Course Project Introduction

Yonghong Tian
yhtian@pku.edu.cn
http://www.pkuml.org

National Engineering Laboratory for Video Technology
School of EE & CS, Peking University

Overview

- Course Outline
  - Please visit goo.gl/NkD5T9 for more details
- Course Code: 04802020
- Course Content
- Teacher
  - Prof. Yonghong Tian yhtian@pku.edu.cn
- TAs
  - Yifan Xiong yfxiong@pku.edu.cn
  - Limeng Qiao qiaolm@pku.edu.cn
- Email
  - pkufml2018@163.com

Outline

- Grading Policy Introduction
- Final Project
- Programming Assignment
- Paper Reading or Examination
- Deep Learning Frameworks

Grading Policy Introduction

☆ For undergraduates and graduates

- Class attendance: **15%**
- 2 Problem assignments: **10% x 2 = 20%**
- Paper reading and Presentation: **15%**
- Final course project: **50%**
  - ✓ **Code implementation**
  - ✓ **Final write-up (formal paper)**
  - ✓ **Bonus points for presentation**
- Late policy
  - ✓ 20% off per day late
  - ✓ Not accepted after 3 late days

Final Project

Overview

- 可选项目列表 (N选1)
  - ☆ 工程型项目
    - △ 网络模型转换
    - △ 网络模型压缩
    - △ 基于区块链的AI开源社区贡献评价策略开发
  - ☆ 竞赛型项目
    - △ 车辆大规模精准搜索

      National Graduate Smart City Contest 2017
    - △ 监控视频场景下的车辆异常检测和重识别

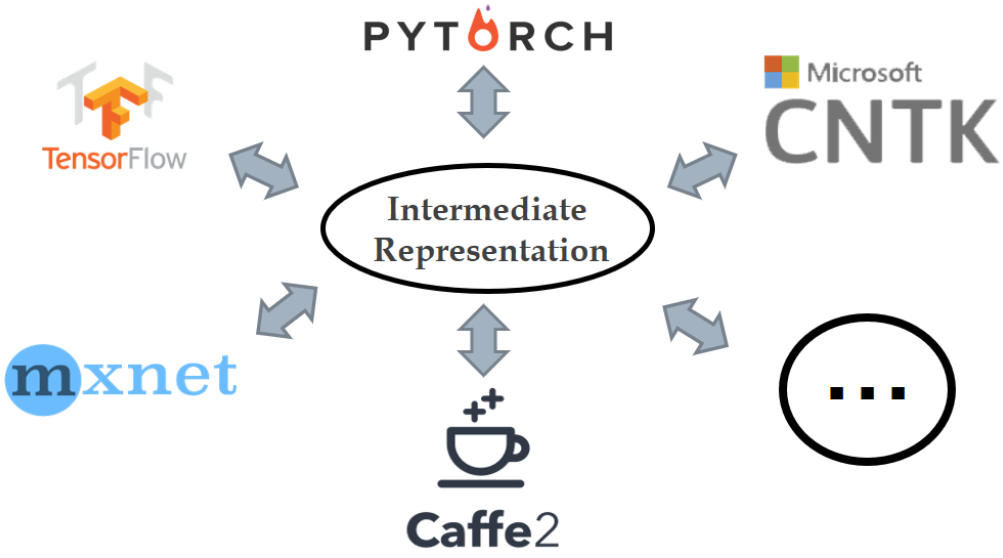
      AVSS2017 Challenge / NVIDIA AI CITY CHALLENGE
    - △ National Graduate Smart City Contest 2018
- 项目要求

  包括但不限于：模型性能+论文+按时完成+…

Project 1: 模型转换

任务背景

从学术界到产业界，目前存在各种各样的深度学习框架来供研究人员研究设计模型，然而对于神经网络的结构，每种框架都有自己的定义，并为模型设定它自己的保存格式，诸如此类的框架差异妨碍了模型之间的交互性操作，导致研究和生产之间产生了的重大延误。对此一个很好的解决方法是**通过一个模型中间表示格式来实现各个框架模型之间的转换**。



Project 1: 模型转换

任务描述

给定一个中间表达(intermediate representation，IR)，选择一个深度学习框架和对应的网络模型，要求完成选定框架模型和中间表达IR之间的模型转换

☆ 参考项目：**MMdnn** from Microsoft

☆ 项目地址：https://github.com/Microsoft/MMdnn

☆ 项目介绍：来自微软的一套能在不同的深度学习框架之间进行交互式操作的工具集，能够用于深度神经网络模型的转换，可视化及诊断等操作

Models	Caffe	Keras	Tensorflow	CNTK	MXNet	PyTorch	CoreML
Inception V1	√	√	√	√	√	x (No LRN)	√
Inception V3	×	√	√	√	√	√	√
Inception V4	√						
ResNet V1 50	×	√	√	o	√	√	√
ResNet V2 152	√	√	√	√	√	√	
VGG 19	√	√	√	√	√	√	√
MobileNet_v1	×	√	√	× (no DepthwiseConv)	×	×	√
Xception	×	√	√	× (no SeparableConv)	×	×	
SqueezeNet	√	√	√	√	√	×	

Project 1: 模型转换

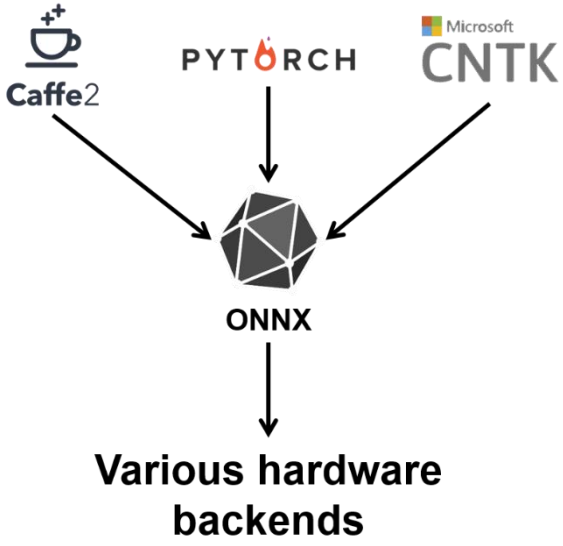
任务描述

给定一个中间表达(intermediate representation，IR)，选择一个深度学习框架和对应的网络模型，要求完成选定框架模型和中间表达IR之间的模型转换。

☆ 参考项目：Open Neural Network Exchange

☆ 项目地址：https://github.com/onnx

☆ 项目介绍：来自Facebook和微软的一套深度神经网络模式表示格式，能够用于深度神经网络模型的转换，可视化及网络优化等操作。



Project 1: 模型转换

参考文献

- Intel nGraph: An Intermediate Representation, Compiler, and Executor for Deep Learning
- Programming with a Differentiable Forth Interpreter
- Differentiable Programs with Neural Libraries
- Diannao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning
- A modern compiler infrastructure for deep learning systems with adjoint code generation in a domain-specific IR
- TVM: End-to-End Optimization Stack for Deep Learning
- ...

Project 2: 网络模型压缩

任务背景

深度神经网络目前已在很多任务中达到了非常可观的性能表现，然而这些网络模型(参数量巨大)往往非常耗费计算资源和内存，从而导致了在终端部署和低延迟需求场景下难以应用的问题，对此一种很好的解决方案就是**在保证模型性能不显著下降的前提下对深度卷积神经网络进行压缩和加速，包括但不限于压缩比、加速比、计算量和运行内存等指标。**

Project 2: 网络模型压缩

任务描述

根据已有的神经网络压缩算法，实现一种新的网络压缩算法并将其与当前的state-of-the-art方法进行比较。

☆ 可参考思路：

1. 参数修剪和共享(parameter pruning)

   探索模型中冗余的部分，并尝试去除冗余和不重要的参数

2. 低秩分解(Low-rank factorization)

   使用矩阵/张量分解来估计深度神经网络中最具信息量的参数

3. 迁移/压缩卷积滤波器(transferred/compact convolutional filters)

   设计特殊结构的卷积滤波器来减少存储和计算的复杂度

4. 知识精炼(knowledge distillation)

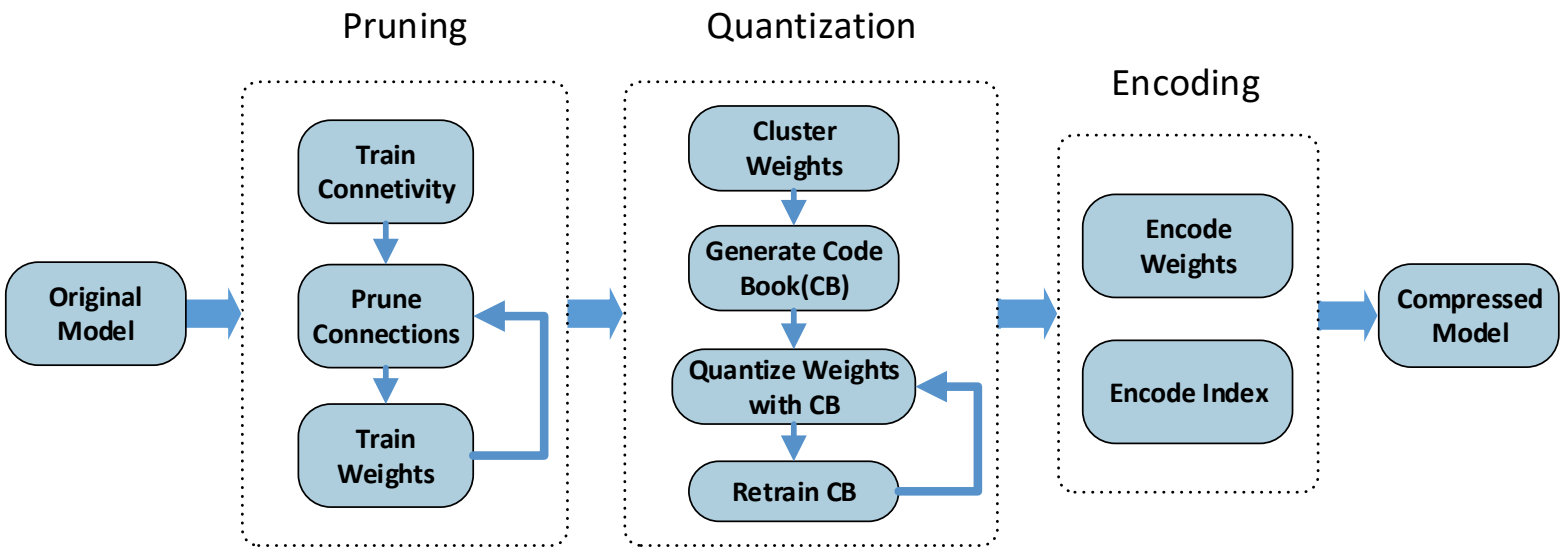
   训练一个更加紧凑的神经网络来再现大型网络的输出性能

Project 2: 网络模型压缩

任务描述

根据已有的神经网络压缩算法，实现一种新的网络压缩算法并将其与当前的state-of-the-art方法进行比较。

☆ 典型的Pipeline参考



Project 2:网络模型压缩

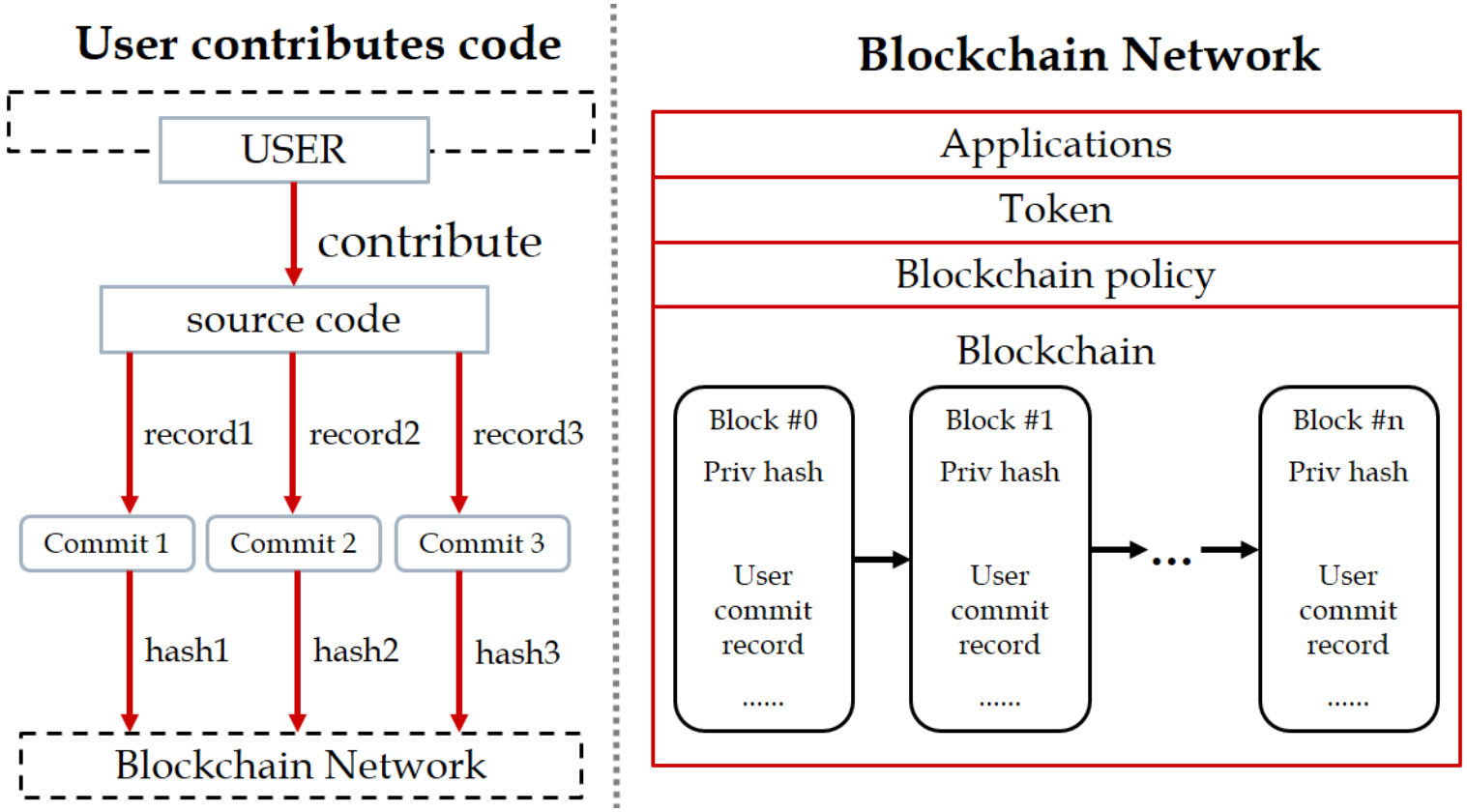
相关论文

- Learning both weights and connections for efficient neural networks
- Network Trimming：A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures
- Pruning Filters for Efficient ConvNets
- ThiNet：A Filter Level Pruning Method for Deep Neural Network Compression
- Channel Pruning for Accelerating Very Deep Neural Networks
- Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning
- Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation
- Compression of deep convolutional neural networks for fast and low power mobile applications
- Accelerating Convolutional Neural Networks for Mobile Applications
- CP-decomposition with Tensor Power Method for Convolutional Neural Networks Compression
- Compressing Deep Convolutional Networks using Vector Quantization
- SqueezeNet：AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size
- …

Project 3: 基于区块链的AI开源社区贡献评价策略开发

任务描述

基于某一开源平台框架，设计并实现一个区块链应用。



Project 4: 车辆大规模精准搜索

任务说明

车辆检索任务与其他对象检索任务类似，可以定义为：给定两部分图片数据ref(车辆图片数据库)和query(测试车辆图片)，前者ref包含N辆车的M张图片(N<=M，即每一辆车可以包含多张不同图片)，目标是对query中每张测试图片在ref中找出所有属于相同车辆的图片并排在查找结果的前面，并采用Mean Average Precision(MAP)@K(K=200)来对算法结果进行评测

注：这里所说的相同车辆并非仅仅是指车辆的型号、颜色相同，而是指在现实世界中的同一辆车

Project 4: 车辆大规模精准搜索

- 问题难点
  - 不同摄像头的拍摄角度、距离、光照环境不同
  - 如何区分相似车辆?
    - 不同车型，不同颜色，但外形相似
    - 相同车型、颜色

different color, different model

different color, same model

same color, different model

same color, same model

Different

Same

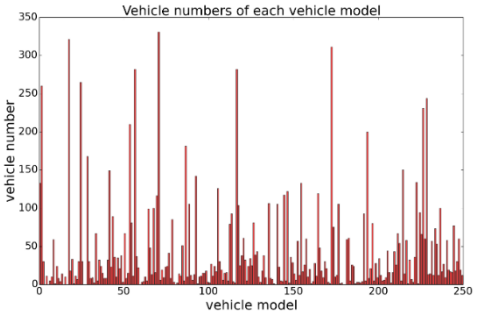
Project 4: 车辆大规模精准搜索

数据集介绍

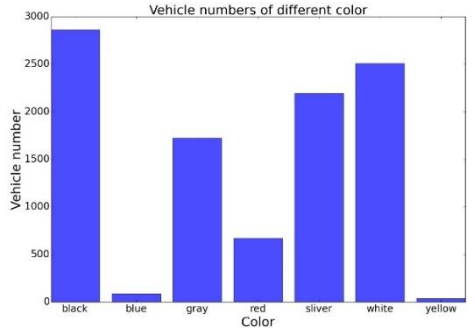
- 使用数据集：VehicleID 数据集
- 数据集内容
  - 采自某真实城市中心区域大量监控摄像头连续一周时间内拍摄的实际车辆数据，同一辆车被拍摄到多次(至少两次)
  - 车辆ID由车牌号作为标准，但在数据集内车牌区域被手动遮挡，不能提供判别信息
  - 考虑到车主隐私，人脸区域被模糊处理
  - 具体车辆图片拍摄角度包括正面和背面
- 数据集大小
  - 训练集：包含5043辆车的39320张图片(7.8 images/vehicle)，每张图片提供车辆ID(VID)、车型(MID)、颜色(CID)三种信息，其中车型类别总共250类，颜色类别总共7类，训练数据不保证每种类别样本数量均衡，仅仅作为辅助属性用于模型训练。
  - 验证集：包含2000辆车的16844张图片，并提供车辆ID(VID)，
  - 测试集：包含5000辆车的40531张图片，不提供任何标注，ref和query事先已经分好，大家需要对每张query中的图片，按相似度把ref数据库中的图片进行排序，并返回对应结果。

Project 4: 车辆大规模精准搜索

数据集图示



数量最多的车型：
别克-凯越
别克-英朗
雪佛兰-科鲁兹
五菱之光
大众-速腾



数量最多的颜色：
黑色
白色
银色
灰色
红色

北京大学

Project 4: 车辆大规模精准搜索

评价指标

- 该任务以MAP@K(K=200)作为技术评价指标，可以分三步来计算最终结果：
  P → AP@K → MAP@K
  - Precision：对于query中的某一个查询图片，参赛者对ref数据中返回的结果进行排序，返回前N个结果，在前n个结果中，精度P(n)=前n个结果中与查询图片是相同车辆的数量/n
  - Average Precision：对于query中的第k张查询图片，其平均精度为

$$AP_k = \sum_{1 \le i \le N \text{且第i个样本为正例}} P_i / \min\{M, N\},$$

    M为ref数据中与该查询图片属于相同车辆的数量
  - Mean Average Precision：所有K张query图片Average Precision的平均值

$$\text{MAP} = \sum_{1 \le k \le K} AP_k / K$$

- 计算示例
  - 目标：计算MAP@10
  - 假设第k张query图片ID为1，ref数据中共有5个ID为1的样本
  - 排序后，前10张图片ID分别为: 1, 3, 1, 2, 9, 8, 1, 3, 2, 4
  - 那么精度P(4) = 2/4 = 0.5
  - 平均精度AP_k@10 = (1/1+2/3+3/7)/min{10, 5} = 0.419
  - $MAP@10 = \sum AP_k / K$
- 其他
  - MAP计算出来应该在0到1之间，值越大，准确率越高

Project 4: 车辆大规模精准搜索

结果格式说明

- 本任务的测试图片文件统一为jpg图片格式
- 结果格式要求：每一小组需要将测试集上结果写入XML文件中，各项标签含如下：
  - evaluateType为任务类型编号，对应于数据集中的任务类型编号（车辆大规模精准搜索的任务编号为6）
  - mediaFile为数据集名称，对应于数据集的名称
  - Item的属性imageName代表query图片名称
  - Item的值为对应测试图片在gallery中的检索结果，即用空格隔开的预测ID序列(列出前K=200个图片的文件名即可，不包含后缀”.jpg”及文件夹路径)

```
<?xml version="1.0" encoding="gb2312"?>
<Message Version="1.0">
    <Info evaluateType="6" mediaFile="vehicle_retrieval_val" />
    <Items>
        <Item imageName="012321 ">
            0292851 0110741 0173591 0092564 0286241 0192567 0340982 …
        </Item>
        <Item imageName="003467 ">
            0387241 0023986 0283751 0230114 9806431 8823012 2389102 …
        </Item>
        <Item imageName="013169">
            3727192 0387654 0007942 0009866 0120397 0485764 1200341 …
        </Item>
        ......
    </Items>
</Message>
```

Project 4: 车辆大规模精准搜索

典型算法

- 提取颜色、轮廓、局部纹理等底层特征，然后通过欧式距离直接进行匹配
- 结合训练数据所标注的车型、颜色信息，计算两张车辆图片是否具有相同属性
- 度量学习（Metric Learning）：尝试学习一个距离矩阵
- 利用深度神经网络尝试预测两张图片相似度（Siamese Network等）

可能的创新点

- **使用车型、颜色信息**：对车型、颜色分别训练一个分类器，得到特定于车型、颜色的特征表达，分别计算车型相似度和颜色相似度，然后融合
- **级联模式**：通过颜色和车型预测，先把ref中不同车型或颜色的样本过滤，然后再比较特征相似度排序
- **采取更优的网络结构**
- **使用精细的判别性特征**：尝试提取局部具有判别性的特征，比如年检标志、特殊装饰灯，区分两个相同车型相同颜色的汽车

Project 5: AVSS2017 Challenge

UA-DETRAC是一个现实场景下的多目标检测和多目标追踪的基准数据集，包含10h的视频数据，25fps，共计包含14w帧数据，8250辆车被人工标注过，这个比赛希望参赛选手能够取探索一些关于对象检测（detection）和追踪（tracking）的方法，并提出一些新的解决思路，详细内容请参考网址 http://detrac-db.rit.albany.edu .

https://www.aicitychallenge.org

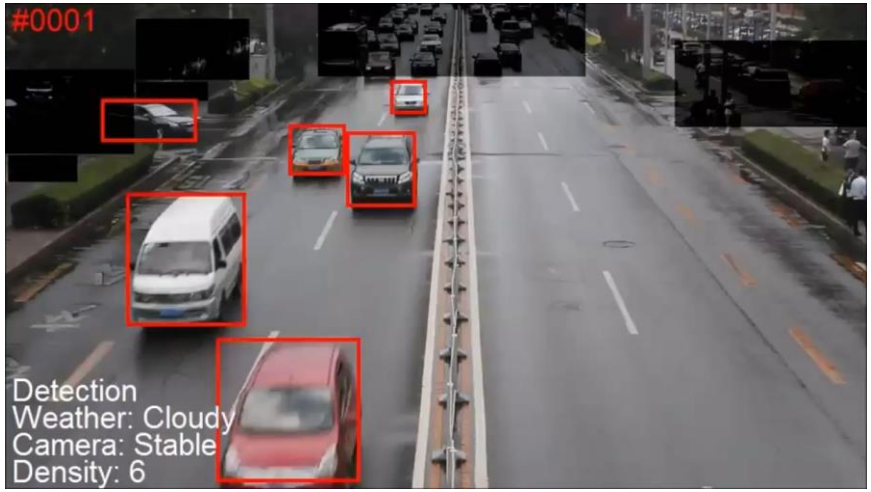
NVIDIA AI CITY CHALLENGE

Contact: nvidiaaicitychallenge@gmail.com

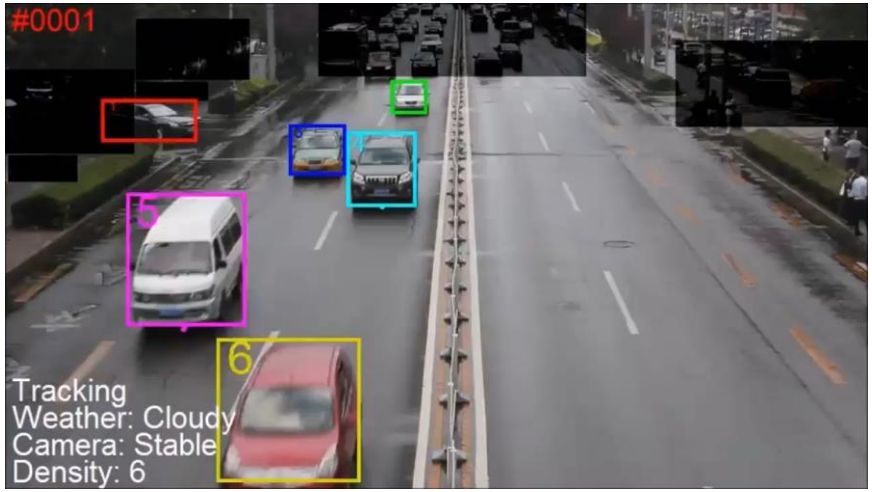
北京大学

Project 5: AVSS2017 Challenge

- 任务说明
  - 比赛分为两个子任务
    - Vehicle Detection
    - Multi-Objects Vehicle Tracking
  - 评测指标
    - 竞赛网址有详细说明，请详细阅读
  - 其他
    - 竞赛方提供数据下载、结果评测、相关（Detection和Tracking）方法基准、相关论文等多项信息，对此项目感兴趣的同学请参考官方网站说明



Detection 示例



Tracking 示例

Project 6: National Graduate Smart City Contest 2018

- 任务说明
  - Action recognition
  - Face recognition under occlusion
  - Hashing for fast search of image

http://www.bigmmchallenge.org

Final Project: 课程报告要求

- **报告主要内容**
  - 摘要---任务描述、创新点
  - 项目意义---车辆精准搜索/检测/追踪/模型压缩的研究意义、困难挑战等
  - 研究现状---简述车辆精准搜索/检测/追踪/模型压缩…等已有算法的优缺点
  - 模型及算法详细说明---包含深度网络的结构、模型介绍、参数说明
  - 课程设计创新点说明
  - 实验结果与分析---在相应评价指标下的实验结果
  - 课程总结与收获
- **其他要求**
  - 项目需要1-3人组队，测试集由比赛方提供
  - 报告文档需要以科研Paper的形式提交，题目和格式不限
  - 课程报告重在实践，关注评价指标，并应突出方法创新性
  - 结果图例与表格展示要求尽量详细，误差分析合理
  - 课程报告内容汇报
  - 请大家在03.31之前把组队信息发至课程邮箱pkufml2018@163.com
    - 邮件主题：姓名1_学号1_姓名2_学号2_姓名3_学号3__Final_Project
    - 邮件内容：小组成员、所选题目、简述基本思路

Paper Reading

Paper Reading & Assignment

- **选读内容要求**
  - 论文应该是CCF-A类的论文，且和机器学习、深度学习领域相关
  - 下周开始Paper Reading，要求每次课不超过5个人，每人8分钟讲解，2分钟提问
  - 请大家把自己选好的论文在03.18 24:00之前发至课程邮箱pkufml2018@163.com，每个人提交两篇论文Title，03.19会在教学网和微信群中公布论文阅读顺序，请大家提前准备PPT
    - 邮件主题：姓名_学号_Paper_Reading
    - 邮件内容：论文题目、发表时间、会议or期刊名称、选题理由
  - 原则上所有分享的论文不能有重复，如重复，按照助教收到邮件的先后顺序选定
- **Assignment**
  - 共计有两次书面作业，作业内容和老师上课讲过的内容有关，每次作业会留2周的时间供大家完成作业
  - 作业发布时间另行通知

Deep Learning Frameworks

Overview

越来越多的深度学习框架…

- ☆ Tensorflow https://www.tensorflow.org/
- ☆ Caffe2 https://caffe2.ai/
- ☆ Pytorch http://pytorch.org/
- ☆ MXNet https://mxnet.apache.org/
- ☆ Caffe http://caffe.berkeleyvision.org/
- ☆ Theano http://deeplearning.net/software/theano/
- ☆ CNTK https://github.com/Microsoft/CNTK
- ☆ Chainer https://chainer.org/
- ☆ Keras https://keras.io/
- ☆ And more…

关于深度学习框架的详细对比可参考：

https://sites.google.com/site/dliftutorial/ (AAAI 2017)

https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

Software	Creator	Software license[a]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)
Caffe	Berkeley Vision and Learning Center	BSD license	Yes	Linux, macOS, Windows[2]	C++	Python, MATLAB	Yes	Under development[3]	Yes	Yes	Yes[4]	Yes	Yes	No	?
Caffe2	Facebook	Apache 2.0	Yes	Linux, macOS, Windows[5]	C++, Python	Python, MATLAB	Yes	Under development[6]	Yes	Yes	Yes[7]	Yes	Yes	No	Yes
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Apache 2.0	Yes	Linux, macOS, Windows, Android (Cross-platform)	C++, Java	Java, Scala, Clojure, Python (Keras), Kotlin	Yes	On roadmap[8]	Yes[9][10]	Computational Graph	Yes[11]	Yes	Yes	Yes	Yes[12]
Dlib	Davis King	Boost Software License	Yes	Cross-Platform	C++	C++	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Intel Data Analytics Acceleration Library	Intel	Apache License 2.0	Yes	Linux, macOS, Windows on Intel CPU[13]	C++, Python, Java	C++, Python, Java[13]	Yes	No	No	Yes	No		Yes		Yes
Intel Math Kernel Library	Intel	Proprietary	No	Linux, macOS, Windows on Intel CPU[14]		C[15]	Yes[16]	No	No	Yes	No	Yes[17]	Yes[17]		No
Keras	François Chollet	MIT license	Yes	Linux, macOS, Windows	Python	Python, R	Only if using Theano as backend	Under development for the Theano backend (and on roadmap for the TensorFlow backend)	Yes	Yes	Yes[18]	Yes	Yes	Yes	Yes[19]
MatConvNet	Andrea Vedaldi, Karel Lenc	BSD license	Yes	Windows, Linux[20] (macOS via Docker on roadmap)	C++	MATLAB, C++,	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes
MATLAB + Neural Network Toolbox	MathWorks	Proprietary	No	Linux, macOS, Windows	C, C++, Java, MATLAB	MATLAB	No	No	Train with Parallel Computing Toolbox and generate CUDA code with GPU Coder[21]	No	Yes[22][23]	Yes[22]	Yes[22]	No	With Parallel Computing Toolbox[24]
Microsoft Cognitive Toolkit	Microsoft Research	MIT license[25]	Yes	Windows, Linux[20] (macOS via Docker on roadmap)	C++	Python (Keras), C++, Command line,[26] BrainScript[27] .NET on roadmap[28])	Yes[29]	No	Yes	Yes	Yes[30]	Yes[31]	Yes[31]	No[32]	Yes[33]
Apache MXNet	Apache Software Foundation	Apache 2.0	Yes	Linux, macOS, Windows,[34][35] AWS, Android,[36] iOS, JavaScript[37]	Small C++ core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl	Yes	On roadmap[38]	Yes	Yes[39]	Yes[40]	Yes	Yes	Yes	Yes[41]
Neural Designer	Artelnics	Proprietary	No	Linux, macOS, Windows	C++	Graphical user interface	Yes	No	No	?	?	No	No	No	?
OpenNN	Artelnics	GNU LGPL	Yes	Cross-platform	C++	C++	Yes	No	Yes	?	?	No	No	No	?
PaddlePaddle	Baidu PaddlePaddle team	Apache 2.0	Yes	Linux, macOS, Android,[42] Raspberry Pi[43]	C++, Go	C/C++, Python	Yes	No	Yes	Yes	Yes[44]	Yes	Yes	No	Yes
PyTorch	Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan	BSD license	Yes	Linux, macOS	Python, C, CUDA	Python	Yes		Yes	Yes	Yes	Yes			Yes
Apache SINGA	Apache Incubator	Apache 2.0	Yes	Linux, macOS, Windows	C++	Python, C++, Java	No	No	Yes	?	Yes	Yes	Yes	Yes	Yes
TensorFlow	Google Brain team	Apache 2.0	Yes	Linux, macOS, Windows[45]	C++, Python	Python (Keras), C/C++, Java, Go, R[46]	No	On roadmap[47] but already with SYCL[48] support	Yes	Yes[49]	Yes[50]	Yes	Yes	Yes	Yes
Theano	Université de Montréal	BSD license	Yes	Cross-platform	Python	Python (Keras)	Yes	Under development[51]	Yes	Yes[52][53]	Through Lasagne's model zoo[54]	Yes	Yes	Yes	Yes[55]
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	BSD license	Yes	Linux, macOS, Windows,[56] Android,[57] iOS	C, Lua	Lua, LuaJIT,[58] C, utility library for C++/OpenCL[59]	Yes	Third party implementations[60][61]	Yes[62][63]	Through Twitter's Autograd[64]	Yes[65]	Yes	Yes	Yes	Yes[66]
Wolfram Mathematica	Wolfram Research	Proprietary	No	Windows, macOS, Linux, Cloud computing	C++	Wolfram Language	No	No	Yes	Yes	Yes[67]	Yes	Yes	Yes	Yes
Leonsil	Leonbud	Apache 2.0	Yes	Linux, Cloud computing	C++	Python	No	No	Yes	No	Yes[68]	No	Yes	No	Yes

Overview

推荐框架

☆ Advice by Feifei Li [1]

1. **Tensorflow** is a safe bet for most projects, not perfect, huge community, wide usage
2. **PyTorch** is best for research. However still new, there can be rough patches
3. Consider **Caffe**, **Caffe2** or **Tensorflow** for production deployment
4. Consider **Tensorflow** or **Caffe2** for mobile

☆ Personal advice:

1. **Python** is the basic skill
2. Choosing the tools for most people, such as **Tensorflow** and **Pytorch**.

[1] link: http://cs231n.stanford.edu/slides/2017/

Deep Learning Frameworks

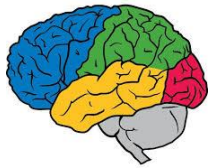
- TensorFlow
- PyTorch

TensorFlow

◆ What is TensorFlow --- Introduction

(1) An open-source machine learning platform for everyone
(2) Fast, flexible, and production-ready
(3) Scales from research to production

Home: https://www.tensorflow.org/



Google Brian[5]

Companies using Tensorflow

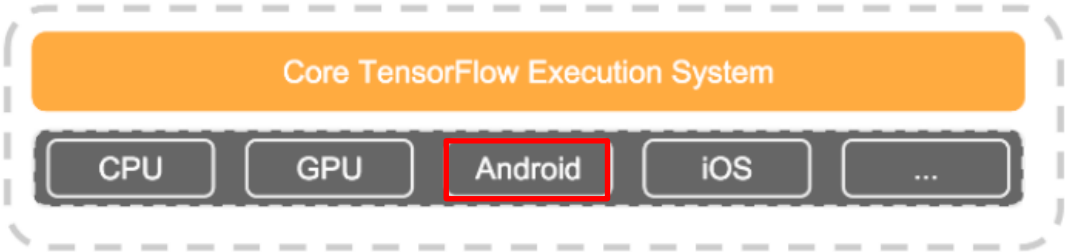
[5] TensorFlow: A system for large-scale machine learning , Jeffrey Dean et al.

TensorFlow

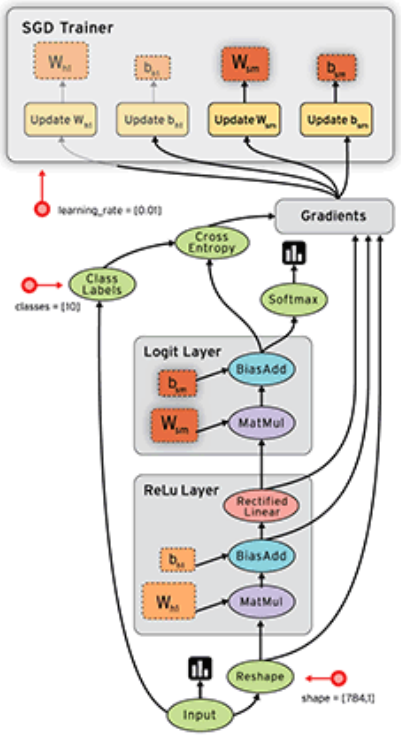
◆ What is TensorFlow --- How does it work?

Tensor means N-dimensional array, **Flow** means the calculation graph for data flow, so **Tensorflow** [6] is the tensor flowing from one to another in graph.

- use data flow graphs to represent a learning model
- Nodes represent mathematical operations
- Edges represent multi-dimensional data arrays(tensors)



Multi-platform



Graph

[6] https://events.withgoogle.com/tensorflow-dev-summit/ , Jeffrey Dean

TensorFlow

◆ TensorFlow examples --- MINST

MNIST is a simple computer vision dataset. Just like programing has "Hello World", it consists of images of handwritten digits like these:

Images of handwritten digits[7]

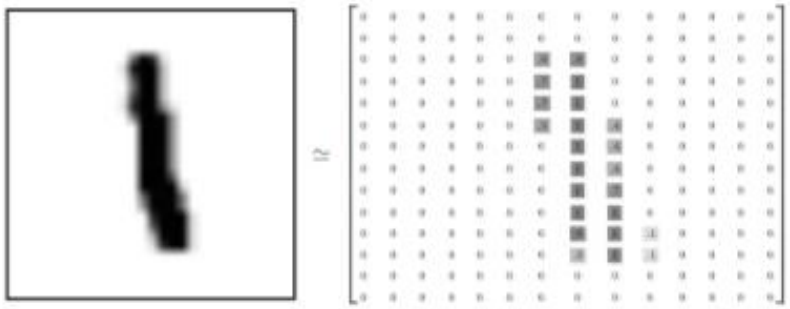
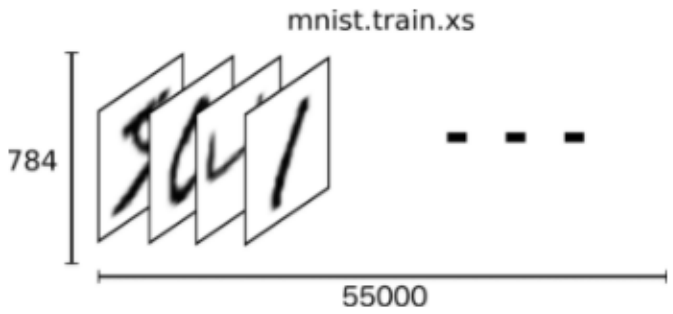


Image array



MNIST train

[7] Yann LeCun's MNIST page: http://yann.lecun.com/exdb/mnist/

TensorFlow

```python
def cnn_model_fn(features, labels, mode):
  """Model function for CNN."""
  # Input Layer
  input_layer = tf.reshape(features["x"], [-1, 28, 28, 1])

  # Convolutional Layer #1
  conv1 = tf.layers.conv2d(
      inputs=input_layer,
      filters=32,
      kernel_size=[5, 5],
      padding="same",
      activation=tf.nn.relu)

  # Pooling Layer #1
  pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[2, 2], strides=2)

  # Convolutional Layer #2 and Pooling Layer #2
  conv2 = tf.layers.conv2d(
      inputs=pool1,
      filters=64,
      kernel_size=[5, 5],
      padding="same",
      activation=tf.nn.relu)
  pool2 = tf.layers.max_pooling2d(inputs=conv2, pool_size=[2, 2], strides=2)
```

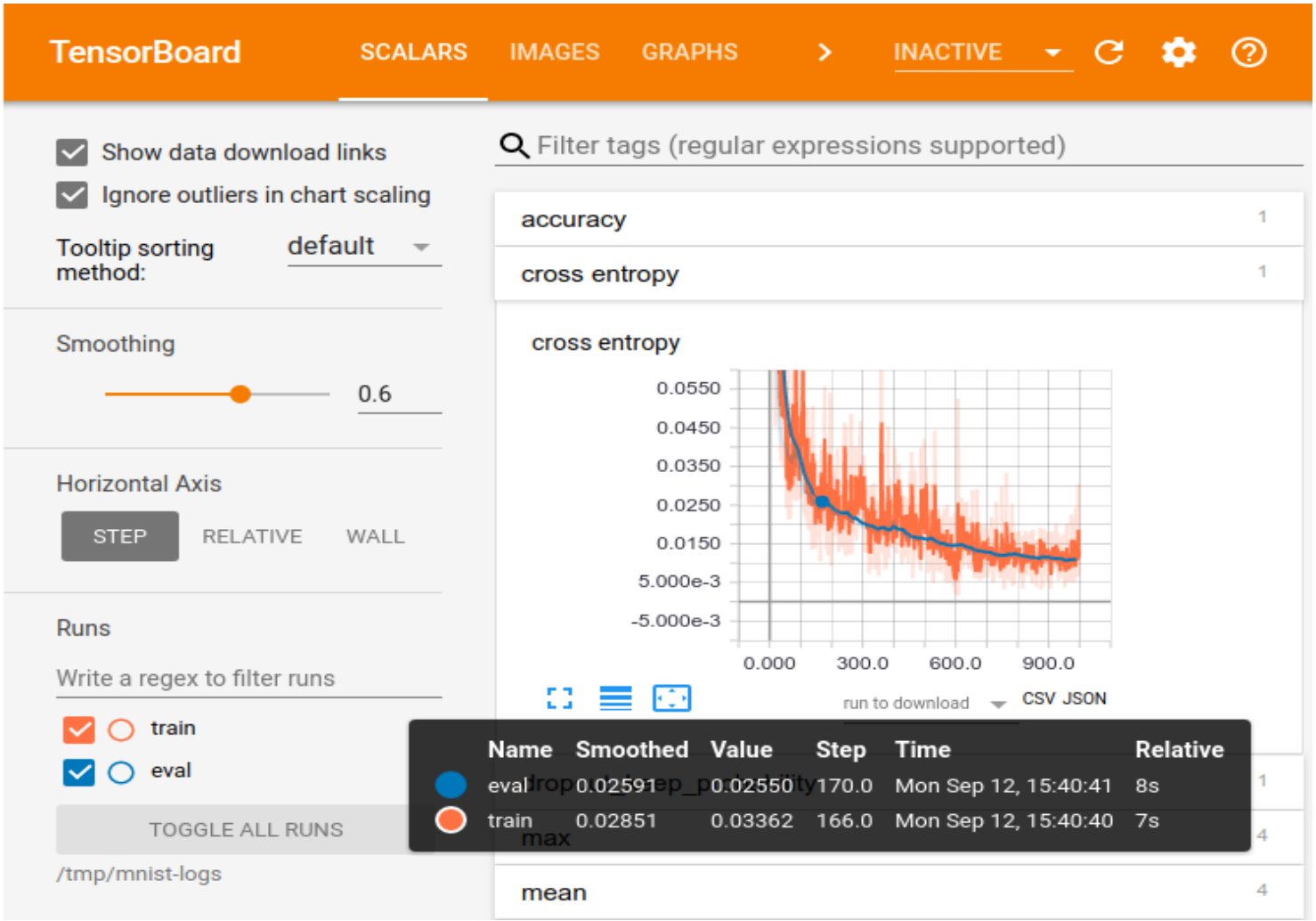
TensorFlow

```
# Dense Layer
pool2_flat = tf.reshape(pool2, [-1, 7 * 7 * 64])
dense = tf.layers.dense(inputs=pool2_flat, units=1024, activation=tf.nn.relu)
dropout = tf.layers.dropout(
    inputs=dense, rate=0.4, training=mode == tf.estimator.ModeKeys.TRAIN)

# Logits Layer
logits = tf.layers.dense(inputs=dropout, units=10)

predictions = {
    # Generate predictions (for PREDICT and EVAL mode)
    "classes": tf.argmax(input=logits, axis=1),
    # Add `softmax_tensor` to the graph. It is used for PREDICT and by the
    # `logging_hook`.
    "probabilities": tf.nn.softmax(logits, name="softmax_tensor")
}

if mode == tf.estimator.ModeKeys.PREDICT:
  return tf.estimator.EstimatorSpec(mode=mode, predictions=predictions)

# Calculate Loss (for both TRAIN and EVAL modes)
loss = tf.losses.sparse_softmax_cross_entropy(labels=labels, logits=logits)
```

TensorFlow

```
# Configure the Training Op (for TRAIN mode)
if mode == tf.estimator.ModeKeys.TRAIN:
  optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001)
  train_op = optimizer.minimize(
      loss=loss,
      global_step=tf.train.get_global_step())
  return tf.estimator.EstimatorSpec(mode=mode, loss=loss, train_op=train_op)

# Add evaluation metrics (for EVAL mode)
eval_metric_ops = {
    "accuracy": tf.metrics.accuracy(
        labels=labels, predictions=predictions["classes"])}
return tf.estimator.EstimatorSpec(
    mode=mode, loss=loss, eval_metric_ops=eval_metric_ops)
```

TensorFlow



PyTorch

◆ What is PyTorch --- Introduction

Scientific computing package for python:

(1) A replace for NumPy for to use the power of GPU

(2) Provides maximum flexibility and speed

Home: http://pytorch.org/

Assistant companies & Universities developing PyTorch

PyTorch

◆ What is PyTorch --- Why it is future star?

PyTorch is more light-weight and easier to get started, so it is best for research.

Numpy

```
import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)

grad_c = 1.0
grad_b = grad_c * np.ones((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a * y
grad_y = grad_a * x
```

Tensorflow

```
import numpy as np
np.random.seed(0)
import tensorflow as tf

N, D = 3, 4

with tf.device('/gpu:0'):
    x = tf.placeholder(tf.float32)
    y = tf.placeholder(tf.float32)
    z = tf.placeholder(tf.float32)

    a = x * y
    b = a + z
    c = tf.reduce_sum(b)

grad_x, grad_y, grad_z = tf.gradients(c, [x, y, z])

with tf.Session() as sess:
    values = {
        x: np.random.randn(N, D),
        y: np.random.randn(N, D),
        z: np.random.randn(N, D),
    }
    out = sess.run([c, grad_x, grad_y, grad_z],
                   feed_dict=values)
    c_val, grad_x_val, grad_y_val, grad_z_val = out
```

PyTorch

```
import torch
from torch.autograd import Variable

N, D = 3, 4

x = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)
y = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)
z = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()

print(x.grad.data)
print(y.grad.data)
print(z.grad.data)
```

CS231n: Convolutional Neural Networks for Visual Recognition in Stanford University by Prof. Feifei Li [9]

PyTorch

◆ What is PyTorch --- How does it work?

- Tensors: live on the CPU or GPU, and accelerate compute by a huge amount
- Dynamic Neural Networks: Tape based autograd

A graph is created on the fly


```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))
```

PyTorch

```
import torch
from torch.autograd import Variable

# N is batch size; D_in is input dimension;
# H is hidden dimension; D_out is output dimension.
N, D_in, H, D_out = 64, 1000, 100, 10

# Create random Tensors to hold inputs and outputs, and wrap them in Variables.
x = Variable(torch.randn(N, D_in))
y = Variable(torch.randn(N, D_out), requires_grad=False)

# Use the nn package to define our model as a sequence of layers. nn.Sequential
# is a Module which contains other Modules, and applies them in sequence to
# produce its output. Each Linear Module computes output from input using a
# linear function, and holds internal Variables for its weight and bias.
model = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out),
)

# The nn package also contains definitions of popular loss functions; in this
# case we will use Mean Squared Error (MSE) as our loss function.
loss_fn = torch.nn.MSELoss(size_average=False)
```

PyTorch

```
learning_rate = 1e-4
for t in range(500):
    # Forward pass: compute predicted y by passing x to the model. Module objects
    # override the __call__ operator so you can call them like functions. When
    # doing so you pass a Variable of input data to the Module and it produces
    # a Variable of output data.
    y_pred = model(x)

    # Compute and print loss. We pass Variables containing the predicted and true
    # values of y, and the loss function returns a Variable containing the
    # loss.
    loss = loss_fn(y_pred, y)
    print(t, loss.data[0])

    # Zero the gradients before running the backward pass.
    model.zero_grad()

    # Backward pass: compute gradient of the loss with respect to all the learnable
    # parameters of the model. Internally, the parameters of each Module are stored
    # in Variables with requires_grad=True, so this call will compute gradients for
    # all learnable parameters in the model.
    loss.backward()

    # Update the weights using gradient descent. Each parameter is a Variable, so
    # we can access its data and gradients like we did before.
    for param in model.parameters():
        param.data -= learning_rate * param.grad.data
```

Q&A