人工智能和聊天机器人

AI | 人工智能

 | 人工神经网络

 | 通过MNIST数据集了解机器学习

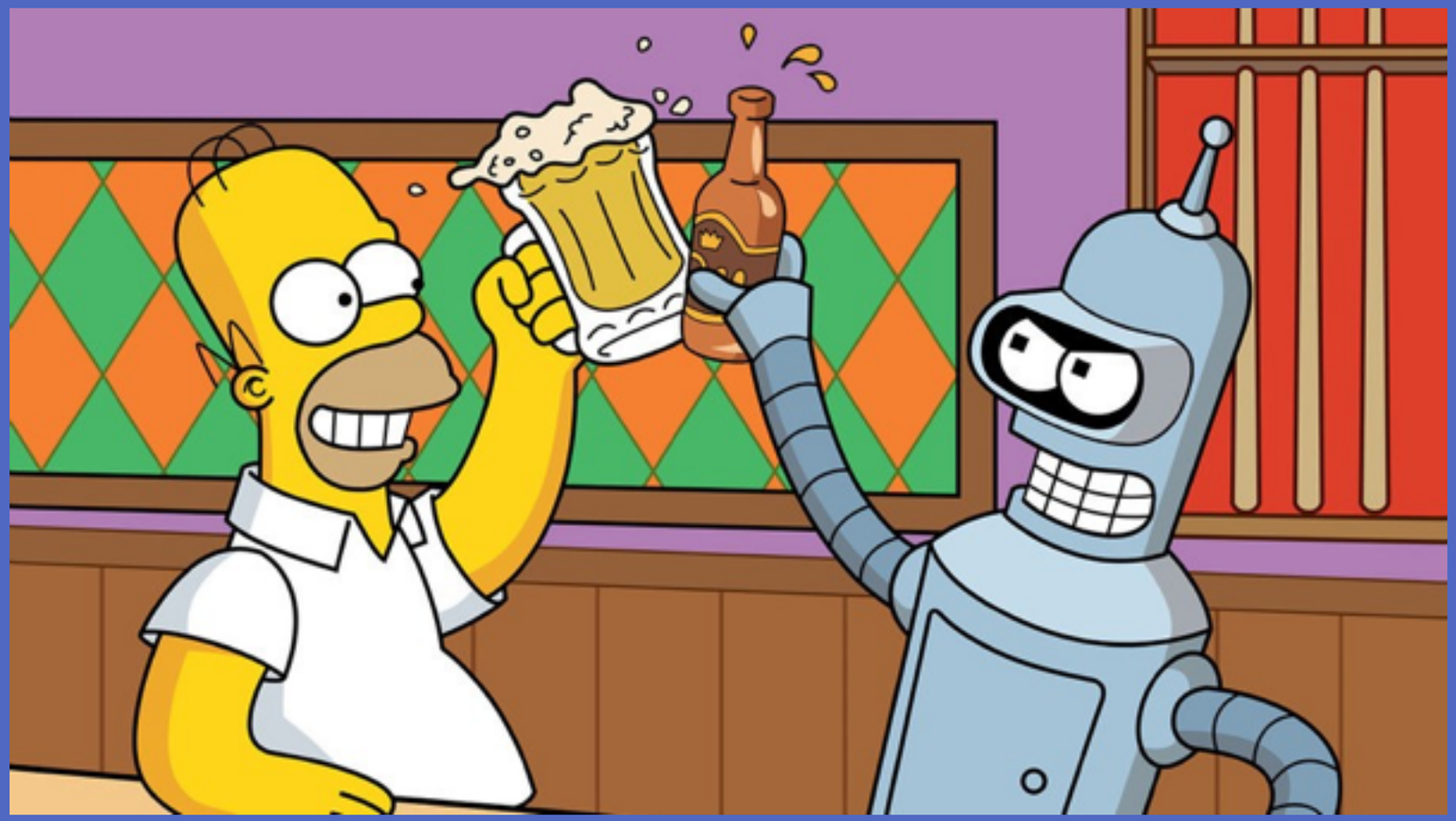
 | Retrieval Chatbot on Tensorflow

 | Botframework和LUIS

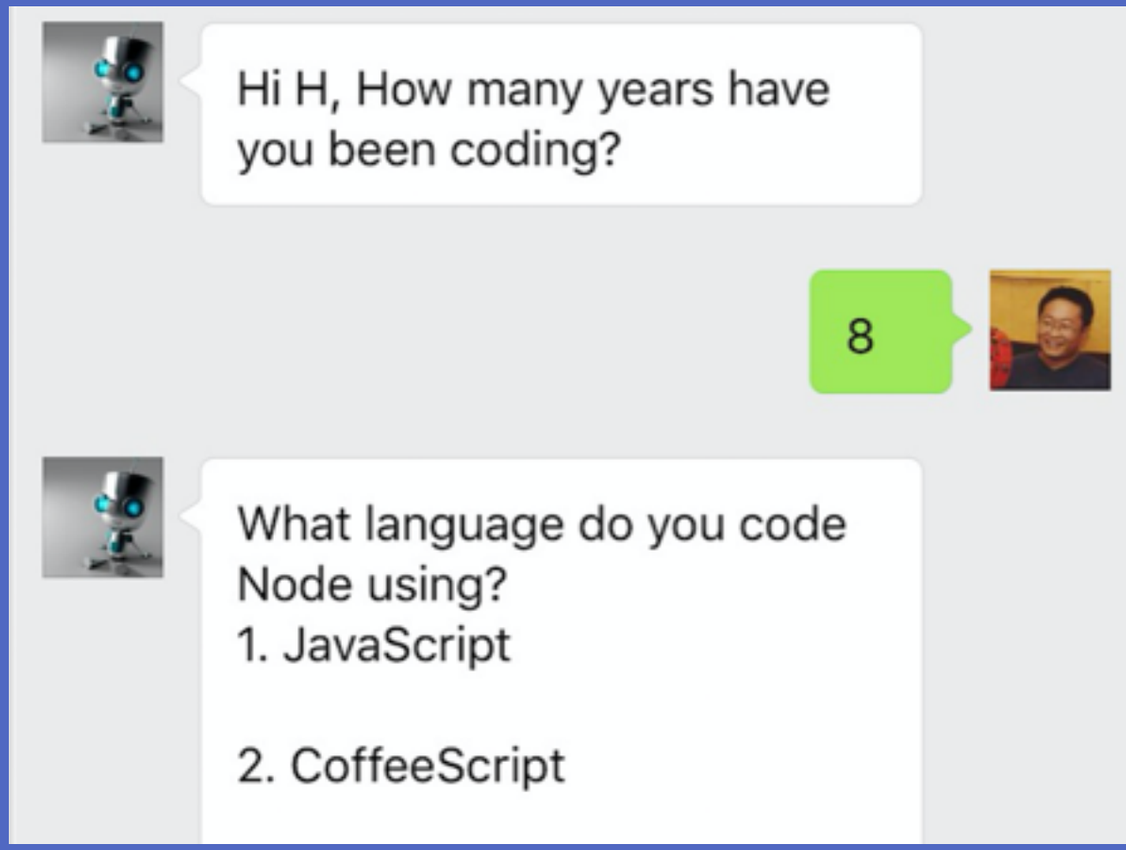
hain

https://rockq.org/user/hain

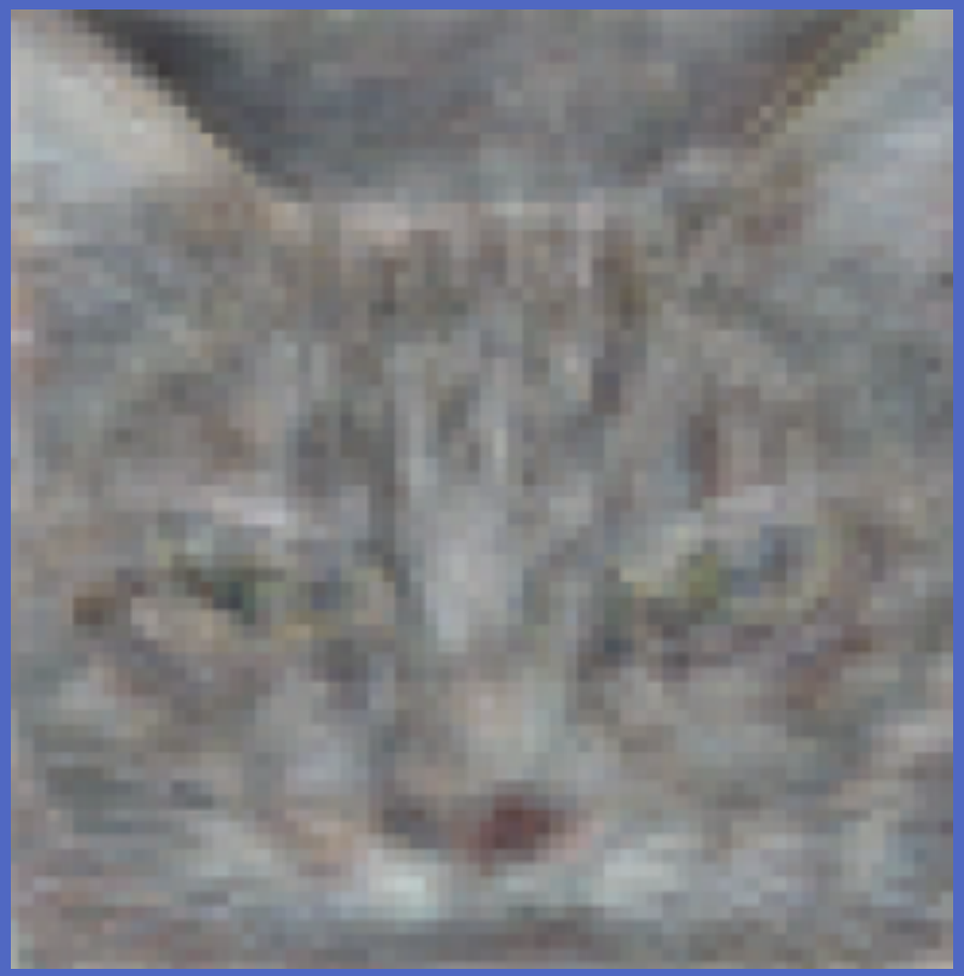
2016/07/20




Google
谷歌
Google 搜索
手气不错
Google.com.hk 使用下列语言： 中文（繁體） English


Hi H, How many years have you been coding?
8
What language do you code Node using?
1. JavaScript
2. CoffeeScript

AI

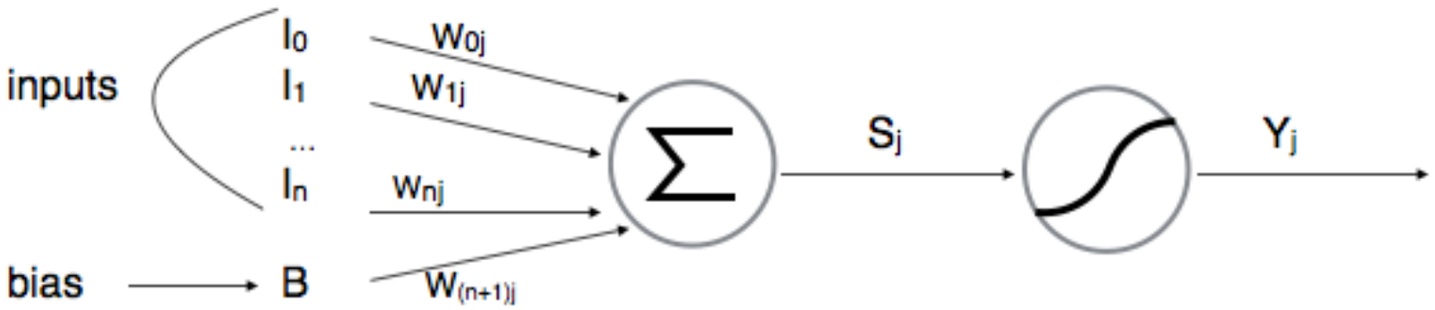


I am a botman !

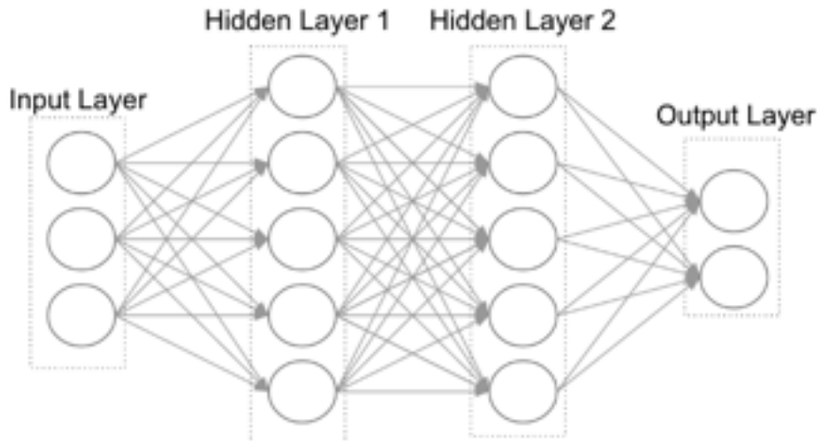
IBM Watson API / AlchemyAPI

腾讯优图

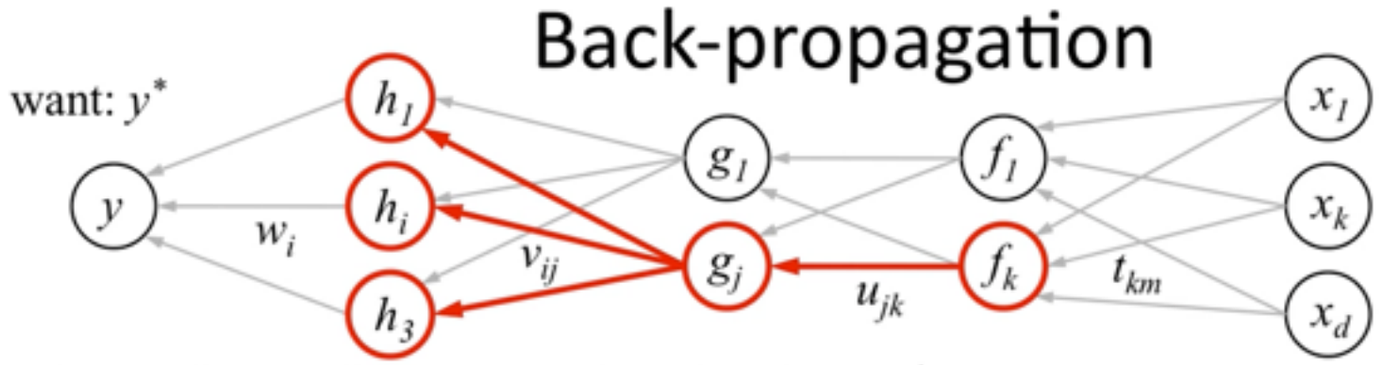
科大讯飞 / 云知声



神经元



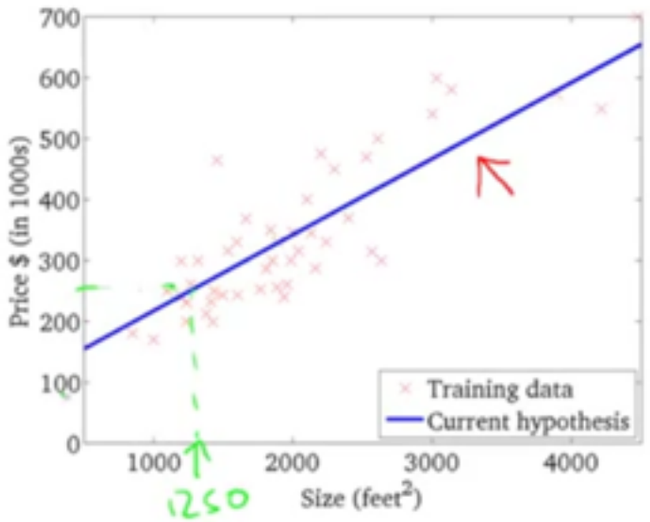
前向传播网络



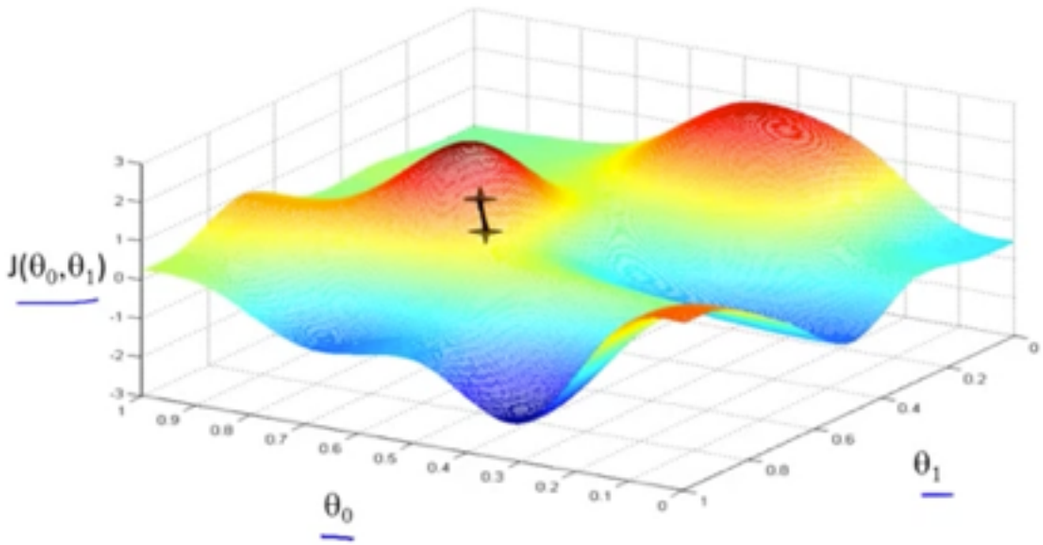
反向传播网络

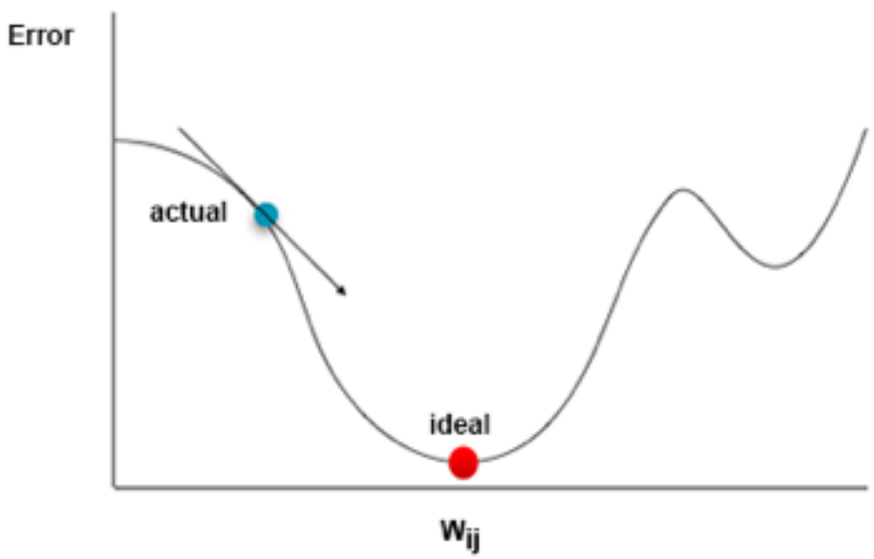
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$y = \theta 0\ x + \theta 1$





Gradient descent algorithm

repeat until convergence {

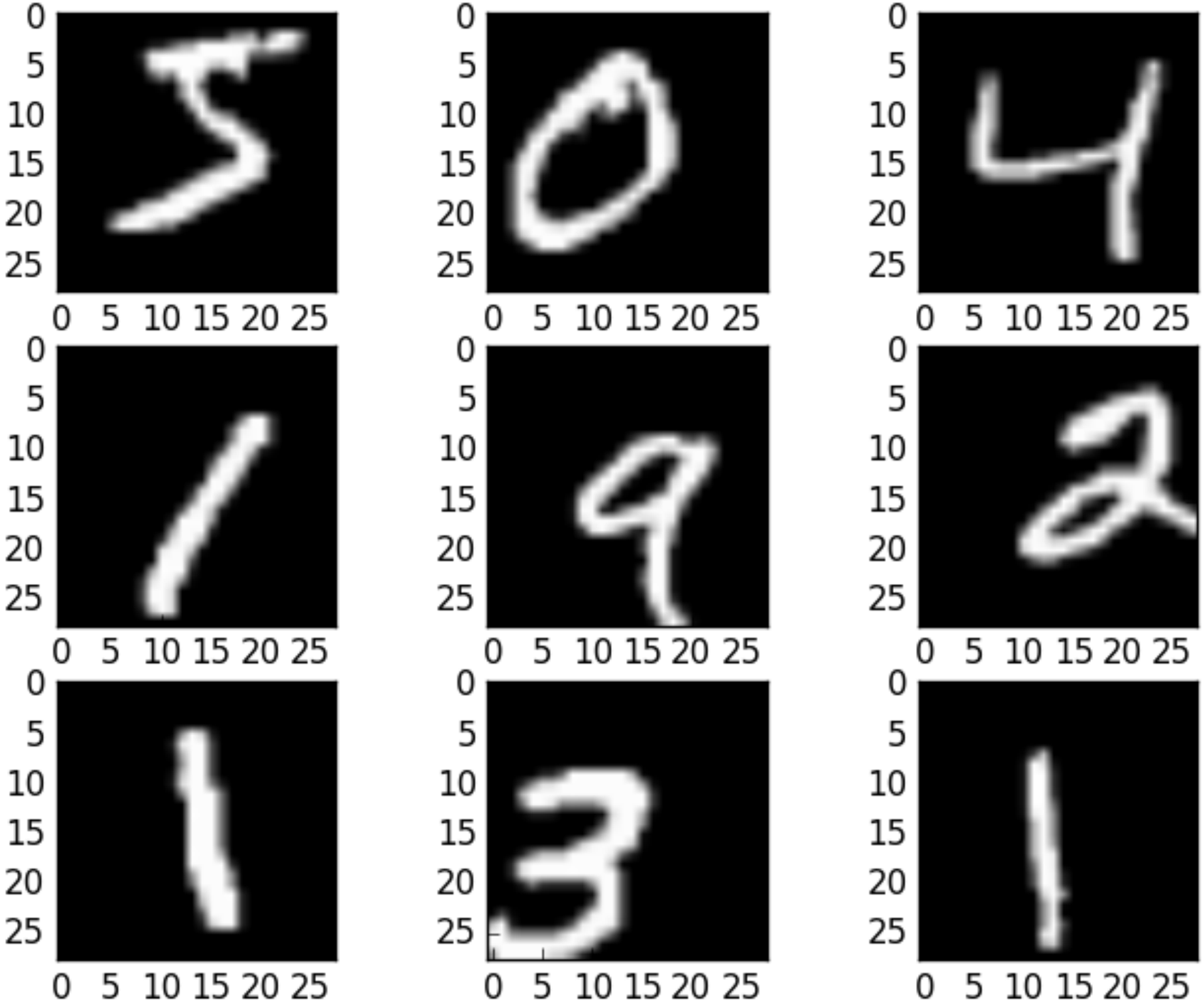
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \text{(for } j = 0 \text{ and } j = 1\text{)}$$

}

Correct: Simultaneous update

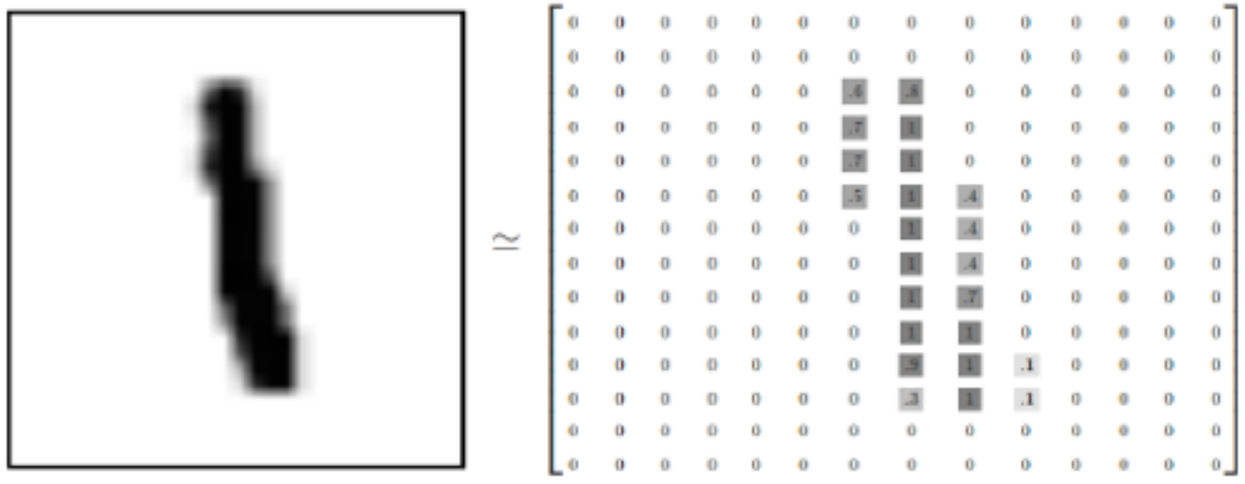
$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

梯度下降原理

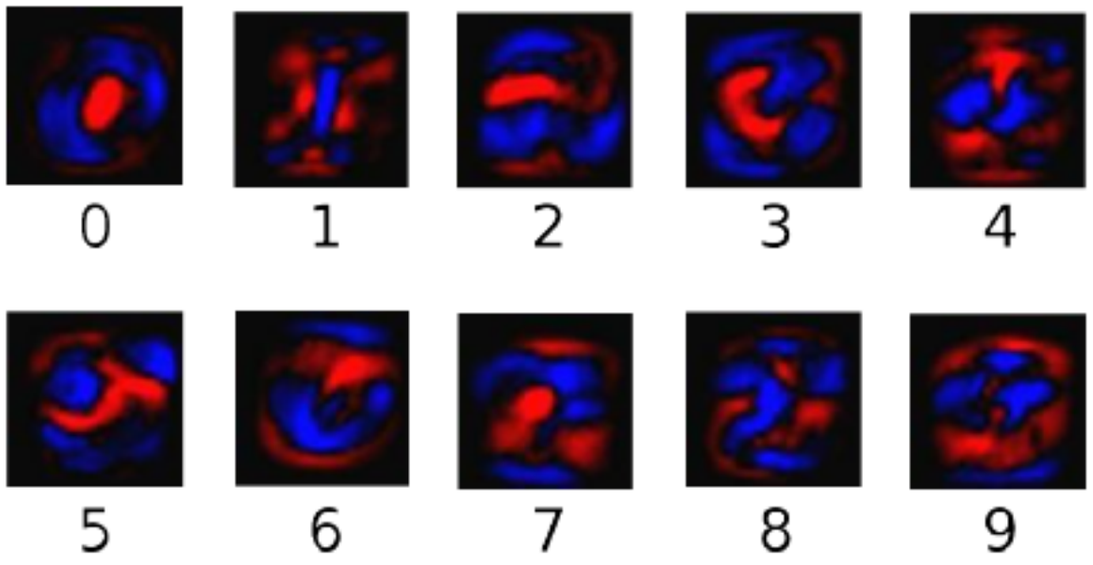


http://yann.lecun.com/exdb/mnist/

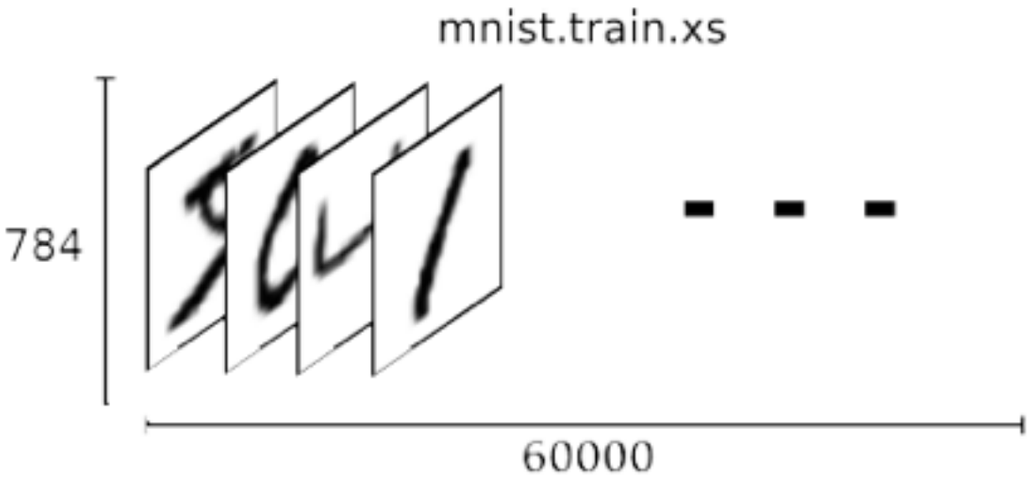
MNIST数据集



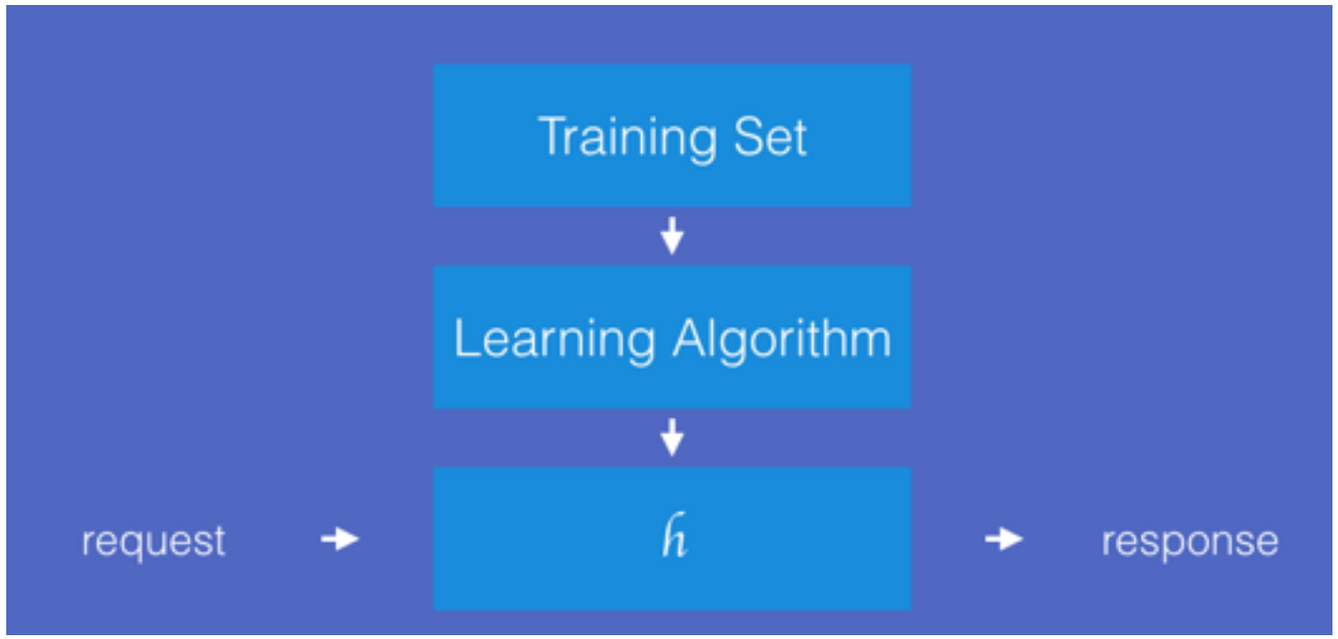
(1)



(4)



(2)



(3)

解决MNIST任务的基本原理

https://github.com/rockq-org/mnist-nnm

git clone git@github.com:rockq-org/mnist-nnm.git

cd mnist-nnm/src && npm install && npm test

```
// Get neural network model.
var nnm = new require('mnist-nnm').Model();
// Setup MNIST Dataset.
var mnist = require('mnist');
var set = mnist.set(700, 20);
// Train neural network.
nnm.train(set.trainingSet, options);

// Validate with test set.
set.testSet.forEach(function(val, index) {
    nnm.validate(val);
});
```

https://github.com/rockq-org/mnist-nnm

git clone git@github.com:rockq-org/mnist-nnm.git

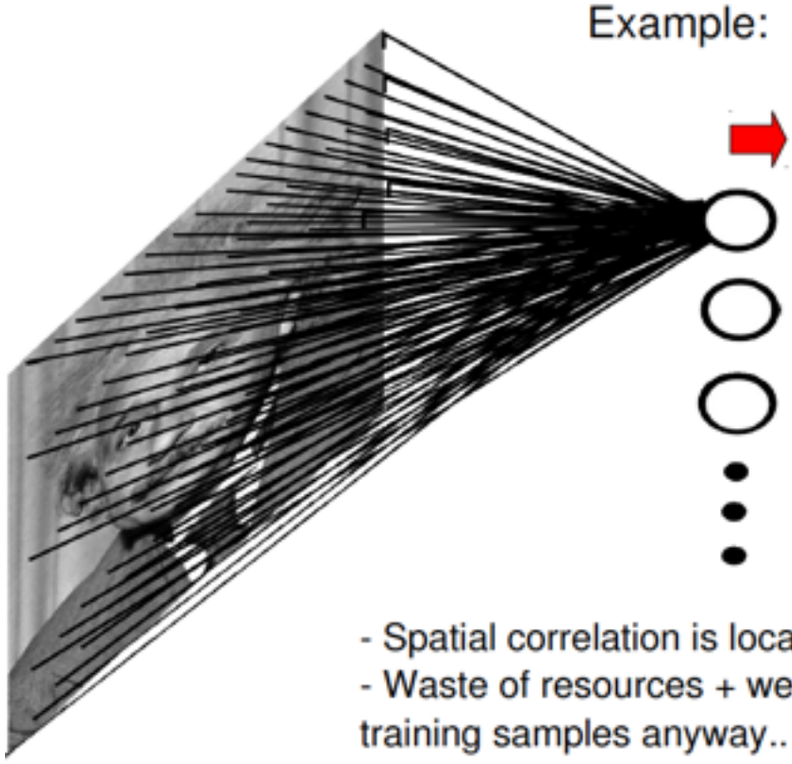
cd mnist-nnm/src && npm install && npm test

```
> mnist-nnm@1.0.0 test /Users/hain/ai/node-mnist-sample/src
> ava "--watch"

[2016-07-22 11:09:23.569] [INFO] nn.test - start to train {"rate":0.1,"iterations":20,"error":0.01
iterations 1 error 6.845912501699674 rate 0.1
iterations 2 error 4.504152991349747 rate 0.1
iterations 3 error 3.028312731669452 7 rate 0.1
iterations 4 error 1.853700781831871 rate 0.1
iterations 5 error 1.3059480795633154 rate 0.1
iterations 6 error 0.9825529508504853 rate 0.1
iterations 7 error 0.7138775146444608 rate 0.1
iterations 8 error 0.6003610502375651 rate 0.1
iterations 9 error 0.4802941631822216 rate 0.1
iterations 10 error 0.34586961128024746 rate 0.1
iterations 11 error 0.3836775495915907 rate 0.1
iterations 12 error 0.36238049755953 99 rate 0.1
iterations 13 error 0.2243141425155864 rate 0.1
iterations 14 error 0.2053262381883071 rate 0.1
```

使用Node.js训练人工神经网络

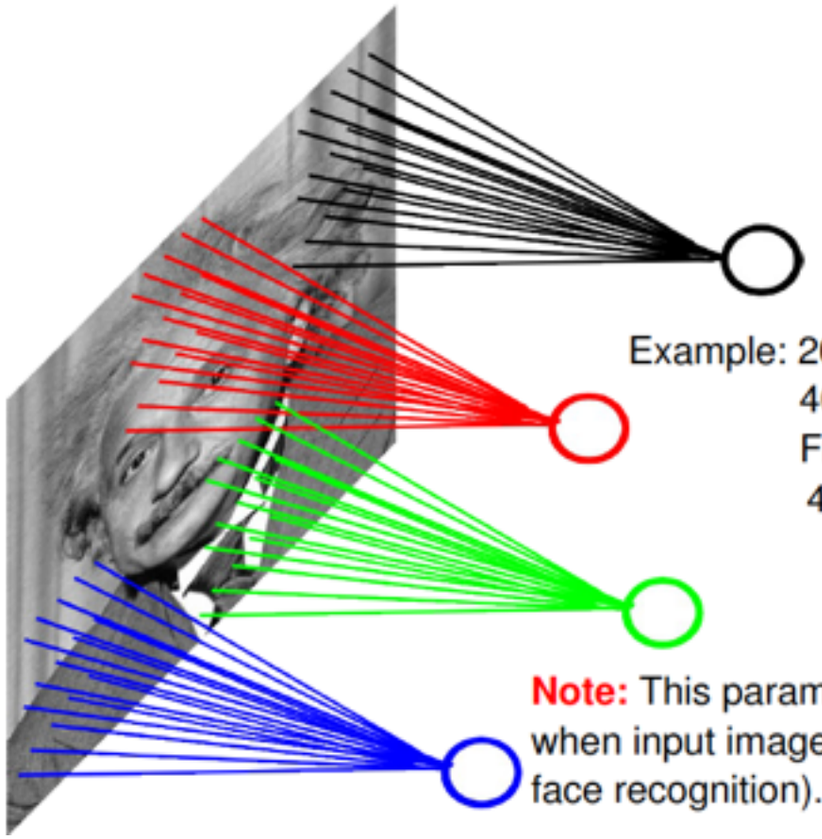
Fully Connected Layer



Example: 200x200 image
40K hidden units
~2B parameters!!!

- Spatial correlation is local
- Waste of resources + we have not enou
training samples anyway..

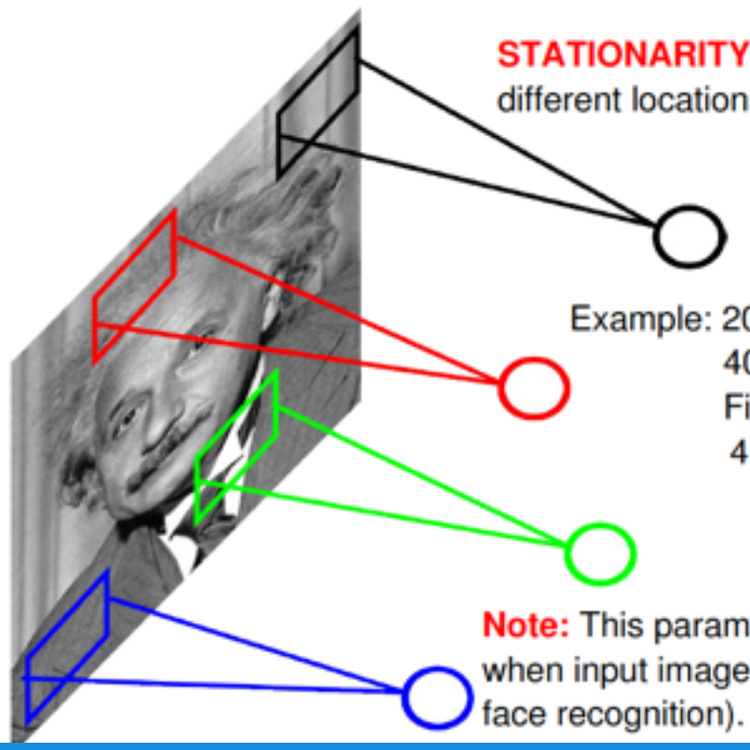
Locally Connected Layer



Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Locally Connected Layer

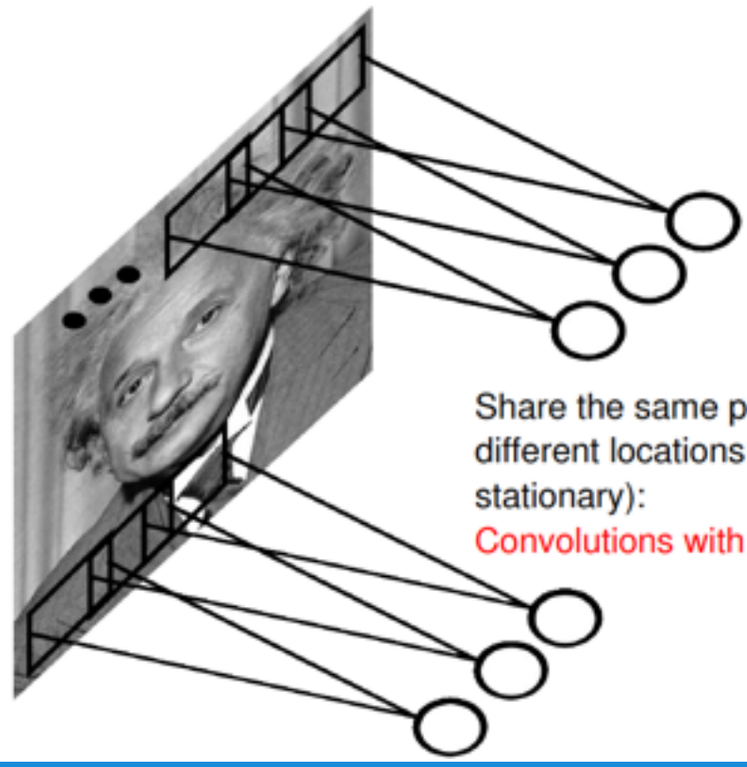


STATIONARITY? Statistics is similar at different locations

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

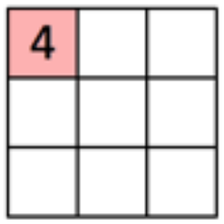
Convolutional Layer



Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

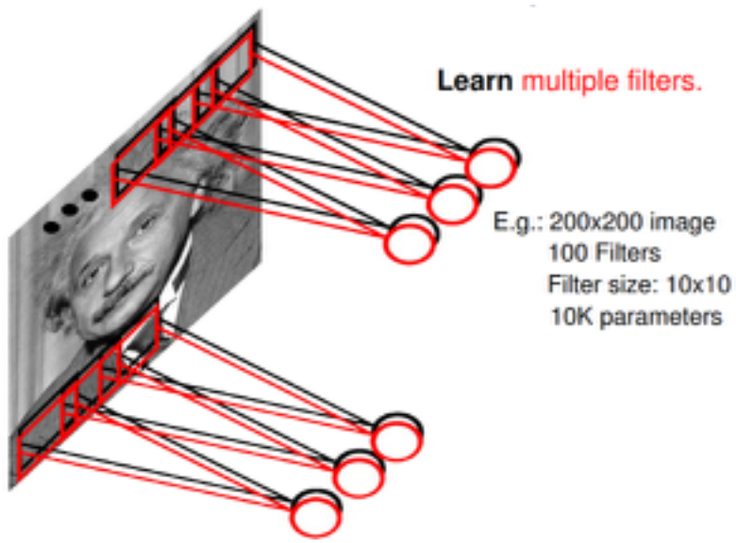
卷积

Image

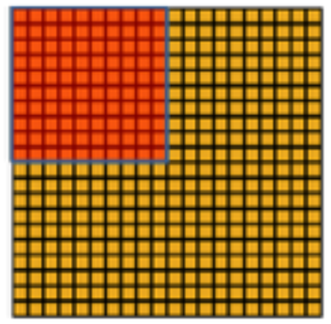


Convolved Feature

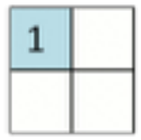
特征提取



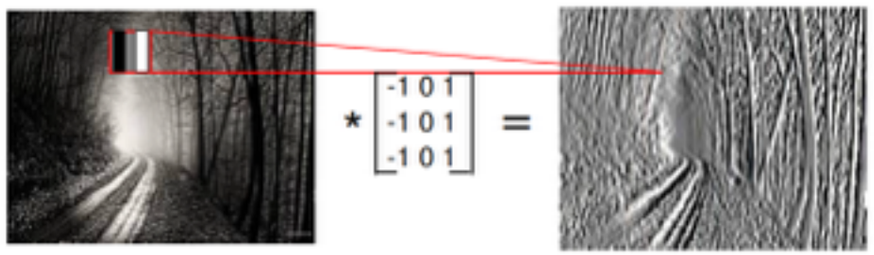
池化

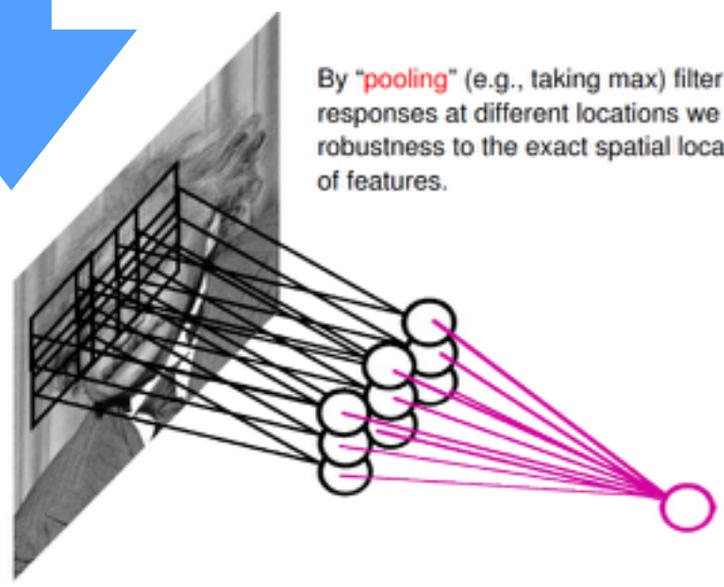


Convolved feature



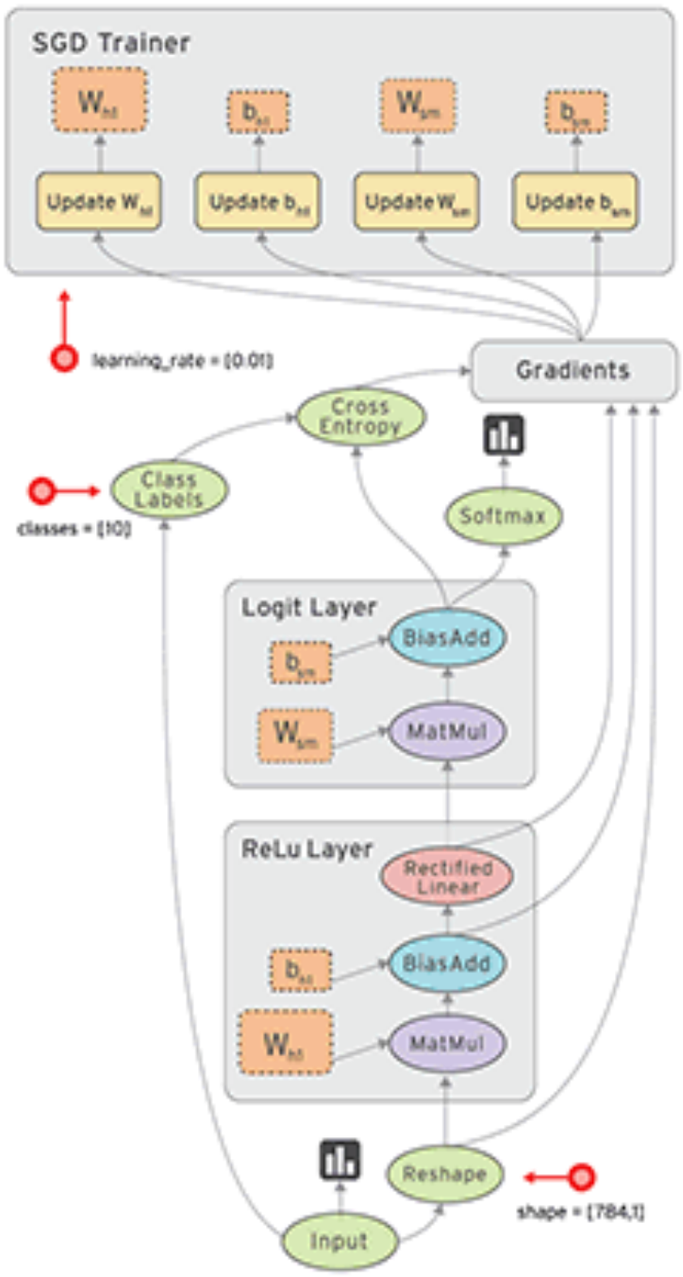
Pooled feature





卷积神经网络

Google TensorFlow



tensorflow

http://www.tensorflow.org github-admin@tensorflow.org

Repositories People 28

Filters Find a repository...

tensorflow

C++ ★ 28,536 11,686

Computation using data flow graphs for scalable machine learning

Updated an hour ago

magenta

Python ★ 2,783 277

Magenta: Music and Art Generation with Machine Intelligence

Updated 7 hours ago

#1 检索的系统

#2 简短的对话

#3 特定的语境

https://github.com/dennybritz/chatbot-retrieval

git clone git@github.com:dennybritz/chatbot-retrieval.git

python udc_train.py

```
INFO:tensorflow:Input queue is exhausted.
INFO:tensorflow:Saving evaluation summary for 0 step: loss = 6.19901, recall_at_5 = 0.4931306
INFO:tensorflow:Step 1: loss = 4.48393
INFO:tensorflow:training step 100, loss = 1.43115 (4.198 sec/batch).
INFO:tensorflow:Step 101: loss = 1.58307
INFO:tensorflow:training step 200, loss = 0.88204 (5.462 sec/batch).
INFO:tensorflow:Step 201: loss = 0.771809
INFO:tensorflow:training step 300, loss = 0.82967 (4.408 sec/batch).
INFO:tensorflow:Step 301: loss = 0.808524
INFO:tensorflow:training step 400, loss = 0.69053 (4.296 sec/batch).
INFO:tensorflow:Step 401: loss = 0.725083
INFO:tensorflow:training step 500, loss = 0.70153 (4.188 sec/batch).
INFO:tensorflow:Step 501: loss = 0.687135
INFO:tensorflow:training step 600, loss = 0.70823 (4.896 sec/batch).
INFO:tensorflow:Step 601: loss = 0.662524
INFO:tensorflow:training step 700, loss = 0.69382 (4.257 sec/batch).
INFO:tensorflow:Step 701: loss = 0.702078
INFO:tensorflow:training step 800, loss = 0.69694 (4.268 sec/batch).
INFO:tensorflow:Step 801: loss = 0.685443
```

使用TensorFlow训练聊天机器人

https://github.com/dennybritz/chatbot-retrieval

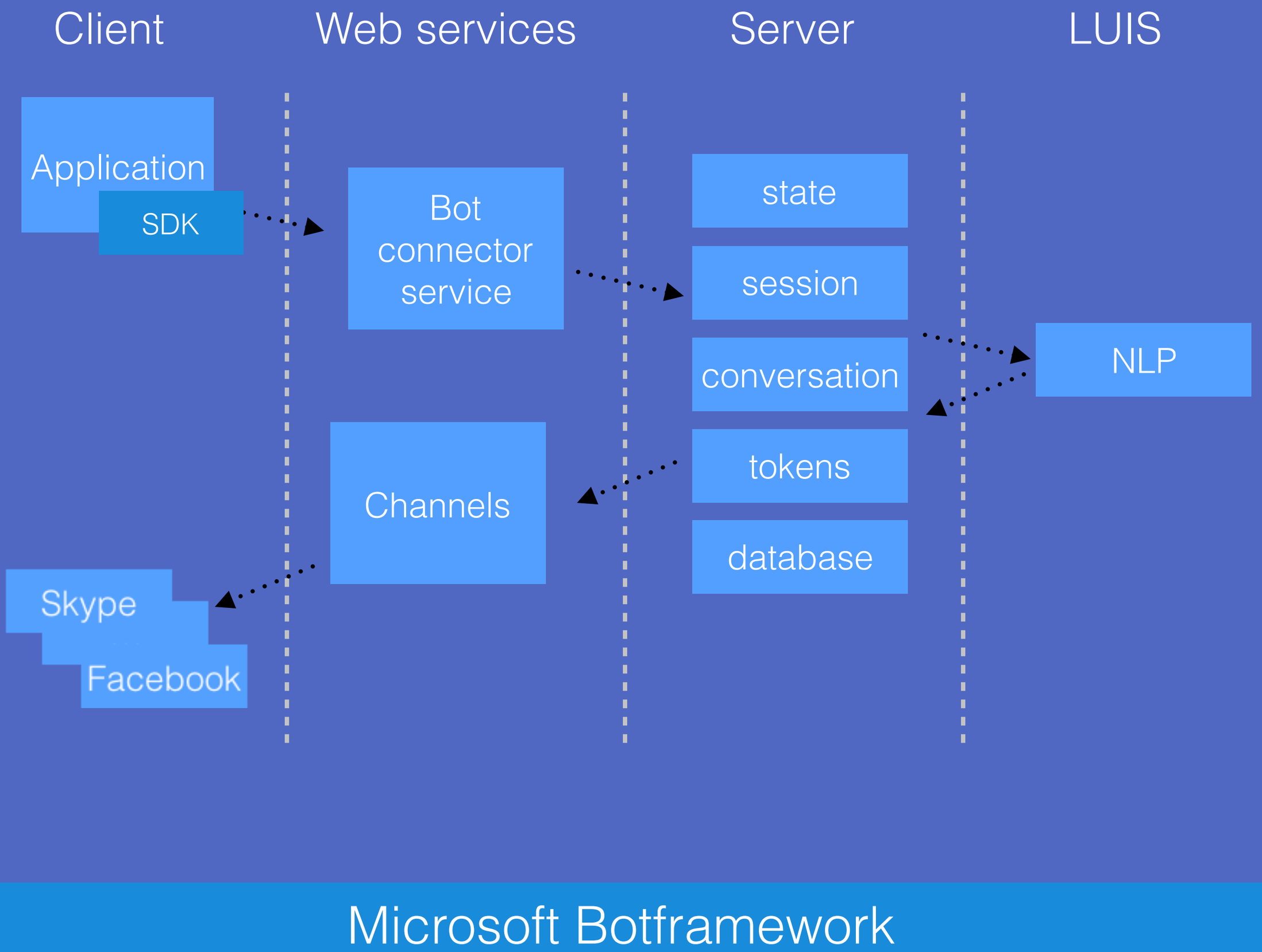
```
git clone git@github.com:dennybritz/chatbot-retrieval.git
python udc_train.py
```

After training for about 20,000 steps:

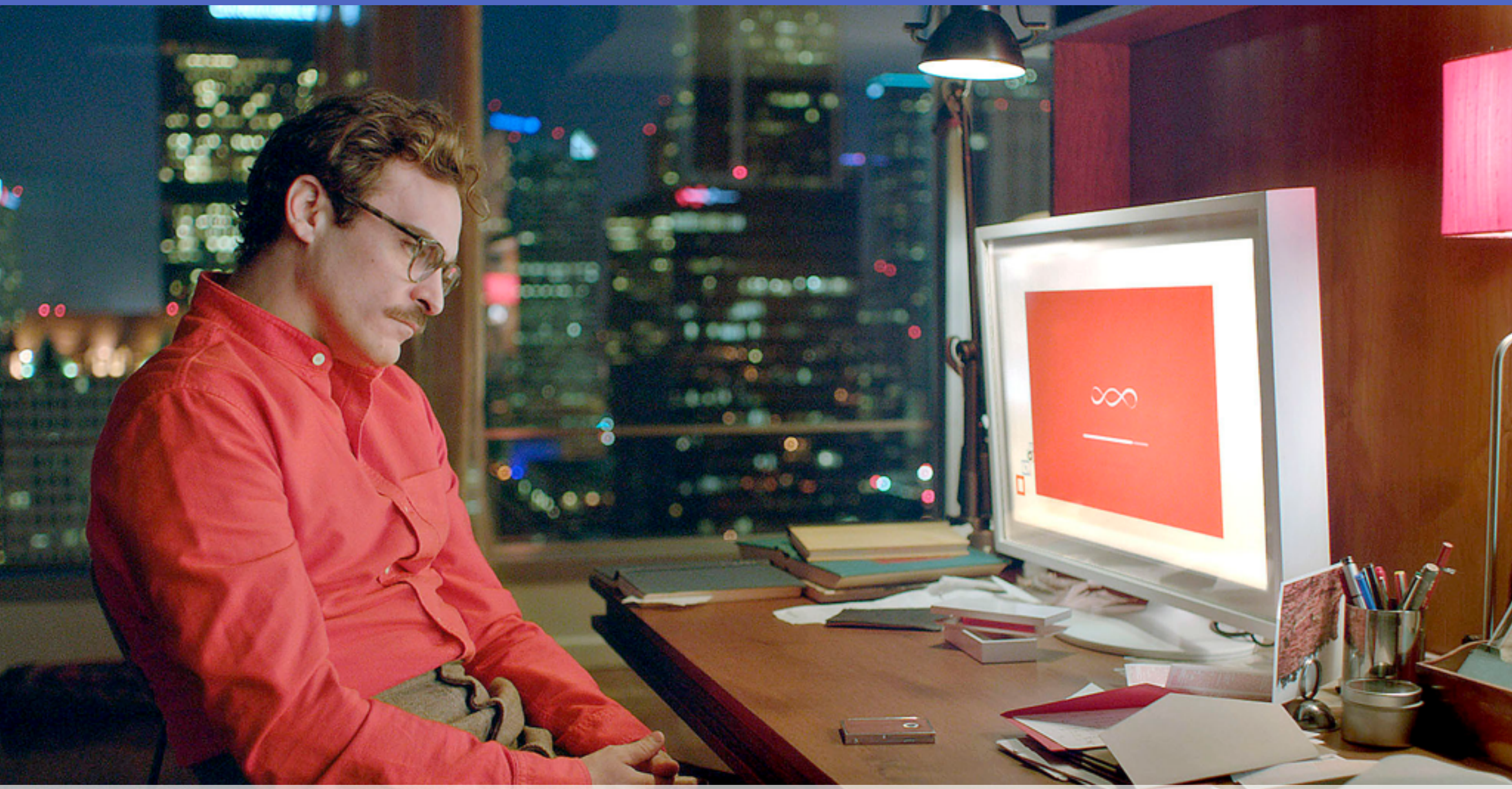
recall_at_1 = 0.507581018519
recall_at_2 = 0.689699074074
recall_at_5 = 0.913020833333

(around an hour on a fast GPU)

使用TensorFlow训练聊天机器人



Microsoft Botframework



Cool, wanna hands on, where to start?

Thank you.