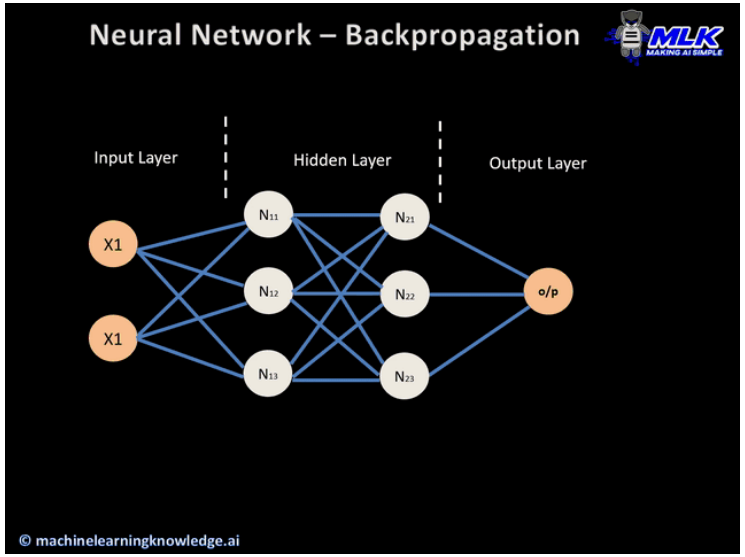
深度学习负载IO模式和优化策略

中国人民大学

鲁蔚征

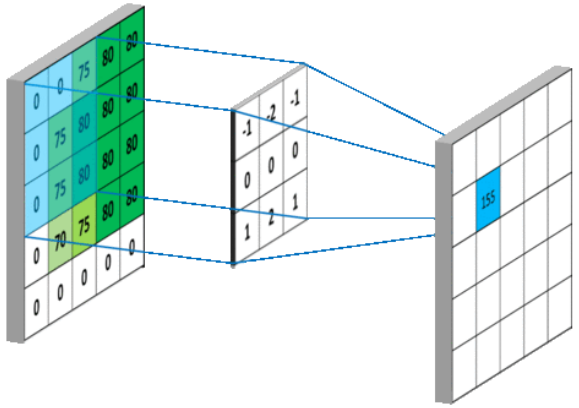
深度学习计算

- 训练和推理
  - 训练：在训练数据集上训练一个模型
  - 推理：使用该模型在未知数据上进行预测
- 典型训练过程
  - 构建神经网络模型
  - 前向：在训练数据集上训练，使用损失函数与标注数据对比，得到损失值
  - 反向：根据损失值更新模型参数

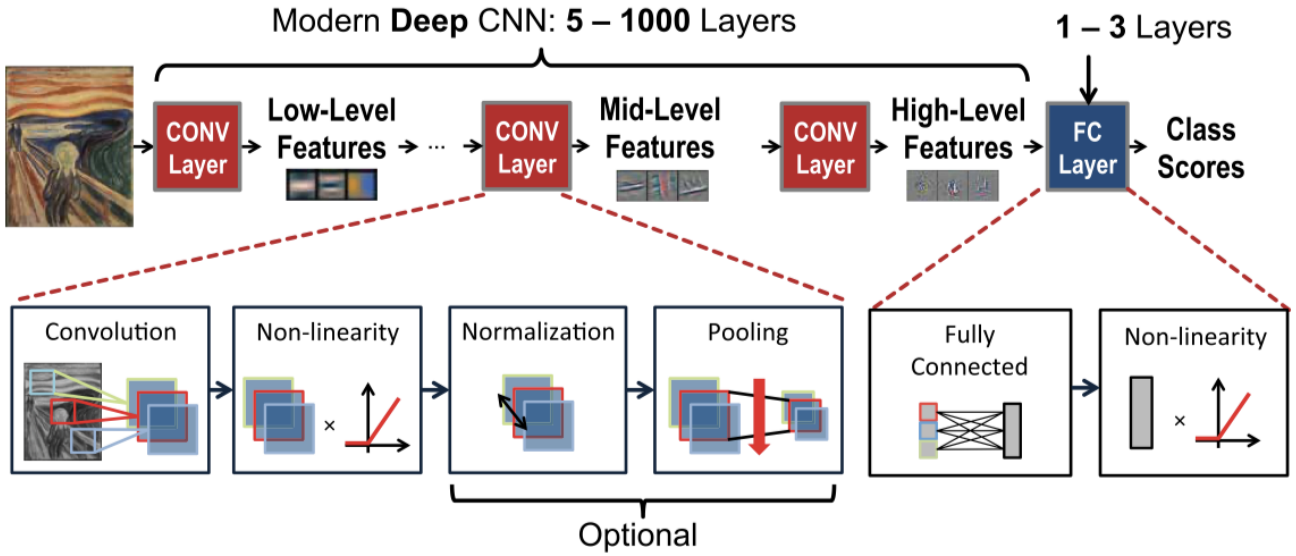


从算子到计算图

- 神经网络中有多种计算：卷积、全连接、归一化、激活函数，被抽象为算子
- 多种算子共同构成计算图



卷积（Convolution）



主流卷积神经网络由多个算子组成

SGD：随机梯度下降

- batch/iteration/step：由于处理器内存有限，一次处理 batch_size 个样本
- shuffle data：每个batch应该从训练集中随机挑选样本，使得模型具有更强的泛化能力，不至于只学到固定模式
- epoch：把训练集所有样本处理完
- 每个样本在一个epoch中被访问一次

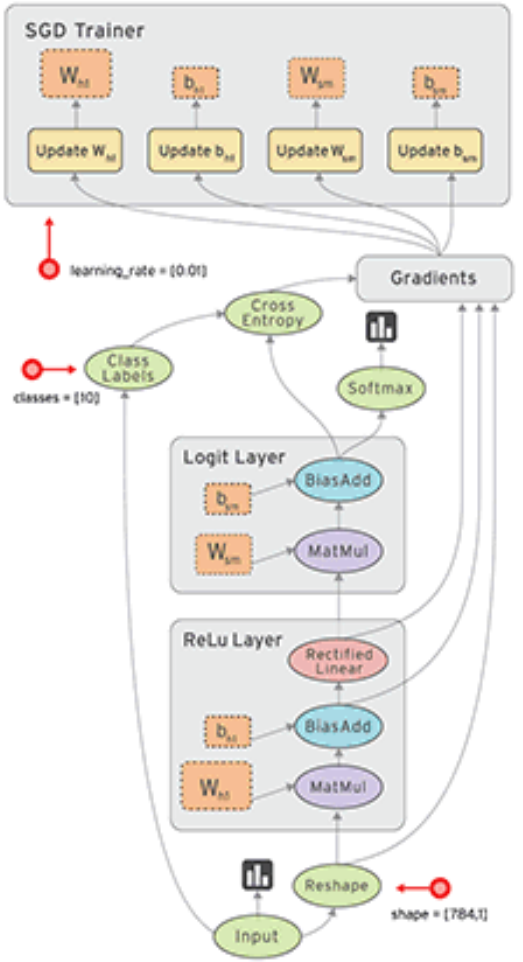
$$steps = \frac{total\ No.of\ examples}{batch\ size}$$

```
epochs = 20
for epoch in range(epochs):
    # shuffle dataset
    for step in enumerate(train_dataset):
        # forward with `batch_size` examples
        # backward and update parameters
```

IO相关

深度学习框架

- 静态图：
  - 代表：TensorFlow、MXNet、MindSpore
  - 计算图的构建与计算分离：事先定义好计算图，然后再对数据进行计算
  - 数据读写和预处理是静态图的一部分，在框架中由C++实现
- 动态图：
  - 代表：PyTorch
  - 计算图的构建和计算同时发生
  - Python友好但牺牲了性能
  - 数据读写依赖Python社区提供的包



IO相关

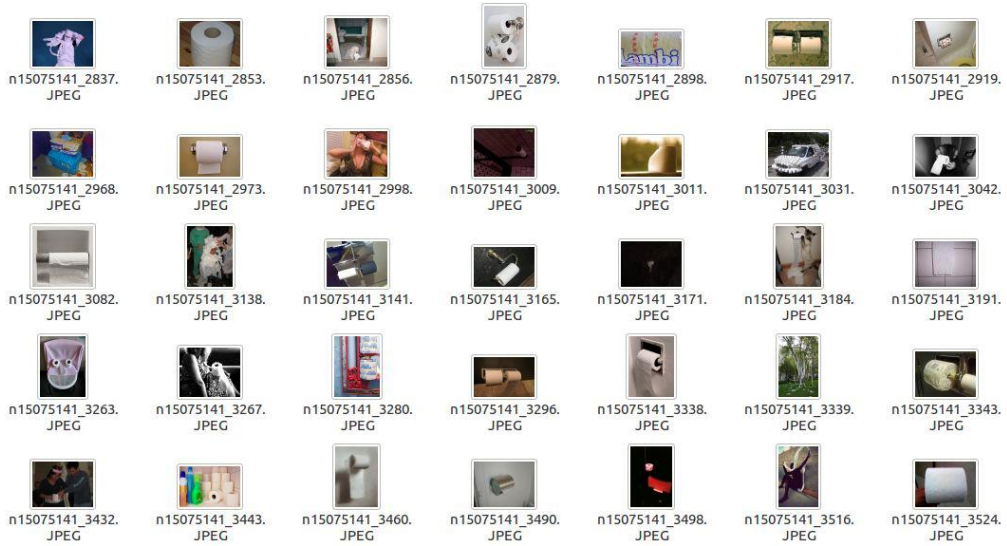
深度学习应用

- 计算机视觉
- 自然语言处理
- 搜索/广告/推荐

	计算机视觉	自然语言处理	搜索/广告/推荐
数据源	图片、视频段	文本	结构化数据
代表性应用	图像分类 目标检测	机器翻译 问答系统	信息流推荐 搜索广告
代表性网络	ResNet YOLO	Transformer BERT	DeepFM DIN

数据

- ImageNet 2012
  - 图像分类领域经典数据集
  - 130万张带标注的图片，约130GB
  - 每张图片为 10K－100K 左右大小
- WMT
  - 机器翻译领域经典数据集
  - 500MB
  - 需先进行文本预处理

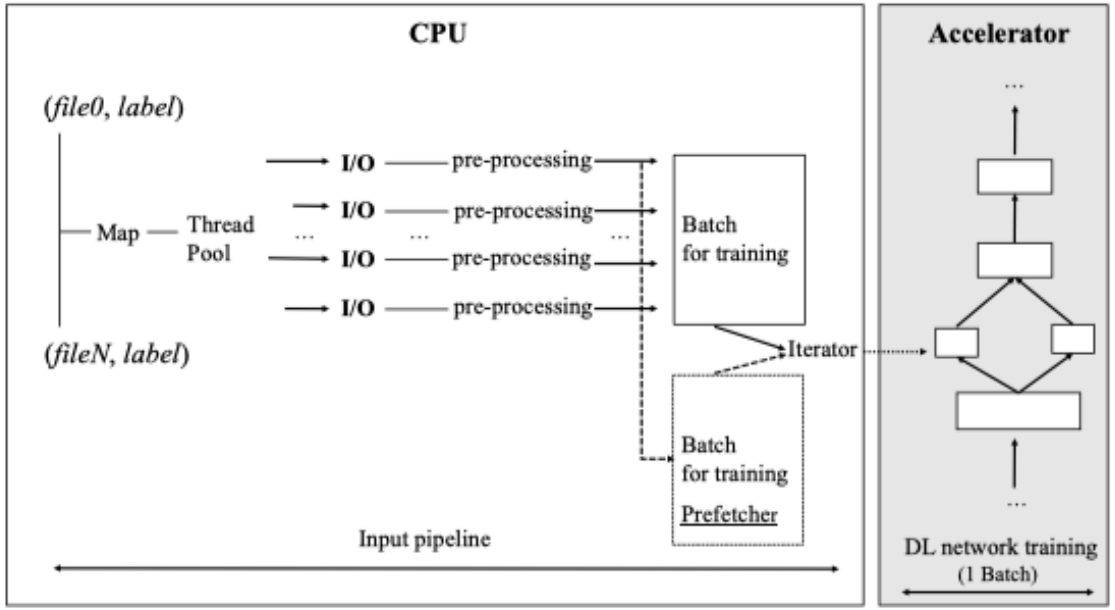


训练前预处理

- 启动深度学习训练任务前，需先对数据进行预处理
- 图片/视频
  - 图片缩放、视频切帧
  - 中小文件居多
- 文本
  - 构建词典，加必要的符号：句子起始符、结束符等
  - 处理后的数据以大文件形式居多
- TensorFlow
  - 将预处理后的数据放入 TensorFlow 预定义好的数据结构中
  - 小文件打包成大文件
- PyTorch
  - 直接读取文件

数据读取与训练时预处理

- 读取数据，进行必要的数据预处理，送入训练流程
- 训练时预处理：数据增广（翻转图片、裁剪图片）



典型训练预处理流程

数据读取：一个TensorFlow案例

- 对ImageNet数据集进行预处理
  - 将纯图片读取成数值格式
  - 多张图片数据打包，生成TFRecord格式文件，一个TFRecord大约1G
- TensorFlow 数据读取流程：
  - C++实现
  - 并行流水线

数据读取：一个PyTorch案例

- 以 PyTorch读取ImageNet数据集为例

获取数据集各个图片的文件名
生成一个Python List

使用Sampler Shuffle List，生成Batch Size组成的训练集

根据文件名，读取文件，调用collate_fn 对单个样本进行必要的预处理

使用Pyhton PIL（Pillow）库
解析图片文件
将文件读取进入内存
Random Access

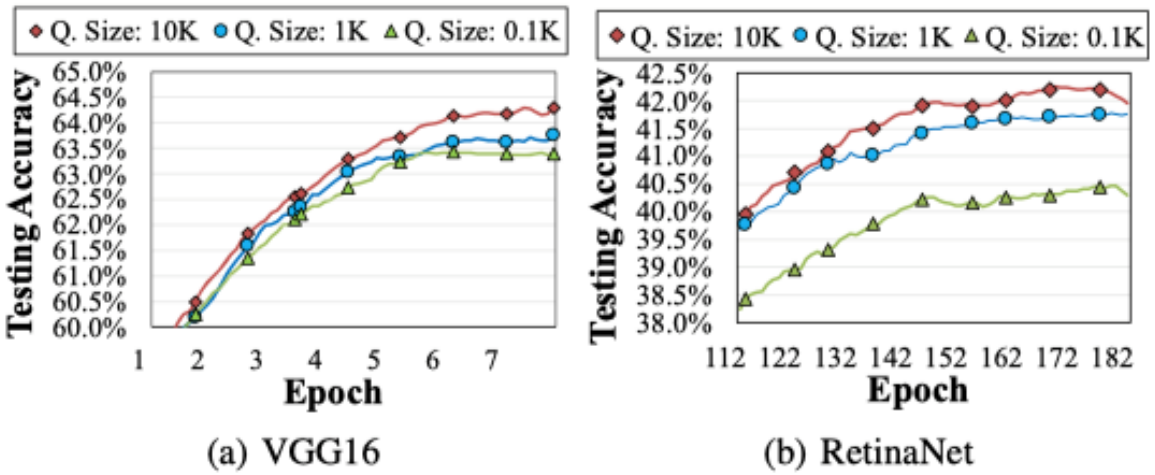
比较：TensorFlow vs. PyTorch

- TensorFlow
  - 需要对数据源进行处理，生成TF定义的一套数据格式
  - TF的数据读取在多线程、流水线等方面进行了优化
  - 不方便调试，不容易看到数据真实值
  - 对小文件打包，Shuffle不是绝对的随机

- PyTorch
  - 没有特定的数据格式，Python可读写
  - Random Access，对存储和文件系统有读写压力
  - 方便调试，可以直接打印某个训练数据

随机性与准确度

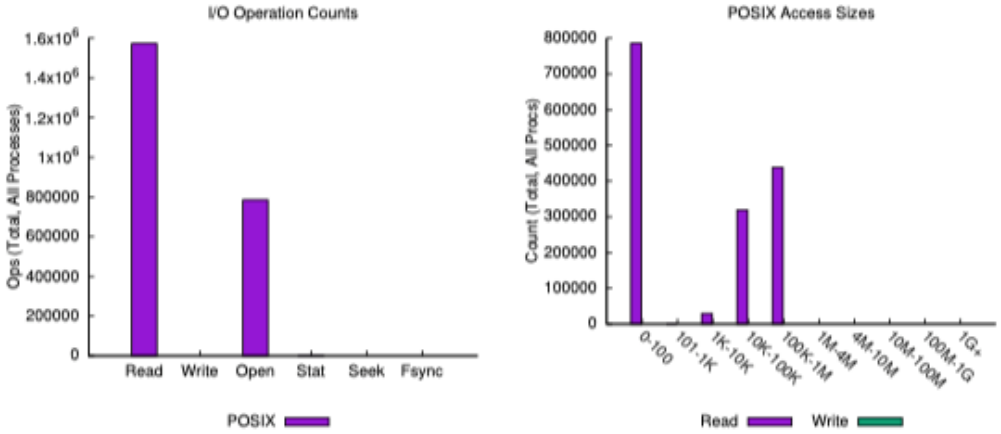
- 随机性越好，准确度越好
- 随机性带来存储访问的压力



Q Size越大，随机性越好，相同的Epoch准确度越高

以小文件读为主的IO特点

- POSIX IO集中在Read和Open上
- 少量的Write一般只用于生成Log，生成Checkpoint
- 小文件读写：Metadata Read
- Metadata Read容易造成性能瓶颈
- 小文件IO：元数据查询、数据传输多次交互

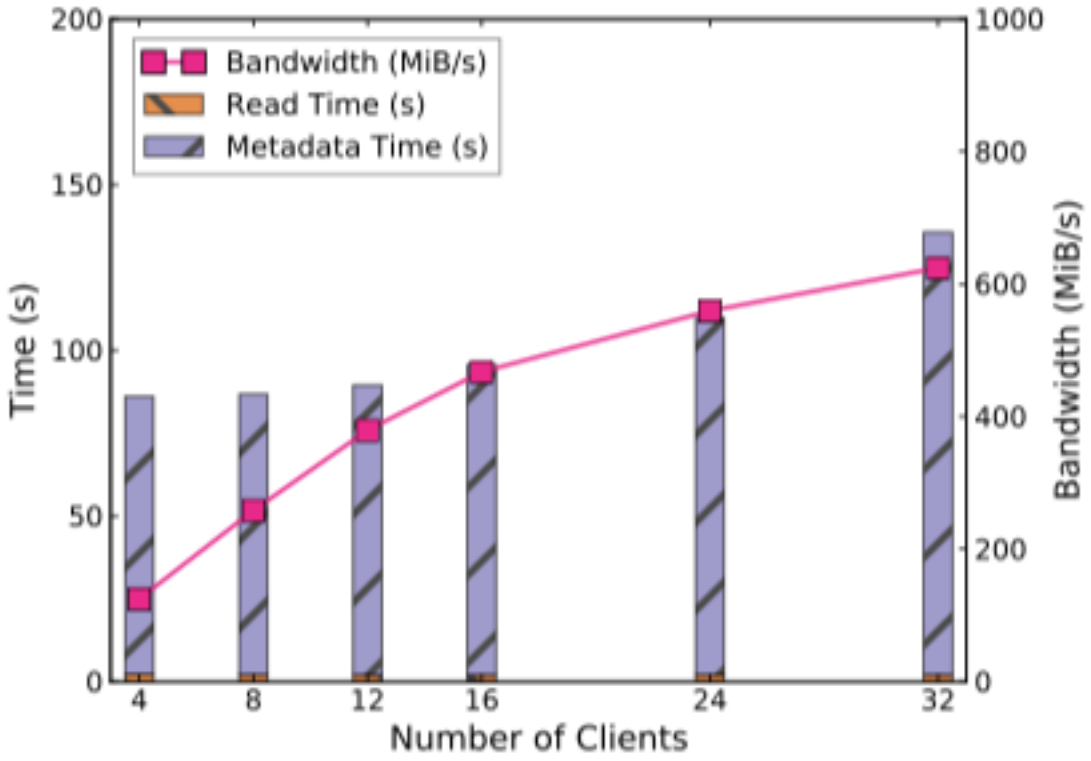


(a) Ops vs. I/O Operations (b) Operation Count vs. File Sizes

使用Darshan对TensorFlow ImageNet进行IO模式分析

小文件IO的横向扩展性

- TensorFlow文件读经过高度优化
- 在并行文件系统上，Metadata占用了大量时间，成为瓶颈
- 随着分布式训练节点数增多，Metadata占比越来越大，制约了吞吐量的线性增长



分布式训练，将节点数由4扩展到32

存储的选择

- 将训练数据集放入闪存甚至内存中
  - 小文件随机读写性能 内存 > 闪存 > 机械硬盘
  - 机械硬盘不太适合深度学习海量小文件训练场景
  - ImageNet场景：本地SSD vs. 机械硬盘存储 端到端训练 1.2-2倍
- 并行文件系统
  - Metadata + Object Storage
  - 聚合带宽高：横向扩展Object Storage带来聚合带宽的增加
  - 海量小文件训练场景，需考虑Metadata访问问题

转变思维模式

- 以往：重视算力、轻视IO
- IO已经是制约AI训练的至关重要的性能瓶颈
- 务必提前规划IO和存储需求

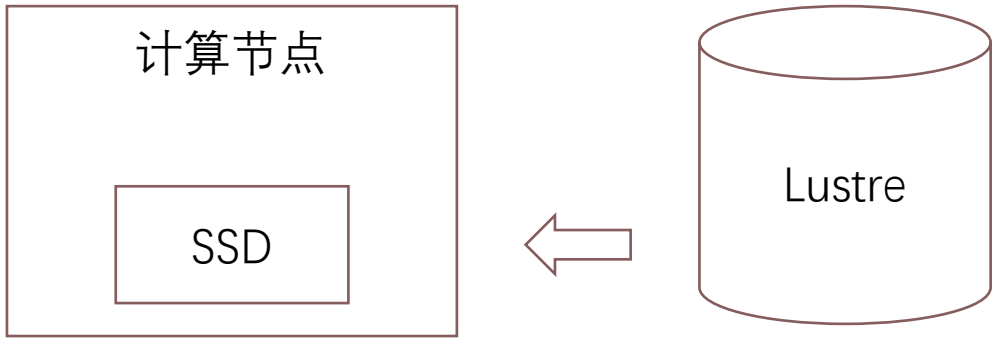
优化方向1：改变深度学习框架IO模式

- PyTorch基于Python进行随机读，有一定性能瓶颈

- 训练前预处理，将小文件打包成大文件
  - 并行文件系统适合大文件读
  - webdataset https://github.com/webdataset/webdataset

优化方向2：充分利用本地SSD

- 充分利用本地SSD
  - Lustre PCC：Persistent Client Caching
  - BeeGFS OnDemand
- AI深度学习训练的计算节点数据只读不写，无需担心一致性问题



优化方向3：内存缓存层

- 目前没有可靠易用的开源软件

- Yue Zhu, et al. Entropy-aware i/o pipelining for large-scale deep learning on hpc systems
- Abhishek Kumar, et al. Quiver: An Informed Storage Cache for Deep Learning

谢谢

https://lulaoshi.info

微信公众号：皮皮鲁的科技星球