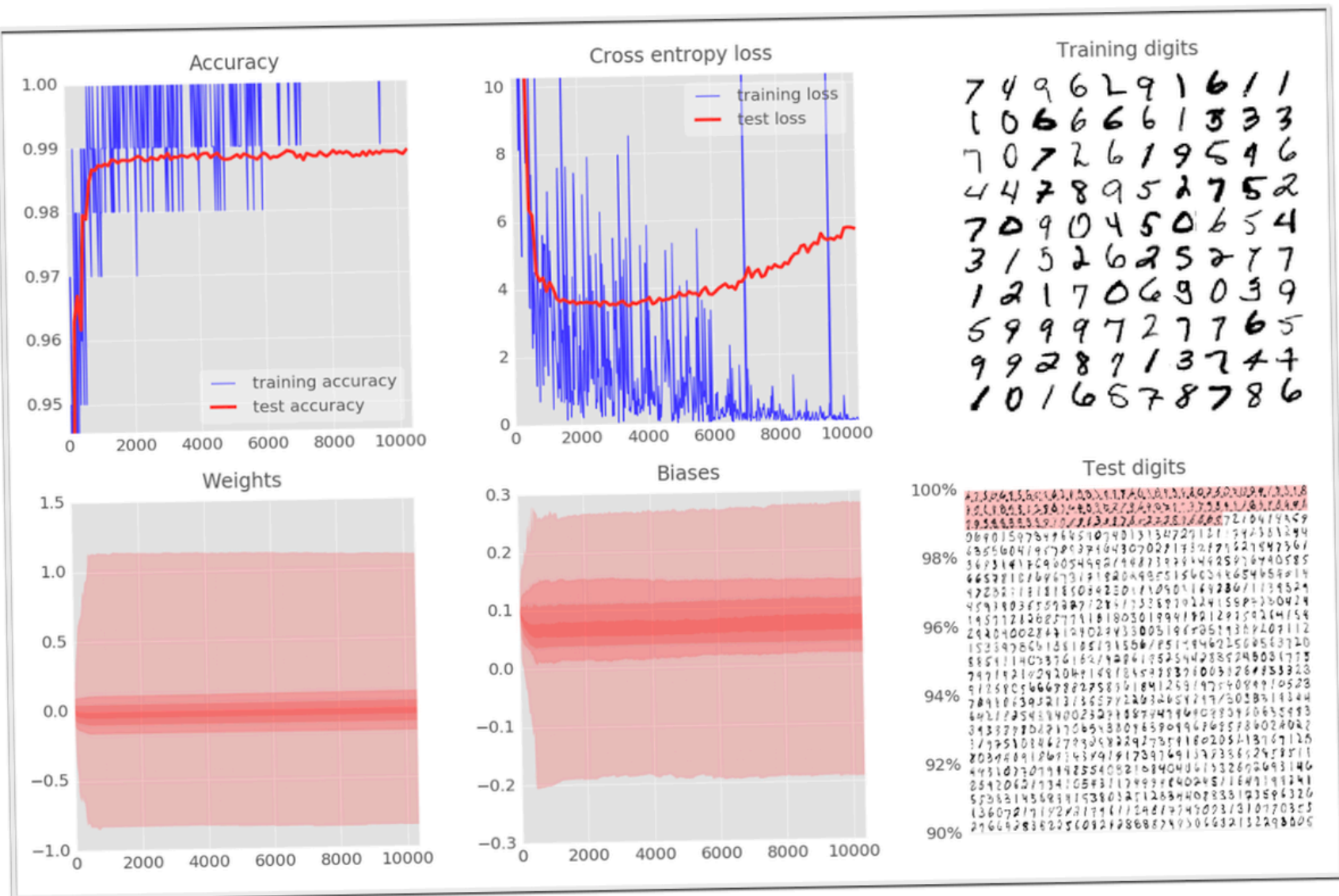
ALL
Convolutional

Bigger Convolutional Neural Network

+ biases on
all layers

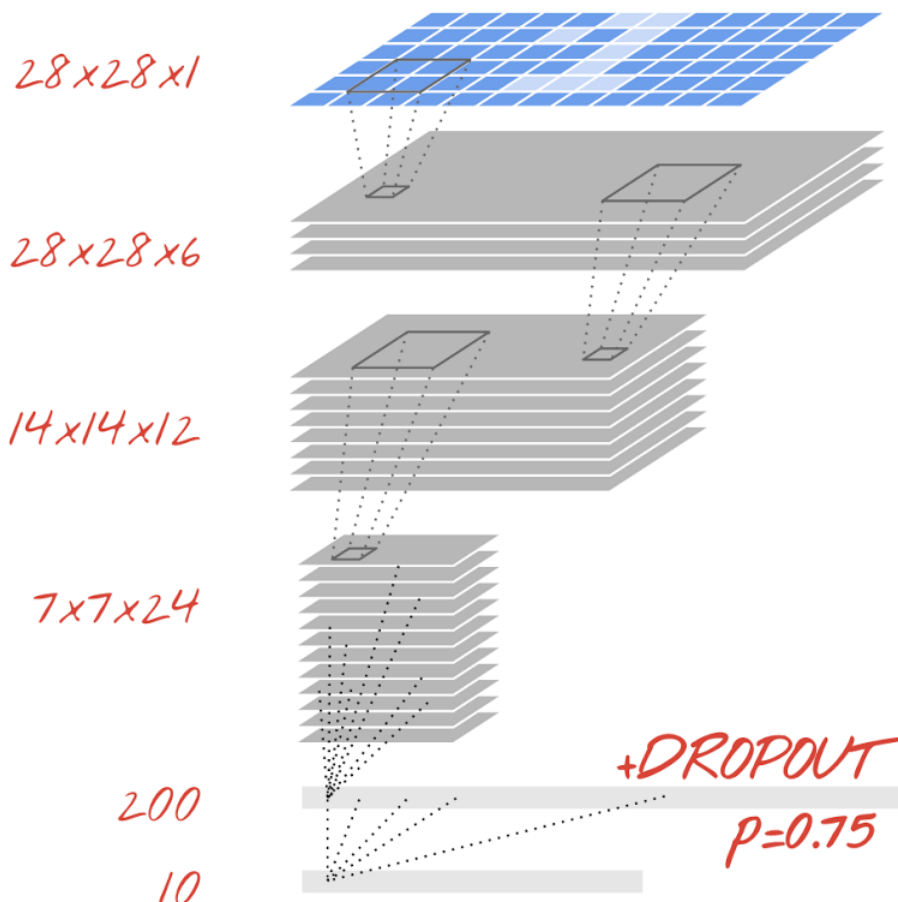


Bigger Convolutional Neural Network



Bigger Convolutional Neural Network + Dropout

+ biases on
all layers



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

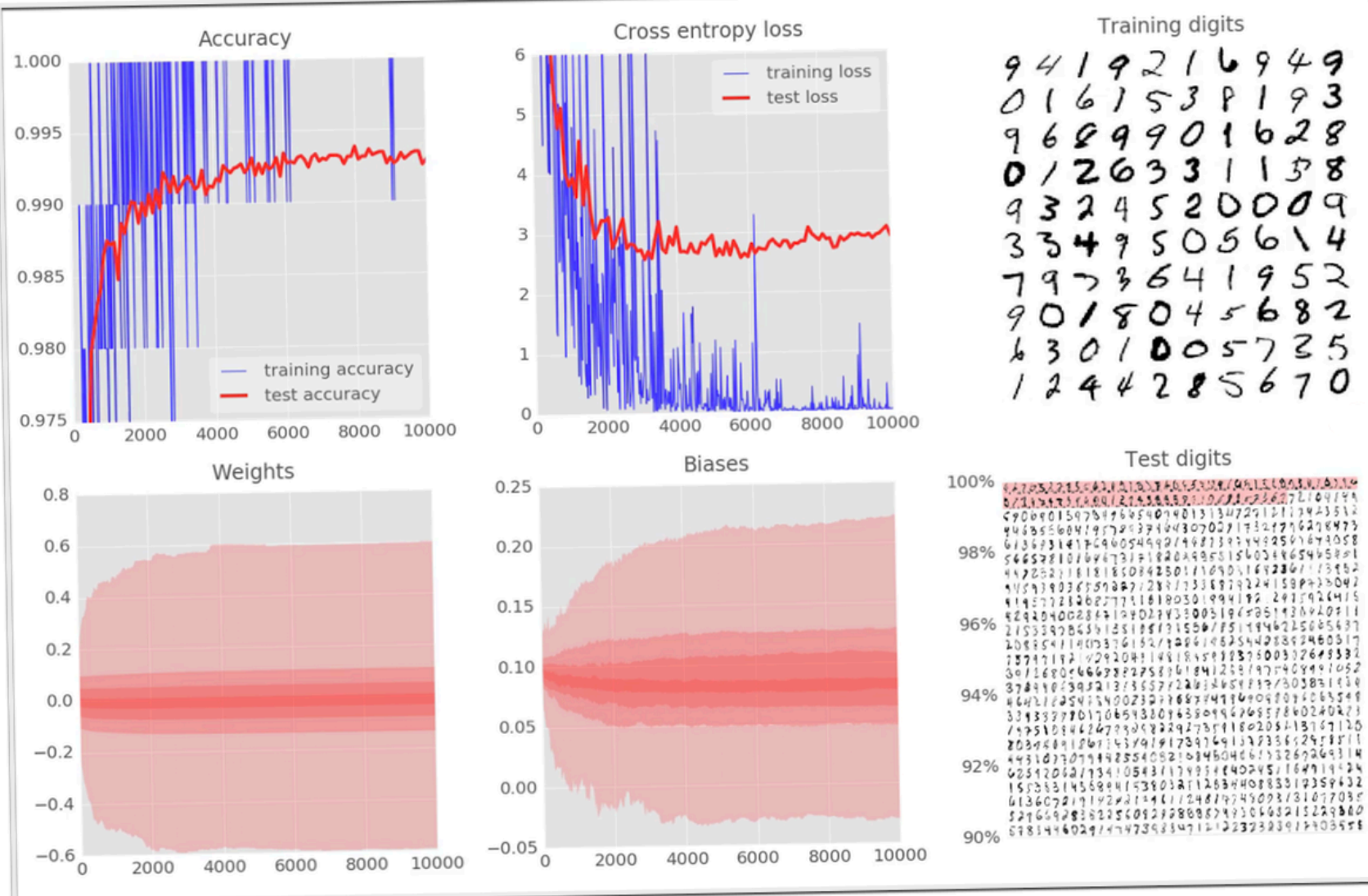
convolutional layer, 12 channels
 $W2[5, 5, 6, 12]$ stride 2

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

fully connected layer
 $W4[7x7x24, 200]$

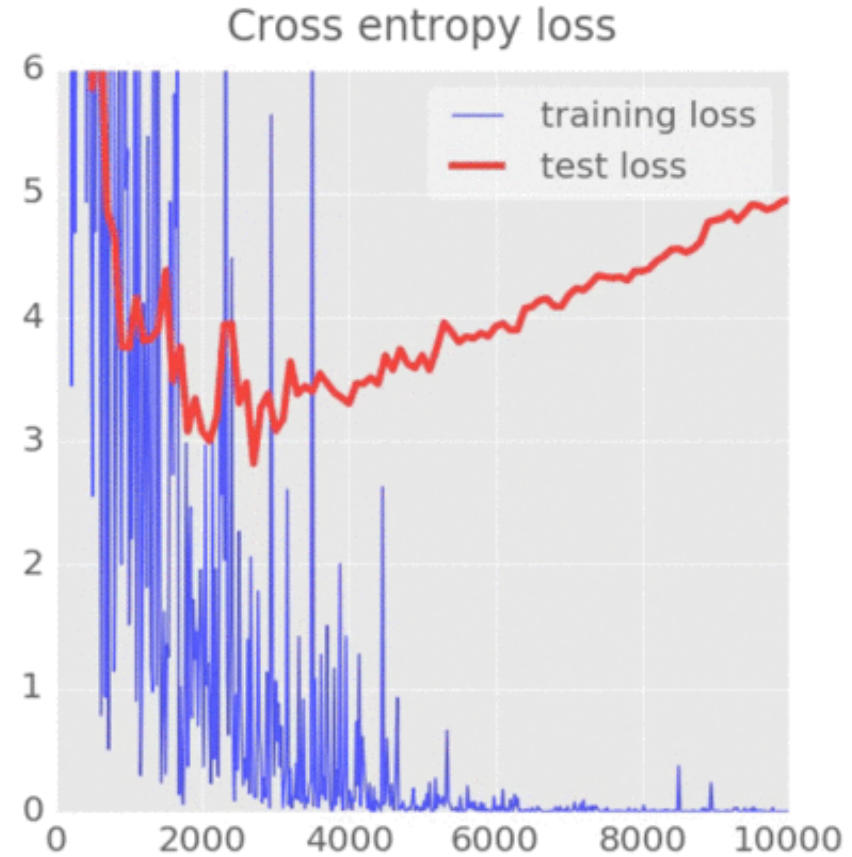
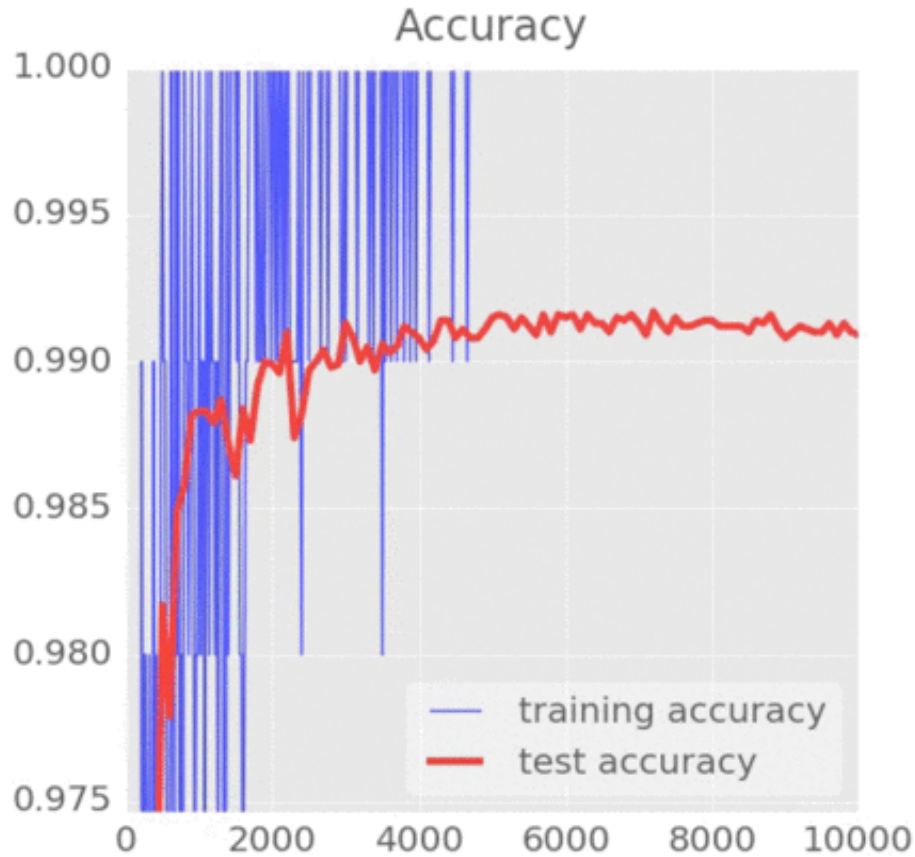
softmax readout layer
 $W5[200, 10]$

TensorFlow MNIST Tutorial



99.3!

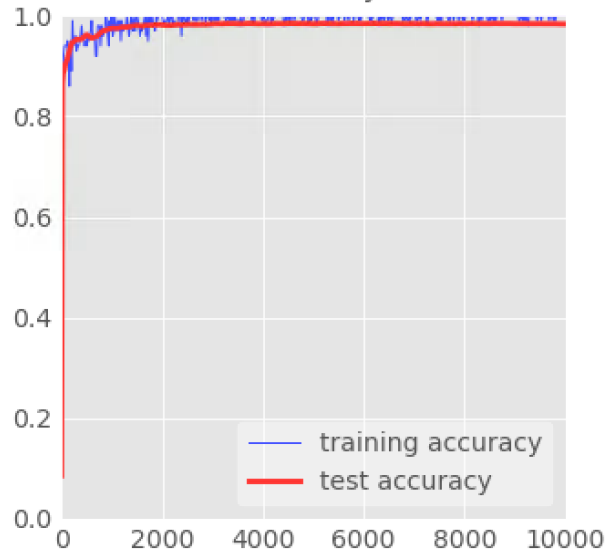
TensorFlow MNIST Tutorial



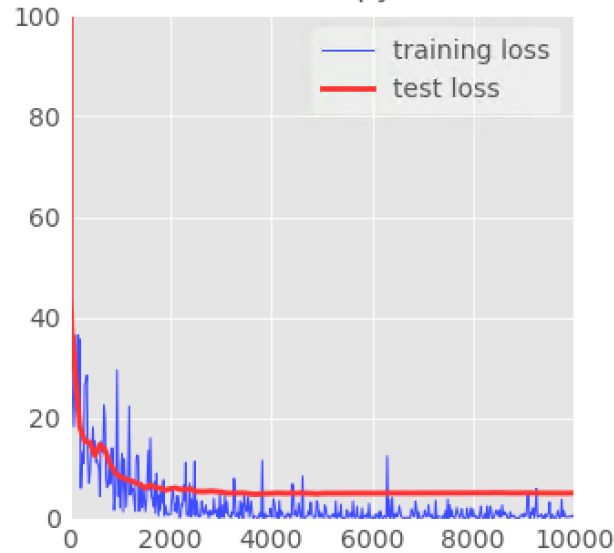
larger convolutional network

TensorFlow MNIST Tutorial

Accuracy



Cross entropy loss

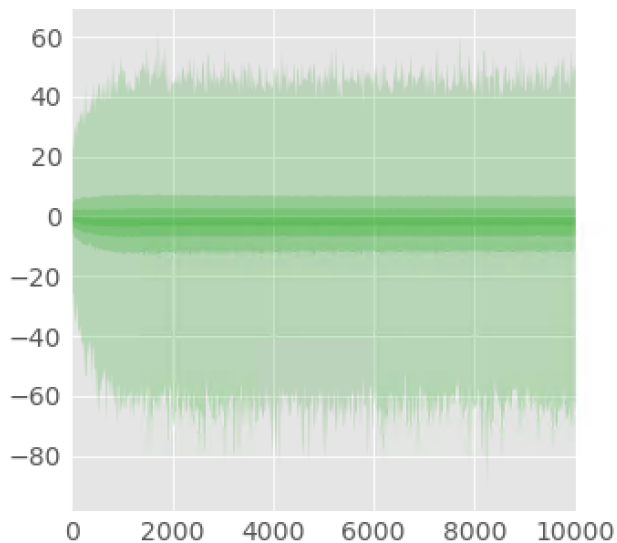


Training digits

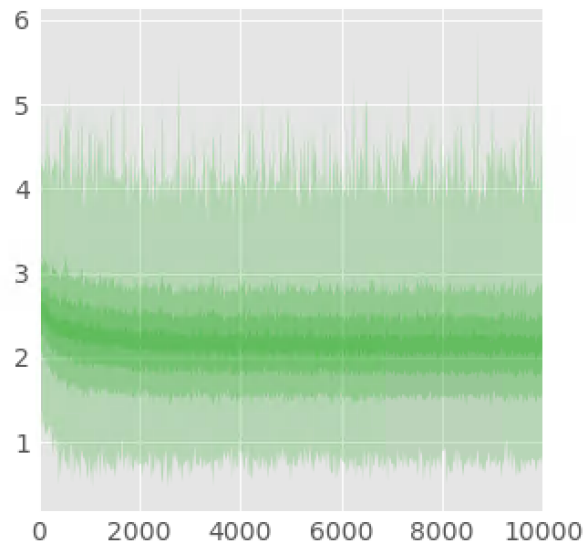
```

6 6 4 3 5 5 0 7 1 0
4 5 4 4 1 2 7 4 3 4
7 4 2 2 4 0 9 2 3 0
9 5 8 0 4 1 4 0 1 0
2 7 5 4 5 7 1 7 5 9
0 9 1 6 4 7 5 5 1 3
7 5 2 9 5 9 2 9 9 4
6 1 8 0 0 7 8 0 2 3
7 1 7 6 0 2 7 1 1 0
5 4 0 6 3 4 4 9 9 3
    
```

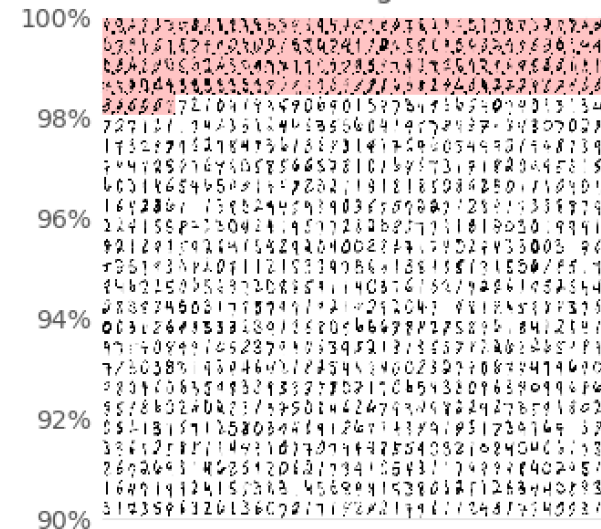
Logits



Max activations across batch



Test digits



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/19PJOJi1vn1kjcctlzNHjRSLbeVI4kd5z>

The screenshot shows a Google Colab notebook interface. At the top, the notebook title is 'tf01_basic_classification.ipynb'. Below the title are navigation options: 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right side, there are icons for 'COMMENT', 'SHARE', and a user profile icon. Below the navigation bar, there are tabs for '+ CODE', '+ TEXT', and 'CELL' (with up and down arrows). On the far right, there are 'CONNECT' and 'EDITING' options. The left sidebar contains a 'Table of contents' with the following items: 'Copyright 2018 The TensorFlow Authors.', 'Licensed under the Apache License, Version 2.0 (the "License");', 'MIT License', 'Train your first neural network: basic classification', 'Import the Fashion MNIST dataset', 'Explore the data', 'Preprocess the data', 'Build the model', 'Setup the layers', 'Compile the model', 'Train the model', 'Evaluate accuracy', and 'Make predictions'. The main content area shows a code cell with the following text: 'Copyright 2018 The TensorFlow Authors.', '↳ 2 cells hidden', and a section header 'Train your first neural network: basic classification'. Below the section header are three links: 'View on TensorFlow.org', 'Run in Google Colab', and 'View source on GitHub'. The text below the links reads: 'This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details, this is a fast-paced overview of a complete TensorFlow program with the details explained as we go. This guide uses [tf.keras](#), a high-level API to build and train models in TensorFlow.' Below this text is a code cell with the following code:

```
1 # memory footprint support libraries/code
2 !ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
3 !pip install gputil
4 !pip install psutil
5 !pip install humanize
6 import psutil
7 import humanize
8 import os
9 import GPUtil as GPU
10 GPUs = GPU.getGPUs()
11 gpu = GPUs[0]
12 def printm():
13     process = psutil.Process(os.getpid())
14     print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Pro
15     print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | Total {3:.0f}MB".format
16     printm()
```

Text Classification

IMDB Movie Reviews

https://colab.research.google.com/drive/1x16h1GhHsLlrLYtPCvCHaoO1W-i_gror

The screenshot shows a Google Colab notebook titled "tf02_basic-text-classification.ipynb". The interface includes a top navigation bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help" menus. On the right, there are "COMMENT", "SHARE", and a user profile icon. Below the navigation bar, there are buttons for "+ CODE", "+ TEXT", "↑ CELL", and "↓ CELL". A "CONNECT" dropdown and "EDITING" button are also visible.

The left sidebar contains a "Table of contents" with the following items:

- Copyright 2018 The TensorFlow Authors.
- Licensed under the Apache License, Version 2.0 (the "License");
- MIT License
- Text classification with movie reviews**
- Download the IMDB dataset
- Explore the data
 - Convert the integers back to words
- Prepare the data
- Build the model
 - Hidden units
 - Loss function and optimizer
- Create a validation set
- Train the model
- Evaluate the model

The main content area shows the following sections:

- Copyright 2018 The TensorFlow Authors.**
 - ↳ 2 cells hidden
- Text classification with movie reviews**

Below the section header, there are three links: "View on TensorFlow.org", "Run in Google Colab", and "View source on GitHub".

The text content reads:

This notebook classifies movie reviews as *positive* or *negative* using the text of the review. This is an example of *binary*—or two-class—classification, an important and widely applicable kind of machine learning problem.

We'll use the [IMDB dataset](#) that contains the text of 50,000 movie reviews from the [Internet Movie Database](#). These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are *balanced*, meaning they contain an equal number of positive and negative reviews.

This notebook uses [tf.keras](#), a high-level API to build and train models in TensorFlow. For a more advanced text classification tutorial using `tf.keras`, see the [MLCC Text Classification Guide](#).

The code cell shows the following code:

```
1 # memory footprint support libraries/code
2 !ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
3 !pip install gputil
4 !pip install psutil
5 !pip install humanize
6 import psutil
7 import humanize
8 import os
9 import GPUtil as GPU
10 GPUs = GPU.getGPUs()
11 gpu = GPUs[0]
12 def printm():
13     process = psutil.Process(os.getpid())
```

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

tf03_basic-regression.ipynb ☆

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CONNECT EDITING

Table of contents Code snippets Files X

▶ 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.

```
1 # memory footprint support libraries/code
2 !ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
3 !pip install gputil
4 !pip install psutil
5 !pip install humanize
6 import psutil
7 import humanize
8 import os
9 import GPUtil as GPU
10 GPUs = GPU.getGPUs()
11 gpu = GPUs[0]
12 def printm():
13     process = psutil.Process(os.getpid())
14     print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Proc size: "
15         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-ot8eFxlKbGwJvXlrjtc>
- run3models.ipynb
 - https://colab.research.google.com/drive/1HQ1GrlqQUUPCct7_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-OI7ZB0MmU>
- 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=fTUwdXUffl8>
- 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=PLiaHhY2iBX9hdHaRr6b7XevZtgZRra1PoU>
- Deep Learning SIMPLIFIED, <https://www.youtube.com/playlist?list=PLjJh1vISEYgvGod9wWiydumYl8hOXixNu>
- 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=llg3gGewQ5U>
- TensorFlow: <https://www.tensorflow.org/>
- Keras: <http://keras.io/>
- Udacity, Deep Learning, https://www.youtube.com/playlist?list=PLAwXtw4SYaPn_OWPFT9uIXLuQrImzHfOV
- <http://p.migdal.pl/2017/04/30/teaching-deep-learning.html>
- <https://github.com/leriomaggio/deep-learning-keras-tensorflow>