



西安电子科技大学  
XIDIAN UNIVERSITY



# 深度学习及其实践

主讲人：刘志

导师：杨淑媛

西安电子科技大学智能感知与图像理解教育部重点实验室

陕西省2011大数据智能感知与计算协同创新中心

国际智能感知与先进计算研究中心

2018年06月20日



# 内容提纲

- 深度学习及平台简介
- 图像分类实践
- 目标检测实践
- 总结

临时WIFI、实验代码、  
数据





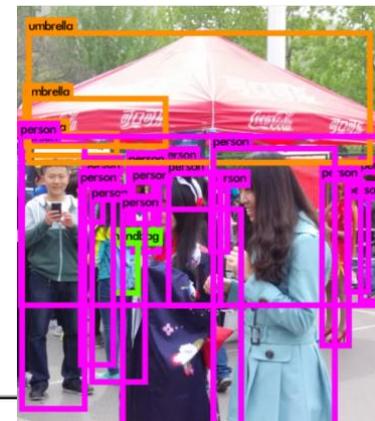
# 内容提纲

- 深度学习及平台简介
- 图像分类实践
- 目标检测实践
- 总结





## 目标检测与识别



YOLO: 训练: VOC2007+2012, 测试: 生活随拍



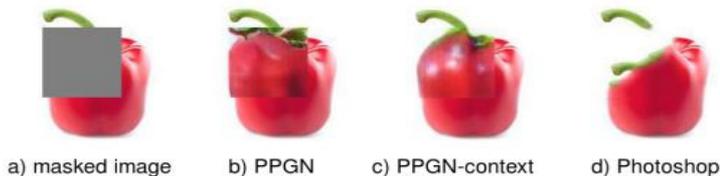
[1] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection[J]. 2016:779-788.

[2] Redmon J, Farhadi A. YOLO9000: Better, Faster, Stronger[J]. 2016.

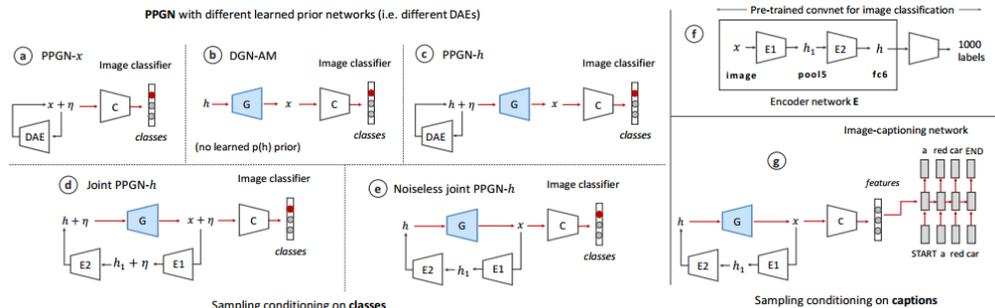




## 图像修复、着色、画风转移



a) masked image    b) PPGN    c) PPGN-context    d) Photoshop



### Image Style Transfer



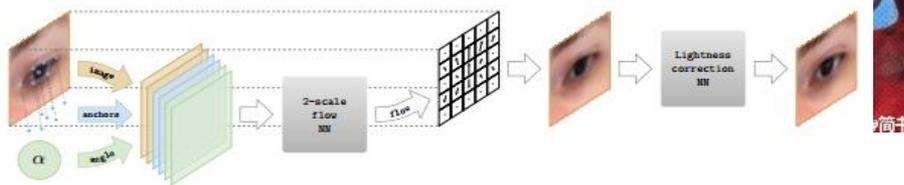
### automatic image colorization



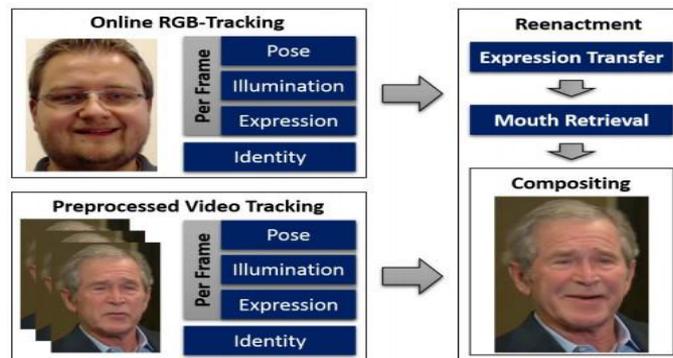


## 图像合成

### DeepWarp



## 角色扮演



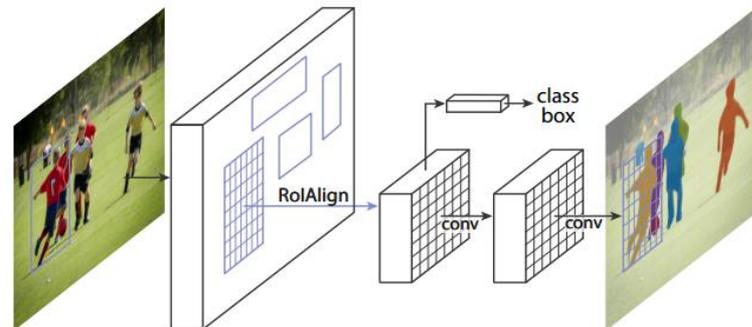
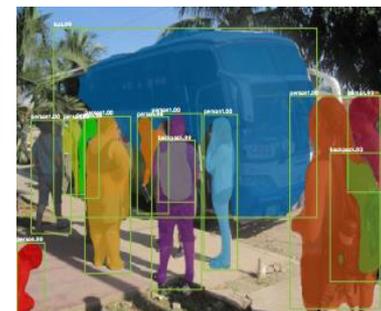
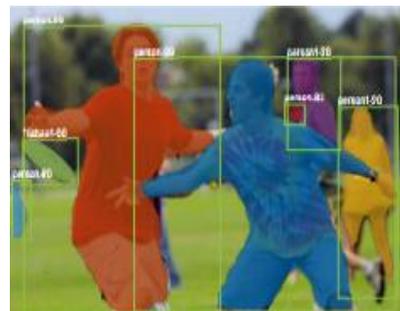
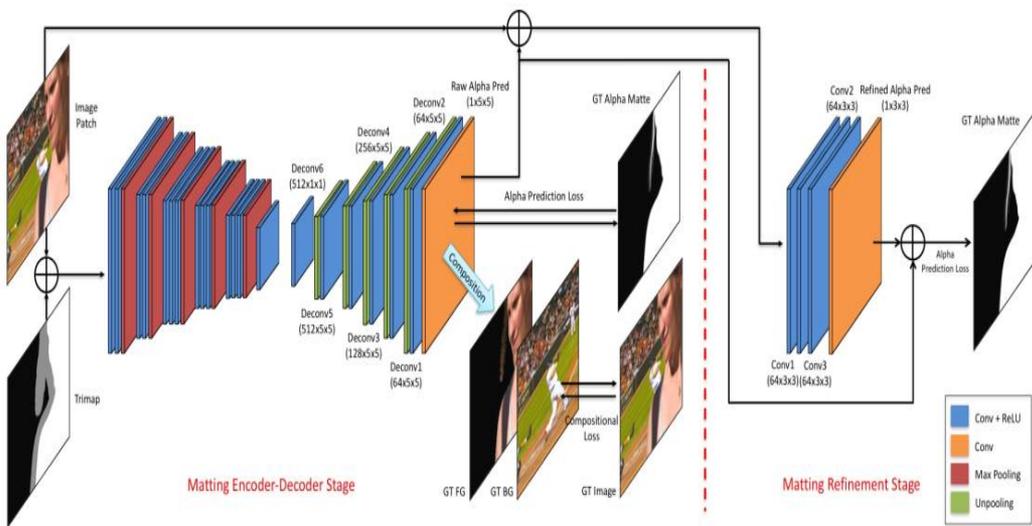
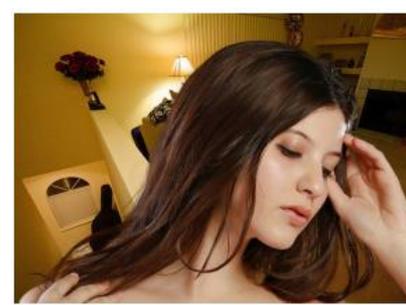
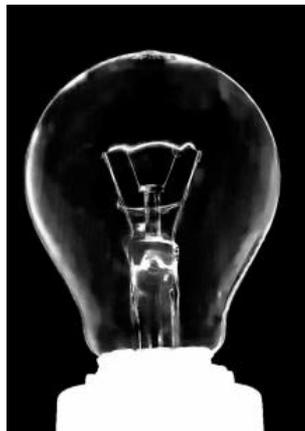
[1] Ganin Y, Kononenko D, Sungatullina D, et al. DeepWarp: Photorealistic Image Resynthesis for Gaze Manipulation[M]// Computer Vision – ECCV 2016. Springer International Publishing, 2016.

[2]Thies J, Zollhofer M, Stamminger M, et al. Face2Face: Real-Time Face Capture and Reenactment of RGB Videos[C]// ACM SIGGRAPH 2016 Emerging Technologies. ACM, 2016:5.





## 从图像级理解到像素级理解-分割与抠图





## 打游戏



## 读懂你的唇语



[1] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning.[J]. Nature, 2015, 518(7540):529-33.

[2] Deep Reinforcement Learning for Flappy Bird

[3] Assael Y M, Shillingford B, Whiteson S, et al. LipNet: End-to-End Sentence-level Lipreading[J]. 2016.





文字



图像

this small bird has a pink breast and crown, and black primaries and secondaries.



this magnificent fellow is almost all black with a red crest, and white cheek patch.



the flower has petals that are bright pinkish purple with white stigma



this white and yellow flower have thin white petals and a round yellow stamen



多伦多大学的 看图讲故事  
(Neural Storyteller) [2]



Later on the eighth day , Billy was a friend of a man who lived on his own . He did n't know how far away they were , and if he was to survive the fall . His mind raced , trying not to show any signs of weakness . The wind ruffled the snow and ice in the snow . He had no idea how many times he was going to climb into the mountains . He told me to stay on the ground for a while , but if I find out what s going on , we should go on foot . Sam and Si Lei joined us in the army .

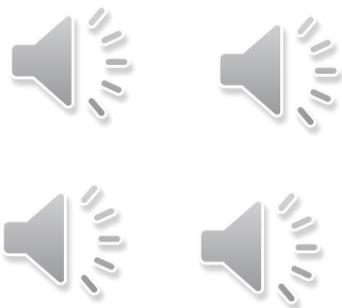
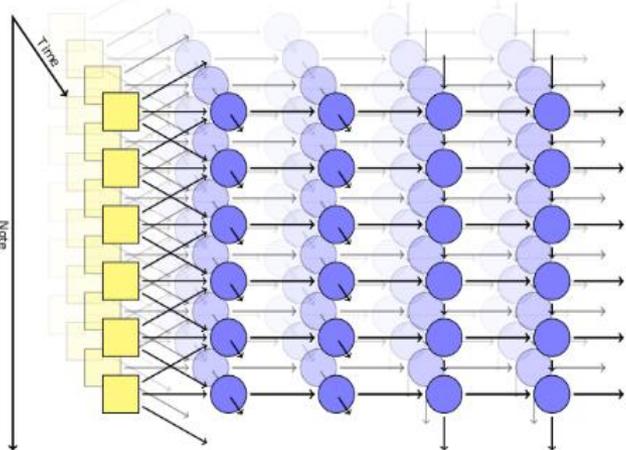
[1] Scott Reed, Zeynep Akata, Xinchun Yan, et al. Generative adversarial text to image synthesis[J]. 2016:1060-1069.

[2] <https://github.com/ryankiros/neural-storyteller>





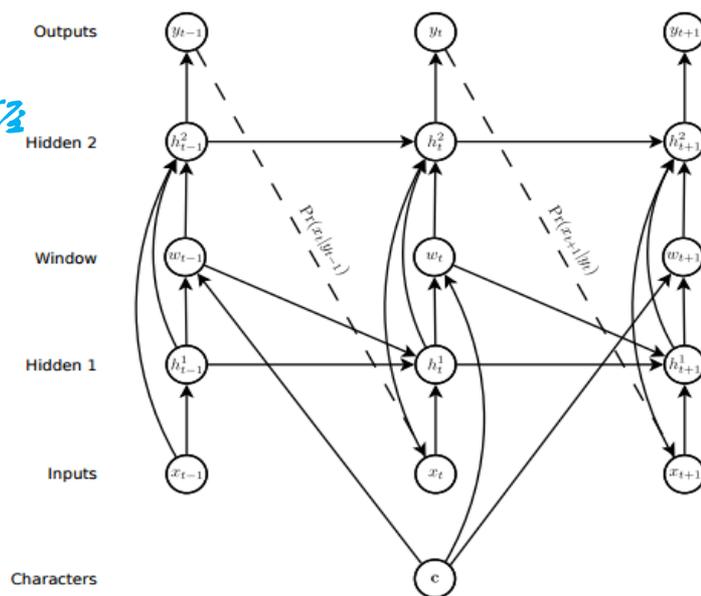
## 音乐创作



使用 Tied Parallel  
Networks 生成和弦音乐

## 书写

使用递归神经网络  
网络书写



## Light-weight Deep Neural Networks

Light-weight Deep Neural Networks

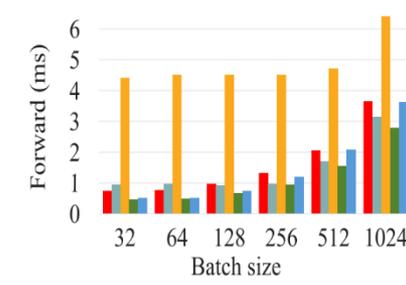
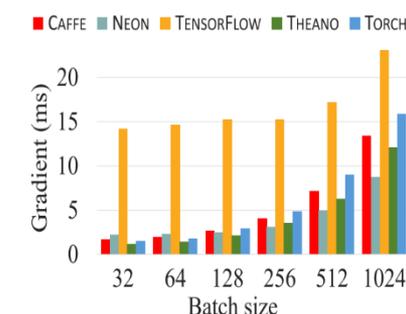
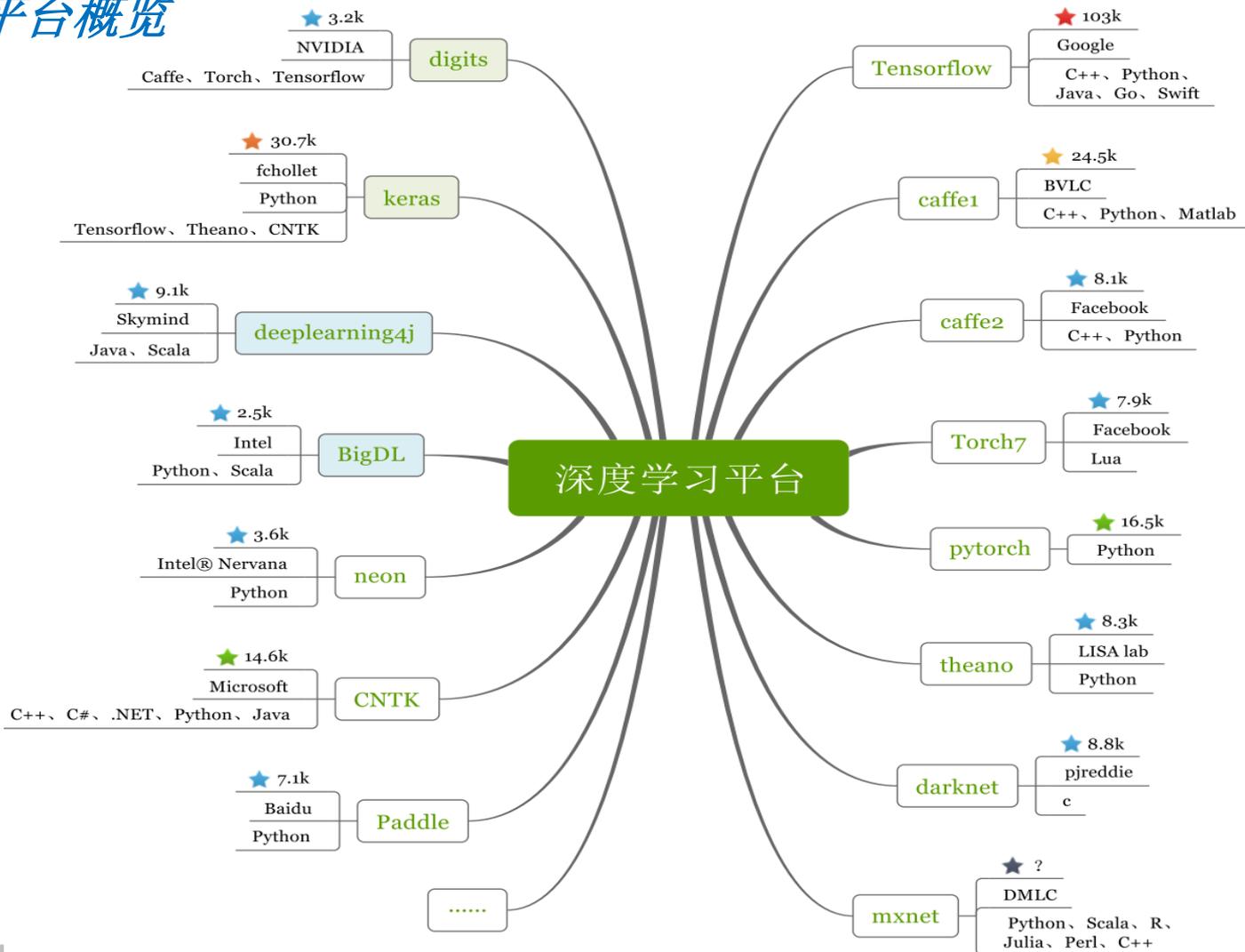
Light-weight Deep Neural Networks

Light-weight Deep Neural Networks





## 平台概览





## TensorFlow

主页: <https://tensorflow.google.cn/>

Python包(CPU): <https://pypi.org/project/tensorflow>

Python包(CPU): <https://pypi.org/project/tensorflow-gpu>

安装CPU版 (python):

```
sudo pip install tensorflow
```

安装GPU版 (python):

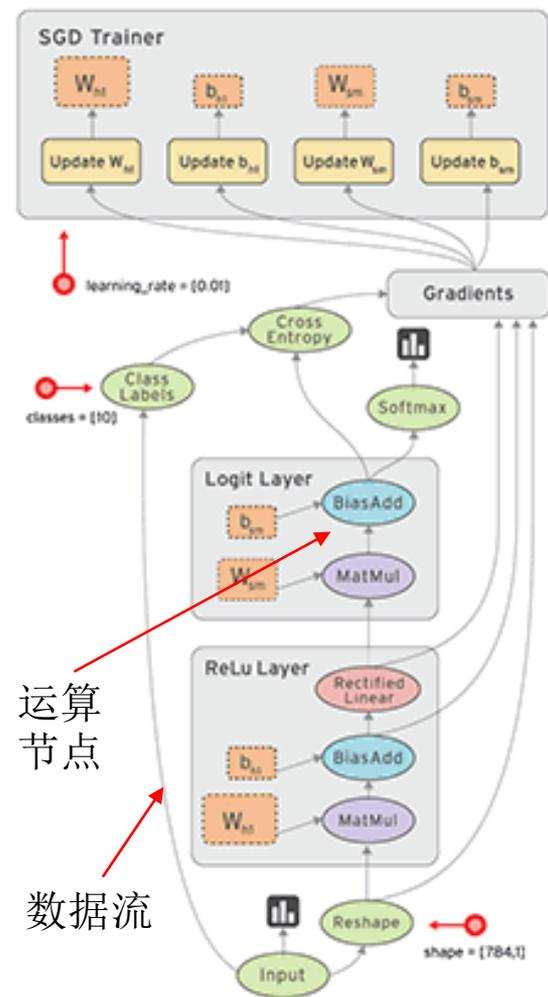
```
sudo pip install tensorflow
```

测试 (python):

```
import tensorflow as tf
```

```
tf.__version__
```

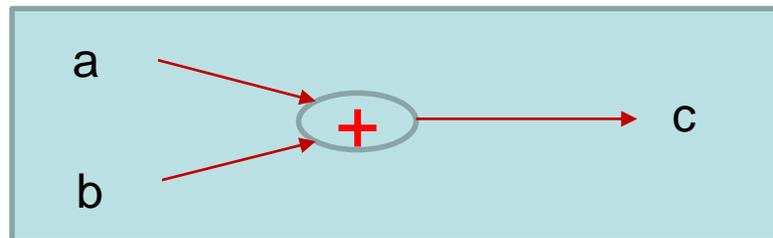
- 基于数据流图
- 灵活性高
- 可移植性强
- 支持自动微分
- 支持多种编程语言
- 安装使用简便
- 文档教程较多





## TensorFlow

$$1 + 2 = 3$$



```
import tensorflow as tf
```

```
# Creates a graph.
```

```
a = tf.constant(1.0, name='a')
```

```
b = tf.constant(2.0, name='b')
```

```
c = tf.add(a, b)
```

```
d = a + b
```

```
print(c)
```

```
print(d)
```

```
# Creates a session
```

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

```
# Runs the op.
```

```
print sess.run(c) → 3.0
```

```
print sess.run(d) → 3.0
```

```
# Creates a graph.
```

```
a = tf.placeholder(tf.float32)
```

```
b = tf.placeholder(tf.float32)
```

```
c = tf.add(a, b)
```

```
d = a + b
```

```
print(c) → Tensor("Add:0", shape=(), dtype=float32)
```

```
print(d) → Tensor("add:0", shape=(), dtype=float32)
```

```
# Creates a session
```

```
with tf.Session() as sess:
```

```
# Runs the op.
```

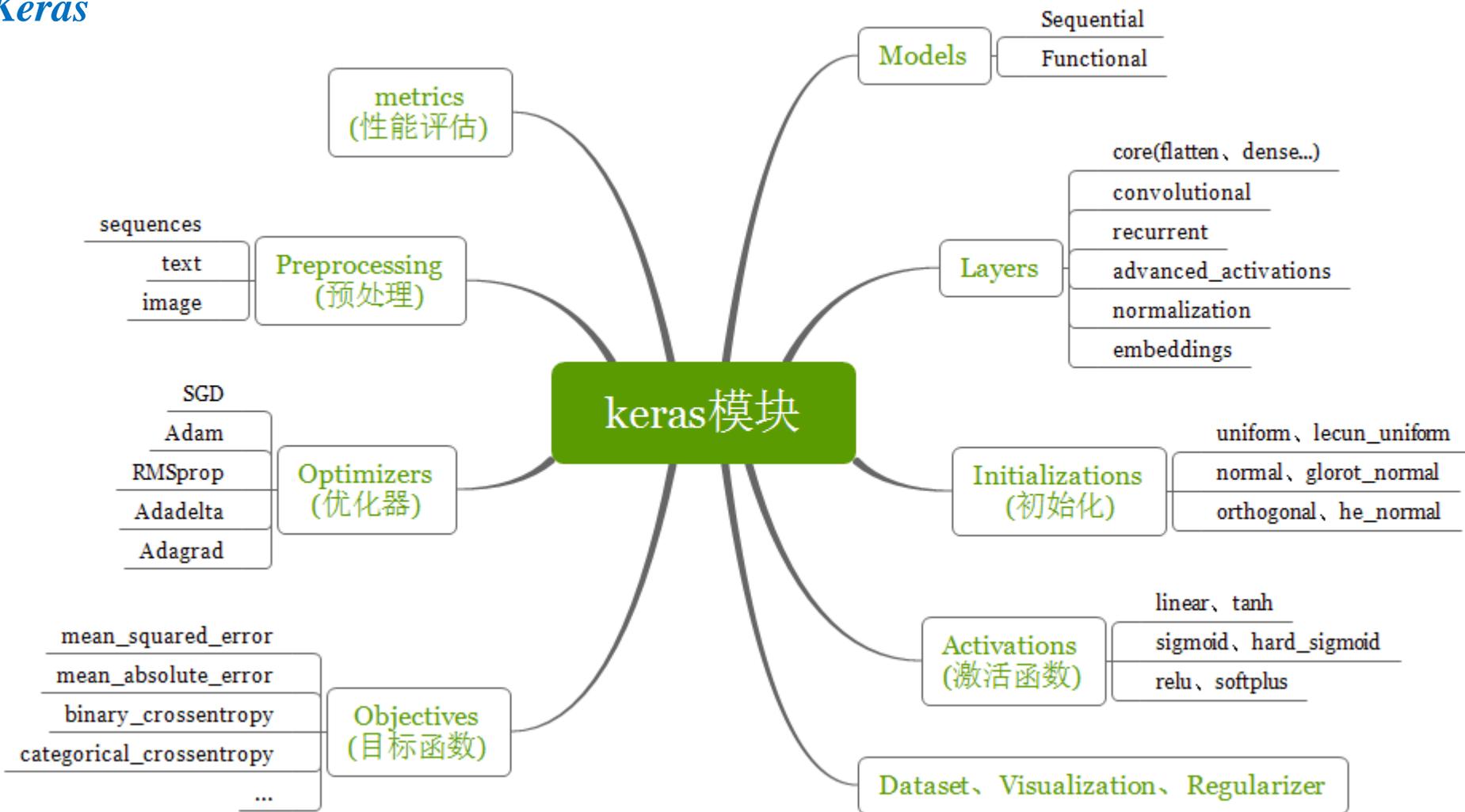
```
print(sess.run(c, feed_dict={a: 1.0, b: 2.0}))
```

```
print(sess.run(d, feed_dict={a: 1.0, b: 2.0}))
```





## Keras





## Keras

## 集成前端

GitHub主页: <https://github.com/keras-team/keras>

英文文档: <https://keras.io/>

中文文档: <http://keras-cn.readthedocs.io/en/latest/>

安装 (python):

```
sudo pip install keras
```

测试 (python):

```
import keras
```

```
Keras.__version__
```

- 支持Tensorflow、Theano、CNTK
- 容易扩展
- 简易和快速的原型设计
- 无缝CPU和GPU切换
- 支持自动微分
- 安装使用简便
- 文档教程较多

```
from keras.models import Sequential  
from keras.layers import Dense, Activation  
from keras.optimizers import SGD
```

```
model = Sequential()
```

```
model.add(Dense(units=64, input_dim=100))  
model.add(Activation("relu"))  
model.add(Dense(units=10))  
model.add(Activation("softmax"))
```

```
model.compile(loss='categorical_crossentropy',  
              optimizer='sgd', metrics=['accuracy'])
```

```
model.compile(loss='categorical_crossentropy',  
              optimizer=SGD(  
                  lr=0.01, momentum=0.9, nesterov=True))
```

```
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

```
model.train_on_batch(x_batch, y_batch)
```

```
loss_and_metrics = model.evaluate(x_test, y_test,  
                                  batch_size=128)
```

```
classes = model.predict(x_test, batch_size=128)
```





## DIGITS 高度集成前端

GitHub主页: <https://github.com/NVIDIA/DIGITS>

英文文档: <https://keras.io/>

中文文档: <http://keras-cn.readthedocs.io/en/latest/>

Caffe



DL4J  
Deeplearning4j

MINERVA

mxnet



K  
KERAS

Microsoft  
CNTK

MatConvNet



theano



### 安装流程:

1. 安装caffe
2. 安装Tensorflow (可选)
3. 安装torch (可选)
4. 安装digits

### 运行:

./digits-devserver

### 浏览器访问:

http://localhost:5000/

- ✓ 功能概览
- ✓ 预处理
- ✓ 分类
- ✓ 目标识别
- ✓ 分割
- ✓ 可视化及分析





# 内容提纲

- 深度学习及平台简介
- 图像分类实践
- 目标检测实践
- 总结





## 基于LeNet的手写体分类

### exp01\_LeNet\_MNIST

**数据集：**手写体数据集 MNIST：含 0~9 十个数字，60000个训练样本，10000个测试样本；

**分类模型：**LeNet-5，一种卷积神经网络。依次包含一个输入层（INPUT）、卷积层（C1）、池化层，也叫下采样层（S2）、卷积层（C3）、下采样层（S4）、卷积层（C5）、全连接层（F6）和一个输出层，由于有0~9个数字，所以输出层的神经元的个数是10。

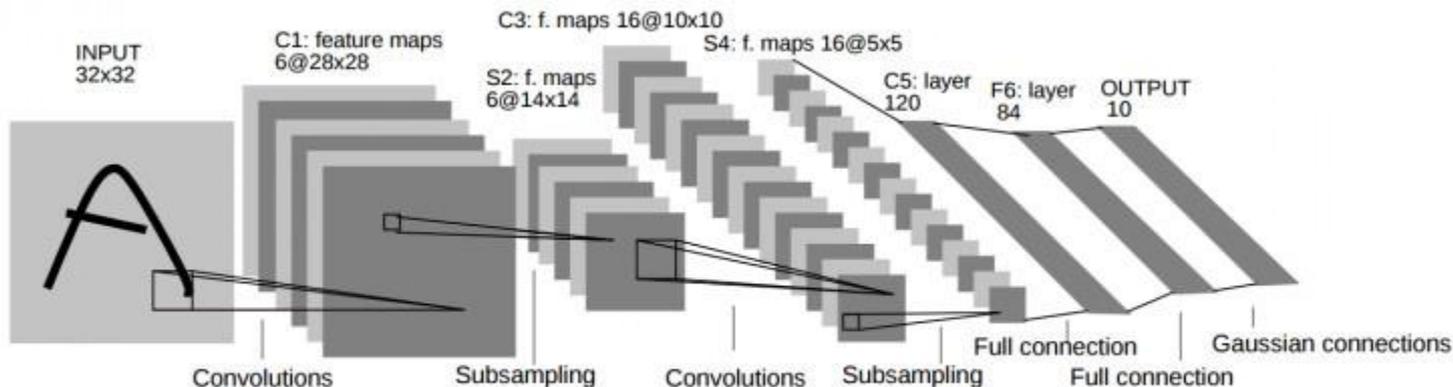
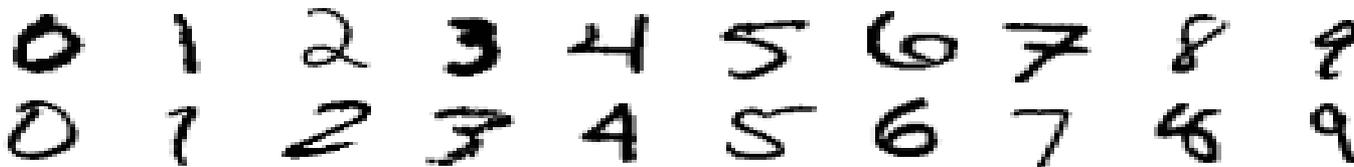


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.





## 基于LeNet的手写体分类 (TF)

exp01\_LeNet\_MNIST/tf

```
batch_size = 64
steps = 10000

# data N-H-W-C
x = tf.placeholder("float", shape=[None, 28, 28, 1])
# label
y_ = tf.placeholder("float", shape=[None, 10])

# 1st layer
conv1_w = tf.get_variable(
    "conv1_w", [5, 5, 1, 32],
    initializer=tf.truncated_normal_initializer(stddev=0.1))
conv1_b = tf.get_variable(
    "conv1_b", [32], initializer=tf.constant_initializer(0.0))
conv1 = tf.nn.conv2d(x, conv1_w, strides=[1, 1, 1, 1], padding='SAME')
relu = tf.nn.relu(tf.nn.bias_add(conv1, conv1_b))

# 2st layer: k=2, stride=2, pad=0
pool1 = tf.nn.max_pool(relu, ksize=[1, 2, 2, 1], strides=[
    1, 2, 2, 1], padding='SAME')
```

```
# 7st layer
# softmax
y_conv = tf.nn.softmax(fc2)

# loss function
cross_entropy = tf.reduce_mean(
    -tf.reduce_sum(y_ * tf.log(y_conv), reduction_indices=[1]))

# Optimizer
# train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)
```

```
step 8832, testing accuracy 0.984375
step 8896, testing accuracy 0.96875
step 8960, testing accuracy 0.984375
step 9024, testing accuracy 0.984375
step 9088, testing accuracy 0.96875
step 9152, testing accuracy 0.96875
step 9216, testing accuracy 0.984375
step 9280, testing accuracy 1
step 9344, testing accuracy 0.96875
step 9408, testing accuracy 0.984375
step 9472, testing accuracy 0.953125
step 9536, testing accuracy 0.984375
step 9600, testing accuracy 1
step 9664, testing accuracy 1
step 9728, testing accuracy 0.984375
step 9792, testing accuracy 1
step 9856, testing accuracy 0.921875
step 9920, testing accuracy 1
step 9984, testing accuracy 0.984375
test accuracy 0.9857
```





## 基于LeNet的手写体分类 (keras)

exp01\_LeNet\_MNIST/keras

Keras代码更为简洁

```
model = Sequential()
model.add(Conv2D(filters=6, kernel_size=(5, 5), padding='valid',
                input_shape=(1, 28, 28), activation='tanh'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(filters=16, kernel_size=(5, 5),
                padding='valid', activation='tanh'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='tanh'))
model.add(Dense(84, activation='tanh'))
model.add(Dense(10, activation='softmax'))
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(optimizer=sgd, loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(trainData, trainLabels, batch_size=64,
          epochs=30, verbose=1, shuffle=True)

plot_model(model, to_file='model.png',
            show_shapes=True, show_layer_names=False)
```

```
ETA: 0s - loss: 0.0497 - acc: 0.9835
ETA: 0s - loss: 0.0493 - acc: 0.9836
ETA: 0s - loss: 0.0496 - acc: 0.9836
ETA: 0s - loss: 0.0493 - acc: 0.9837
ETA: 0s - loss: 0.0496 - acc: 0.9836
ETA: 0s - loss: 0.0494 - acc: 0.9837
ETA: 0s - loss: 0.0495 - acc: 0.9836
ETA: 0s - loss: 0.0495 - acc: 0.9836
ETA: 0s - loss: 0.0495 - acc: 0.9835
ETA: 0s - loss: 0.0496 - acc: 0.9835
ETA: 0s - loss: 0.0495 - acc: 0.9835
ETA: 0s - loss: 0.0496 - acc: 0.9835
ETA: 0s - loss: 0.0494 - acc: 0.9835
ETA: 0s - loss: 0.0493 - acc: 0.9835
ETA: 0s - loss: 0.0492 - acc: 0.9836
ETA: 0s - loss: 0.0492 - acc: 0.9835
ETA: 0s - loss: 0.0493 - acc: 0.9835
ETA: 0s - loss: 0.0494 - acc: 0.9835
ETA: 0s - loss: 0.0498 - acc: 0.9833
ETA: 0s - loss: 0.0510 - acc: 0.9829
ETA: 0s - loss: 0.0521 - acc: 0.9826
4s 60us/step - loss: 0.0521 - acc: 0.9826
```





## 基于LeNet的手写体分类 (digits)

### 1. 创建数据集

**New Image Classification Dataset**

这里填写图像信息

Image Type: Grayscale

Image size (Width x Height): 28 x 28

Resize Transformation: Squash (图像预处理方式)

Use Image Folder | Use Text Files (数据集信息)

Training Images: /DataSets/mnist/train (填写你的数据集路径)

Minimum samples per class: 2 (每类最小采样样本数)

Maximum samples per class: (每类最大采样样本数 (留白))

% for validation: 25 (训练集中用于验证的百分比)

% for testing: 0 (训练集与测试集分开)

Separate validation images folder

Separate test images folder

Test Images: DataSets/mnist/test (填写你的测试集路径)

Minimum samples per class: 2

Maximum samples per class: 1

DB backend: LMDB

Image Encoding: PNG (lossless)

Group Name:

Dataset Name: MNIST (起个名字)

Create (创建数据集任务)

exp01\_LeNet\_MNIST/digits

MNIST  
Owner: charming

Clone Job Delete Job

Job Information

Job Directory: /var/lib/digits/jobs/20170613-213539-ct64

Image Dimensions: 28x28 (Width x Height)

Image Type

Job Status Done

- Initialized at 09:35:39 PM (1 second)
- Running at 09:35:41 PM (31 seconds)
- Done at 09:36:12 PM (Total - 32 seconds)

Create DB (train)

Input File (before shuffling): train.txt

DB Creation log file: create\_train\_db.log

Image Count

Image Mean: 3

Explore the db

类别





## 基于LeNet的手写体分类 (digits)

exp01\_LeNet\_MNIST/digits

### 2. 创建模型

创建分类模型，命名为：MNIST Classification，选择LeNet网络，设置训练参数。

### New Image Classification Model

**选择数据集**

Select Dataset

MNIST

MNIST Details:

- Done 09:36:12 PM
- Image Size: 28x28
- Image Type: GRAYSCALE
- DB backend: Imdb
- Create DB (train): 45002 images
- Create DB (val): 14998 images
- Create DB (test): 10000 images

**Solver Options**

Training epochs: 30 (训练代数)

Snapshot interval (in epochs): 1 (每隔多少代一次快照, 即每隔几代保存一次训练模型)

Validation interval (in epochs): 1 (验证间隔)

Random seed: [none]

Batch size: [network defaults] (批大小, 即每次送入网络的图片数量)

Batch Accumulation: [none]

Solver type: Stochastic gradient descent (SG) (优化方法)

Base Learning Rate: 0.01 (学习率)

Show advanced learning rate options:

**Data Transformations**

Subtract Mean: Image

Crop Size: none

**Python Layers**

Server-side file:

Use client-side file:

### 选择配置分类网络

Standard Networks Previous Networks Pretrained Networks Custom Network (这里可以自定义网络)

Network	Details	Intended image size
LeNet	Original paper [1998]	28x28 (gray) <a href="#">Customize</a>
AlexNet	Original paper [2012]	256x256
GoogLeNet	Original paper [2014]	256x256

输入 GPU ID:

或者选择GPU

Use this many GPUs (next available):

or

Select which GPU[s] you would like to use

- #0 - GeForce GTX 1080 Ti (10.9 GB memory)
- #1 - GeForce GTX 1080 (7.92 GB memory)

Group Name:

Model Name: MNIST Classification

**Create** 创建分类模型任务 (job)





## 基于LeNet的手写体分类 (digits)

### 3. 训练模型

exp01\_LeNet\_MNIST/digits

MNIST Classification [✕](#)  
Owner: charming

[Clone Job](#) [Abort Job](#) [Delete Job](#)

**Job Directory**  
/var/lib/digits/jobs/20170613-221731-99ec  
Disk Size  
0 B  
Network (train/val)  
train\_val.protobxt  
Network (deploy)  
deploy.protobxt  
Network (original)  
original.protobxt  
Solver  
solver.protobxt  
Raw caffe output  
caffe\_output.log

**Dataset**

MNIST  
Done 09:36:12 PM

Image Size  
28x28  
Image Type  
GRAYSCALE  
DB backend  
lmdb  
Create DB (train)  
45002 images  
Create DB (val)  
14998 images  
Create DB (test)  
10000 images

**Job Status Running**

- Initialized at 10:17:31 PM (1 second)
- Running at 10:17:32 PM

Train Caffe Model Running ▾

60%

Estimated time remaining: 19 seconds

- Initialized at 10:17:31 PM (1 second)
- Running at 10:17:32 PM

**Hardware**

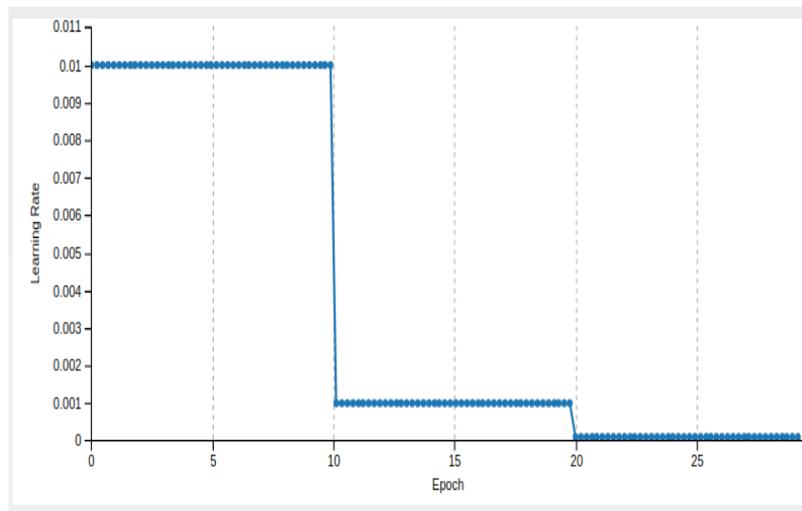
GeForce GTX 1080 Ti (#0)

Memory  
442 MB / 10.9 GB (4.0%)  
GPU Utilization  
50%  
Temperature  
55 °C

GeForce GTX 1080 (#1)

Memory  
429 MB / 7.92 GB (5.3%)  
GPU Utilization  
91%  
Temperature  
58 °C

模型创建完成后，DIGITS 开始训练网络，并实时显示训练损失、验证损失、验证正确率、学习率变化、GPU 资源占用等信息。





### 基于LeNet的手写体分类 (digits)

#### 4. 测试模型

**Trained Models**

Select Model: Epoch #30 [Download Model] [Make Pretrained Model]

**Test a single image**

Image Path:  你的路径

Upload image: [Browse...]

Show visualizations and statistics

[Classify One]

单幅图分类, 可以可视化数据、权重、隐藏层输出

**Test a list of images**

Upload Image List: [Browse...]

Accepts a list of filenames or urls (you can use your val.txt file)

Image folder (optional):  你的路径

Relative paths in the text file will be prepended with this value before reading

Number of images use from the file:

Leave blank to use all

[Classify Many]

多幅图像分类 可以查看混淆矩阵等信息

Number of images to show per category:  top 5

[Top N Predictions per Category]

每类的TopN分类预测图示

**MNIST Classification** Image Classification Model

6

数字 6 预测结果

Predictions	Percentage
6	100.0%
0	0.0%
5	0.0%
4	0.0%
8	0.0%

**data**

Activation

标准化的数据

**Statistics**

Data shape: [ 1 28 28]  
Mean: 10.1212  
Std deviation: 58.5525

**Visualization**

**scaled**

Activation

**Statistics**

Data shape: [ 1 28 28]  
Mean: 0.126515  
Std deviation: 0.731907





exp01\_LeNet\_MNIST/digits

## 基于LeNet的手写体分类 (digits)

### 4. 测试模型

#### MNIST Classification Image Classification Model

Summary

Top-1 accuracy  
99.13%

Top-5 accuracy  
99.99%

Confusion matrix

	0	1	2	3	4	5	6	7	8	9	Per-class accuracy
0	978	0	0	0	0	0	0	2	0	0	99.8%
1	0	1131	0	0	0	1	1	2	0	0	99.65%
2	1	0	1024	0	1	0	1	4	0	1	99.22%
3	0	0	1	1003	0	3	0	1	1	1	99.31%
4	0	0	1	0	975	0	2	0	1	3	99.29%
5	2	0	0	5	0	882	1	0	1	1	98.88%
6	4	2	0	0	1	3	948	0	0	0	98.96%
7	0	2	4	0	0	0	0	1020	1	1	99.22%
8	4	0	2	0	0	2	1	0	962	3	98.77%
9	0	0	0	2	7	5	0	3	2	990	98.12%

All classifications

Path	Ground truth	Top predictions
1 /home/liu/DataSets/mnist/test/7/00000.png	7	7 <span>100.0%</span> 3 <span>0.0%</span> 1 <span>0.0%</span> 4 <span>0.0%</span> 2 <span>0.0%</span>
2 /home/liu/DataSets/mnist/test/2/00001.png	2	2 <span>100.0%</span> 1 <span>0.0%</span> 0 <span>0.0%</span> 6 <span>0.0%</span> 7 <span>0.0%</span>

#### TopN Image Classification

Owner: chaming

Close Job Delete Job

Job Status Done

- Initialized at 10:29:33 PM (1 second)
- Running at 10:29:34 PM (6 seconds)
- Done at 10:29:40 PM (Total - 7 seconds)

Infer Model Done

Notes

None

#### Top N Predictions per Category

Category	Top images for this category
0	00 00 0
1	11 11 1
2	22 22 2
3	33 33 3





# 内容提纲

- 深度学习及平台简介
- 图像分类实践
- 目标检测实践
- 总结



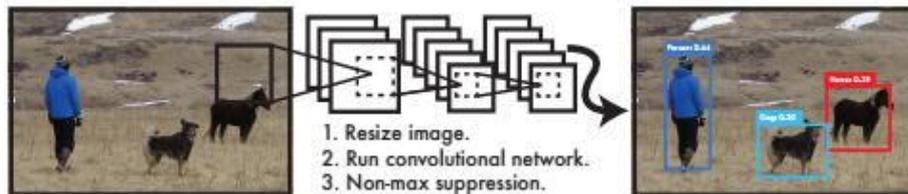


## 基于YOLOv3的目标检测 (darknet平台)

## exp02\_YOLO\_Detection

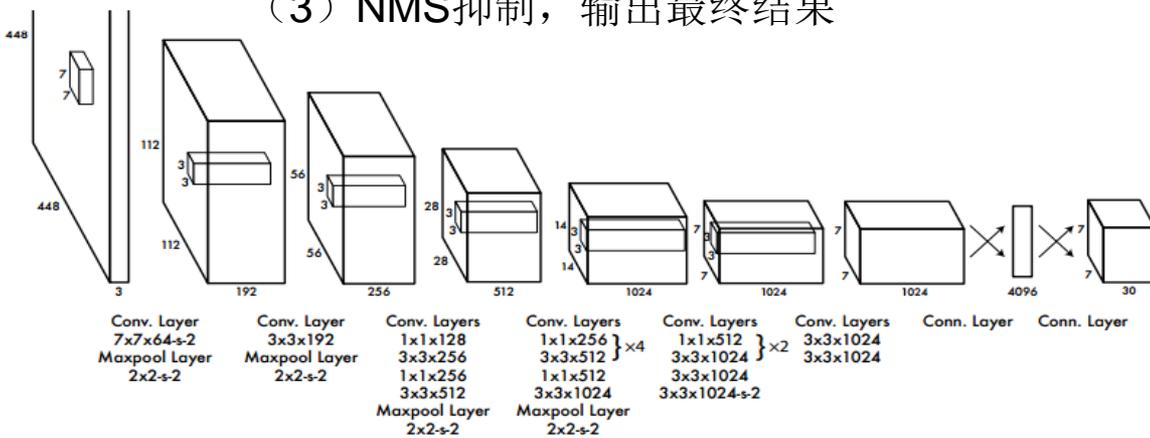
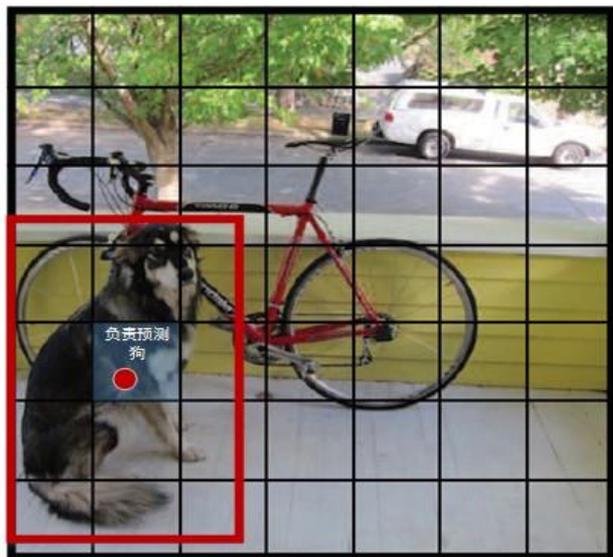
数据集: VOC2007, 含20类, 训练集 (5011幅), 测试集 (4952幅), 共计9963幅图;

网络: YOLOv3, 采用多尺度预测 (类FPN); 采用更好的基础分类网络 (类ResNet) 和分类器。



YOLO利用单个卷积神经网络, 将目标检测问题转换为直接从图像中提取**bounding boxes**和类别概率的回归问题, 分为3个阶段:

- (1) 将图像缩放到448\*448
- (2) 通过神经网络进行检测和分类
- (3) NMS抑制, 输出最终结果





## 基于YOLOv3的目标检测 (darknet平台)

exp02\_YOLO\_Detection

详细教程

Darknet主页:

<https://pjreddie.com/darknet/>

darknet源码下载:

<https://github.com/pjreddie/darknet>

修改配置 (Makefile):

GPU=1

CUDNN=1

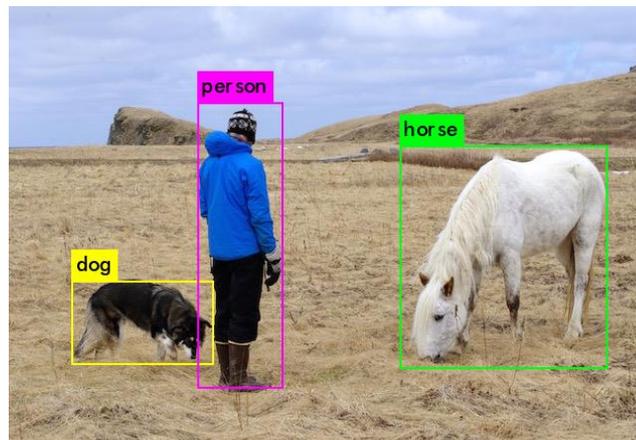
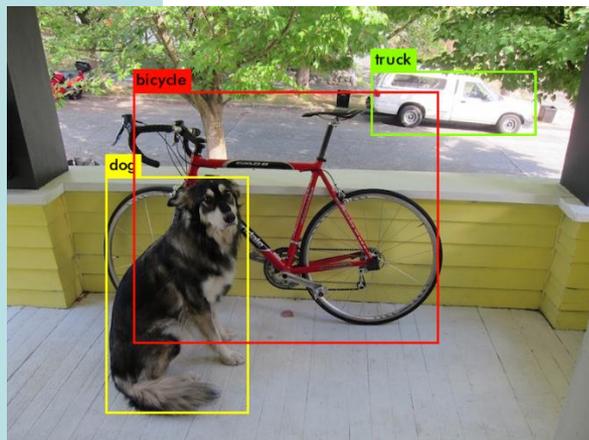
OPENCV=0

OPENMP=0

DEBUG=0

构建安装:

make -j\$(nproc)





# 内容提纲

- 深度学习及平台简介
- 图像分类实践
- 目标检测实践
- 总结





- 简要介绍了深度学习的应用；
- 简要对比了常见深度学习平台；
- 精通一种平台，了解多个平台；
- 具体使用哪个平台，根据需求选择。





谢谢!

敬请批评与指正!

