西安電子科技大學
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

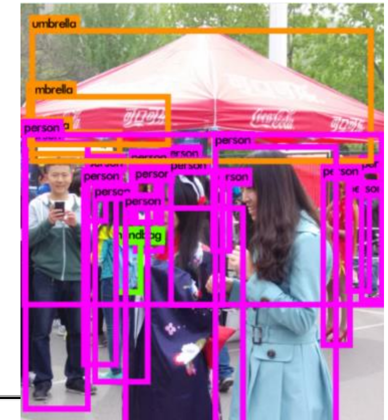
- 深度学习及平台简介
- 图像分类实践
- 目标检测实践
- 总结

临时WIFI、实验代码、数据

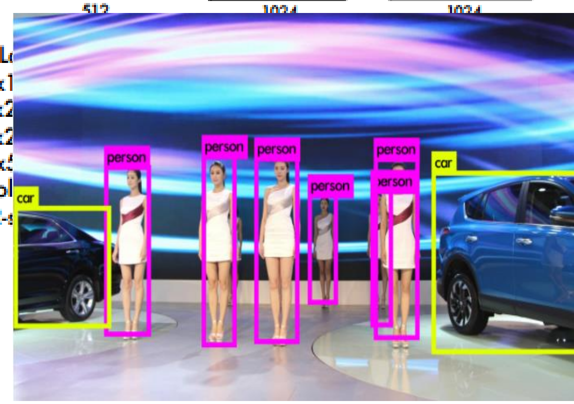
内容提纲

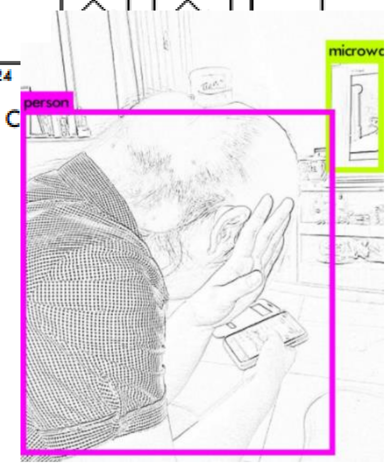
- 深度学习及平台简介
- 图像分类实践
- 目标检测实践
- 总结

目标检测与识别



YOLO：训练：VOC2007+2012，测试：生活随拍

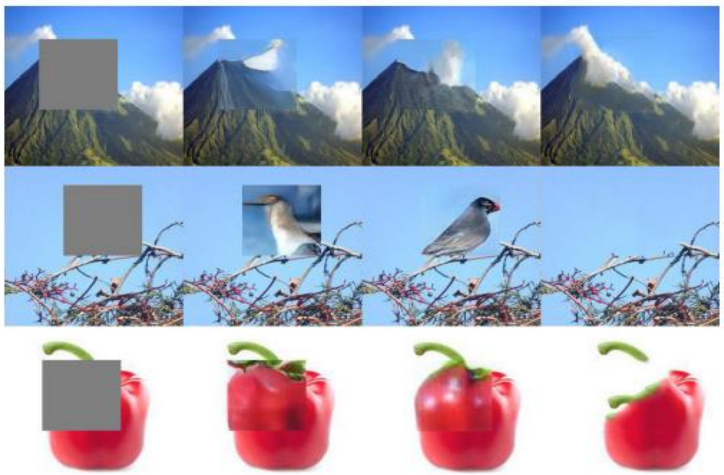




[1] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection[J]. 2016:779-788.
[2] Redmon J, Farhadi A. YOLO9000: Better, Faster, Stronger[J]. 2016.

深度学习及平台简介

图像修复、着色、画风转移



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

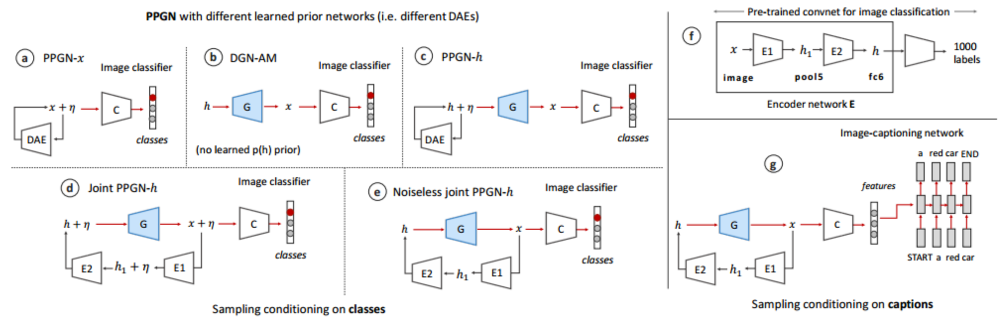


Image Style Transfer

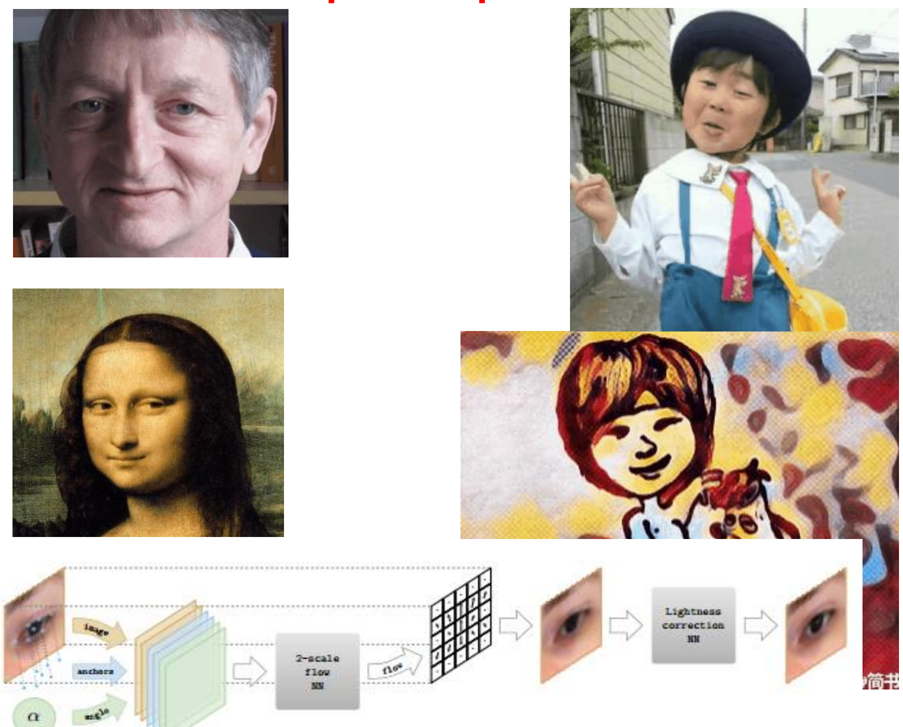
automatic image colorization

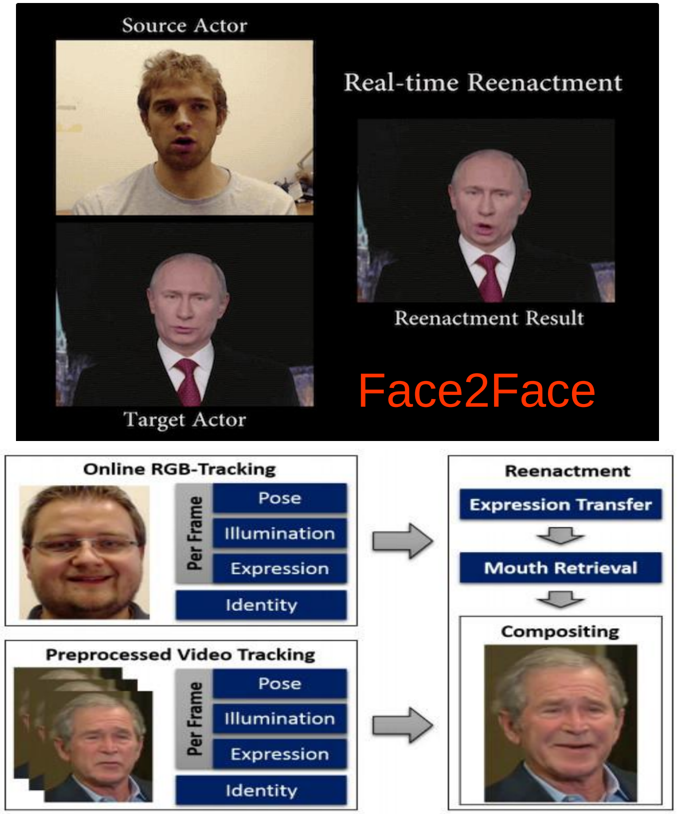
深度学习及平台简介

图像合成

角色扮演

DeepWarp



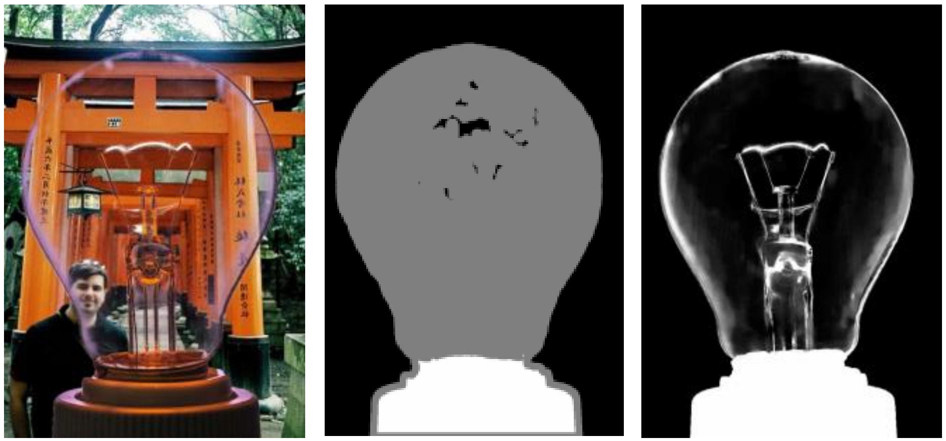


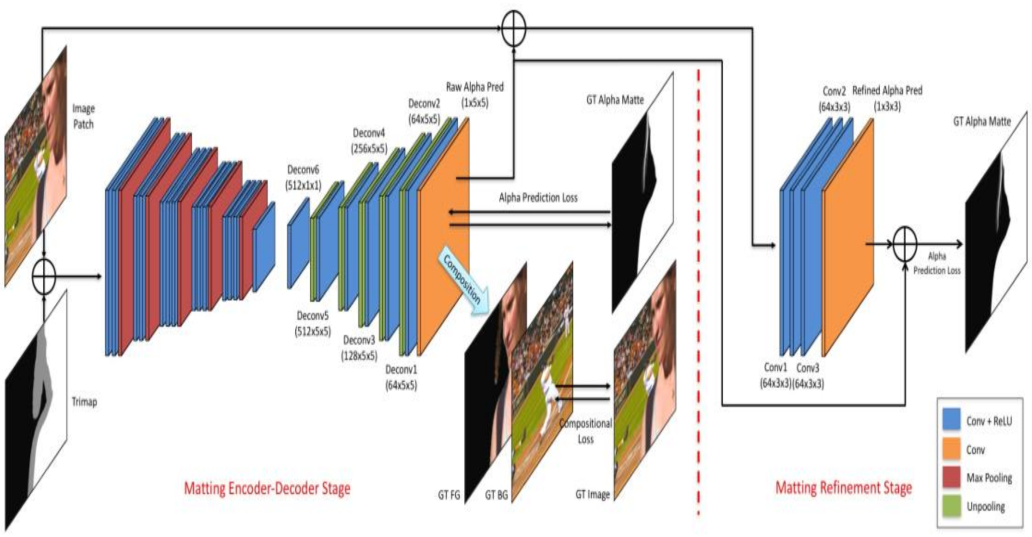
[1] Ganin Y, Kononenko D, Sungatullina D, et al. DeepWarp: Photorealistic Image Resynthesis for Gaze Manipulation[M]// Computer Vision – ECCV 2016. Springer International Publishing, 2016.

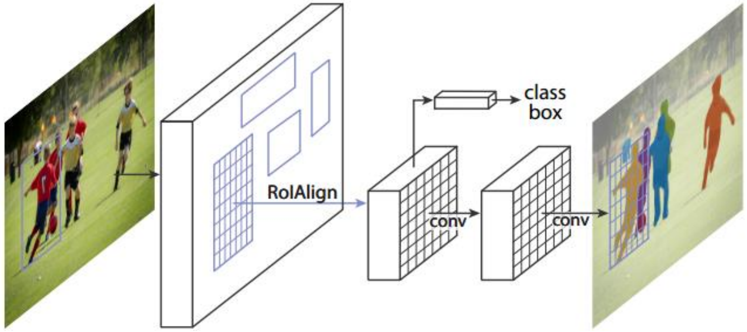
[2]Thies J, Zollhofer M, Stamminger M, et al. Face2Face: Real-Time Face Capture and Reenactment of RGB Videos[C]// ACM SIGGRAPH 2016 Emerging Technologies. ACM, 2016:5.

深度学习及平台简介

从图像级理解到像素级理解-分割与抠图







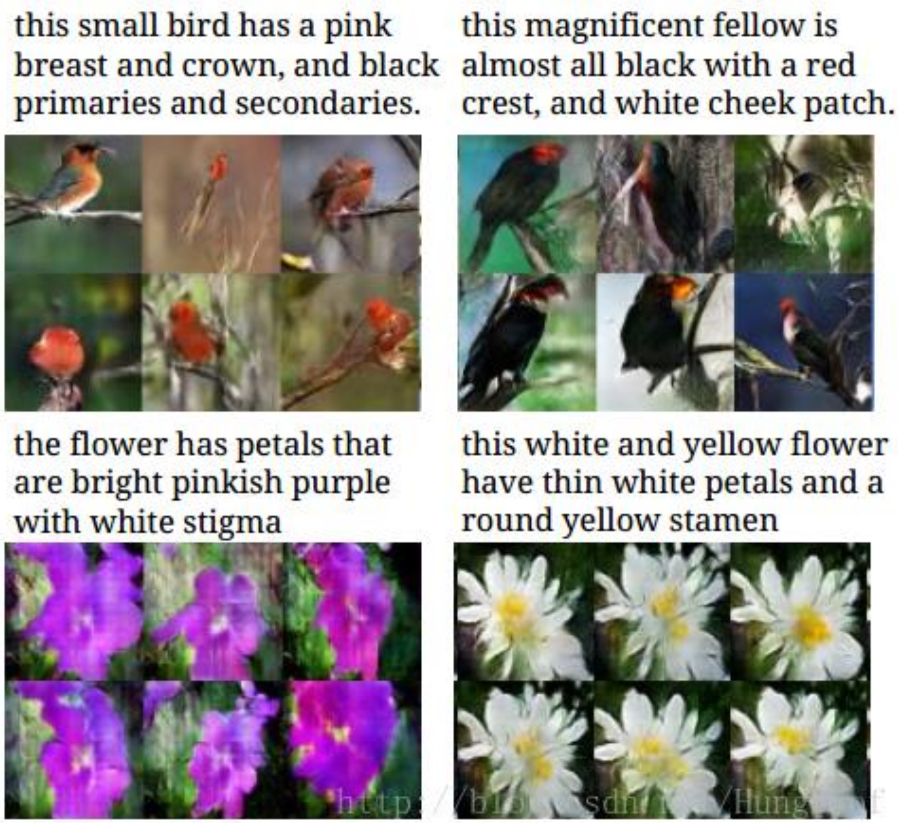
打游戏

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.

深度学习及平台简介

文字 ⟷ 图像



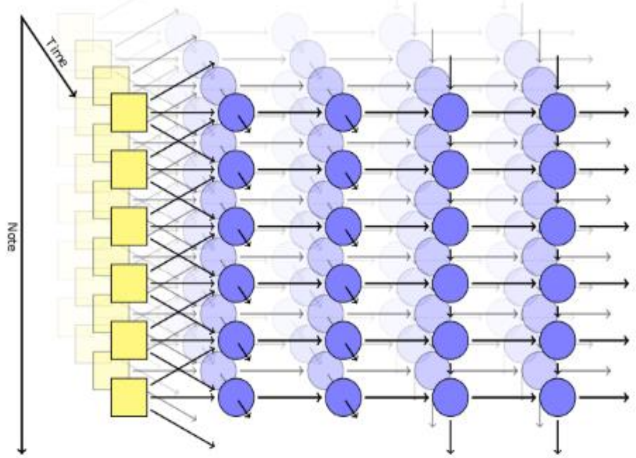
多伦多大学的 **看图讲故事**

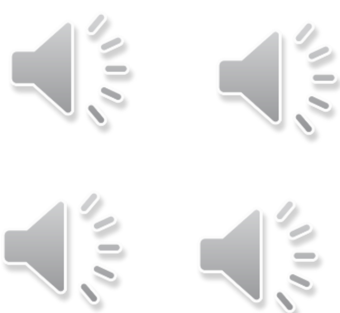
（Neural Storyteller）[2]

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, Xinchen Yan, et al. Generative adversarial text to image synthesis[J]. 2016:1060-1069.
[2] https://github.com/ryankiros/neural-storyteller

音乐创作

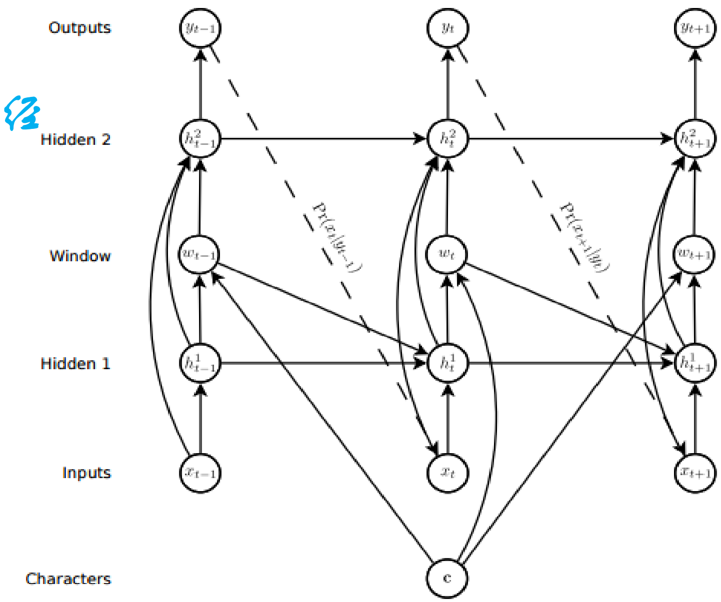




使用 Tied Parallel Networks 生成和弦音乐

书写

使用递归神经网络书写



Light-weight Deep Neural Networks

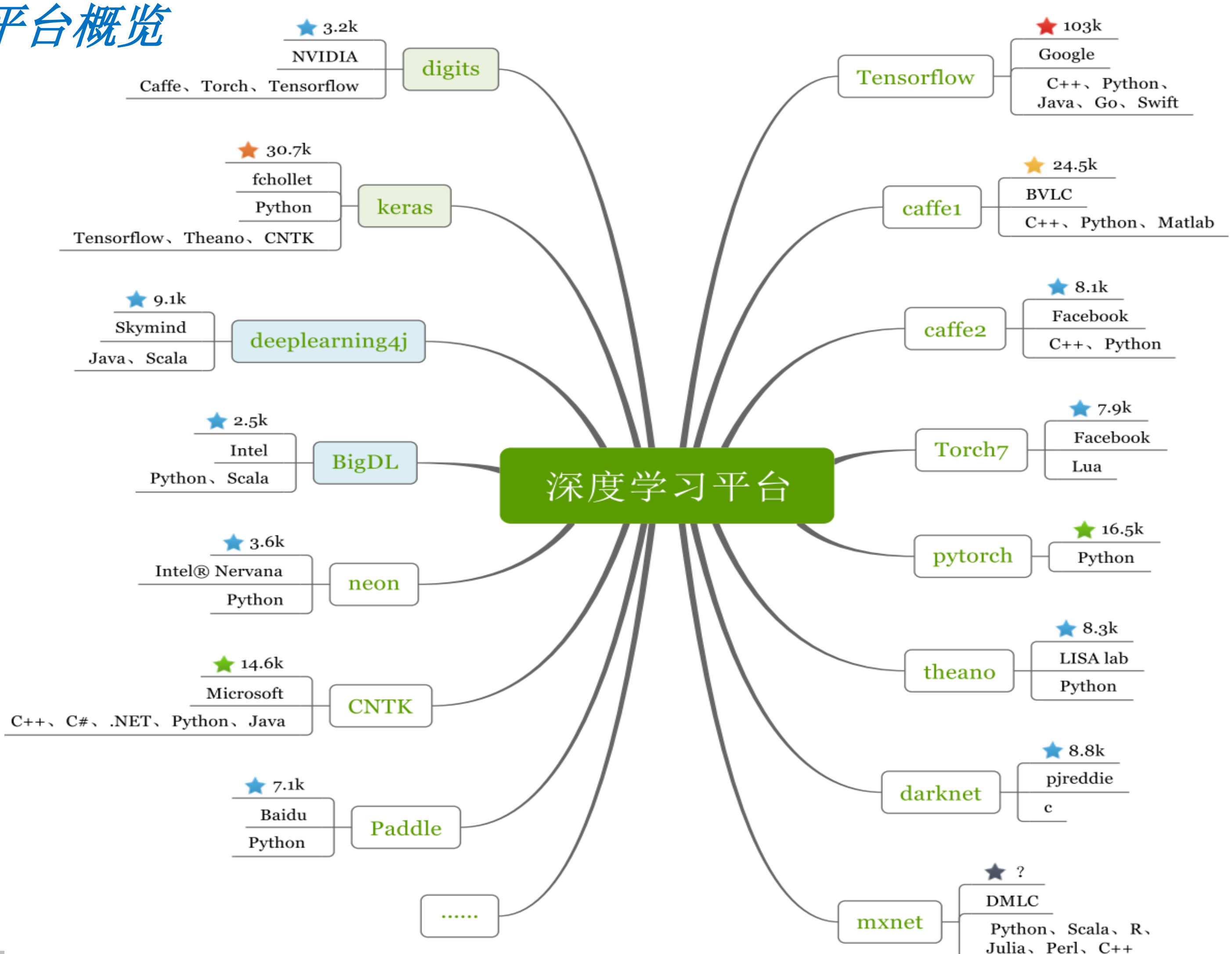
Light-weight Deep Neural Networks

Light-weight Deep Neural Networks

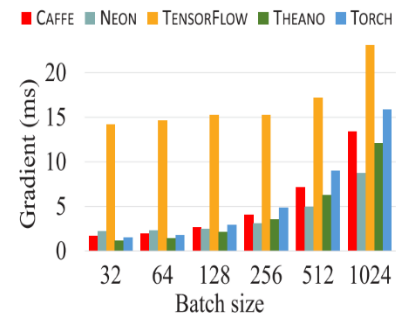
Light-weight Deep Neural Networks

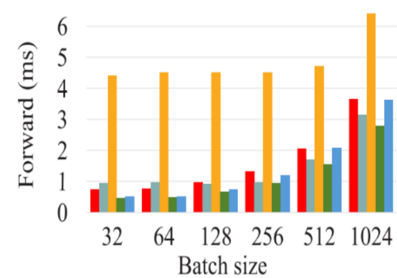
深度学习及平台简介

平台概览



more





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)：

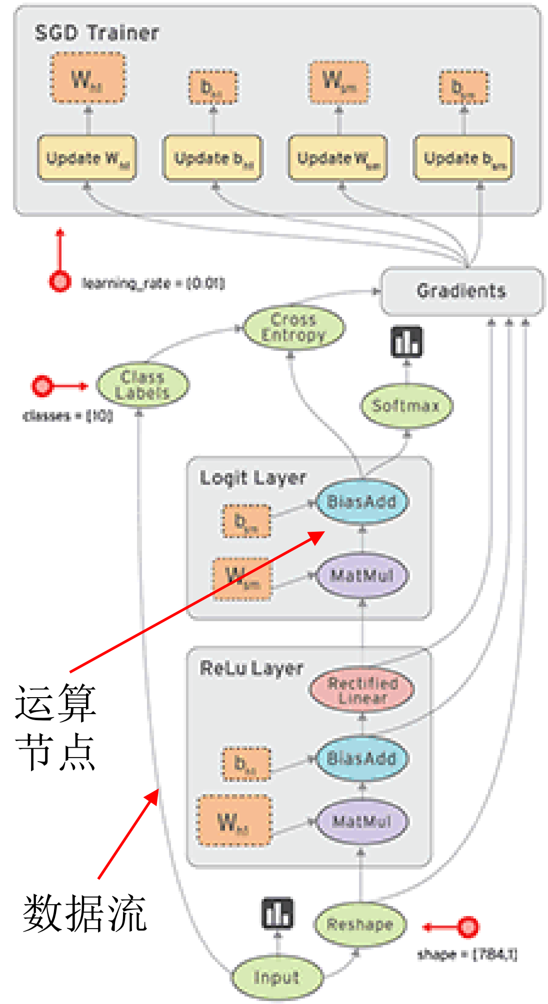
sudo pip install tensorflow

测试(python)：

import tensorflow as tf

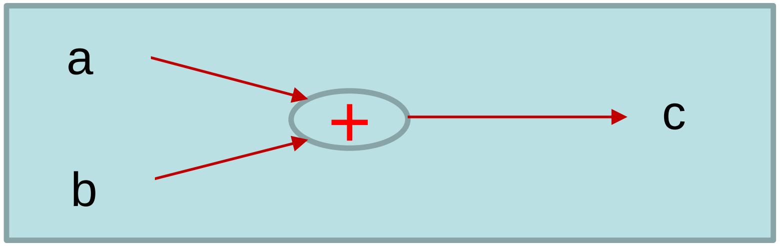
tf.__version__

- 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



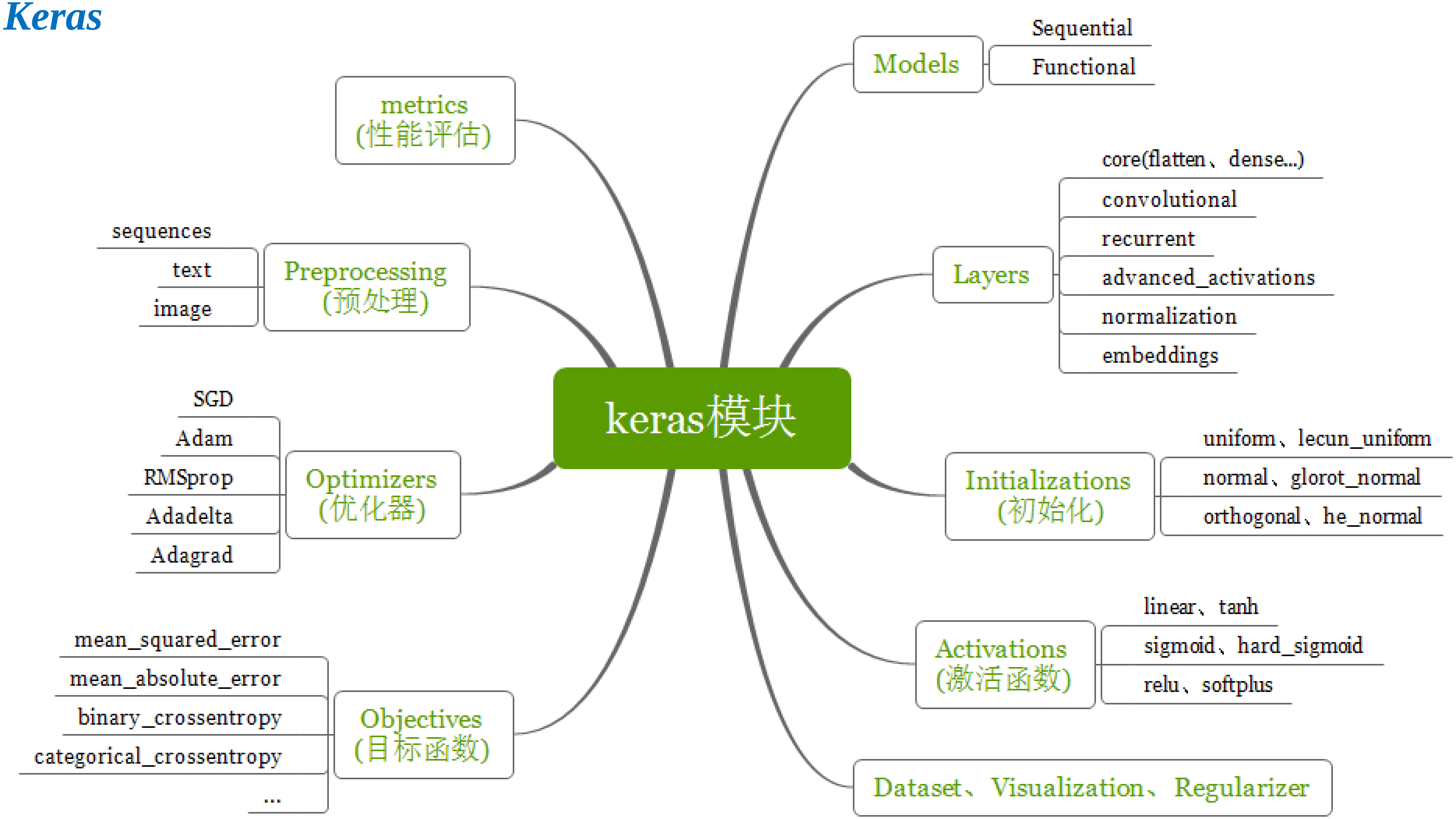

```
import tensorflow as tf

# 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)
print sess.run(d)
```

```
3.0
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)
print(d)

# 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}))
```

```
Tensor("Add:0", shape=(), dtype=float32)
Tensor("add:0", shape=(), dtype=float32)
```

Keras


keras模块
Models
Sequential
Functional
Layers
core(flatten、dense...)
convolutional
recurrent
advanced_activations
normalization
embeddings
Initializations (初始化)
uniform、lecun_uniform
normal、glorot_normal
orthogonal、he_normal
Activations (激活函数)
linear、tanh
sigmoid、hard_sigmoid
relu、softplus
Dataset、Visualization、Regularizer
metrics (性能评估)
Preprocessing (预处理)
sequences
text
image
Optimizers (优化器)
SGD
Adam
RMSprop
Adadelta
Adagrad
Objectives (目标函数)
mean_squared_error
mean_absolute_error
binary_crossentropy
categorical_crossentropy
...

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

Caffe Chainer DL4J Deeplearning4j MINERVA mxnet Purine KERAS Microsoft CNTK MatConvNet TensorFlow theano torch

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/

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

- 第一步：准备并创建数据集
- 第二步：设计并创建任务模型
- 第三步：训练测试模型

内容提纲

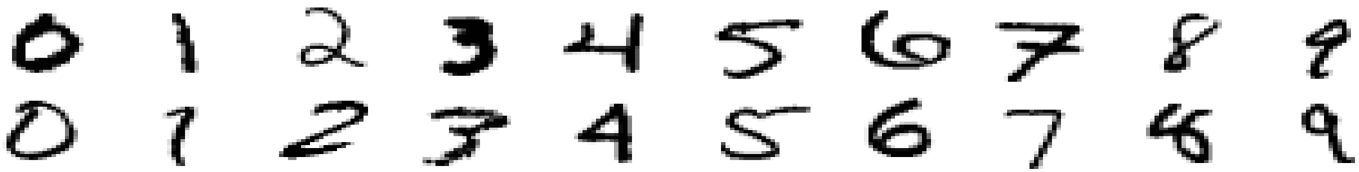
- 深度学习及平台简介
- 图像分类实践
- 目标检测实践
- 总结

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。



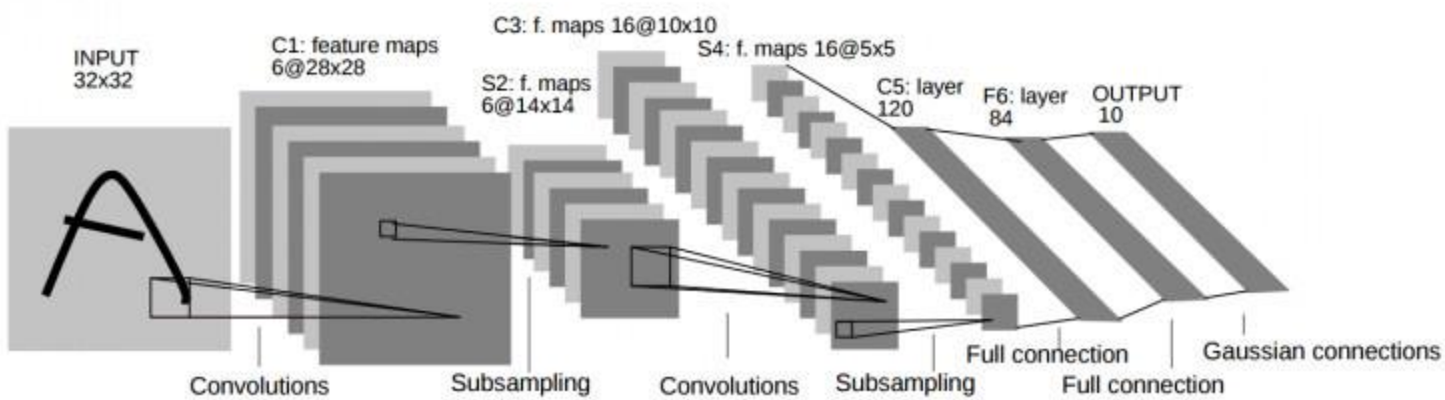


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

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')
relu1 = tf.nn.relu(tf.nn.bias_add(conv1, conv1_b))

# 2st layer: k=2, stride=2, pad=0
pool1 = tf.nn.max_pool(relu1, 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

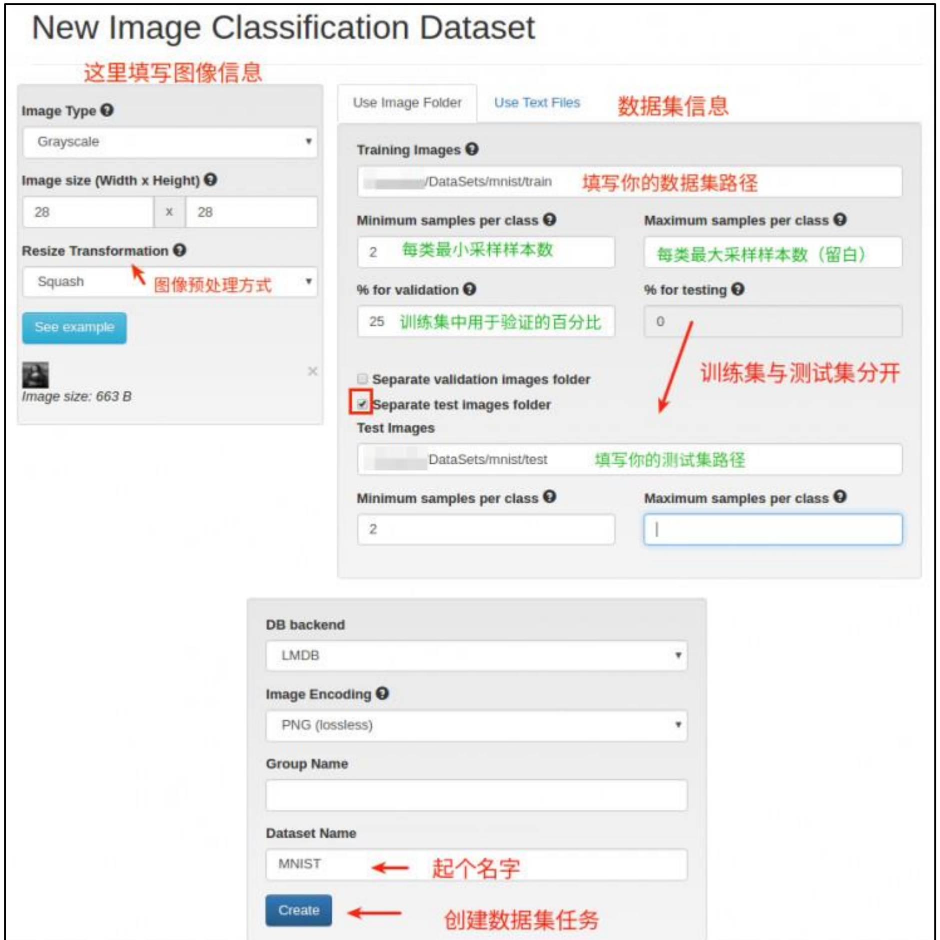
```
model = Sequential()
model.add(Conv2D(filters=6, kernel_size=(5, 5), padding='valid',
                 input_shape=(1, 28, 28), activation='tanh'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(filters=16, kernel_size=(5, 5),
                 padding='valid', activation='tanh'))
model.add(MaxPooling2D(pool_size=(2, 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)
```

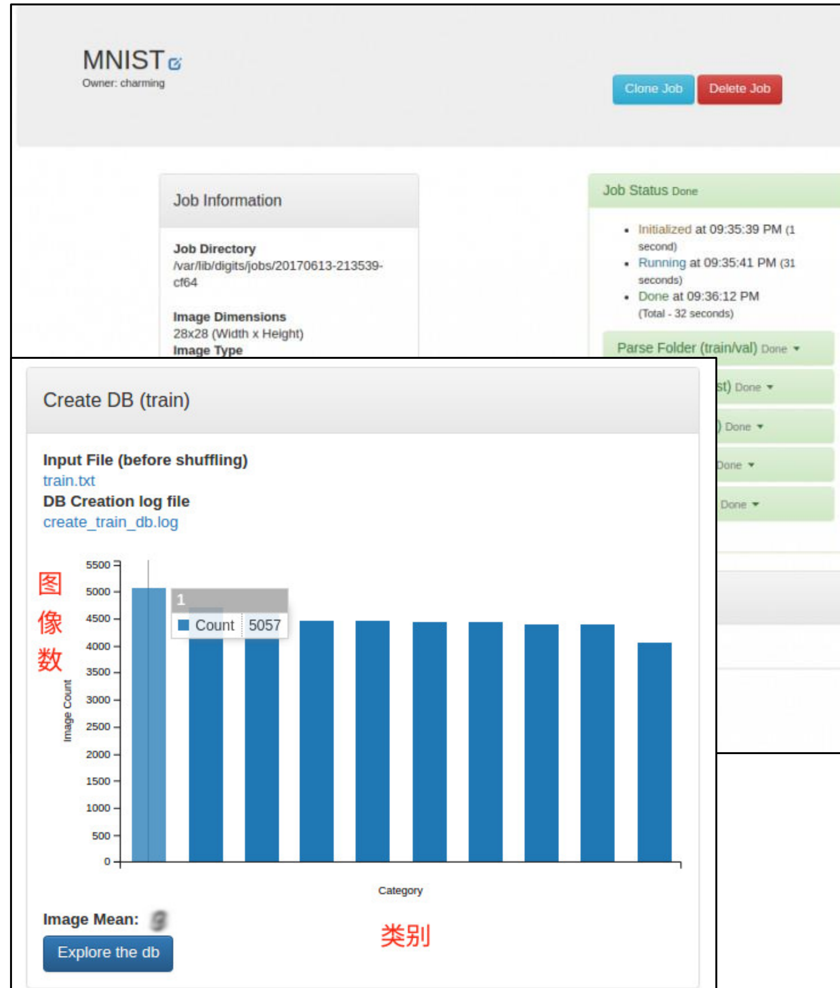
```
ETA: 0s - loss: 0.0497 - acc: 0.9835
ETA: 0s - loss: 0.0493 - acc: 0.9836
ETA: 0s - loss: 0.0496 - acc: 0.9836
ETA: 0s - loss: 0.0493 - acc: 0.9837
ETA: 0s - loss: 0.0496 - acc: 0.9836
ETA: 0s - loss: 0.0494 - acc: 0.9837
ETA: 0s - loss: 0.0495 - acc: 0.9836
ETA: 0s - loss: 0.0495 - acc: 0.9836
ETA: 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）

exp01_LeNet_MNIST/digits

1. 创建数据集



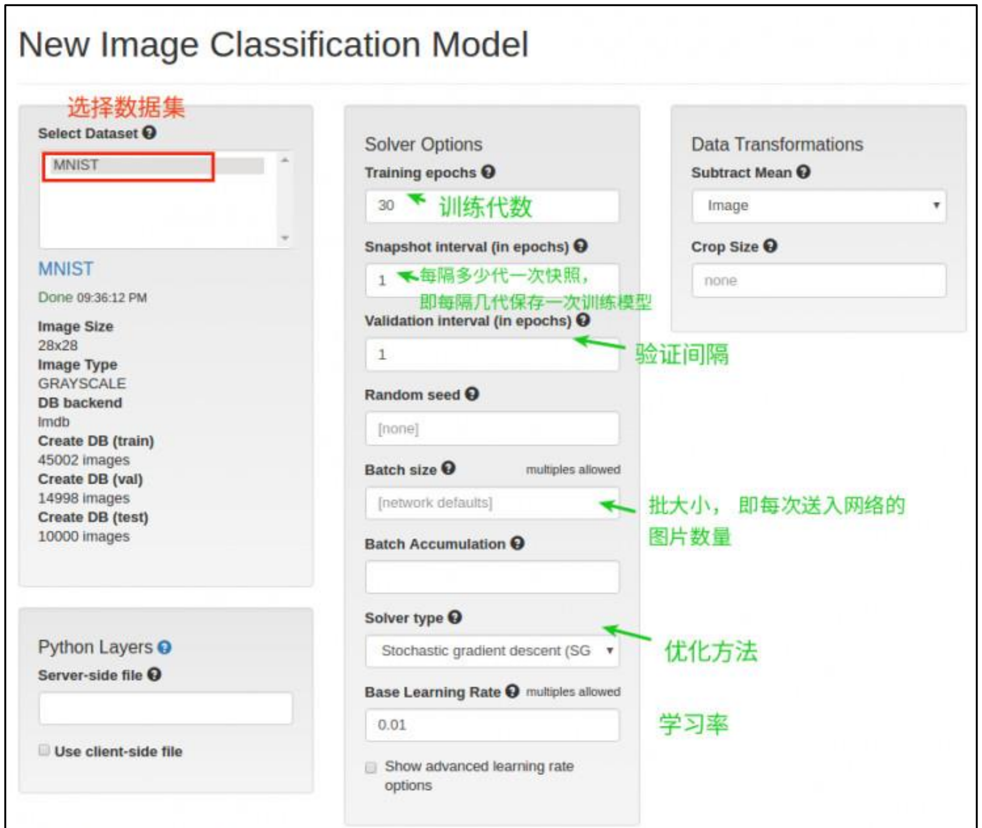


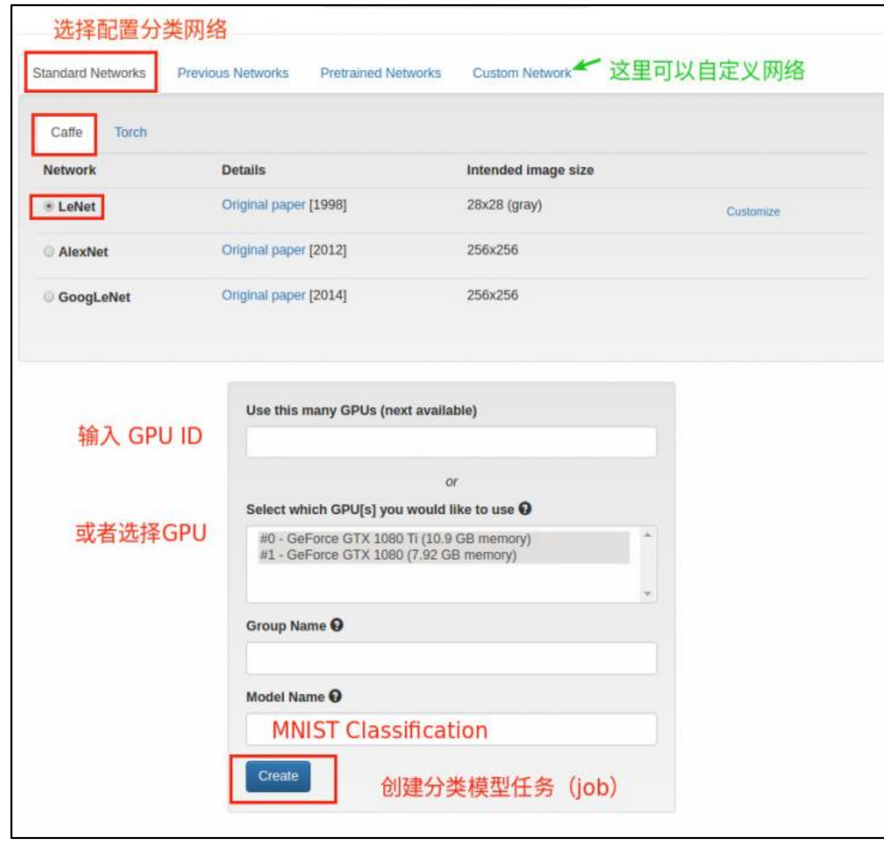
基于LeNet的手写体分类（digits）

exp01_LeNet_MNIST/digits

2. 创建模型

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

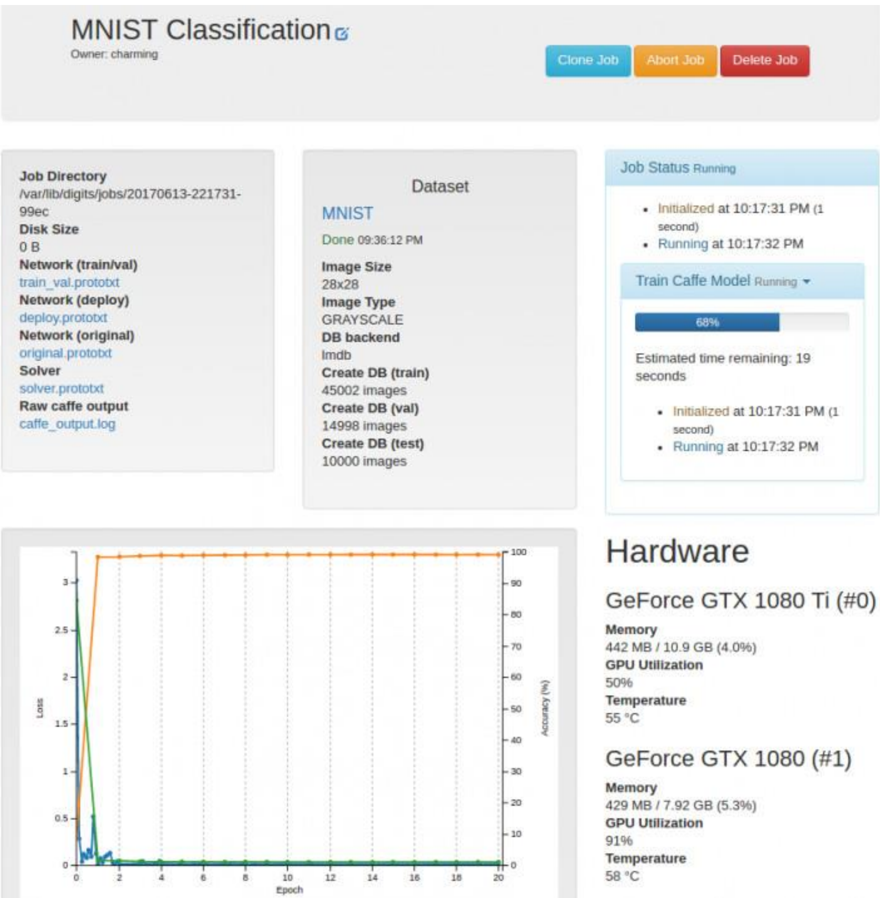




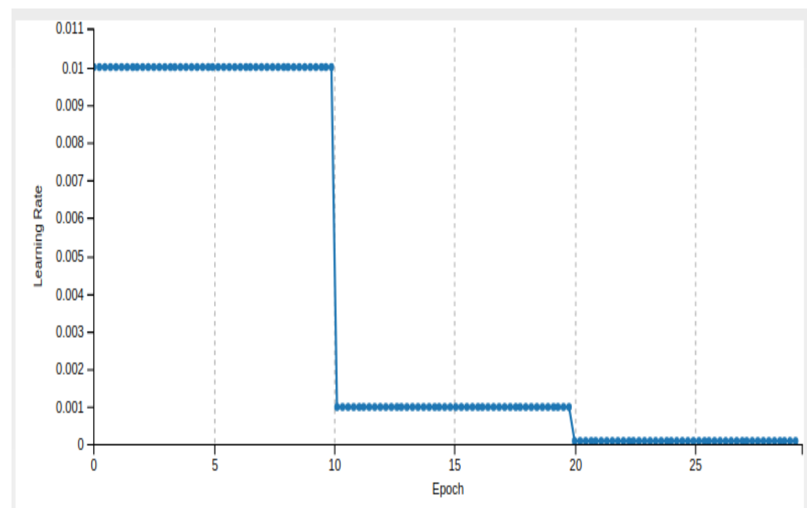
基于LeNet的手写体分类（digits）

exp01_LeNet_MNIST/digits

3. 训练模型



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。

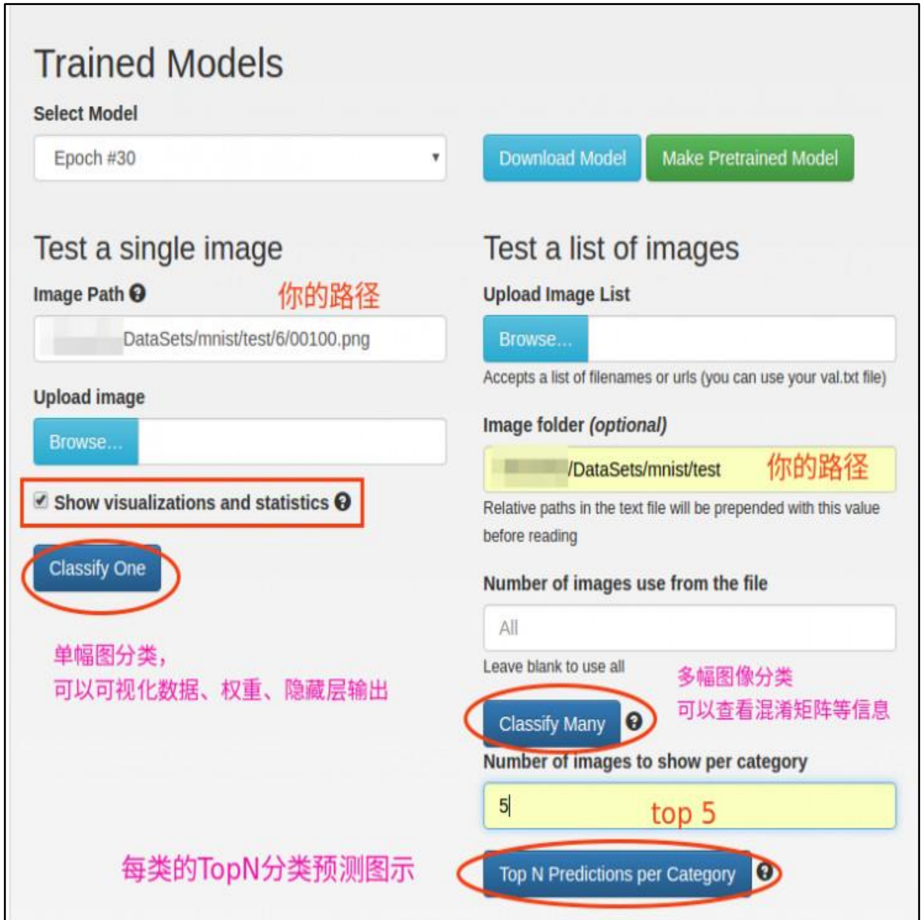


图像分类实践

基于LeNet的手写体分类（digits）

exp01_LeNet_MNIST/digits

4. 测试模型



基于LeNet的手写体分类（digits）

exp01_LeNet_MNIST/digits

4. 测试模型

MNIST Classification Image Classification Model

Job Status Done

- Initialized at 10:28:11 PM (1 second)
- Running at 10:28:12 PM (6 seconds)
- Done at 10:28:19 PM
  (Total - 7 seconds)

Infer Model Done

Notes

None

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	100.0%	3	0.0%	1	0.0%	4	0.0%	2	0.0%
2	/home/liu/DataSets/mnist/test/2/00001.png	2	2	100.0%	1	0.0%	0	0.0%	6	0.0%	7	0.0%

TopN Image Classification
Owner: charming

Clone Job | Delete Job

Top N Predictions per Category

Job Status Done

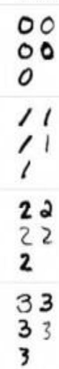
- Initialized at 10:29:33 PM (1 second)
- Running at 10:29:34 PM (6 seconds)
- Done at 10:29:40 PM
  (Total - 7 seconds)

Infer Model Done

Notes

None

Category	Top images for this category
0	
1	
2	
3	



内容提纲

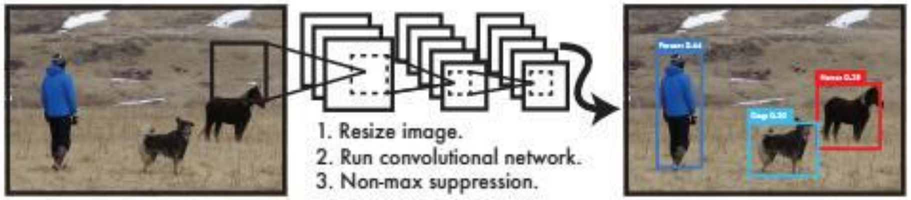
- 深度学习及平台简介
- 图像分类实践
- 目标检测实践
- 总结

目标检测实践

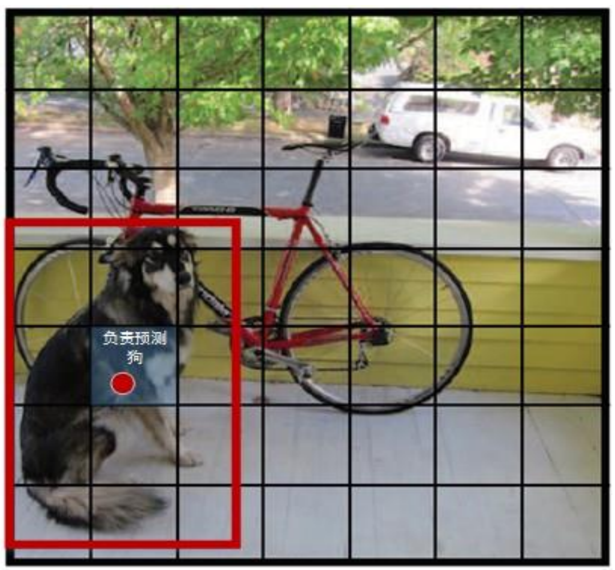
基于YOLOv3的目标检测（darknet平台）

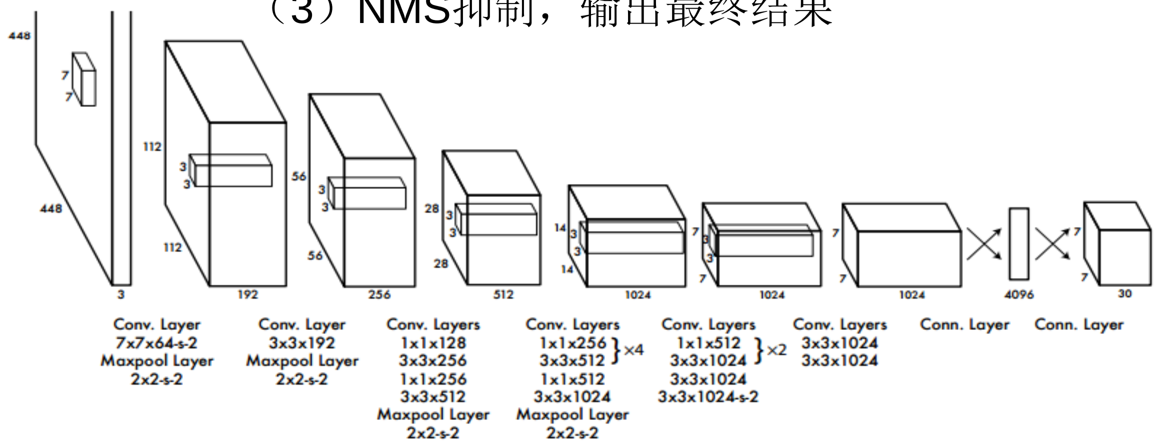
exp02_YOLO_Detection

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）通过神经网格进行检测和分类
（3）NMS抑制，输出最终结果





目标检测实践

基于YOLOv3的目标检测（darknet平台）

exp02_YOLO_Detection

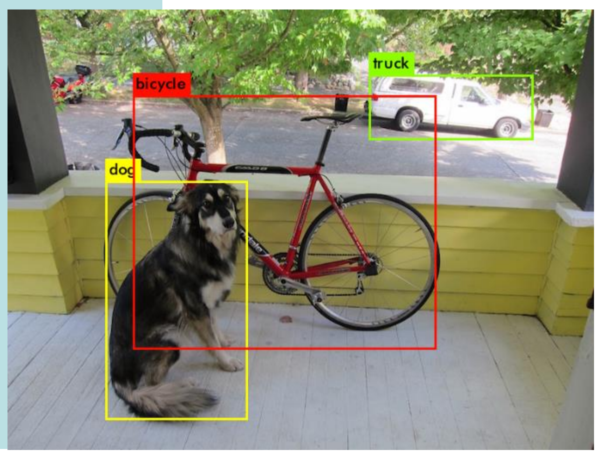
详细教程

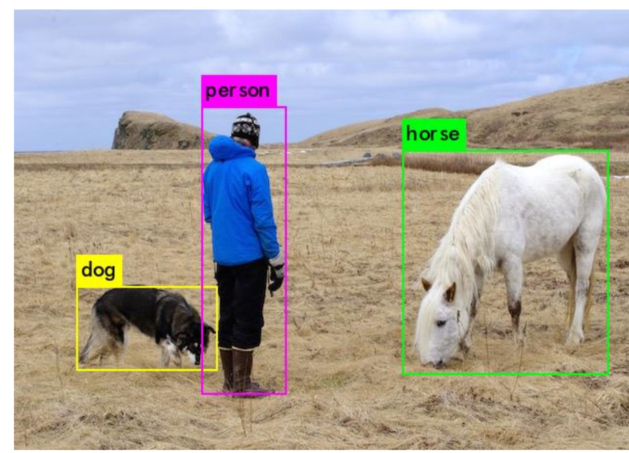
Darknet主页：
https://pjreddie.com/darknet/

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

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！