

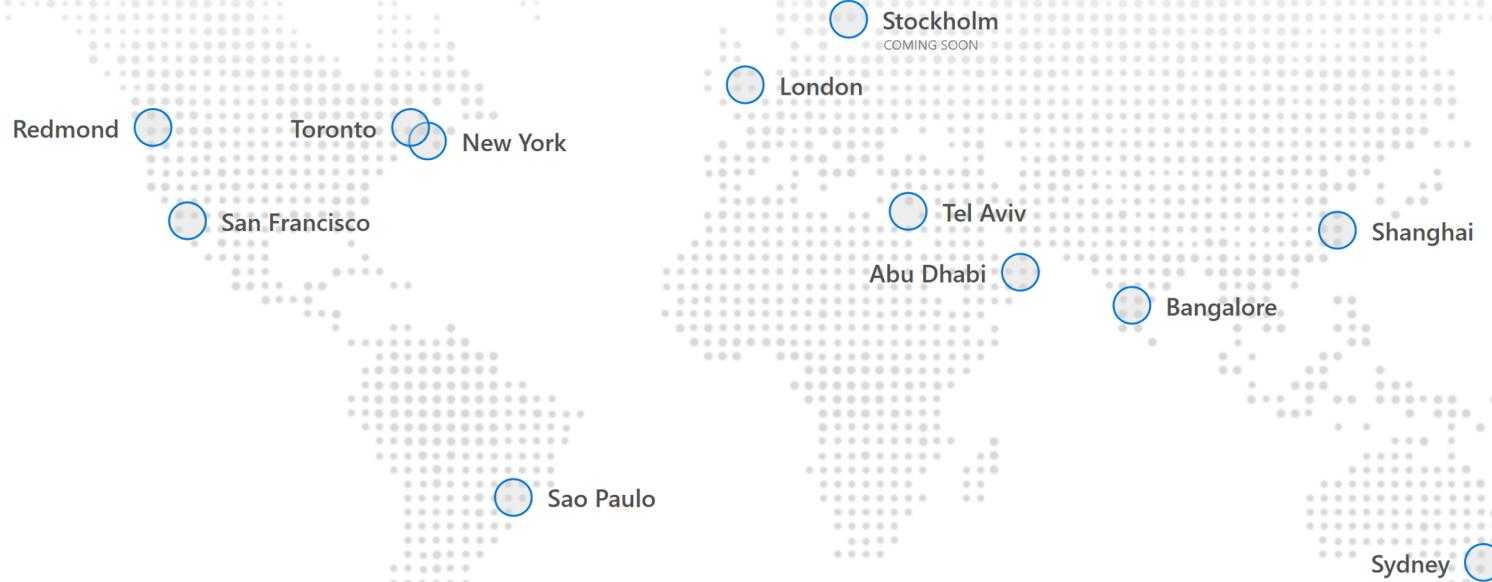
Reactor

一起学人工智能系列
- PyTorch 介绍

2021-11-03



Map



个人介绍



Kinfey Lo – (卢建晖)

Microsoft Cloud Advocate

前微软MVP、Xamarin MVP和微软RD，拥有超过10年的云原生、人工智能和移动应用经验，为教育、金融和医疗提供应用解决方案。 Microsoft Ignite , TechEd 会议讲师，Microsoft AI 黑客马拉松教练，目前在微软，为技术人员和不同行业宣讲技术和相关应用场景。



爱编程(Python , C# , TypeScript , Swift , Rust , Go)

专注于人工智能，云原生，跨平台移动开发

Github : <https://github.com/kinfey>

Email : kinfeylo@microsoft.com Blog : <https://blog.csdn.net/kinfey>

Twitter : @Ljh8304

一. PyTorch 介绍



PyTorch

<https://pytorch.org/>

开源机器学习框架，可加快从研究原型设计到生产部署的过程。

开发场景

使用 TorchScript 在 Eager 和图形模式之间无缝转换，并使用 TorchServe 加快迭代。

分布式

研究和生产中的可扩展分布式训练和性能优化由 `torch.distributed` 后端实现。

应用场景和库都齐备

丰富的工具和库生态系统扩展了 PyTorch 并支持计算机视觉、NLP 等领域的开发。

云端支持

PyTorch 在主要云平台上得到很好的支持，提供无缝开发和非常好的扩展性能。



PyTorch 安裝

<https://pytorch.org/>

PyTorch Build	Stable (1.10)	Preview (Nightly)	LTS (1.8.2)
Your OS	Linux	Mac	Windows
Package	Conda	Pip	LibTorch Source
Language	Python	C++ / Java	
Compute Platform	CUDA 10.2	CUDA 11.3	ROCM 4.2 (beta) CPU
Run this Command:	<pre>pip3 install torch==1.10.0+cpu torchvision==0.11.1+cpu torchaudio==0.10.0+cpu -f https://download.pytorch.org/wheel/cpu/torch_stable.html</pre>		

基于机器学习环境的配置

轻松搭建机器学习/深度学习开发环境

<https://blog.csdn.net/kinfey/article/details/117635067>

从开发者角度玩Windows 11

<https://blog.csdn.net/kinfey/article/details/120614677>



二. PyTorch 基础介绍



Tensor 张量介绍

张量是一种特殊的数据结构，与数组和矩阵非常相似。在 PyTorch 中，我们使用张量对模型的输入和输出以及模型的参数进行编码。

张量类似于 NumPy 的 ndarray，不同之处在于张量可以在 GPU 或其他硬件加速器上运行。事实上，张量和 NumPy 数组通常可以共享相同的底层内存，从而无需复制数据张量也针对自动微分进行了优化如果您熟悉 ndarrays，那么您将熟悉 Tensor API。



Tensor 张量介绍

```
data = [[1, 2],[3, 4]]  
x_data = torch.tensor(data)  
print(x_data)
```

张量运算

有超过100种张量相关的运算操作, 例如转置、索引、切片、数学运算、线性代数、随机采样等。更多的运算可以在这里查看。

所有这些运算都可以在GPU上运行(相对于CPU来说可以达到更高的运算速度)。



示例



三. Datasets 和 DataLoaders



Datasets 和 Dataloaders

处理数据样本的代码可能会变得混乱且难以维护；我们理想地想要我们的数据集代码与我们的模型训练代码分离，以获得更好的可读性和模块化。

PyTorch 提供了两种数据原语：`torch.utils.data.DataLoader` 和 `torch.utils.data.Dataset`

允许您使用预加载的数据集以及您自己的数据。

`Dataset` 存储样本及其相应的标签，`DataLoader` 包装了一个可迭代对象
`Dataset` 可以轻松访问样本。

PyTorch 域库提供了许多预加载的数据集（例如 FashionMNIST），这些数据集子类 `torch.utils.data.Dataset` 并实现特定于特定数据的功能。

它们可用于对您的模型进行原型设计和基准测试。你可以找到他们
此处：

图像数据集 <https://pytorch.org/vision/stable/datasets.html>

文本数据集 <https://pytorch.org/text/stable/datasets.html>

音频数据集 <https://pytorch.org/audio/stable/datasets.html>



示例



四 . Transforms



Transforms 数值变换

数据并不总是以其所需的最终处理形式出现训练机器学习算法。 我们使用 **transforms*** 来执行一些处理数据并使其适合训练。

所有 TorchVision 数据集都有两个参数（`transform` 以修改特征和 `target_transform` 修改标签）接受包含转换逻辑的可调用对象。<https://pytorch.org/vision/stable/transforms.html> 模块提供几种常用的开箱即用转换。



示例



五. 神经网络创建



构建神经网络

神经网络由对数据执行操作的层/模块组成。`torch.nn` 命名空间提供了您需要的所有构建块建立自己的神经网络。

PyTorch 中的每个模块都是 `nn.Module` 的子类。

神经网络是一个模块，由其他模块（层）组成。

这种嵌套结构允许轻松构建和管理复杂的架构。



示例



六. autograd



使用“torch.autograd”自动微分

在训练神经网络时，最常用的算法是反向传播。

在这个算法中，参数（模型权重）是根据损失函数的梯度进行调整到给定的参数。

为了计算这些梯度，PyTorch 有一个内置的微分引擎称为“`torch.autograd`”。它支持任何梯度的自动计算

计算图。

考虑最简单的一层神经网络，输入为“`x`”，参数 `w` 和 `b`，以及一些损失函数。



示例



七. 优化模型参数



优化模型参数

现在我们有了模型和数据，是时候通过优化数据上的参数来训练、验证和测试我们的模型了。训练模型是一个迭代过程；在每次迭代中*epoch*，模型对输出进行猜测，计算其猜测中的误差*loss*，收集误差相对于其参数的导数（如我们在模块中看到的），并使用梯度下降**优化**这些参数。



示例



八. 模型应用



示例



九. 实例



示例



Microsoft Learn上的人工智能

<https://docs.microsoft.com/zh-cn/learn>



PyTorch 基础知识

<https://aka.ms/PytorchStudy>



TensorFlow 基础知识

<https://aka.ms/TensorflowLearn>

Q&A





Reactor

Thank You!