又拍云深度学习实践

叶靖 (yejingx) @ 又拍云 杭州

2017.6.24 OpenTalk Shenzhen

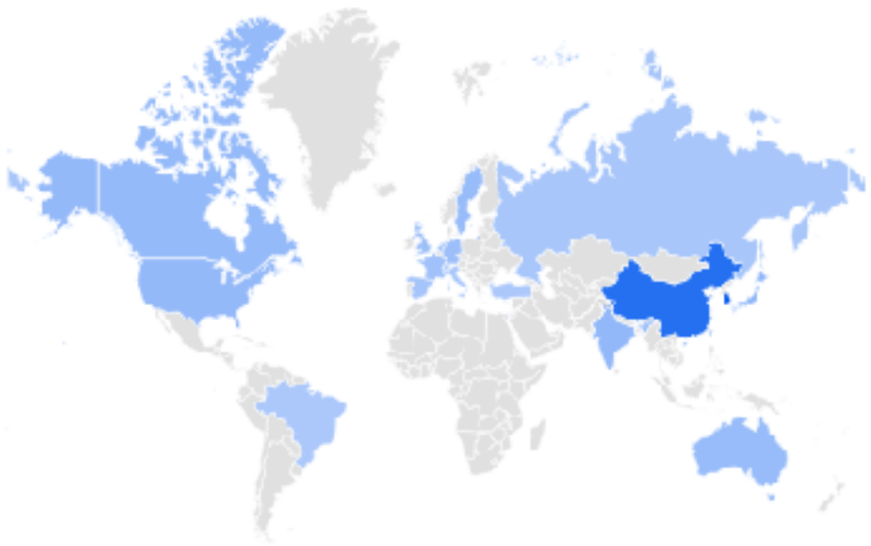
Google Trends

deep learning

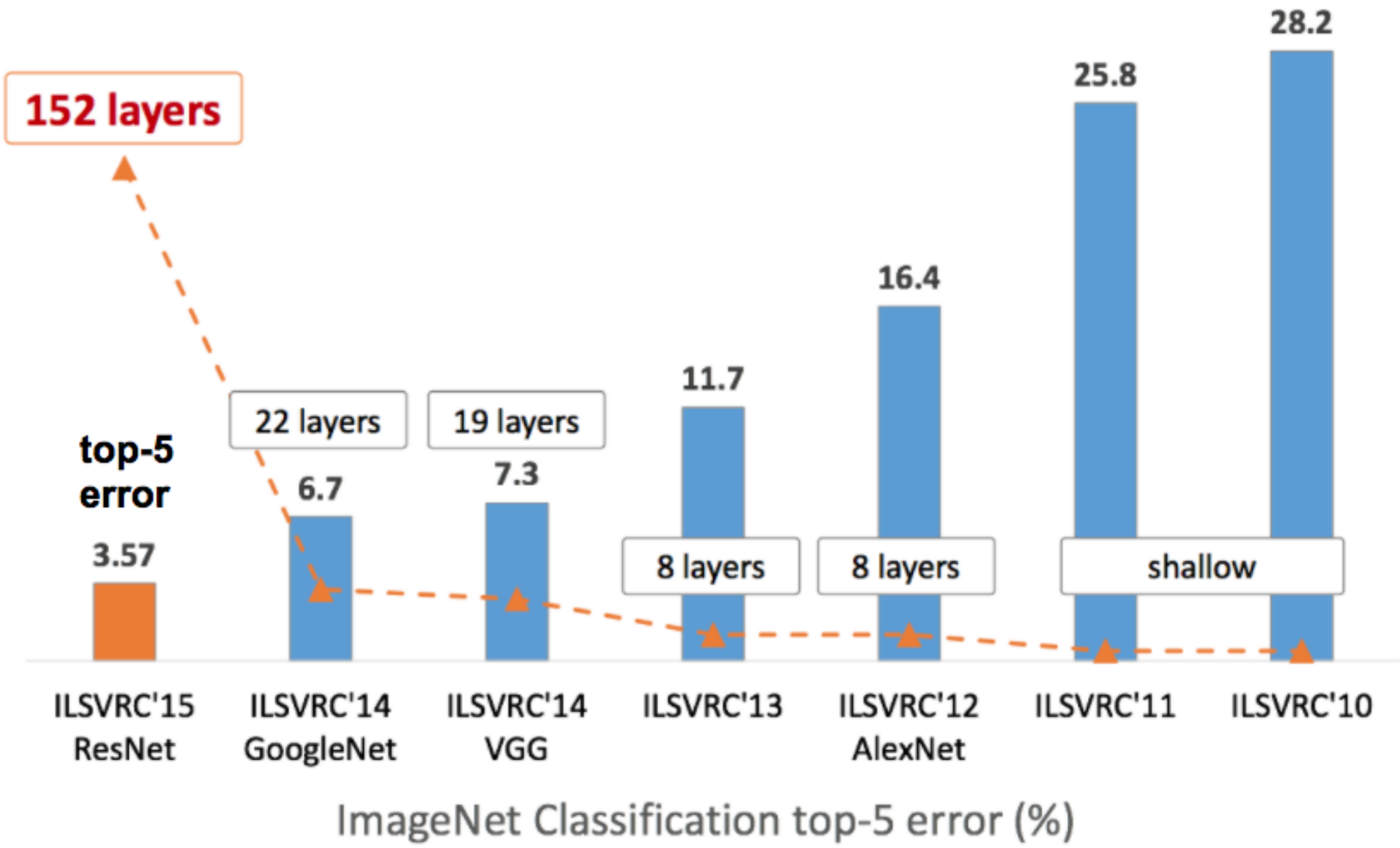
热度随时间变化的趋势

按区域显示的搜索热度

区域

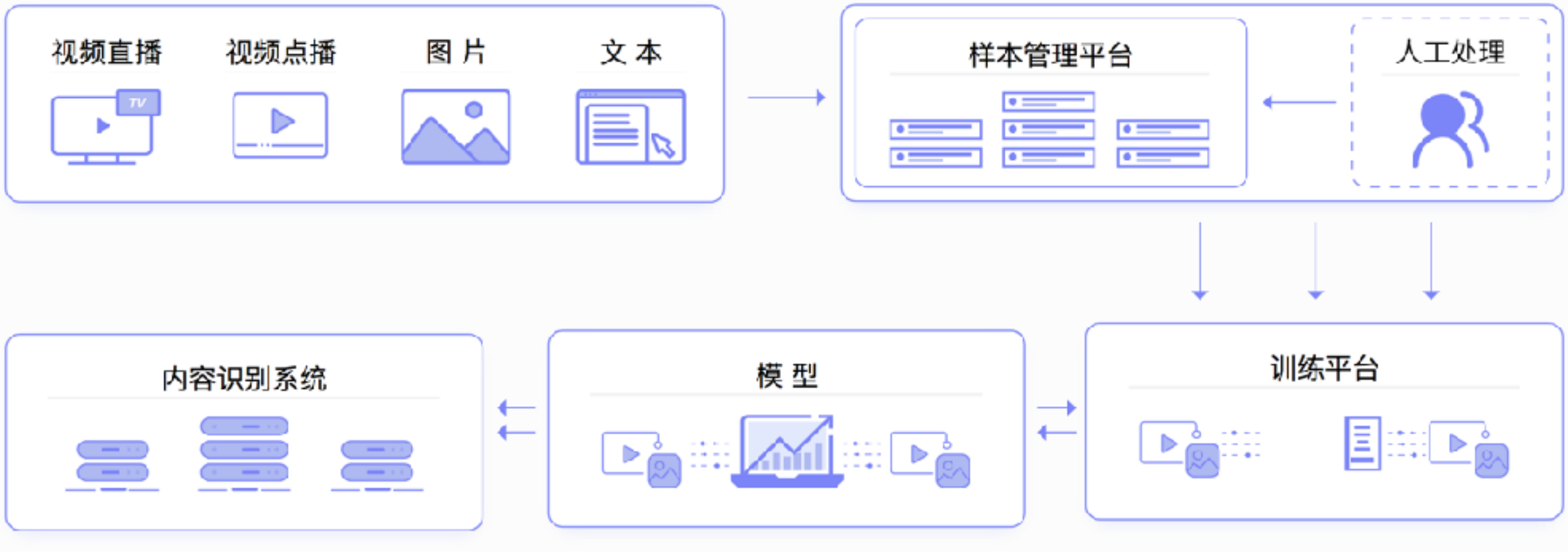


1	韩国	100
2	中国	77
3	新加坡	51
4	香港	48
5	以色列	36
6	台湾	30



数据来源: https://arxiv.org/pdf/1512.03385.pdf

图片鉴别

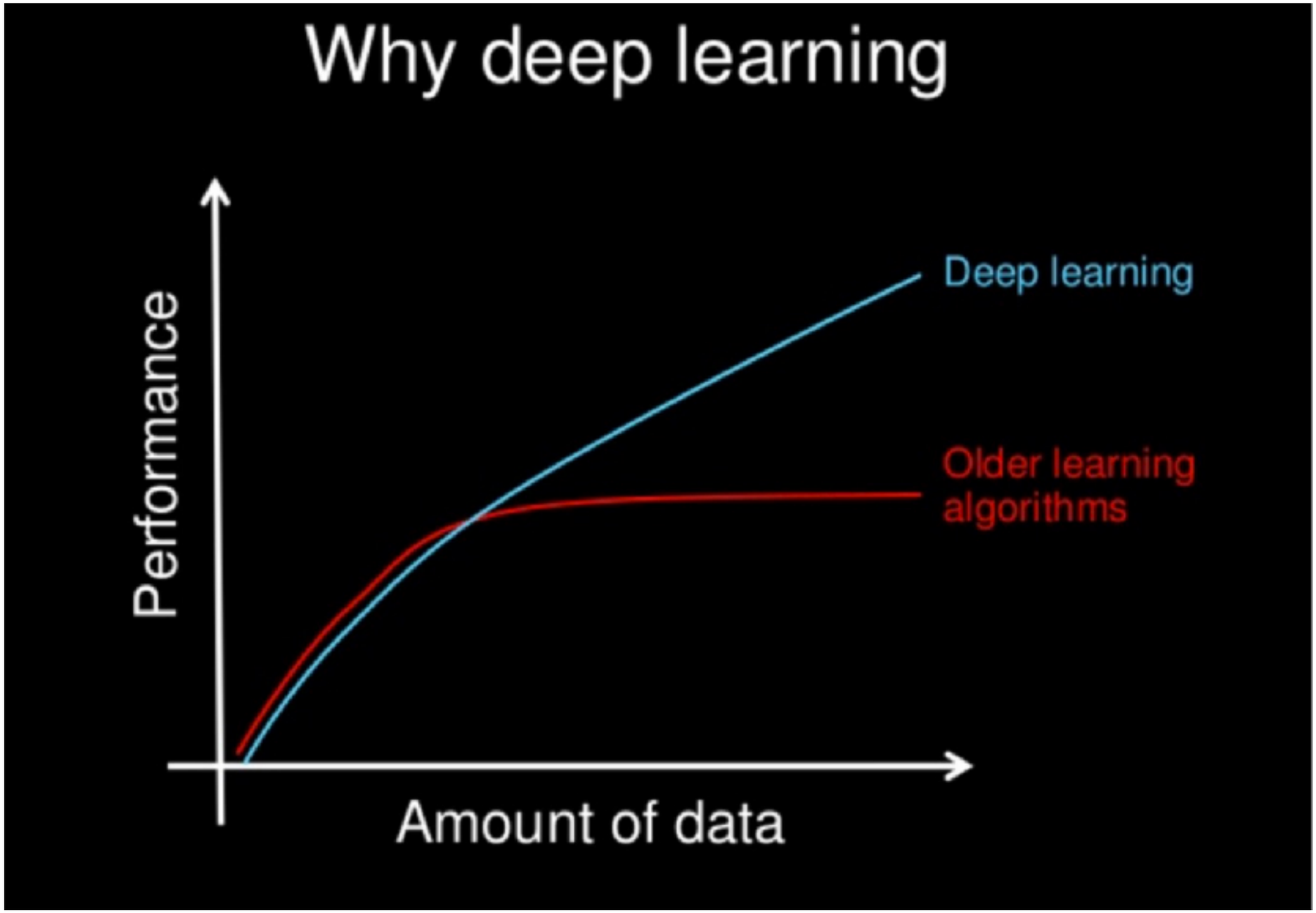


快速入门

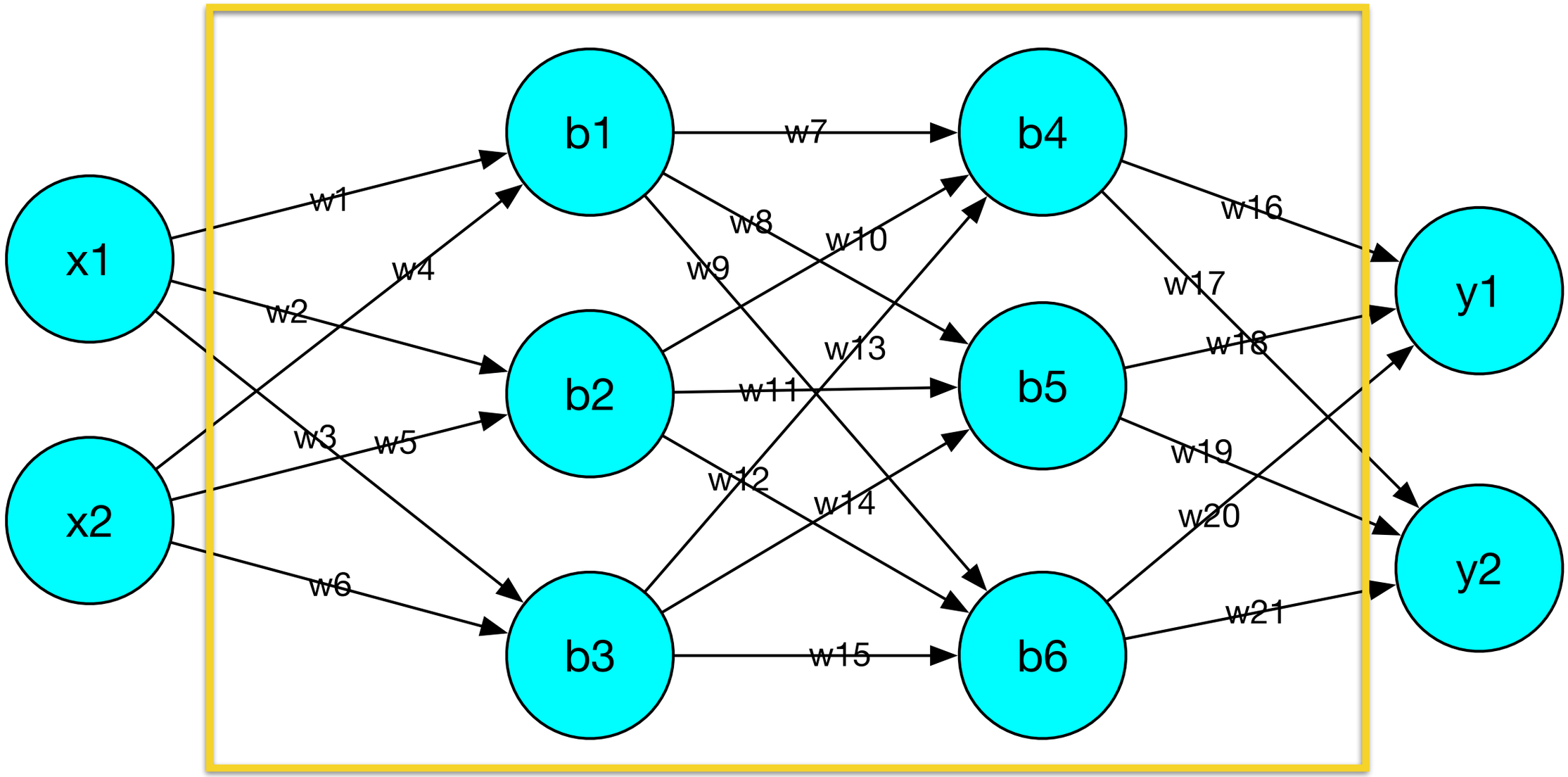
分类器

垃圾邮件过滤、手写数字识别、图片分类、鉴黄...

- 朴素贝叶斯
- KNN
- SVM
- 人工神经网络
- ...

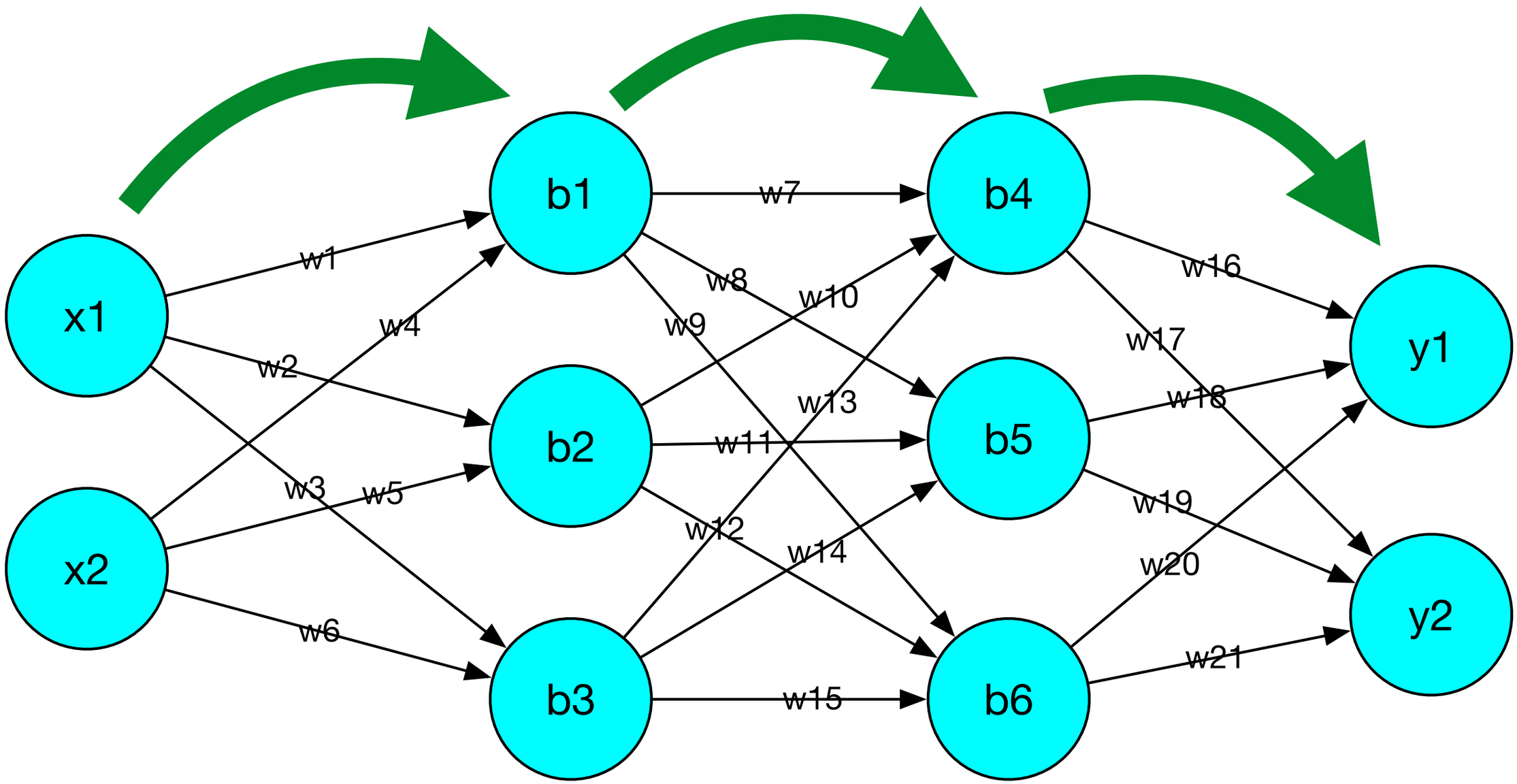


人工神经网络



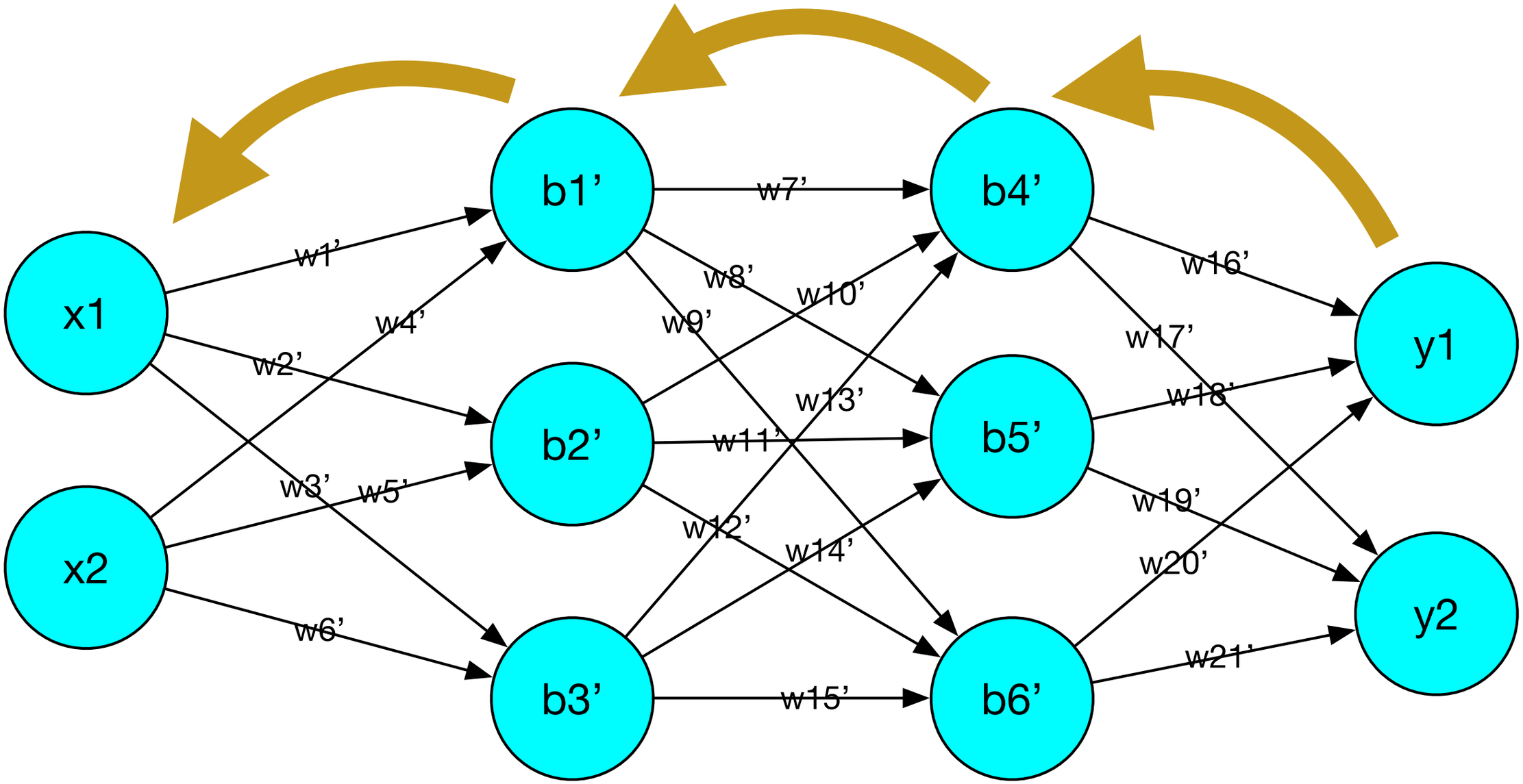
输入层　　隐层　　隐层　　输出层

前向传播



输入层　　隐层　　隐层　　输出层

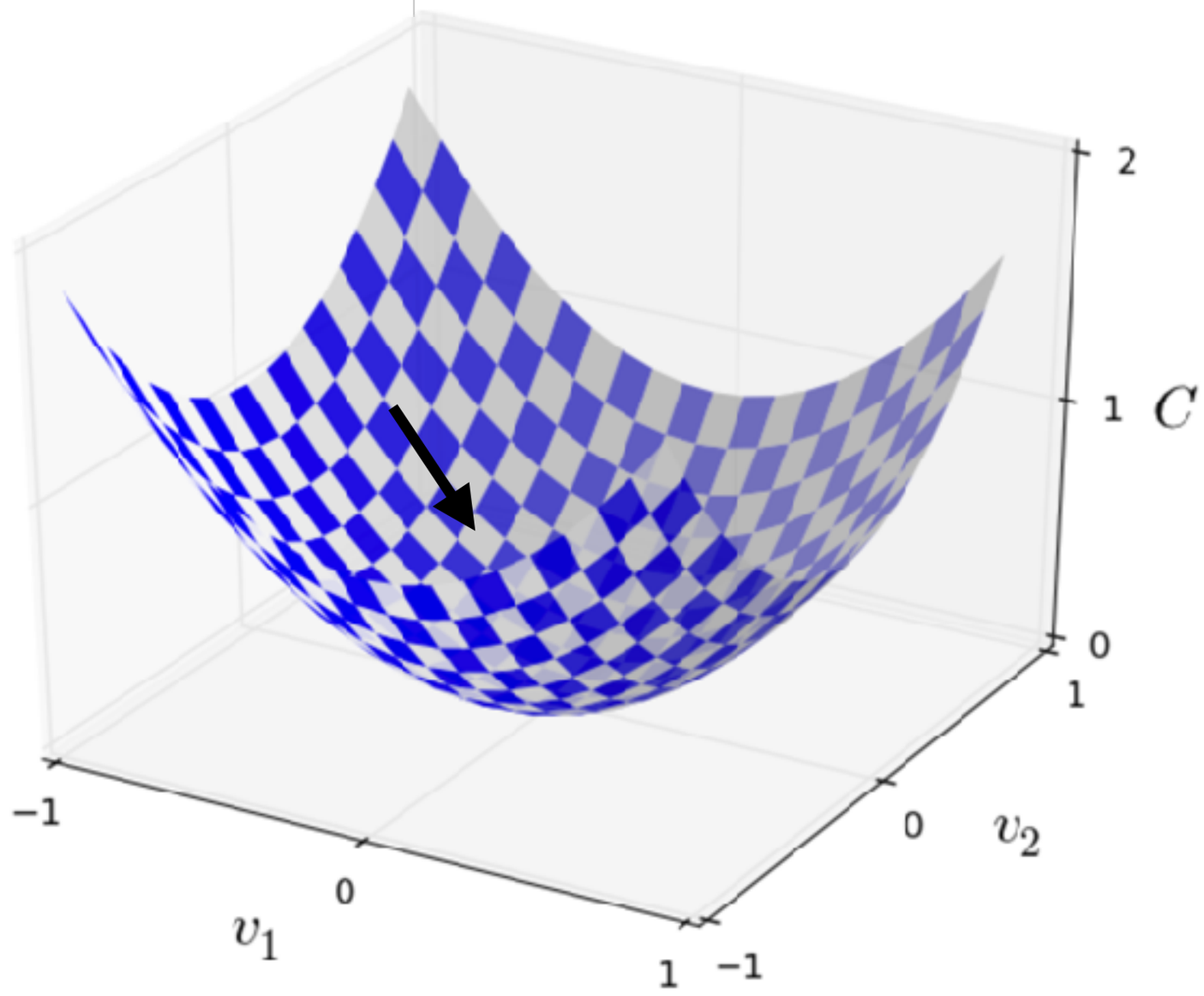
反向传播



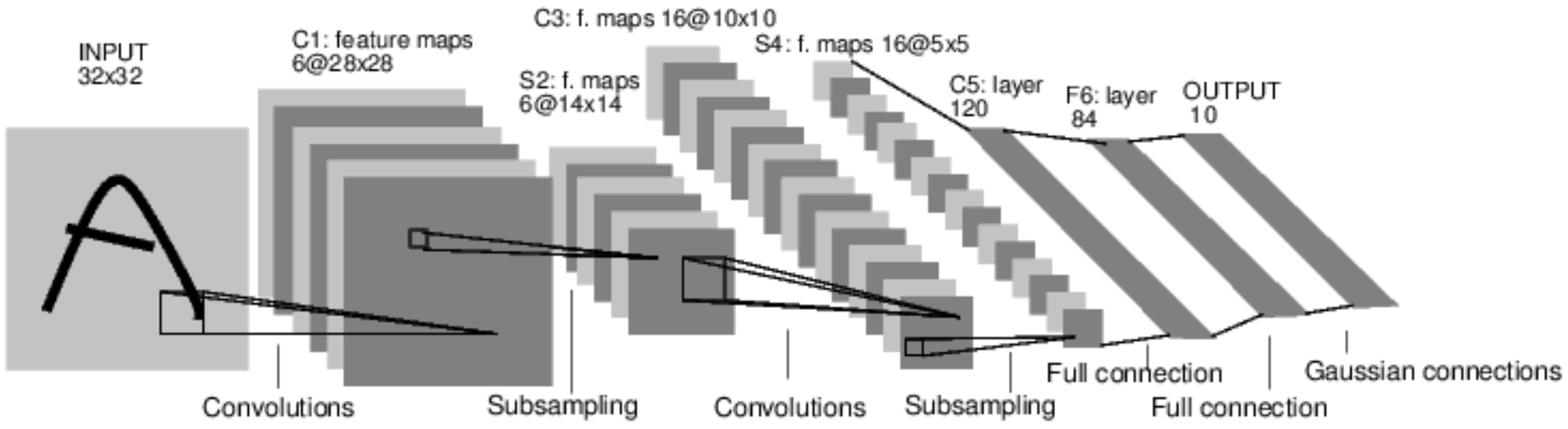
输入层　　隐层　　隐层　　输出层

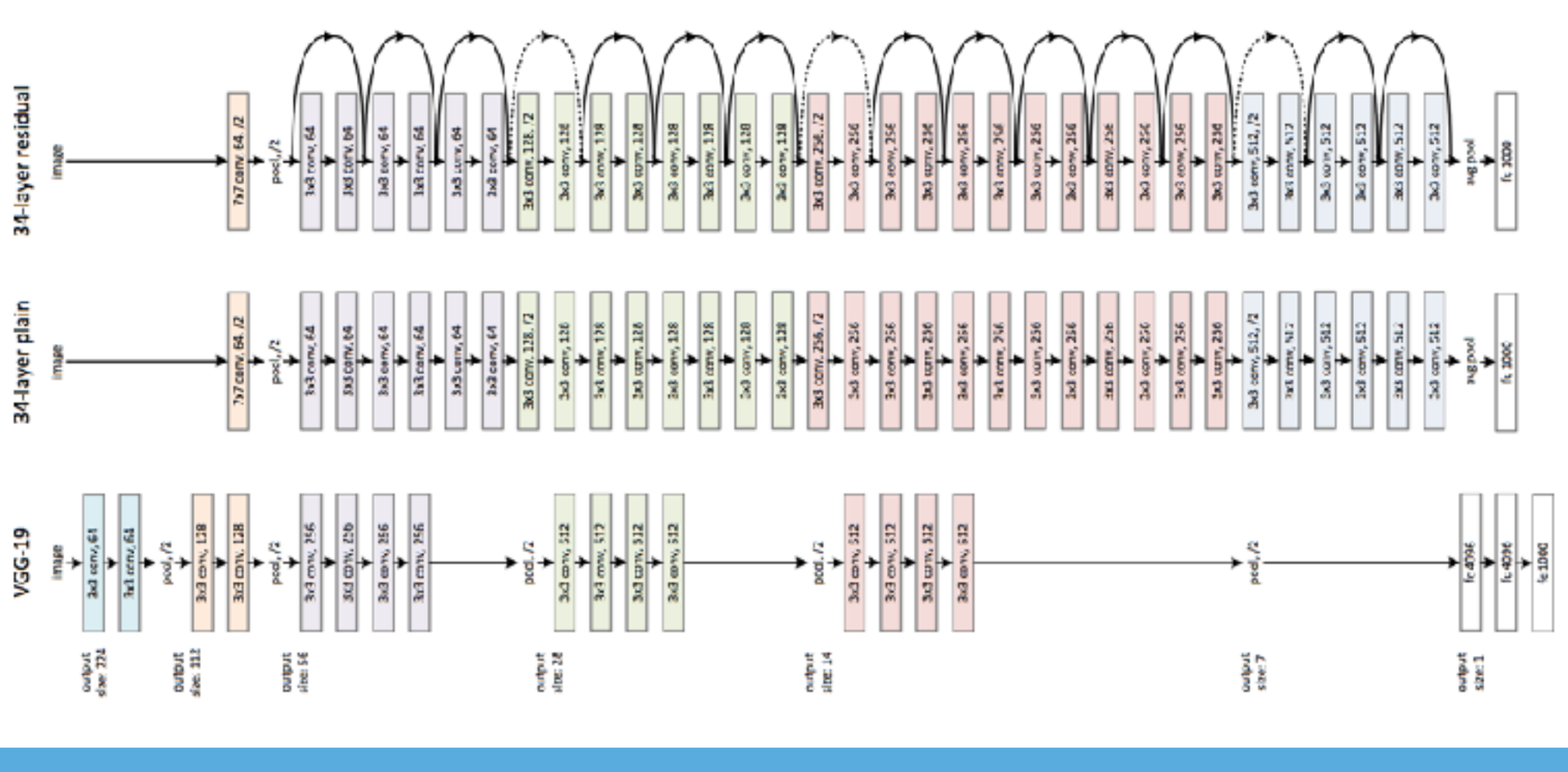
梯度下降

$$C(w,b)=\frac{1}{2n}\sum(h(x)-y)^2$$



真实的网络


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections


VGG-19
34-layer plain
34-layer residual
image
3x3 conv, 64
7x7 conv, 64, /2
pool, /2
3x3 conv, 128
3x3 conv, 128, /2
3x3 conv, 256
3x3 conv, 256, /2
3x3 conv, 512
3x3 conv, 512, /2
fc 4096
fc 4096
fc 1000
avg pool
fc 1000

工具

GPU vs CPU

ALL NUMBERS IN [r	milliseconds (ms)	ms	ms	ms	ms	ms	ms	ms	ms	
	Intel i2600k (3.4GHz)	Intel Xeon E5620 (2.4GI	Intel Xeon E5-1650 v3	GTX560	GTX570	GTX670	GTX680	GTX770	Titan X-M	Algorithm setting
matchTemplate	424.0	673.0	185.3	19.90	16.31	14.67	15.22	12.96	9.01	3000x3000, 32FC1, templ 125x125, C
minMaxLoc	531.0	252.5	15.1	3.72	2.49	3.03	2.72	2.25	1.83	8000x8000, 32F
remap	91.0	167.8	20.1	5.70	3.25	4.19	3.60	3.10	0.95	4000x4000, 8UC4, INTER_LINEAR, B
dft	561.0	945.0	121.8	19.01	14.07	12.43	12.99	9.15	5.79	4000x4000, 32FC2, complex-to-compl
cornerHarris	317.0	512.0	228.4	71.89	47.99	37.21	34.74	31.30	13.69	4000x4000, 32FC1, BORDER_REFLE
integral	13.3	27.2	6.5	14.54	7.93	8.56	8.50	3.13	2.06	4000x4000, 8UC1
norm	72.6	100.7	15.1	3.23	2.20	2.10	2.03	2.35	1.46	4000x4000, 32FC4, NORM_INF
meanShift	1395.0	1163.0	1179.7	44.21	23.63	16.13	16.50	18.37	7.52	800x800, 8UC3 vs 8UC4
BruteForceMatcher	169.3	269.9	70.9	13.70	7.20	6.16	6.07	5.93	1.89	radiusMatch
magnitude	16.4	27.8	13.4	2.60	1.56	1.45	1.43	1.10	0.73	4000x4000, 32FC1
add	13.0	22.0	11.9	2.03	1.39	1.22	1.25	1.20	0.73	4000x4000, 32FC1
log	36.7	55.9	8.8	2.88	1.61	1.23	1.20	0.80	0.50	4000x4000, 32F
mulSpectrums	55.9	80.1	36.9	3.16	2.96	2.46	2.49	2.26	1.44	4000x4000
resize	33.3	53.0	1.5	0.42	0.40	0.22	0.23	?	0.17	3000x3000, 8UC4, down
cvtColor	267.9	398.3	38.0	3.39	2.29	3.54	3.64	2.30	0.79	4000x4000, 8UC4, CV_HSV2BGR
erode	18.8	36.2	11.3	18.25	10.06	8.84	8.74	22.20	3.76	4000x4000
threshold	9.6	15.8	4.4	1.61	0.99	0.86	0.89	0.82	0.50	4000x4000, 32FC1, THRESH_TRUNC
pow	53.0	89.4	10.3	1.55	1.14	0.80	0.83	0.74	0.50	4000x4000, 32F
projectPoints	7.3	11.6	5.9	0.50	0.47	0.23	0.25	0.38	0.08	260307
solvePnPRansac	2542.0	3821.0		123.94	120.70	116.90	172.10	149.70		265786
GaussianBlur	58.5	89.2	48.4	18.43	9.35	13.10	11.07	7.60	2.95	4000x4000, 8UC4
filter2D	493.8	866.0	174.0	126.50	85.34	57.18	55.08	68.30	25.69	ksize = 15, 2048x2048, 8UC4
pyrDown	2.7	4.5	2.1	0.60	0.38	0.29	0.29	0.36	0.06	1000x1000, 8UC4
pyrUp	9.9	23.1	7.2	2.92	1.65	1.13	1.12	1.00	0.27	1000x1000, 8UC4
equalizeHist	11.0	20.7	3.2	3.17	1.62	2.22	2.16	1.69	0.42	3000x3000
reduce	4.2	8.2	2.1	0.84	0.55	0.59	0.58	0.26	0.14	3000x3000, dim = 1
AVERAGE (ms)	277.2	374.4	88.9	19.6	14.1	12.2	14.1	14.0	3.3	
SPEEDUP	1.0	0.7	3.1	14.2	19.6	22.8	19.7	19.8	83.6	

数据来源: http://www.timzaman.com/2012/05/opencv-gpu-cuda-performance-comparison/

GPU 的不足

- 不易扩展，需要针对 GPU 重写算法
- 对硬件依赖高 (Nvidia, CUDA, cuDNN)
- 显存小 (8G, 12G)
- 从内存拷贝数据慢

机器配置

项目	型号
OS	ubuntu 16.04
CPU	Xeon E5-2620 CPU X2
内存	SK DDR4-R-ECC 16G X4
显卡	丽台公版 GTX 1070 X4
硬盘	4T X4
SSD	480G
主板	华硕 Z10PE-D8 WS
佳航	航嘉 1000W

深度学习框架

- Caffe
- TensorFlow
- Torch
- Theano
- Paddle

Caffe

Deep learning framework by BAIR

Created by
Yangqing Jia
Lead Developer
Evan Shelhamer

View On GitHub

Caffe

Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by Berkeley AI Research (BAIR) and by community contributors. Yangqing Jia created the project during his PhD at UC Berkeley. Caffe is released under the BSD 2-Clause license.

Check out our web image classification demo!

Why Caffe?

Expressive architecture encourages application and innovation. Models and optimization are defined by configuration without hard-coding. Switch between CPU and GPU by setting a single flag to train on a GPU machine then deploy to commodity clusters or mobile devices.

Extensible code fosters active development. In Caffe's first year, it has been forked by over 1,000 developers and had many significant changes contributed back. Thanks to these contributors the framework tracks the state-of-the-art in both code and models.

Speed makes Caffe perfect for research experiments and industry deployment. Caffe can process **over 60M images per day** with a single NVIDIA K40 GPU*. That's 1 ms/image for inference and 4 ms/image for learning and more recent library versions and hardware are faster still. We believe that Caffe is among the fastest convnet implementations available.

Community: Caffe already powers academic research projects, startup prototypes, and even large-scale industrial applications in vision, speech, and multimedia. Join our community of brewers on the caffe-users group and Github.

* With the ILSVRC2012-winning SuperVision model and prefetching IO.

Caffe 优缺点

- 性能好
- 支持 python 和 C++ 接口
- 非常容易入门

- model zoo
- 很难扩展
- 单机
- 安装太复杂

Caffe 基本概念

- train_val.prototxt
  - 输入、输出
  - 网络结构
- solver.prototxt
  - 学习率(base_lr)，权重衰减(weight_decay)，迭代次数(max_iter) ...
- deploy.prototxt
- .caffemodel

TensorFlow

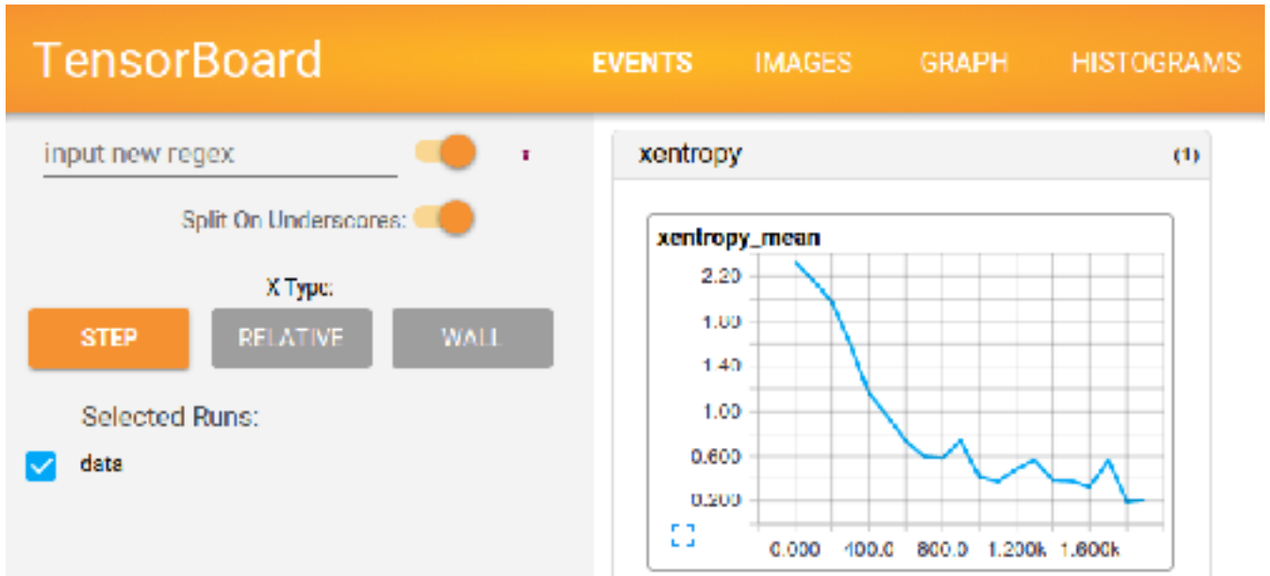
Linux CPU	Linux GPU	Mac OS CPU	Windows CPU	Android
build passing	build passing	build passing	build passing	build passing

TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them. This flexible architecture lets you deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code. TensorFlow also includes TensorBoard, a data visualization toolkit.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well.

TensorFlow 优缺点

- 支持多 GPU，分布式
- TensorBoard
- 社区活跃、完善的文档
- 功能强大，容易扩展
- 模型没有 caffe 直观
- 需要一定算法功底



实践

三要素

- 数据
  - ImageNet
  - 爬虫/人工标注
  - 好的样本 == 成功了一半!
- 模型
  - model zoo
  - AlexNet, CaffeNet, GoogLeNet, VGG, ResNet ...
  - 修改
- 计算

数据

```
/home/data2/images/f85/f8547f90452d92678946faf0d8ec1268.jpg 0
/home/data2/images/f67/f67c51cfa43373d7b1c8ed5b93f0b1ef.jpg 0
/home/data2/images/100/100f5ca5b3753ccb69812570a7a67061.jpg 1
/home/data2/images/d61/d6152911487982fb2a8e21e6c10a4143.jpg 1
/home/data2/images/6c3/6c3235389c2d04cfb29133921ab8006b.jpg 0
/home/data2/images/2eb/2eb731fcd3f314f49c81b097bd34bf7f.jpg 0
/home/data2/images/1d5/1d55f595e2511b777faa88a04ec3a8ca.jpg 0
/home/data2/images/f38/f380a0a6cb1ce1896cc4d16aae7507f7.jpg 1
/home/data2/images/fcc/fcc3790196762d1f820ebb0d76cfb2ba.jpg 0
/home/data2/images/cf9/cf909c03f69c1a9b4c74c256c151161e.jpg 1
/home/data2/images/332/332e7dc686827d244b02fcfdaf38969f.jpg 0
/home/data2/images/4ed/4edf36d752531e17df5cfd7cea8d85e1.jpg 1
/home/data2/images/b59/b5946be934b6c45b4a8bbe52c0665805.jpg 0
/home/data2/images/c23/c238414f90e83fd7980c62b0ea7db725.jpg 0
/home/data2/images/615/615603362df037f6e1fbf473fefa9e5b.jpg 1
/home/data2/images/b8f/b8f25da6a59205231d9d83d655d68803.jpg 1
/home/data2/images/9f2/9f2626cf195823a24f57b1d63c5b7935.jpg 1
/home/data2/images/f44/f4417e1ef9f46ee15bd6986baa6ed736.jpg 0
/home/data2/images/553/55302e2b4c0c3608647180718d5121c9.jpg 0
/home/data2/images/56b/56b5b84a9eb01ba2d503c68db55f2ffd.jpg 1
/home/data2/images/b67/b6700de45d00226accbac7d035f49208.jpg 1
/home/data2/images/be5/be569dfc85e011188990f5677a6bf1f2.jpg 0
/home/data2/images/84f/84f3b8b0374e5c7a6444f49e010982c2.jpg 1
/home/data2/images/8f4/8f49999658ee5df4ef0367137b76576d.jpg 1
/home/data2/images/a37/a37eb55418d5d45425f09319942e3dbb.jpg 1
/home/data2/images/71d/71d25d56fc34d3efd8ff0a6b3845c528.jpg 1
/home/data2/images/511/51154a88b6b8d06585ca945317c38b70.jpg 1
/home/data2/images/102/1022c22950cb0e6c2a974edddb76d2d0.jpg 1
/home/data2/images/e61/e61fda385105e683981656265c616fe2.jpg 0
/home/data2/images/5d4/5d4ee48857e5a8a66ed188d2c48e6974.jpg 0
```

0 正常

1 色情

模型

```
models/
├── bvlc_alexnet
│   ├── deploy.prototxt
│   ├── readme.md
│   ├── solver.prototxt
│   └── train_val.prototxt
├── bvlc_googlenet
│   ├── deploy.prototxt
│   ├── quick_solver.prototxt
│   ├── readme.md
│   ├── solver.prototxt
│   └── train_val.prototxt
├── bvlc_reference_caffenet
│   ├── deploy.prototxt
│   ├── readme.md
│   ├── solver.prototxt
│   └── train_val.prototxt
├── bvlc_reference_rcnn_ilsvrc13
│   ├── deploy.prototxt
│   └── readme.md
└── finetune_flickr_style
    ├── deploy.prototxt
    ├── readme.md
    ├── solver.prototxt
    └── train_val.prototxt
```

Model-Zoo

- Berkeley-trained models
- Network in Network model
- Models from the BMVC-2014 paper "Return of the Devil in the Details: Delving Deep into Convolutional Nets"
- Models used by the VGG team in ILSVRC-2014
- Places-CNN model from MIT.
- GoogLeNet GPU implementation from Princeton.
- Fully Convolutional Networks for Semantic Segmentation (FCNs)
- CaffeNet fine-tuned for Oxford flowers dataset
- CNN Models for Salient Object Subitizing.
- Deep Learning of Binary Hash Codes for Fast Image Retrieval
- Places_CNDS_models on Scene Recognition
- Models for Age and Gender Classification.
- GoogLeNet_cars on car model classification
- ParseNet: Looking wider to see better
- SegNet and Bayesian SegNet
- Conditional Random Fields as Recurrent Neural Networks
- Holistically-Nested Edge Detection
- CCNN: Constrained Convolutional Neural Networks for Weakly Supervised Segmentation
- Emotion Recognition in the Wild via Convolutional Neural Networks and Mapped Binary Patterns
- Facial Landmark Detection with Tweaked Convolutional Neural Networks
- Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
- ResNets: Deep Residual Networks from MSRA at ImageNet and COCO 2015
- Pascal VOC 2012 Multilabel Classification Model
- SqueezeNet: AlexNet-level accuracy with 50x fewer parameters
- Mixture DCNN
- CNN Object Proposal Models for Salient Object Detection
- Deep Hand: How to Train a CNN on 1 Million Hand Images When Your Data Is Continuous and Weakly Labelled
- Multimodal Compact Bilinear Pooling for VQA
- Pose-Aware CNN Models (PAMs) for Face Recognition
- Learning Structured Sparsity in Deep Neural Networks
- Neural Activation Constellations: Unsupervised Part Model Discovery with Convolutional Networks
- Inception-BN full ImageNet model
- ResFace101: ResNet-101 for Face Recognition
- DeepYeast
- ImageNet pre-trained models with batch normalization
- ResNet-101 for regressing 3D morphable face models (3DMM) from single images
- Cascaded Fully Convolutional Networks for Biomedical Image Segmentation
- Deep Networks for Earth Observation
- Supervised Learning of Semantics-Preserving Hash via Deep Convolutional Neural Networks
- Striving for Simplicity: The All Convolutional Net

模型 -- 以 AlexNet 为例

models/bvlc_alexnet/train_val.prototxt

修改输入:

```
name: "AlexNet"
layer {
  name: "data"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TRAIN
  }
  transform_param {
    mirror: true
    crop_size: 227
    mean_value: 104
    mean_value: 117
    mean_value: 123
  }
  data_param {
    source: "/imagenet/train_lmdb"
    batch_size: 256
    backend: LMDB
  }
}
```

```
name: "AlexNet"
layer {
  name: "data"
  type: "ImageData"
  top: "data"
  top: "label"
  include {
    phase: TRAIN
  }
  transform_param {
    mirror: true
    crop_size: 227
    mean_value: 104
    mean_value: 117
    mean_value: 123
  }
  image_data_param {
    source: "/home/data2/train.txt"
    batch_size: 256
    shuffle: true
    new_height: 256
    new_width: 256
  }
}
```

模型 -- 以 AlexNet 为例

models/bvlc_alexnet/train_val.prototxt

修改输出:

```
layer {
  name: "fc8"
  type: "InnerProduct"
  bottom: "fc7"
  top: "fc8"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 1000
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

```
layer {
  name: "fc8"
  type: "InnerProduct"
  bottom: "fc7"
  top: "fc8"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
```

训练

models/bvlc_alexnet/solver.prototxt

```
net: "models/bvlc_alexnet/train_val.prototxt"
test_iter: 1000
test_interval: 1000
base_lr: 0.01
lr_policy: "step"
gamma: 0.1
stepsize: 100000
display: 20
momentum: 0.9
weight_decay: 0.0005
max_iter: 100000
snapshot_prefix: "models/alexnet"
```

> caffe train -solver solver.prototxt -gpu 0

测试

> caffe test -model **models/bvlc_alexnet/train_val.prototxt** -weights **models/alexnet_iter_10000.caffemodel** -gpu 0 -iterations 100

```
I0621 12:24:21.344616 10346 caffe.cpp:290] Running for 100 iterations.
I0621 12:24:34.368952 10346 caffe.cpp:313] Batch 0, accuracy = 1
I0621 12:24:34.371574 10346 caffe.cpp:313] Batch 0, loss = 0.0107442
I0621 12:24:36.789677 10346 caffe.cpp:313] Batch 1, accuracy = 1
I0621 12:24:36.789738 10346 caffe.cpp:313] Batch 1, loss = 0.00435053
I0621 12:24:39.975857 10346 caffe.cpp:313] Batch 2, accuracy = 1
I0621 12:24:39.978317 10346 caffe.cpp:313] Batch 2, loss = 0.0444124
I0621 12:24:43.221068 10346 caffe.cpp:313] Batch 3, accuracy = 0.9
I0621 12:24:43.223600 10346 caffe.cpp:313] Batch 3, loss = 0.100746
I0621 12:24:46.158462 10346 caffe.cpp:313] Batch 4, accuracy = 1
I0621 12:24:46.158524 10346 caffe.cpp:313] Batch 4, loss = 0.0410968
I0621 12:24:49.837234 10346 caffe.cpp:313] Batch 5, accuracy = 1
I0621 12:24:49.842905 10346 caffe.cpp:313] Batch 5, loss = 0.0284642
I0621 12:24:52.938685 10346 caffe.cpp:313] Batch 6, accuracy = 1
I0621 12:24:52.938875 10346 caffe.cpp:313] Batch 6, loss = 0.011008
I0621 12:24:56.460505 10346 caffe.cpp:313] Batch 7, accuracy = 1
```

python 接口

```
# load the model
net = caffe.Net('models/bvlc_reference_alexnet/deploy.prototxt',
                'models/alexnet_iter_10000.caffemodel',
                caffe.TEST)

# load input and configure preprocessing
transformer = caffe.io.Transformer({'data': net.blobs['data'].data.shape})
transformer.set_mean('data', np.load('python/caffe/imagenet/ilsvrc_2012_mean.npy').mean(1).mean(1))
transformer.set_transpose('data', (2,0,1))
transformer.set_channel_swap('data', (2,1,0))
transformer.set_raw_scale('data', 255.0)

# since we classify only one image, we change batch size from 10 to 1
net.blobs['data'].reshape(1,3,227,227)

# load the image in the data layer
im = caffe.io.load_image('examples/images/cat.jpg')
net.blobs['data'].data[...] = transformer.preprocess('data', im)

# compute
out = net.forward()

# out = net.forward_all(data=np.asarray([transformer.preprocess('data', im)]))

# predicted predicted class
print out['prob'].argmax()
```

Fine-turning

- 样本数据少
- 训练慢
- 二次调优

> caffe train -solver solver.prototxt -gpu 0 **-weights pretrained.caffemodel**

集群模式

> caffe train -solver solver.prototxt **-gpu 0,1,2,3**

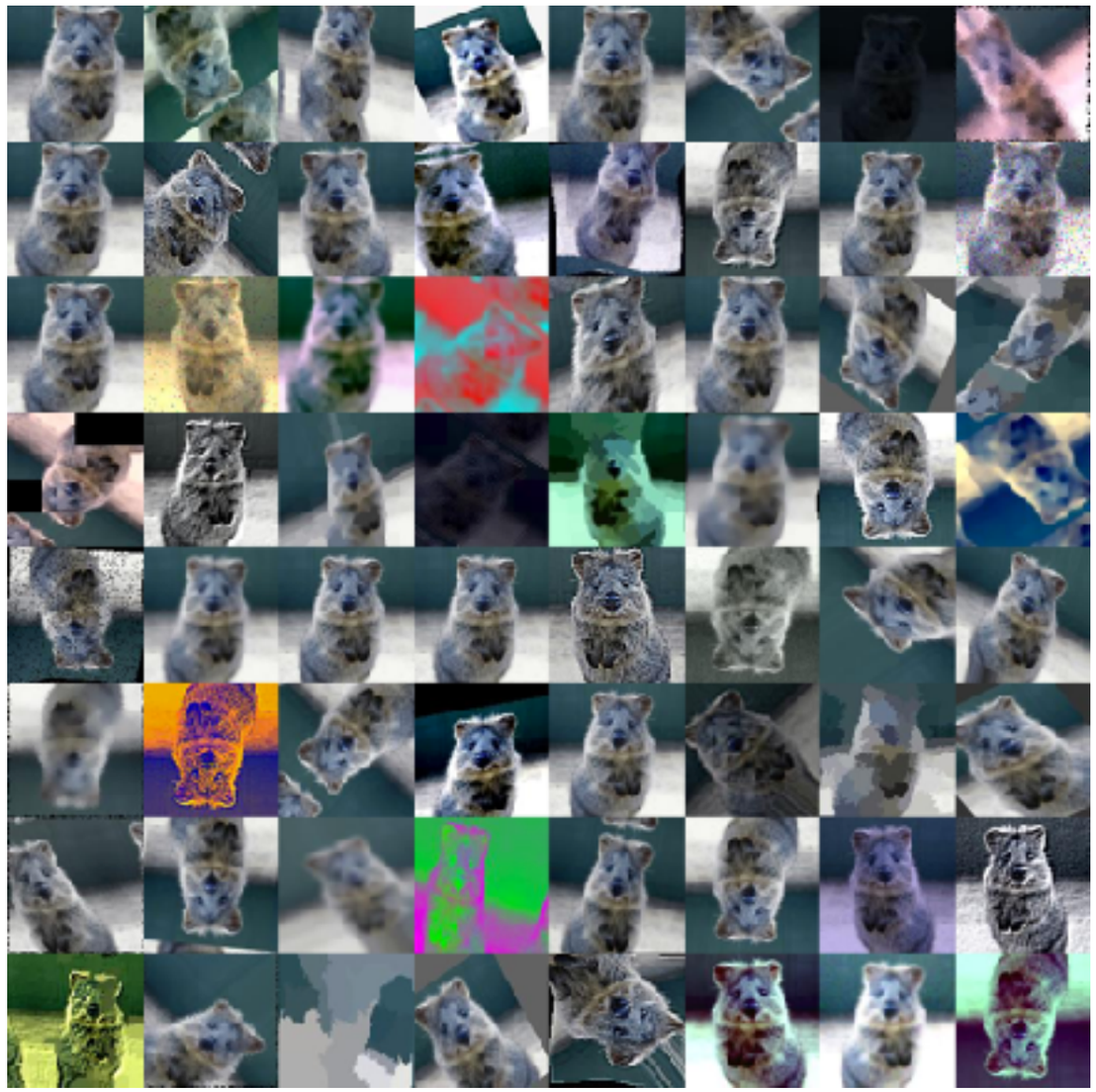
- CaffeOnSpark
- TensorFlow

Data Augmentation

增加正样本数量

- 旋转/翻转
- 随机裁剪/缩放
- 对比度/亮度调整
- 噪声
- 扭曲
- ...

https://github.com/aleju/imgaug

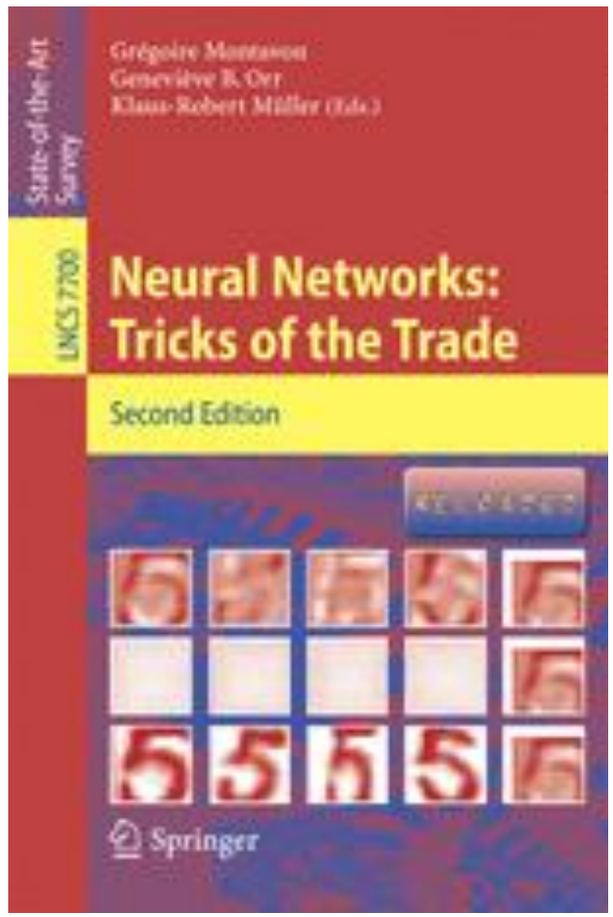


Mesos+Docker部署

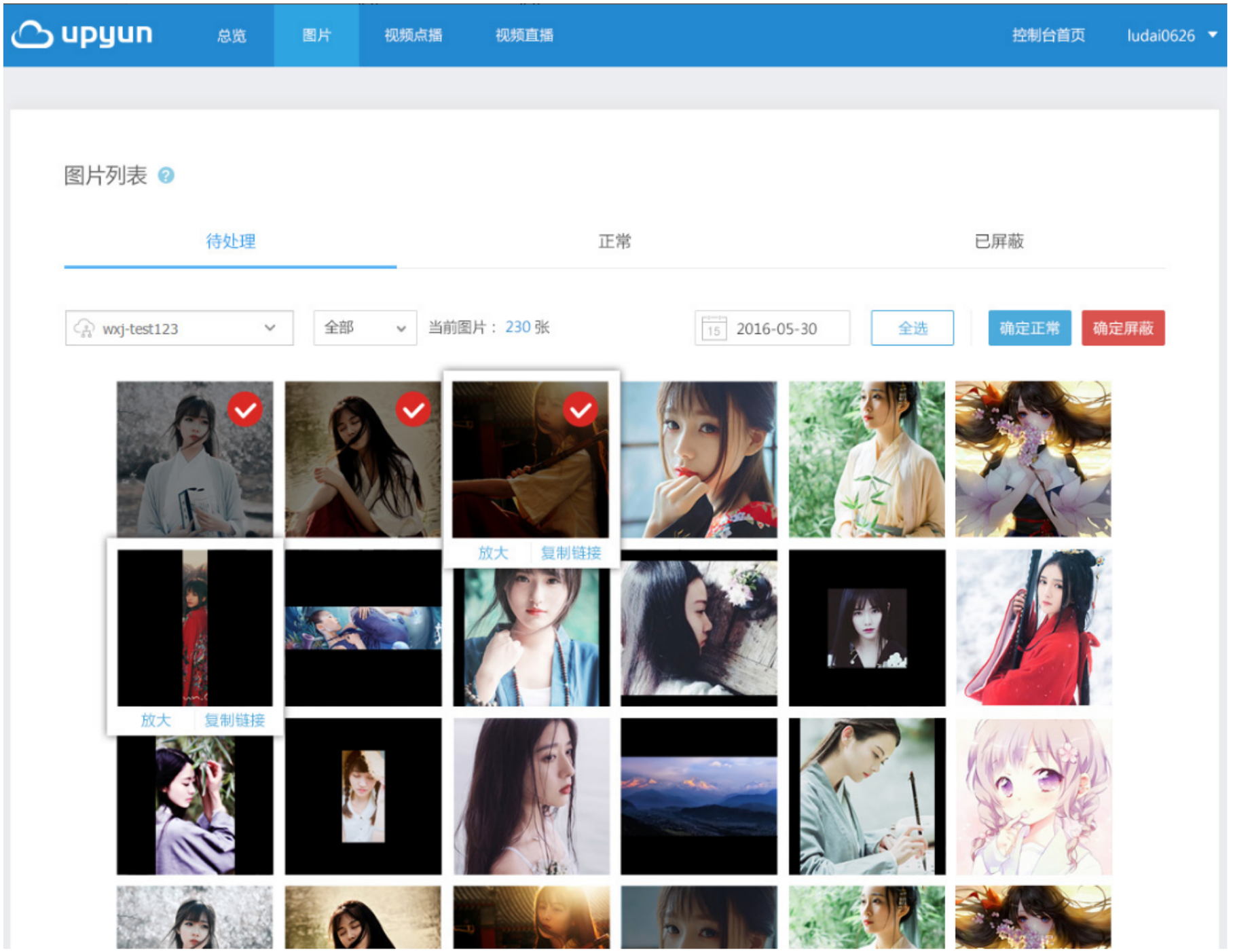
- CPU 模式
  - 简单、不受机器限制
  - 速度慢，单核处理一张图片需要 250ms
  - 异步处理
- GPU 模式
  - CPU 模式 x8, GTX 1060 3G显存
  - mesos: --docker=nvidia-docker
  - 同步处理

一些心得

- Pillow-SIMD 规换 PIL
- export CUDA_VISIBLE_DEVICES=0,1
- 样本越多越好
- batch_size
- base_lr, weight_decay 等超参设定
  - 不能太大，否则会发散；也不能太小
- **try again, again and again!**

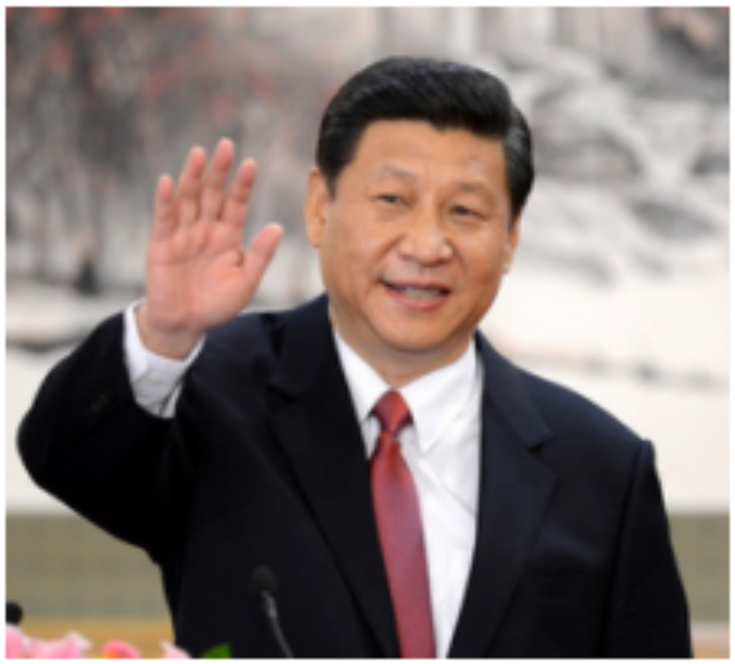


图片鉴黄


upyun
总览
图片
视频点播
视频直播
控制台首页
ludai0626
图片列表
待处理
正常
已屏蔽
wxj-test123
全部
当前图片： 230 张
2016-05-30
全选
确定正常
确定屏蔽
放大
复制链接
放大
复制链接

人脸识别

图片：



检测结果：

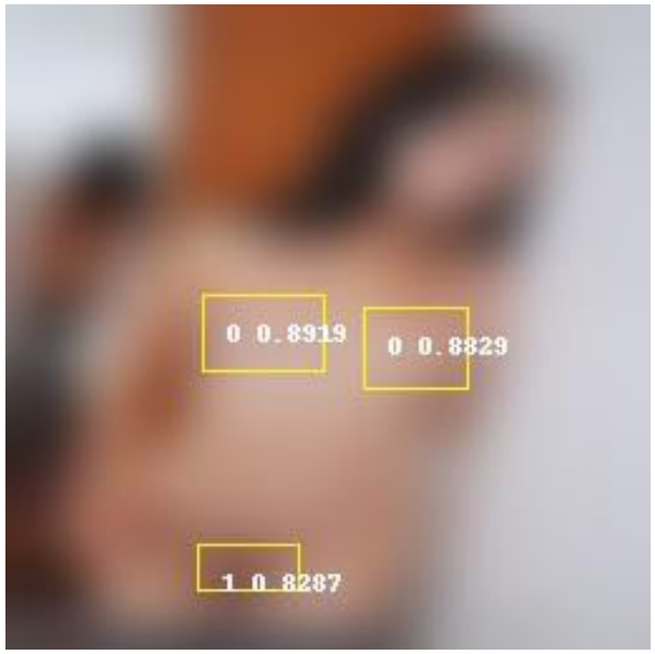
```
人物： 习近平
相似度(越小越相似)： 0.3339
用时： 699.0ms
```

图片：

检测结果：

```
人物： 李克强
相似度(越小越相似)： 0.4036
用时： 476.0ms
```

图片马赛克



- 目标检测（Object Detection）
- R-CNN, Fast R-CNN, Faster R-CNN ...

Thanks

Q & A