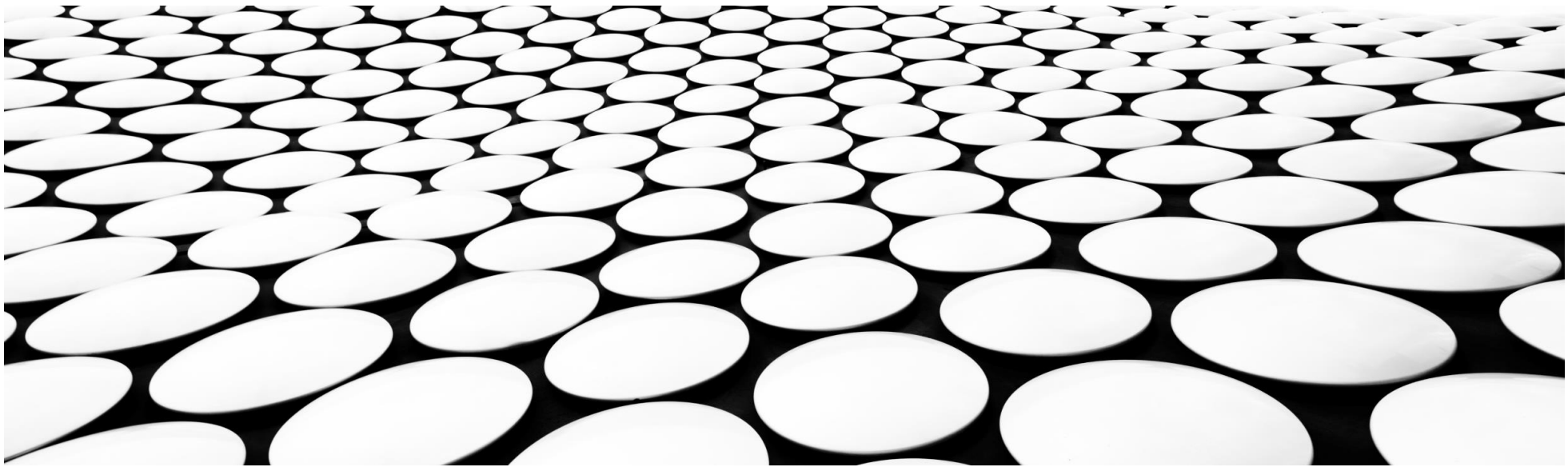
深度学习

邱怡轩



今天的主题

- 常用优化方法简介
- 练习：卷积神经网络建模与训练
- 神经网络训练实践

优化方法（改进SGD）

改进 SGD

- SGD 虽然具有较好的理论性质
- 但在实际中会遇到各种挑战，如：
  - 确定合适的学习率 η，过大导致优化不收敛，过小耗费大量迭代次数
  - 对每个参数使用了相同的学习率 η

Adagrad

- 核心思想是对每一个参数计算一个单独的 η
- 令 $g_i^{(k)}$ 为第 i 个参数在第 k 次迭代的导数
- SGD 即为 $\theta_i^{(k+1)} = \theta_i^{(k)} - \eta \cdot g_i^{(k)}$
- Adagrad 为 $\theta_i^{(k+1)} = \theta_i^{(k)} - \frac{\eta}{\sqrt{G_i^{(k)} + \varepsilon}} \cdot g_i^{(k)}$
- 其中 $G_i^{(k)} = \left(g_i^{(1)}\right)^2 + \cdots + \left(g_i^{(k)}\right)^2$

依赖之前梯度类加

整个学习率自动下降

Adagrad

- 优点:
  - 学习率自动衰减
  - 每个参数使用自适应的学习率
- 缺点:
  - 学习率衰减非常快，后期动力不足

RMSprop

- 对 Adagrad 加以改进
- 使用滑动平均计算 G_i
- $$G_i^{(k)} = \gamma G_i^{(k-1)} + (1-\gamma)\left(g_i^{(k)}\right)^2$$

越近期比重越大

- $$\theta_i^{(k+1)} = \theta_i^{(k)} - \frac{\eta}{\sqrt{G_i^{(k)} + \varepsilon}} \cdot g_i^{(k)}$$

Adadelta

- 与 RMSprop 独立地对 Adagrad 加以改进
- 加入分子的滑动平均，保持量纲一致
- $G_i^{(k)} = \gamma G_i^{(k-1)} + (1-\gamma)\left(g_i^{(k)}\right)^2$
- $\theta_i^{(k+1)} = \theta_i^{(k)} - \frac{\sqrt{D_i^{(k)}+\varepsilon}}{\sqrt{G_i^{(k)}+\varepsilon}} \cdot g_i^{(k)}$

对分子也做平均

$\Delta\theta_i^{(k)}$

- 分子为 $D_i^{(k)} = \gamma D_i^{(k-1)} + (1-\gamma)\left(\Delta\theta_i^{(k)}\right)^2$

Adam

移动方向

- 对梯度也进行滑动平均
- $m_i^{(k)} = \beta_1 m_i^{(k-1)} + (1-\beta_1) g_i^{(k)}$
- $v_i^{(k)} = \beta_2 v_i^{(k-1)} + (1-\beta_2) \left(g_i^{(k)}\right)^2$
- 修正偏差 $\hat{m}_i^{(k)} = m_i^{(k)} / (1-\beta_1^k)$，$\hat{v}_i^{(k)} = v_i^{(k)} / (1-\beta_2^k)$
- $\theta_i^{(k+1)} = \theta_i^{(k)} - \frac{\eta}{\sqrt{\hat{v}_i^{(k)}} + \varepsilon} \cdot \hat{m}_i^{(k)}$

对比

- 理论性质的研究尚在进行
- 实际应用中往往比 SGD 更快收敛
- 前期用自适应的方法，后期用 SGD

- 见 `lec8-optimizer.ipynb`

Ali Rahimi 的演讲

https://www.bilibili.com/video/BV1BW411Y78t
13:27-16:00

练习

- 第4次作业
- 见 `lec8-convnet-training.ipynb`

计算环境

计算效率

- 神经网络的计算效率由众多不同的因素决定

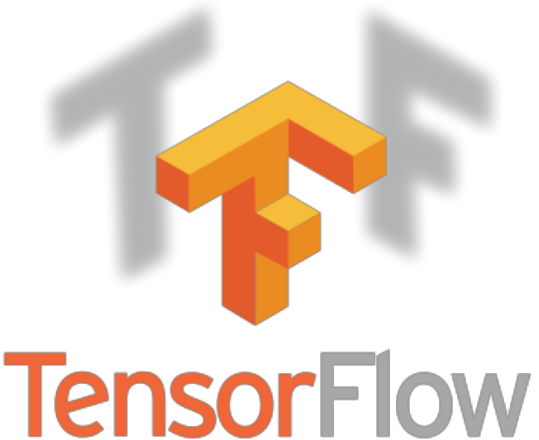
- 算法收敛速度
- 软件实现
- 硬件设备
- ……

软件

- 主流的深度学习框架都对核心运算，如矩阵乘法、卷积等进行了高度优化

Tensorflow

- https://www.tensorflow.org/
- 免费、开源
- 主要由 Google 开发维护



PyTorch

- https://pytorch.org/
- 免费、开源
- 主要由 Facebook 开发维护

MXNet

- https://mxnet.apache.org/
- 免费、开源
- 由陈天奇、李沐等人发起
- 当前主要由 Apache 软件基金会和 Amazon 团队开发维护

MindSpore

- https://www.mindspore.cn/
- 免费、开源
- 由华为开发维护

对比

- 不同的软件框架之间并无绝对的优劣之分
- 语法通常非常相似
- 个人使用可依喜好选择
- 商业应用往往考虑兼容性和维护成本等因素

硬件

- 像绝大多数应用程序一样，神经网络模型可以运行在 CPU 上
- 随着深度学习的流行，更多专用设备如 GPU、TPU 等被用来加速计算

硬件

- GPU 的优势在于可以高度并行，特别适合神经网络的结构

	GeForce RTX 4090	GeForce RTX 4080	GeForce RTX 4070 Ti
GPU Engine Specs:			
NVIDIA CUDA® Cores	16384	9728	7680
Boost Clock (GHz)	2.52	2.51	2.61
Base Clock (GHz)	2.23	2.21	2.31
Technology Support:			
Ray Tracing Cores	3rd Generation	3rd Generation	3rd Generation
Tensor Cores	4th Generation	4th Generation	4th Generation
NVIDIA Architecture	Ada Lovelace	Ada Lovelace	Ada Lovelace

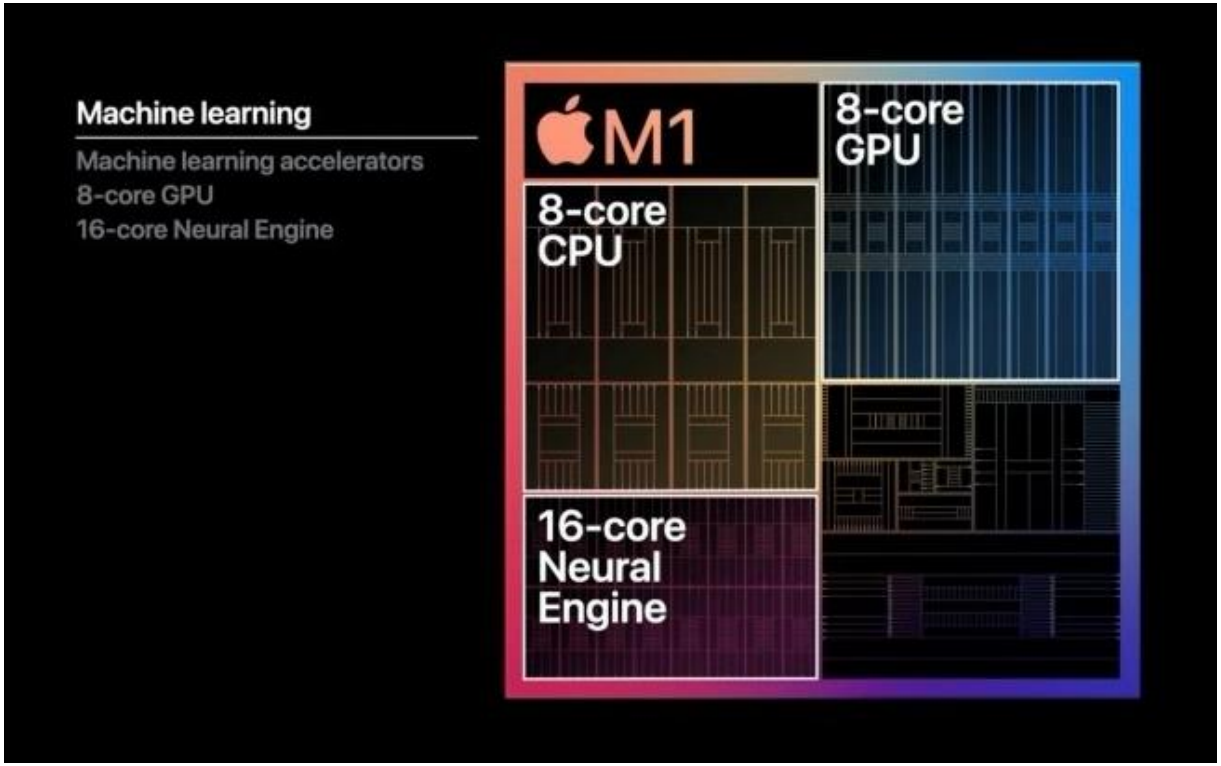
硬件

- 但并非 GPU 上运行的神经网络就一定比 CPU 上快
- 数据传输到 GPU 的计算核心需要时间
- GPU 单核的性能一般不如 CPU
- 数据量少、网络简单时并行效果不明显

- 对于较复杂的网络，GPU 的运算效率往往有很大的提升

硬件

- 一些集成芯片上还带有专门的神经网络处理器



Google Colab

- https://colab.research.google.com/
- 免费计算资源
- 可使用 GPU