深蓝学院
shenlanxueyuan.com

ImageNet 分类实践

主讲人 洪振

中国科学院自动化研究所
模式识别国家重点实验室

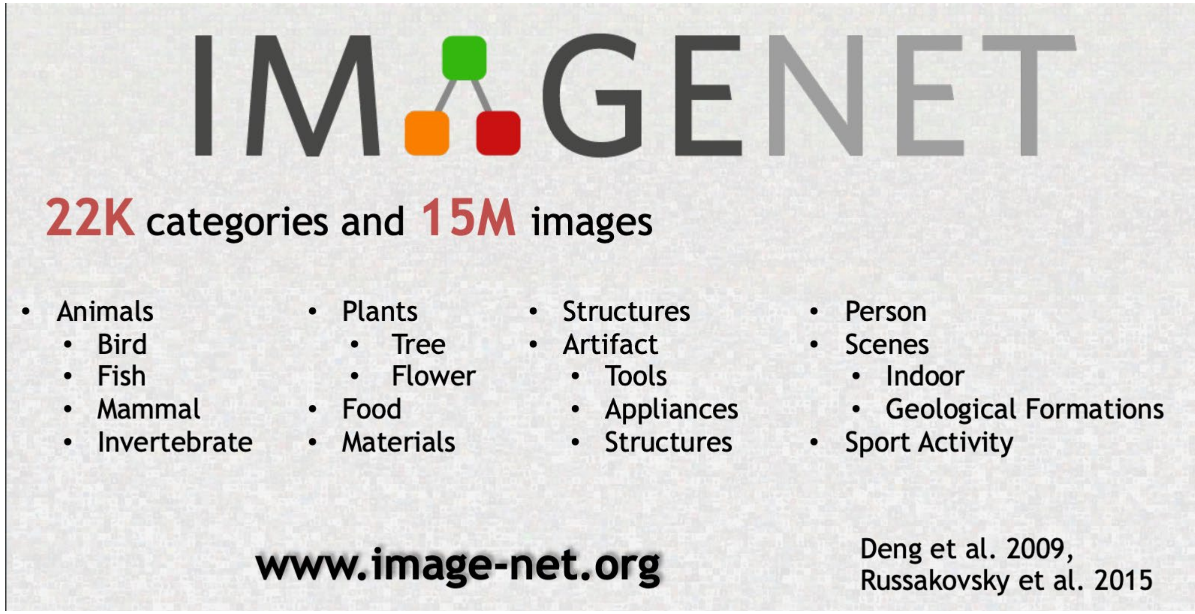
课程内容

- ✓**ImageNet简介**
- ✓**Demo演示: TinyImageNet**

ImageNet简介

□ ImageNet和ILSVRC

- □ ImageNet是一个超过15 million的图像数据集，大约有22,000类
- □ ILSVRC全称ImageNet Large-Scale Visual Recognition Challenge，从2010年开始举办到2017年最后一届，使用ImageNet数据集的一个子集，总共有1000类，120w训练图片



IMAGENET

22K categories and 15M images

- Animals
  - Bird
  - Fish
  - Mammal
  - Invertebrate
- Plants
  - Tree
  - Flower
- Food
- Materials
- Structures
- Artifact
  - Tools
  - Appliances
  - Structures
- Person
- Scenes
  - Indoor
  - Geological Formations
- Sport Activity

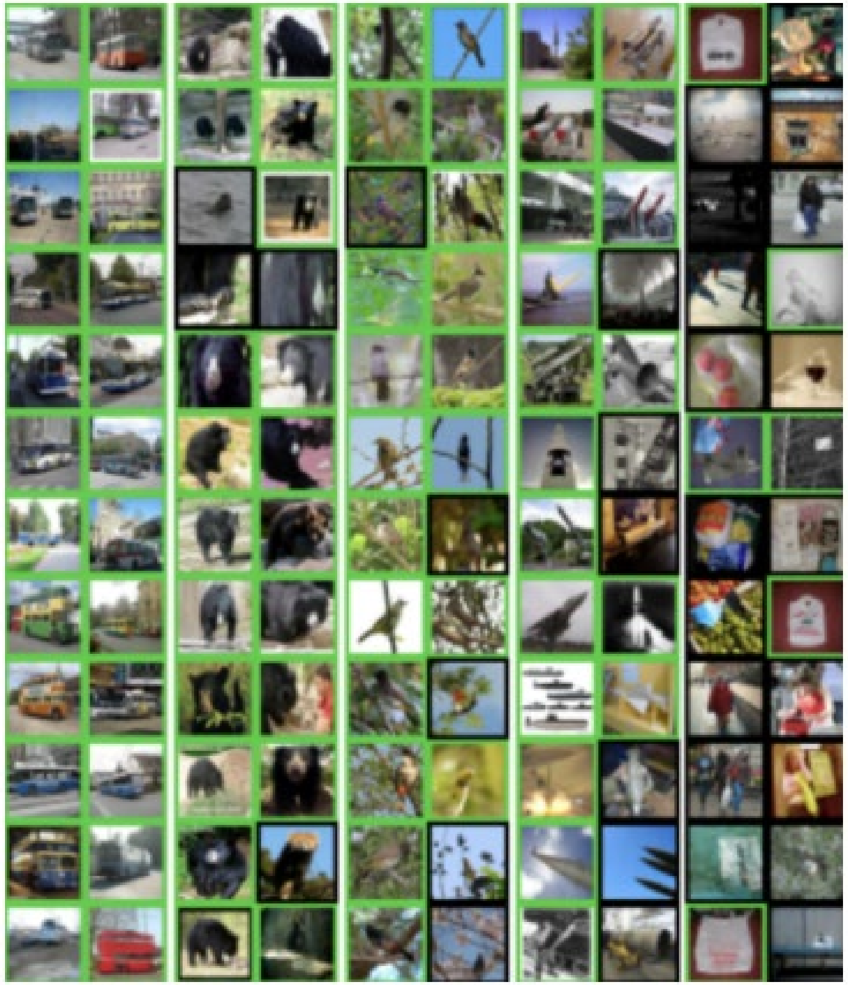
www.image-net.org

Deng et al. 2009,
Russakovsky et al. 2015

ImageNet数据集百度云链接
链接：https://pan.baidu.com/s/1-TE-FJlBa88vxTEzP1Ju9Q
提取码：caza

ImageNet简介

- TinyImageNet
  - 200类，每类有500张训练样本、50张测试样本和50张验证样本
  - 数据结构与ImageNet相同，每张图片下采样至64x64, 数据集大小237M
  - 斯坦福cs231课程采用作为案例，方便快速进行算法学习



- 下载：

```
wget http://cs231n.stanford.edu/tiny-imagenet-200.zip
unzip tiny-imagenet-200.zip
rm tiny-imagenet-200.zip
```

https://arxiv.org/pdf/1707.08819.pdf

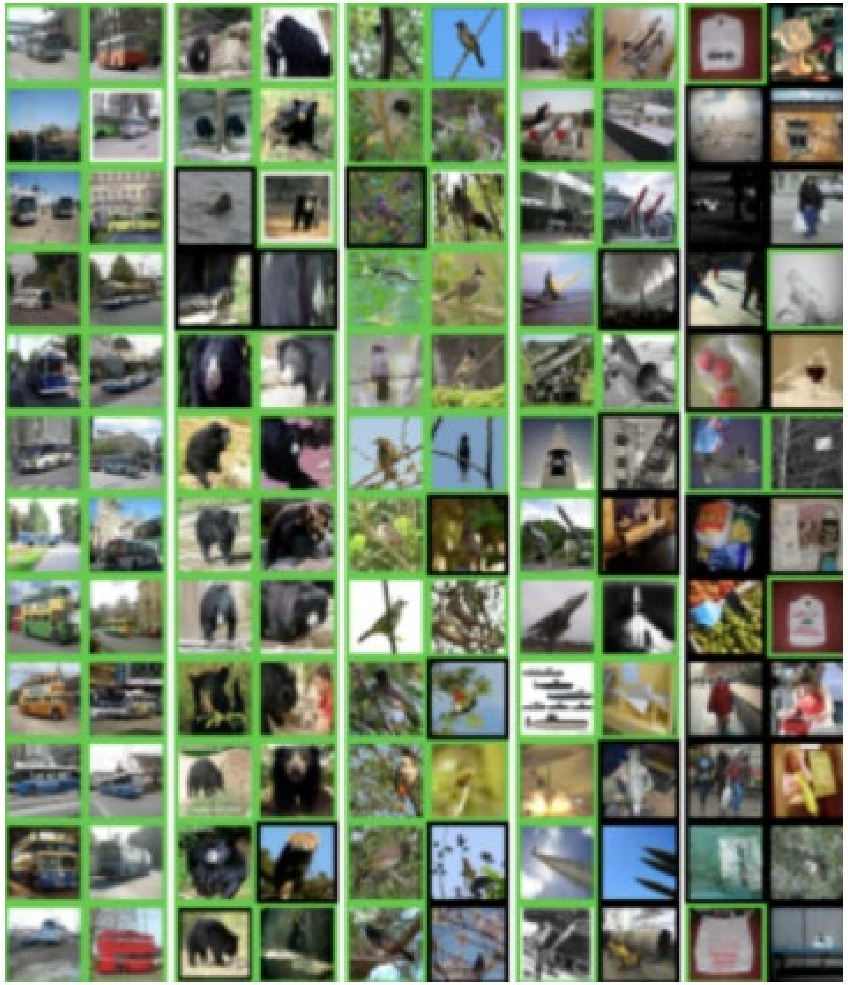
ImageNet使用

□ Github： https://github.com/DennisHanyuanXu/Tiny-ImageNet

```
pip install -r requirements.txt
```

□ 其它安装方法：

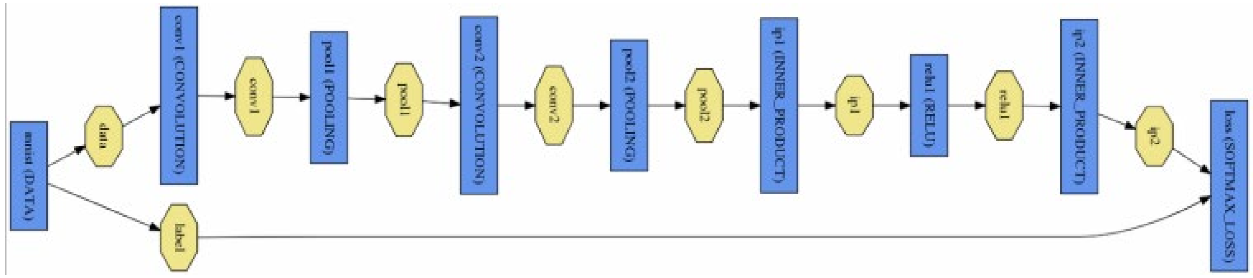
- □ conda install xxx # https://anaconda.org/
- □ pip install xxxx   # https://pypi.org/



Pytorch 构建神经网络

设计网络(前向传播)

准备数据 →



设计损失函数（计算loss）

反向传播，更新参数

- □ 准备数据：Dataset+DataLoader
- □ 网络设计：torch.nn.module
- □ 损失函数：torch.nn.CrossEntropyLoss, torch.nn.MSELoss
- □ 参数更新：torch.optim

optimizer = optim.SGD(net.parameters(), lr=args.lr, momentum=0.9, weight_decay=1e-4)

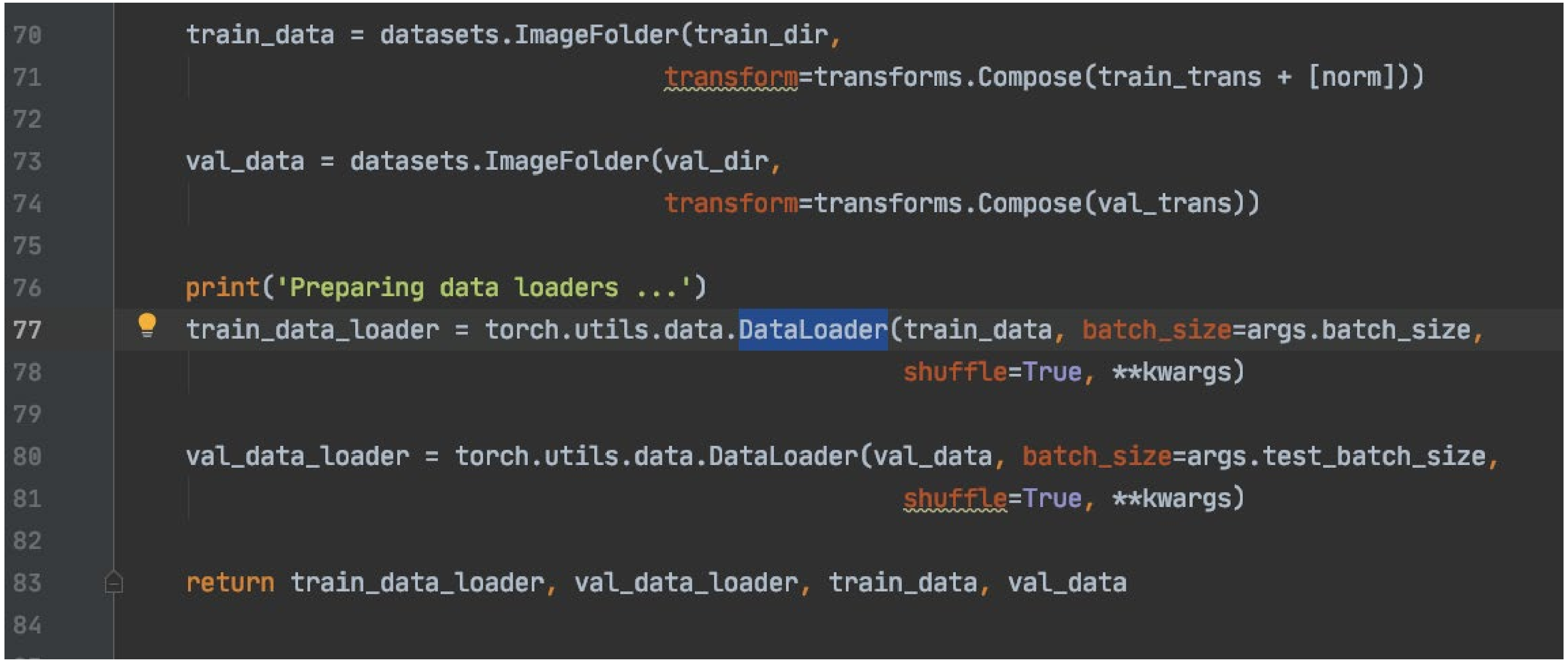
Pytorch—数据准备

- □ Dataset

  采用torchvision中 ImageFolder 类

- □ DataLoader

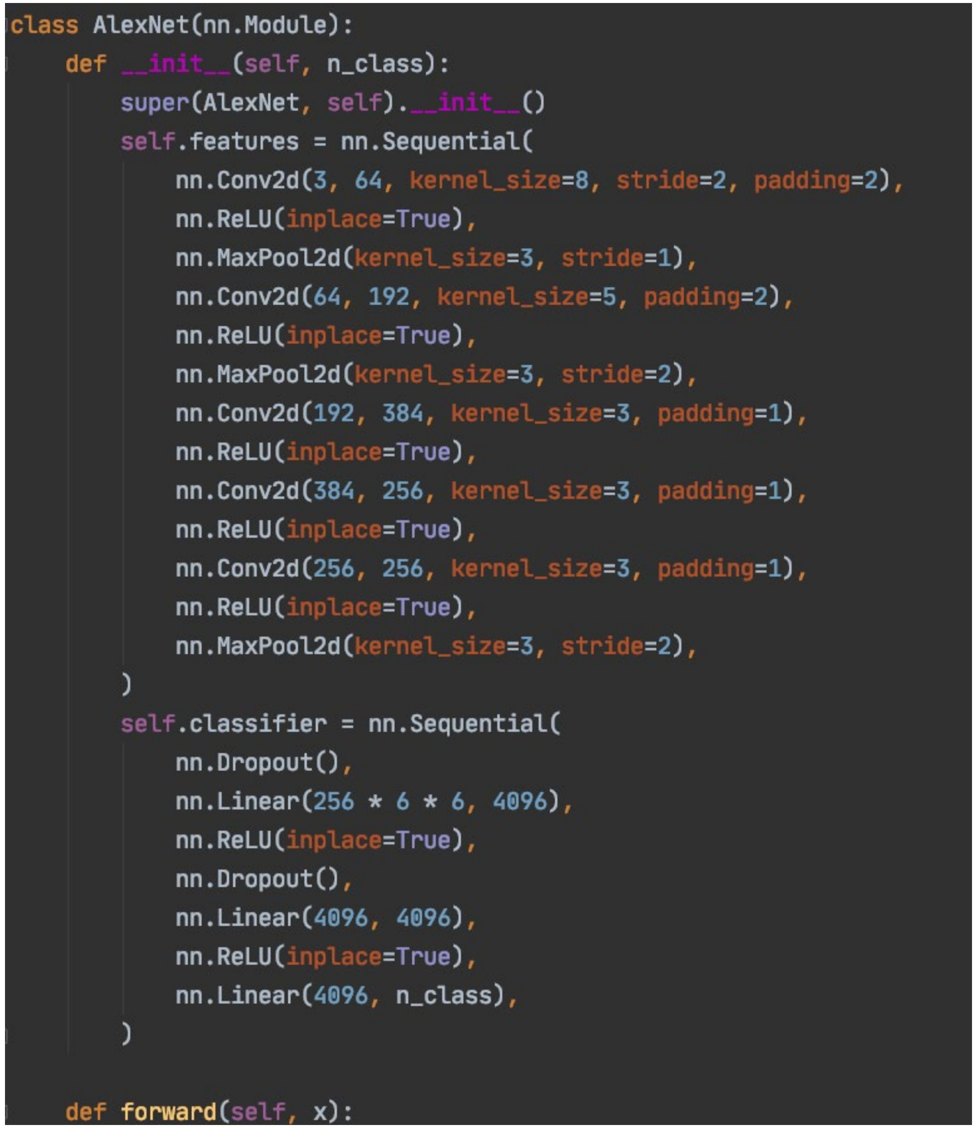
  采用torchvision中 默认DataLoader 类



```
    train_data = datasets.ImageFolder(train_dir,
                                    transform=transforms.Compose(train_trans + [norm]))

    val_data = datasets.ImageFolder(val_dir,
                                    transform=transforms.Compose(val_trans))

    print('Preparing data loaders ...')
    train_data_loader = torch.utils.data.DataLoader(train_data, batch_size=args.batch_size,
                                                    shuffle=True, **kwargs)

    val_data_loader = torch.utils.data.DataLoader(val_data, batch_size=args.test_batch_size,
                                                    shuffle=True, **kwargs)

    return train_data_loader, val_data_loader, train_data, val_data
```

网络训练

- □ 网络： AlexNet
- □ 损失函数：nn.CrossEntropyLoss
- □ 优化器Optimizer：SGD / Adam



```python
class AlexNet(nn.Module):
    def __init__(self, n_class):
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=8, stride=2, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=1),
            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
        )
        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Linear(256 * 6 * 6, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),
            nn.Linear(4096, n_class),
        )

    def forward(self, x):
```

感谢各位聆听

Thanks for Listening