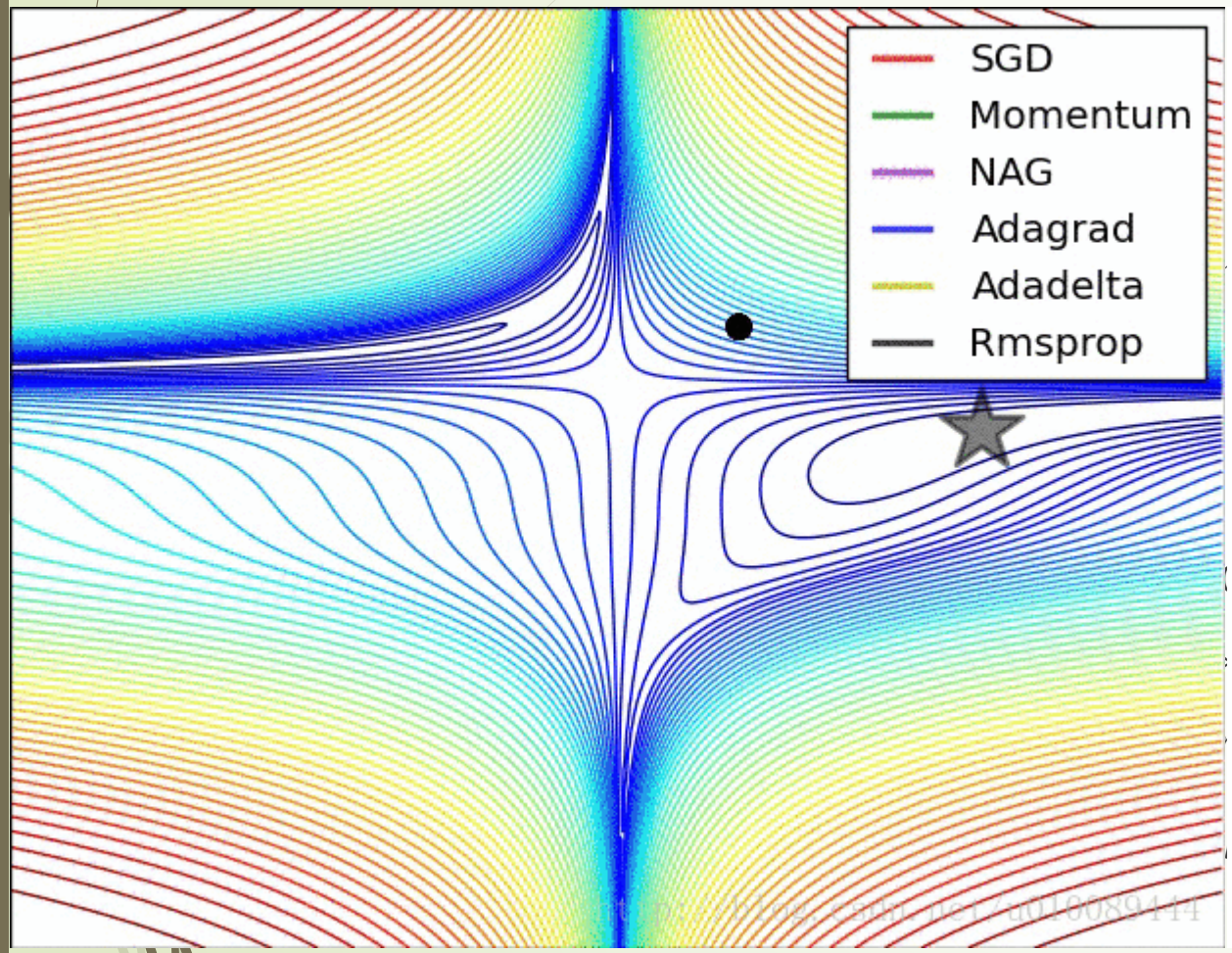
深度学习课程设计
---预备知识

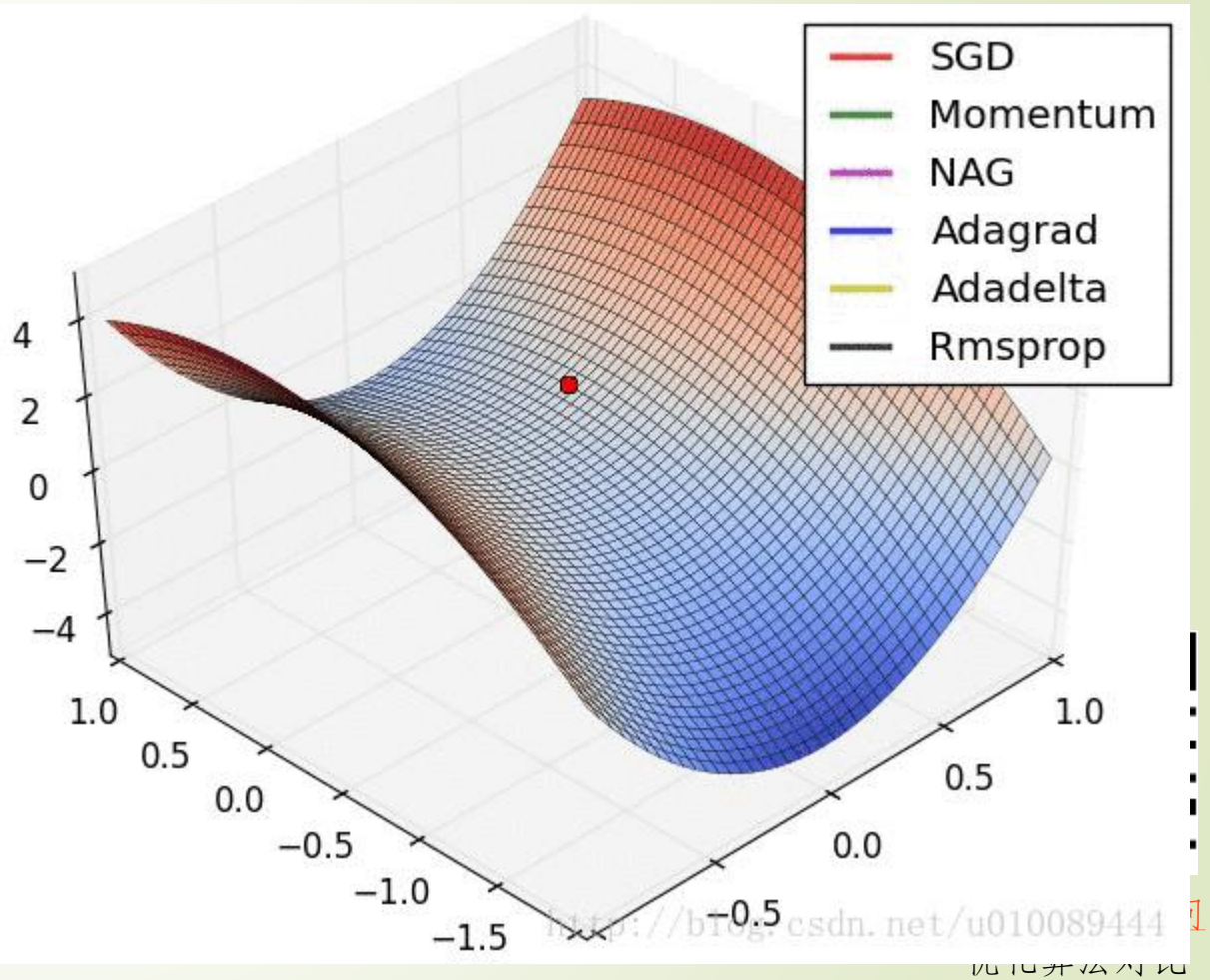
王昊
wanghao@ise.neu.edu.cn

部分深度学习框架介绍(字典序)

姓名	语言	单位	社会关系	优点	缺点
Caffe	C++/Python /Matlab	BVLC		速度快、模块化 源码写得十分优雅	灵活性较差，写代码难度高 需要手写C++/CUDA正向反向
Caffe2	C++/Python	FB	看名字就知道是Caffe的儿子	基于Caffe改进，速度更快，模块化更好	还没流行起来就被并入PyTorch
Keras	Python	GMNA*	坐拥TF/CNTK/Theano /MXNet等几大后台	High-level API简单统一整合多种后端，Google/MS/Nvidia/Amazon支持，借鉴Torch	使用不够灵活，难以实现自定义功能
MatConvNet	Matlab	VLFeat		基于Matlab语言	最新版本为2017年8月发布 灵活性较差
PyTorch	Python/C++	FB	集Caffe2与Torch于一身	上手容易、代码简单灵活 动态计算图、自动求导	运行效率相比TF略低
TensorFlow	C++/Python /Go/Java/…	Google	亲儿子Tensorboard可视化利器	大厂出品，更新快，优化好 现已加入动态计算图豪华套餐	入门较难，API繁多(据说2.0在精简)
Theano	Python	MILA**		深度学习框架鼻祖之一 集成Numpy，计算稳定性好	2017年9月宣布停止重大更新
Torch7	Lua	FB		速度快(Lua-JIT可达C的80%) 可直接调用C、可移植性好	Lua受众较少，学习成本高
* GMNA: Google, Microsoft, NVIDIA, Amazon AWS ** MILA: Montréal Institute for Learning Algorithms of Université de Montréal(蒙特利尔大学), Scientific Director: Yoshua Bengio					
Chainer CNTK Deeplearning4J Dlib H2O Lasagne Leaf MXNet Neon PaddlePaddle PyLearn等如有兴趣请自行探索					

PyTorch基本介绍 – DL常见概念





PyTorch基本介绍 – DL常见概念

- Iteration / Epoch
- Mini-batch / Batch Size
- Loss Function (Cost Function)
- Learning Rate
- Supervised / Unsupervised Learning
- Array / Tensor
- Computing Graph
- …….

PyTorch基本介绍 – 常用接口

- torch
  - `>>> import torch`
  - `torch.set_num_threads(num)  # useful command for limiting threads`
  - `torch.set_default_tensor_type(type)  # e.g., type = torch.float32`
  - `torch.zeros/ones/eye/rand/randn/randperm`

    `# 全0/全1/对角线元素为1/[0,1)均匀分布/N(0,1)高斯分布/[0,n-1]随机排序`
  - `torch.split/cat/squeeze/unsqueeze/stack`

    `# 切分/连接/去除维度/插入维度/堆叠`
  - `torch.abs/add/sub/mul/div/pow/exp/neg/sqrt/sin/dot`
  - `torch.mean/sum/median/max/min/std/var/eq/lt/gt/le/ge/`

PyTorch基本介绍 – 常用接口

- torch.Tensor
  - `torch.[Float/Double/Byte/Char/Short/Int/Long]Tensor`
  - `torch.cuda.[Float/Double/Half/Byte/Char/Short/Int/Long]Tensor`
  - `t.cuda() / t.to(device) / t.is_cuda()`
  - `t.abs() ↔ torch.abs(t)`

    `# NOTE: t.fun_() is in-place operation same as t.fun()`
  - `t.transpose(dim0, dim1) / t.permute(*dims)  # t.permute(2,0,1) HWC→CHW`
  - `t.type_as(t1)`
  - `t.view(*shape) / t.contiguous()  # tensor.view 要求内存连续`

PyTorch基本介绍 – 常用接口

- torch.nn
  - nn.Module *# base class of all network layers*
    - .cpu|.cuda / .train|.eval / .forward / .parameters / .zero_grad
  - nn.**Sequential** / nn.ModuleList
  - nn.**Conv2d**/ConvTransposed?d/MaxPool?d/AvgPool?d
  - nn.ReLU/ELU/LeakyReLU/PReLU/Sigmoid/Tanh/Softplus/Softmax
  - nn.BatchNorm2d/SyncBatchNorm/InstanceNorm?d/LayerNorm/GroupNorm/
  - nn.RNN/LSTM/GRU
  - nn.**Linear**
  - nn.Dropout
  - nn.DataParallel

PyTorch基本介绍 – 常用接口

- torch.nn
  - nn.L1Loss/MSELoss/BCELoss/CrossEntropyLoss/NLLLoss
  - nn.PixelShuffle/nn.Upsample/UpsamplingNearest2d/UpsamplingBilinear2d
  - nn.functional *# functions of previously mentioned layers*
- torch.optim
  - optim.SGD/Adam/…
    - optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
    - optimizer = optim.Adam([

```
                {'params': model.base.parameters()},
                {'params': model.classifier.parameters(), 'lr': 1e-3}
            ], lr=1e-2, momentum=0.9)
```

PyTorch基本介绍 – 常用接口

```
>>> x = torch.randn((1, 1), requires_grad=True)
>>> with torch.autograd.profiler.profile() as prof:
...     y = x ** 2
...     y.backward()
>>> # NOTE: some columns were removed for brevity
... print(prof)
-----------------------------------  ---------------  ---------------
Name                                          CPU time        CUDA time
-----------------------------------  ---------------  ---------------
PowConstant                                142.036us          0.000us
N5torch8autograd9GraphRootE                 63.524us          0.000us
PowConstantBackward                        184.228us          0.000us
MulConstant                                 50.288us          0.000us
PowConstant                                 28.439us          0.000us
Mul                                         20.154us          0.000us
N5torch8autograd14AccumulateGradE           13.790us          0.000us
N5torch8autograd5CloneE                      4.088us          0.000us
```

PyTorch基本介绍 – 常用接口

- torch.utils
  - utils.checkpoint / torch.save|torch.load
  - utils.data
    - .Dataset
    - .DataLoade
  - utils.model_zoo.load_url
  - utils.tensorboard (currently experimental)

PyTorch基本介绍 – 常用接口

- torchvision
  - `torchvision.datasets (some are listed below)`
    - `MNIST / Fashion-MNIST / COCO / LSUN / CIFIAR / VOC / Cityscapes / Imagenet-12`
  - `torchvision.models`
    - `AlexNet / VGG / ResNet / DenseNet / Inception / GoogleNet / ShuffleNet`
  - **`torchvision.transforms`**
    - `Compose` *`# Compose a series of transform operations`*
    - `CenterCrop/RandomCrop/RandomHorizontalFlip/RandomRotation/Resize`
    - `Normalize/ToPILImage/ToTensor/`
  - `torchvision.utils.save_image`

常见数据集介绍

- MNIST http://yann.lecun.com/exdb/mnist/
- Fashion-MNIST https://github.com/zalandoresearch/fashion-mnist
- MS COCO http://cocodataset.org/#overview
- LSUN http://lsun.cs.princeton.edu/
- ImageNet http://image-net.org/
- CIFAR https://www.cs.toronto.edu/~kriz/cifar.html
- Pascal VOC http://host.robots.ox.ac.uk/pascal/VOC/
- Cityscapes https://www.cityscapes-dataset.com/

常见数据集介绍 – MNIST

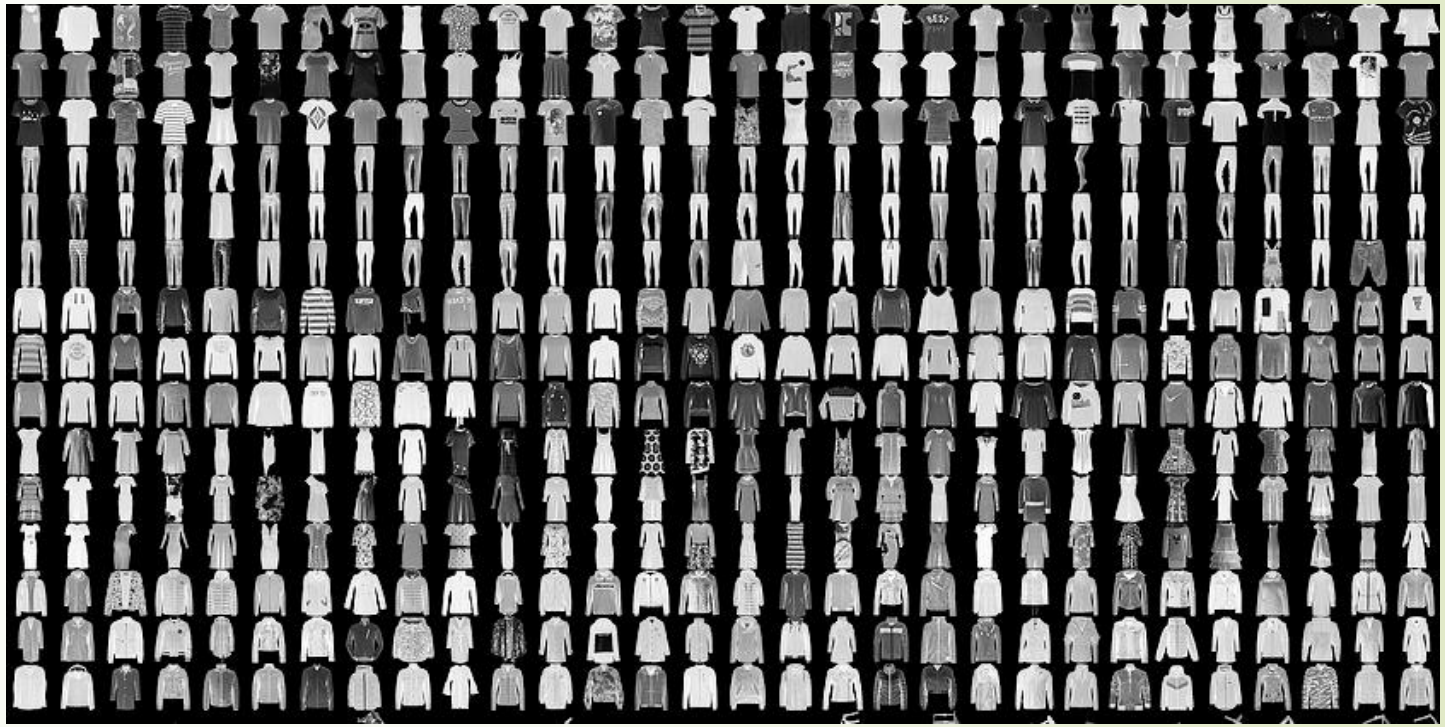
- 10 classes
- 60,000 training set
- 10,000 test set

- Task
  - Classification
  - Generation

常见数据集介绍 – Fashion-MNIST

- 10 classes
- 60k training set
- 10k test set

- Task
  - Classification
  - Generation



常见数据集介绍 – MS COCO

- COCO 2015 as e.g.
- 80 classes
- Over 200k images



- Task
  - Detection
  - Caption
  - Segmentation

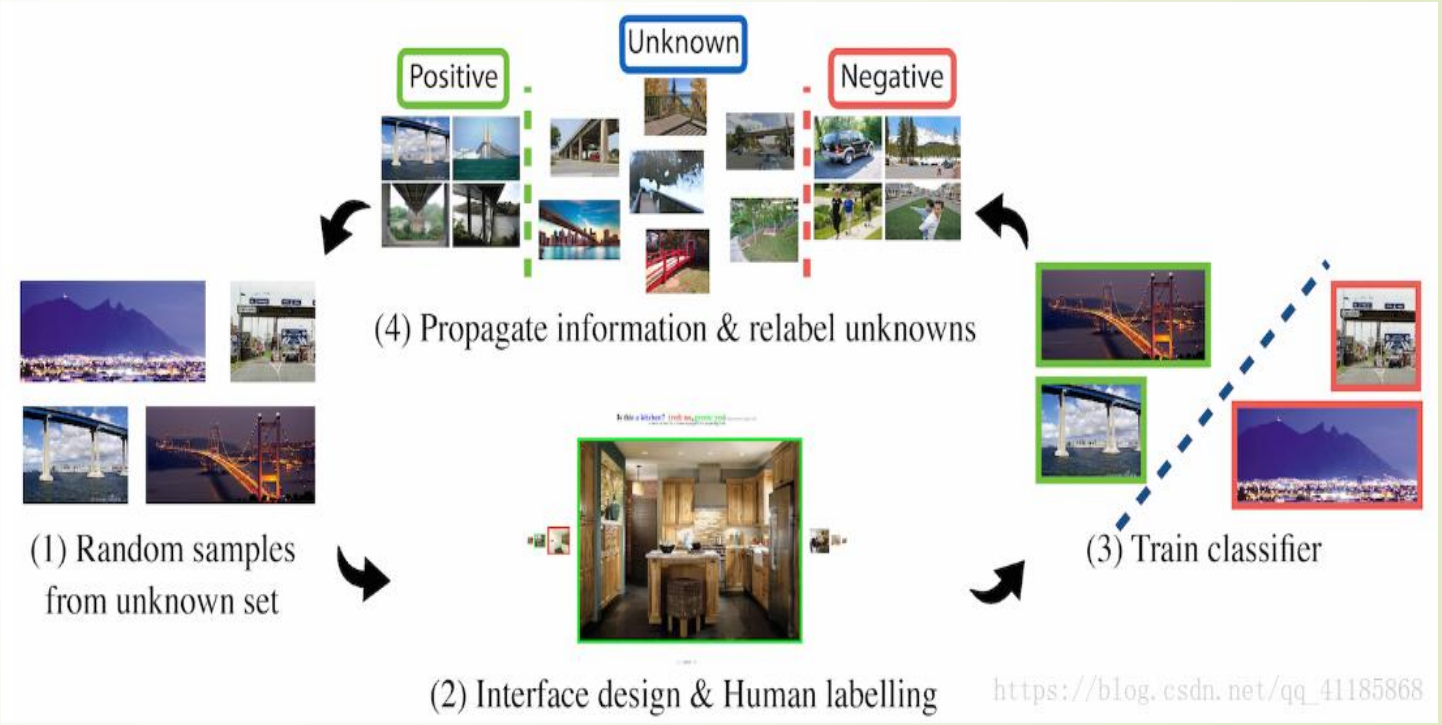

The man at bat readies to swing at the pitch while the umpire looks on.


A large bus sitting next to a very tall building.

常见数据集介绍 – LSUN

- 10 scenes/20 objects
- ~1M images

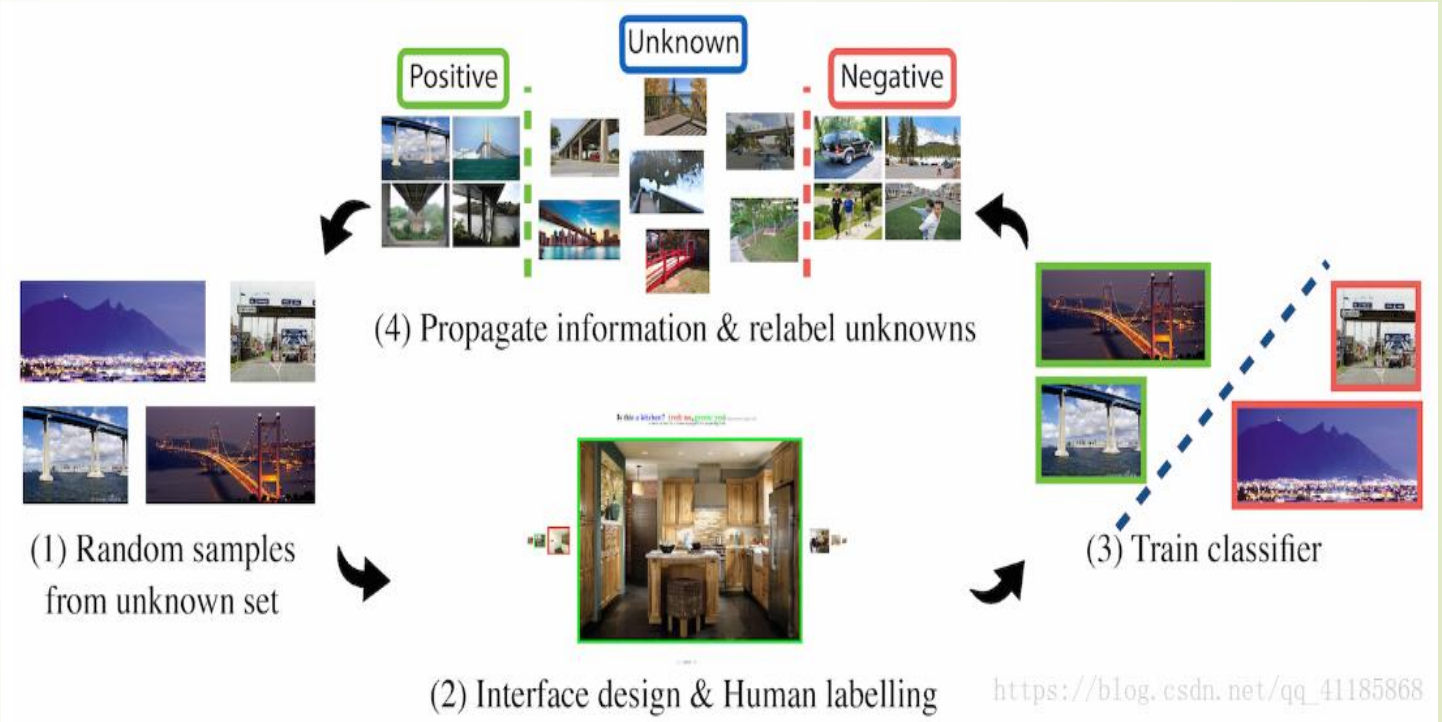
- Task
  - Classification
  - Generation
  - Saliency Prediction



常见数据集介绍 – ImageNet

- 1000 classes
- >14M images

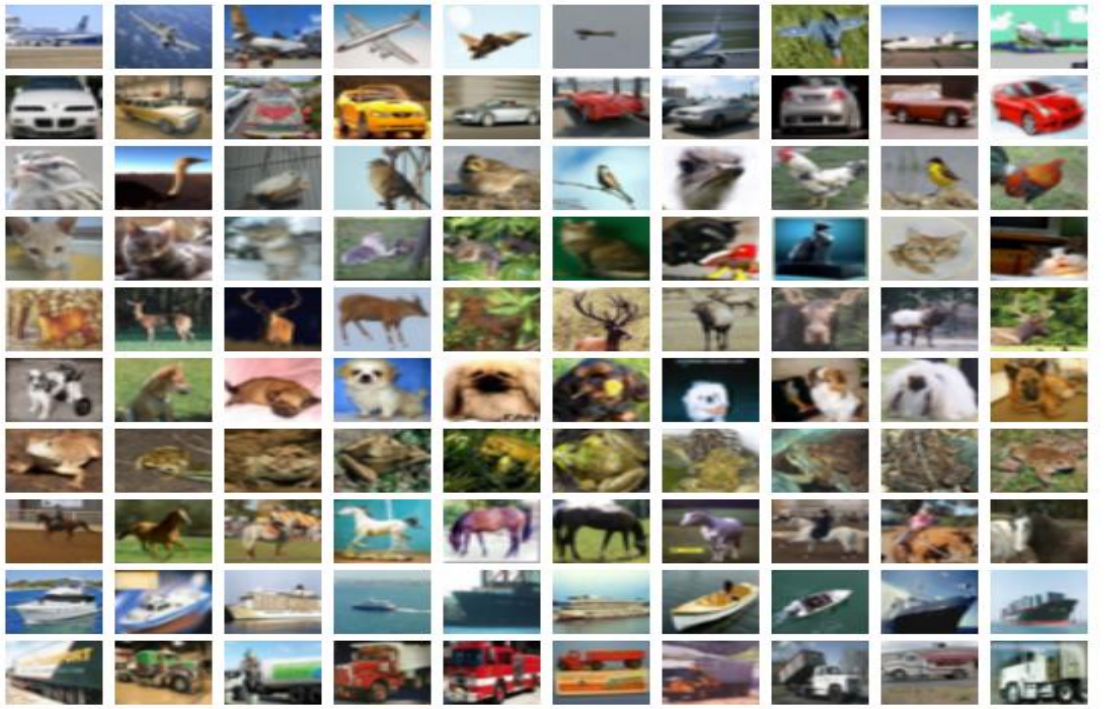
- Task
  - Classification
  - Generation
  - Detection
  - …
- Pre-training



常见数据集介绍 – Cifar

- 10/100 classes
- Cifar-10 as e.g.
  - 50k training images
  - 10k test images
- Task
  - Classification
  - Generation

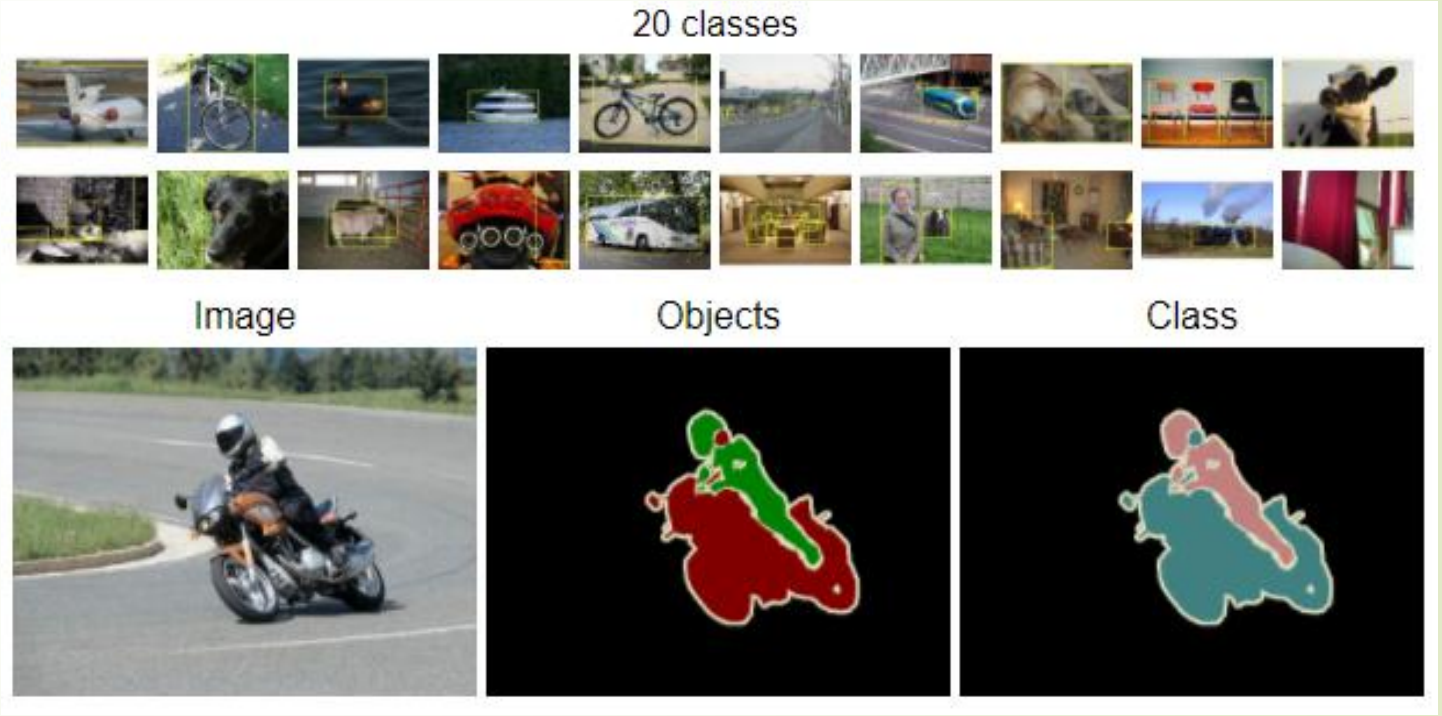
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



常见数据集介绍 – Pascal VOC

- 20 classes
- >10k images

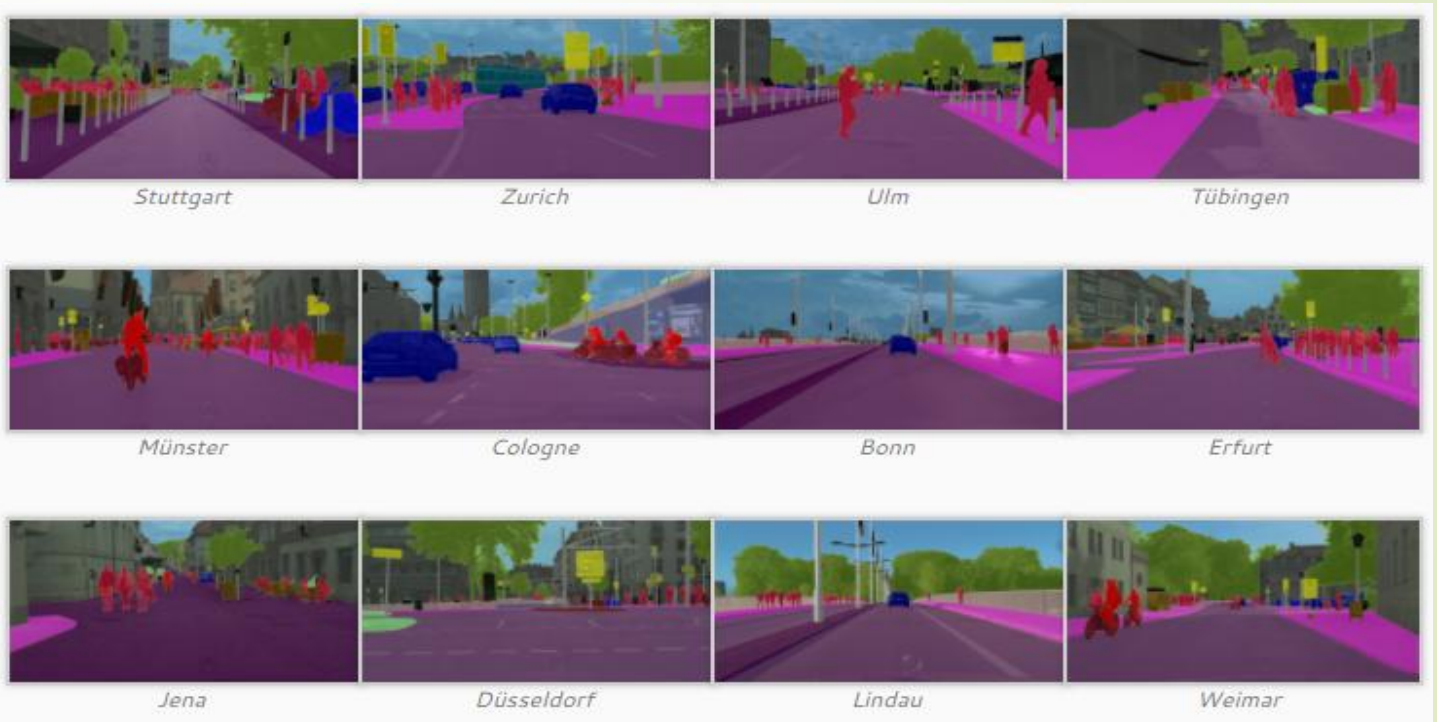
- Task
  - Classification
  - Detection
  - Segmentation



常见数据集介绍 – Cityscapes

- 30 classes
- 5k pixel-level
- 20k weakly anaotated

- Task
  - Segmentation
  - Generation



环境配置 – Python

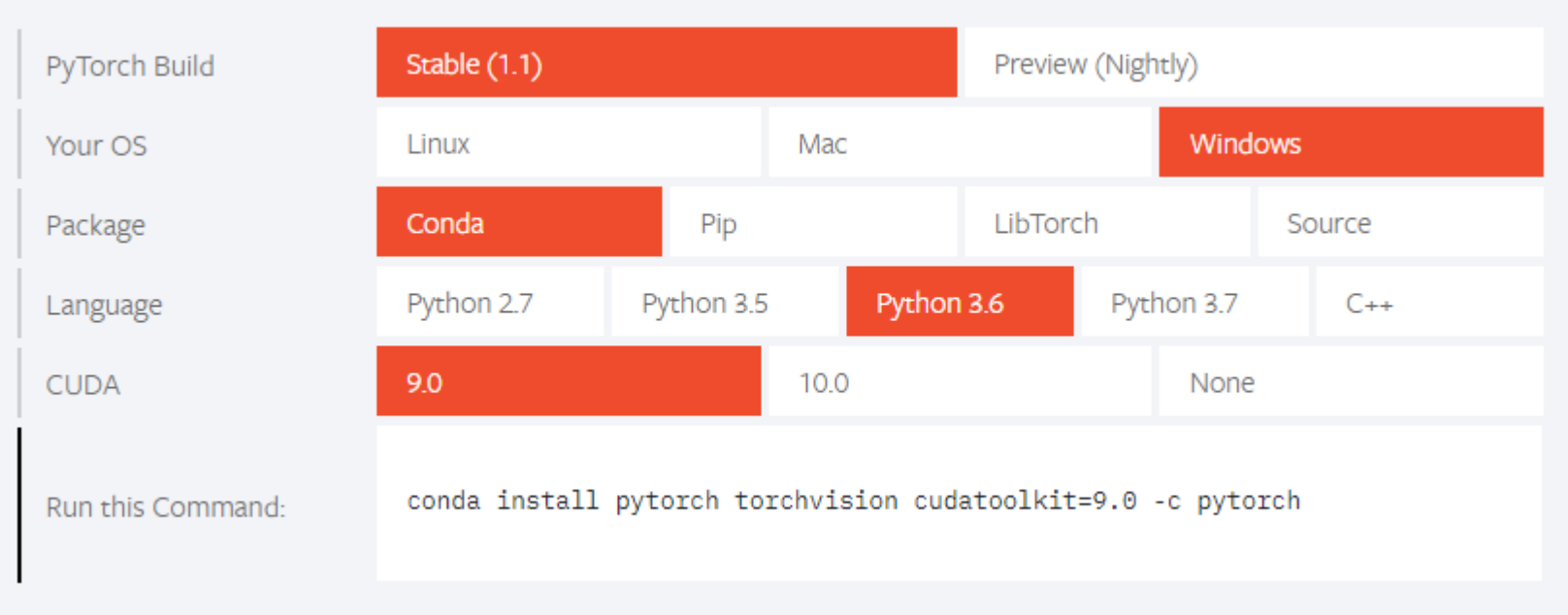
- Python 现行版本 http://python.org
  - Python 2.7将于2020年1月1日停止支持，建议使用Python 3.x
  - Python 3.5 ~ Python 3.7为当前活跃版本
- Python 安装与版本管理 http://anaconda.org
  - 常用Anaconda, PyEnv, pip(自带)等，建议使用Anaconda维护多虚拟环境，命令举例：
    - conda create –n py3 python=3.6 (创建Python版本为3.6，名字为py3的虚拟环境)
    - conda install pytorch-cpu torchvision-cpu –n py3 –c pytorch (在py3环境下安装PyTorch CPU版本)
- 深度学习常用第三方包
  - numpy, opencv, matplotlib, PIL, scipy等
    - python –m pip install numpy opencv-python matplotlib pillow scipy
    - conda install numpy opencv matplotlib pillow scipy

环境配置 – 计算加速

- 深度学习加速
  - NVIDIA GPU + CUDA (最常见)
  - Google TPU (Google 开发，据说配合TensorFlow使用速度很快) https://cloud.google.com/tpu/
  - FPGA (嵌入式)
  - NPU (移动端)
  - ….
- CUDA安装 https://developer.nvidia.com/cuda-toolkit&https://developer.nvidia.com/cudnn
  - 支持最广泛 cuda 8.0 + cudnn v5.1 (支持Torch、MatConvNet等)
  - 较新较稳定 cuda 9.x + cudnn v6/7
  - 最新版本 cuda10.x + cudnn v7 (支持Turing架构最新特性，RTX系列显卡Tensor Core加速)

环境配置 – PyTorch安装

- 参考https://pytorch.org
  - 建议安装前配置第三方pypi/anaconda源



资源推荐

- PyTorch文档
  - https://pytorch.org/docs/stable/index.html (EN) / https://pytorch-cn.readthedocs.io/zh/latest/ (ZH)
- 矩阵求导
  - 简单教程 https://github.com/LynnHo/Matrix-Calculus
  - 求导网站 http://www.matrixcalculus.org/
- Python教程
  - 菜鸟教程 https://www.runoob.com/python/python-tutorial.html
- PyTorch教程
  - 莫烦系列 https://morvanzhou.github.io/tutorials/machine-learning/torch/

深度学习课程设计(一)

王昊

wanghao@ise.neu.edu.cn

实验1 卷积神经网络实现

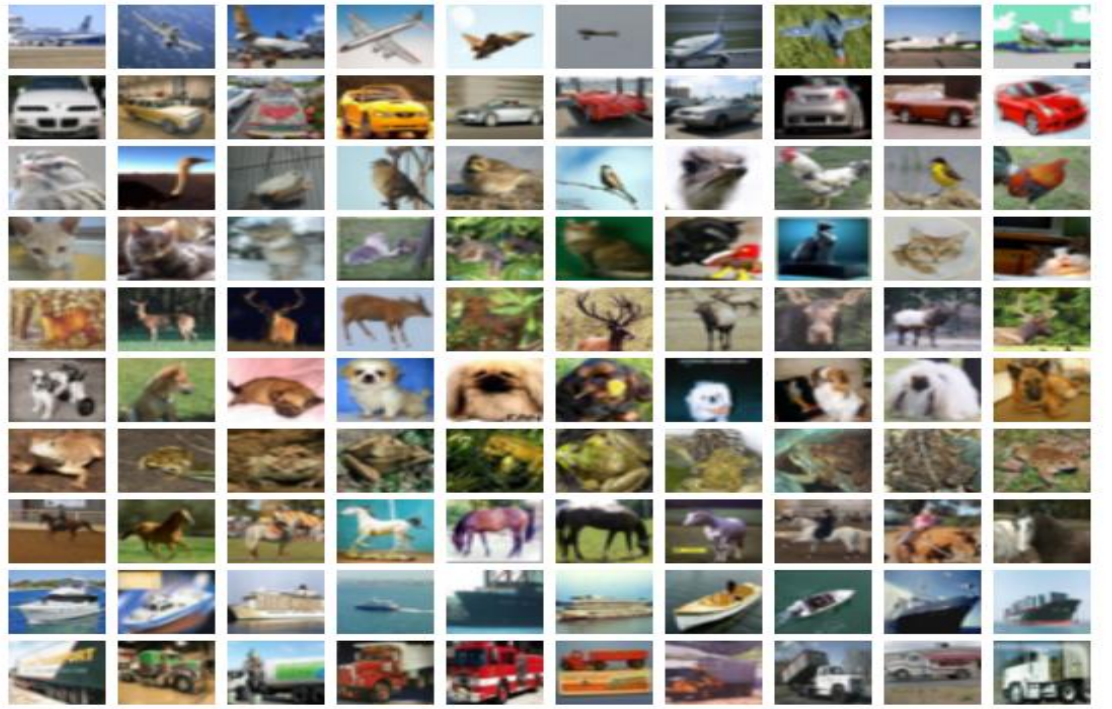
- 基于PyTorch实现AlexNet [1]结构
- 在Cifar-10数据集上进行验证
- 如有条件，尝试不同参数的影响，尝试其它网络结构
- 请勿使用torchvision.models.AlexNet
- 作业模板：参考群内作业模板即可。
- 提交时间：13周周四中午12:00前，BB平台电子版

[1] Krizhevsky A , Sutskever I , Hinton G . ImageNet Classification with Deep Convolutional Neural Networks[C]// NIPS. Curran Associates Inc. 2012.

常见数据集介绍 – Cifar

- 10/100 classes
- Cifar-10 as e.g.
  - 50k training images
  - 10k test images
- Task
  - Classification
  - Generation

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



深度学习课程设计(二)

王昊

wanghao@ise.neu.edu.cn

实验2 常用网络结构实现

- 基于PyTorch实现VGG结构
  - VGG要求实现VGG-11（Conv部分按论文实现，Classifier直接一层全连接即可）；
  - 基于VGG进行训练方式对比（LR对比三组及以上【此时选用任一种优化器】，优化器对比SGD与Adam【选用LR对比时的最佳LR】）
- 在Cifar-10数据集上进行验证

实验2 常用网络结构实现

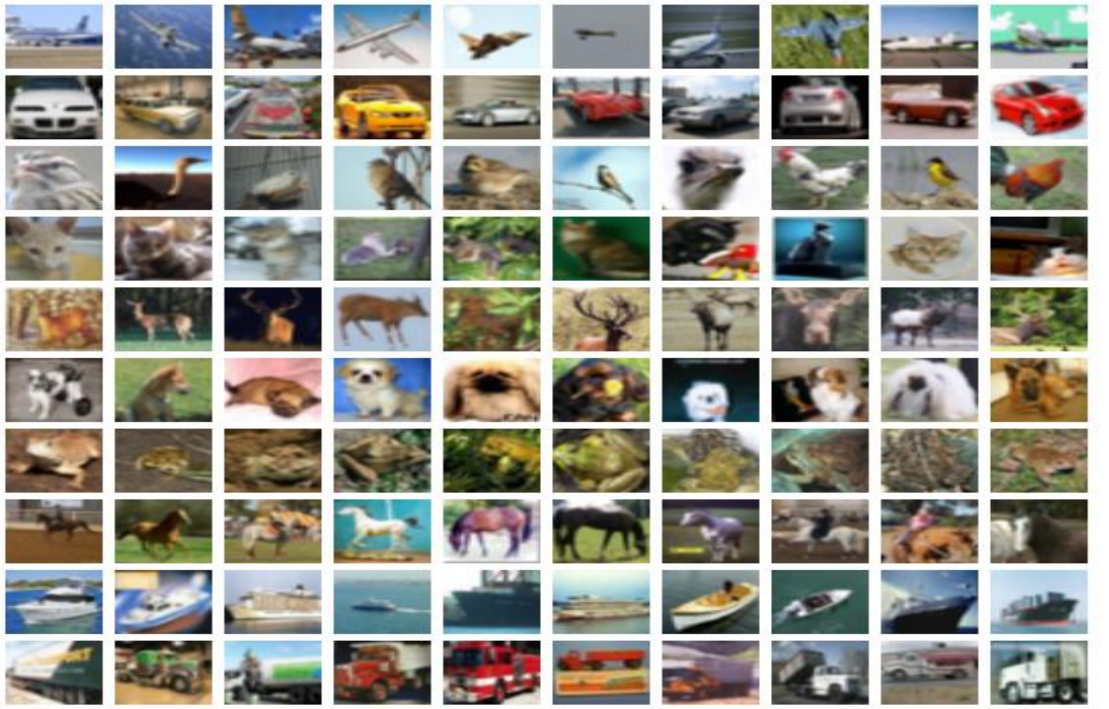
- 要求基于CUDA实现
  - 可设定是否使用GPU，默认参数设定为GPU
- 要求提交实验报告
  - VGG论文搜索过程记录（给出文字描述即可）
  - 实验结果及相应对比分析（要求有测试集准确率曲线）
  - 如有精力，可自行实现dataset（基于opencv或Pillow）

提交时间：15周周四中午12:00前，BB平台电子版

常见数据集介绍 – Cifar

- 10/100 classes
- Cifar-10 as e.g.
  - 50k training images
  - 10k test images
- Task
  - Classification
  - Generation

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



环境配置及背景知识

- OpenCV
  - `python -m pip install opencv-python`　　`>>> import cv2`
- Pillow
  - `python -m pip install pillow`　　`>>> from PIL import Image`
- Dataset（继承 torch.utils.data.Dataset）
  - `def __init__(self, args)`
  - `def __len__(self)`
  - `def __getitem__(self, index)`
    `return {'image': image, 'label': label}`

环境配置及背景知识

Dataset简单示例

```
def __getitem__(self, index):
    image = cv2.imread(self.paths[index])
    label = self.labels[index]
    return {'image': image, 'label': label}
def __len__(self):
    return len(self.paths)
```

评分标准

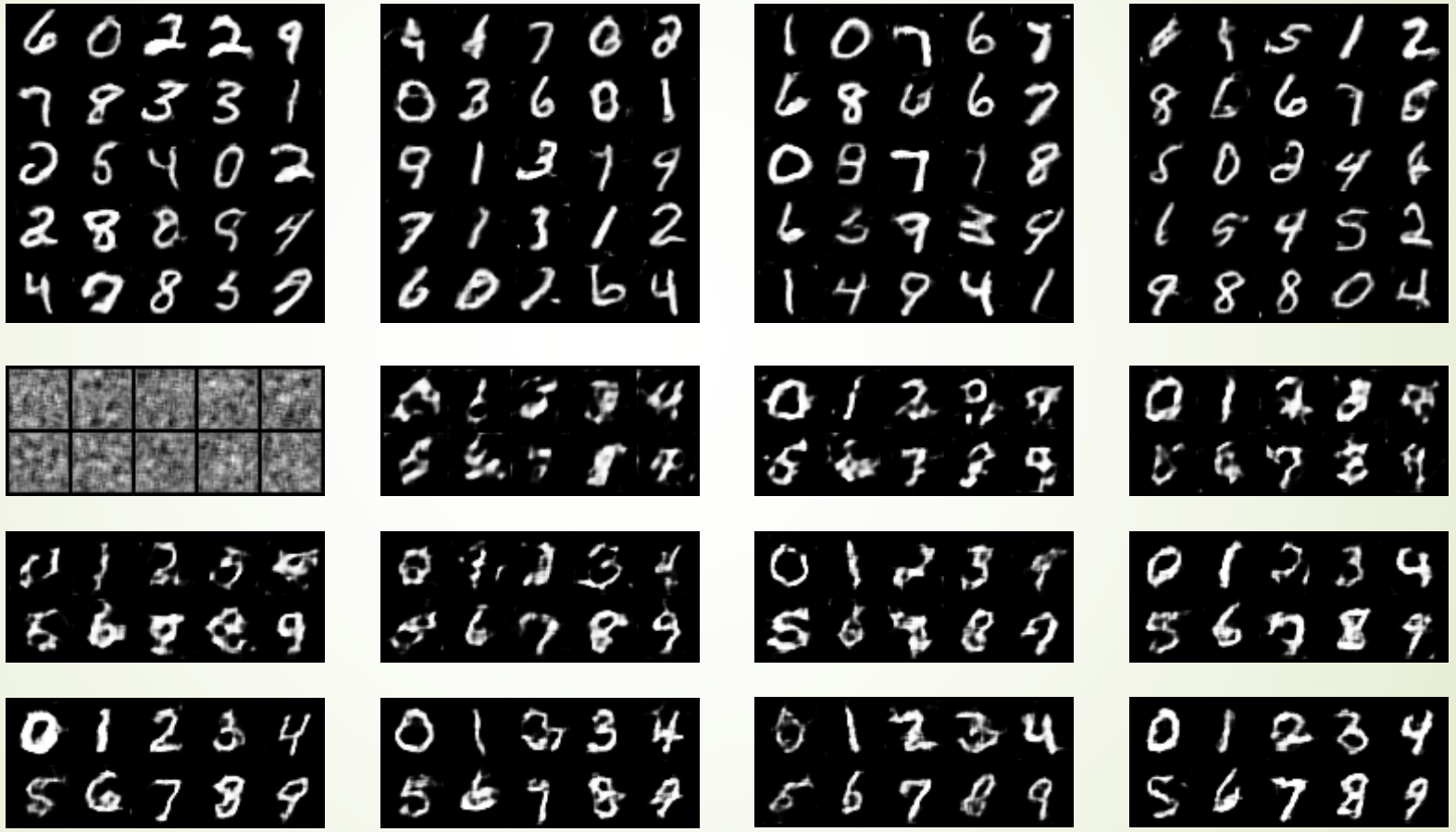
- VGG-11 （结构、性能符合要求）
- VGG-11对比实验 （符合要求）
- Cuda运行（能使用cuda运行 ）
- 手动实现dataset （正确实现）
- 报告可读性/完整性/格式
- 论文搜索（从搜索引擎到论文PDF的完整过程）

深度学习课程设计(三)

崔建江

cuijianjiang@ise.neu.edu.cn

生成对抗网络 (GAN)



实验3 生成对抗网络

- 基于PyTorch实现生成对抗网络
  - 拟合给定分布
  - 要求可视化训练过程
  - 实验报告
  - 对比GAN、WGAN、WGAN-GP（稳定性、性能）
  - 对比不同优化器的影响

提交时间：17周周六中午12:00前，BB平台电子版