Building Your First PyTorch Solution

INSTALLING PYTORCH ON A LOCAL MACHINE

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Exploring PyTorch installation options

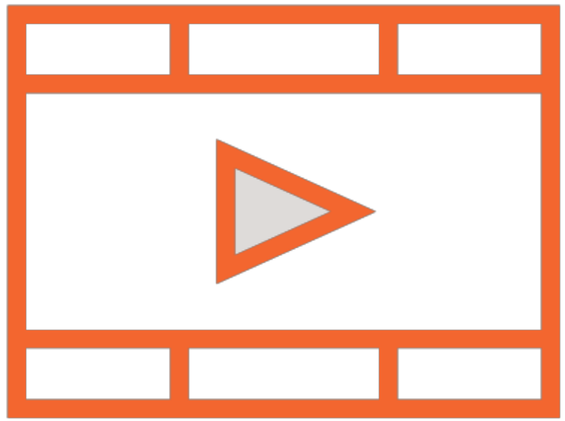
GPU support in PyTorch using CUDA

Installing PyTorch on CPU-only machines

Installing PyTorch on GPU-enabled machines

Prerequisites and Course Outline

Prerequisites

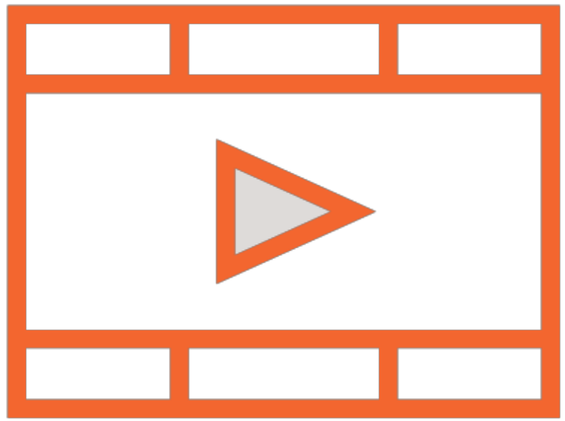


Programming in Python

Familiarity with basic machine learning concepts

Familiarity with basic building blocks of PyTorch

Prerequisites Courses



Building Your First scikit-learn Solution

Foundations of PyTorch

Course Outline

Installing PyTorch on CPU and GPU enabled machines

Linear regression using a single neuron

Building a regression model

Building a classification model

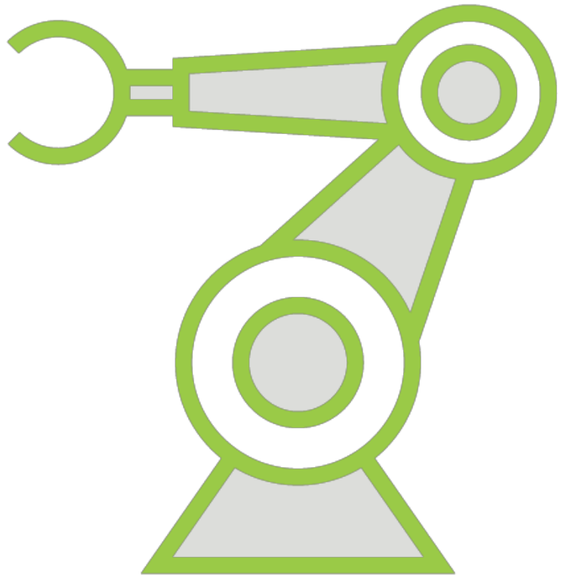
PyTorch, CUDA, and GPUs

GPU (Graphics Processing Unit)

Specialized chips with highly parallel architecture that makes them an order of magnitude faster than CPUs for some deep learning applications

PyTorch Tensors have been architected to make optimal use of GPUs for massively parallel computations

GPUs for ML

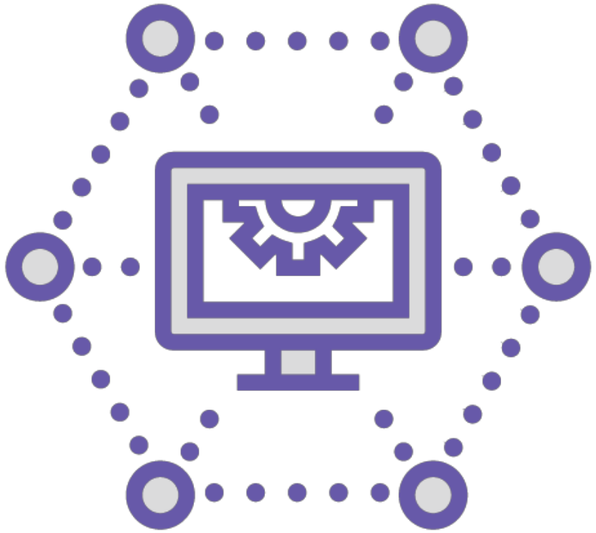


Usage of GPUs has gone far beyond video/graphics processing

Widely used in Big Data and Machine Learning applications

Speedup of 10-50X where parallelization yields big wins

CUDA



Nvidia is a major maker of GPUs

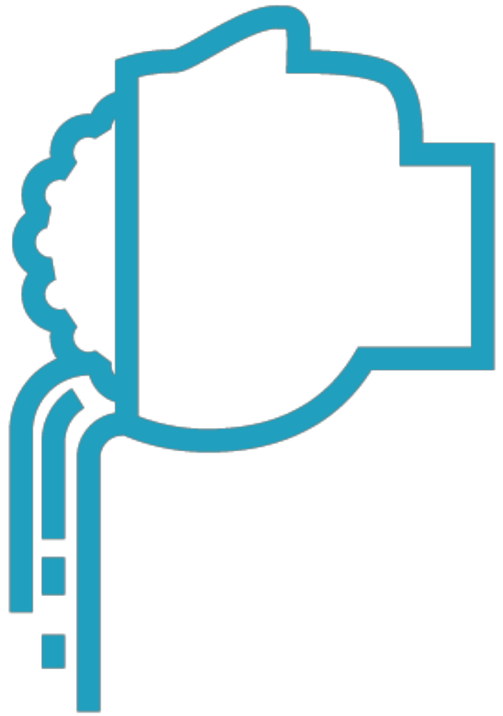
Devised CUDA, a parallel computing platform and API

A standard for general purpose (non-graphics) users of GPUs

Initially acronym for “Compute Unified Device Architecture”

Now a standalone term, not an acronym

CUDA and PyTorch

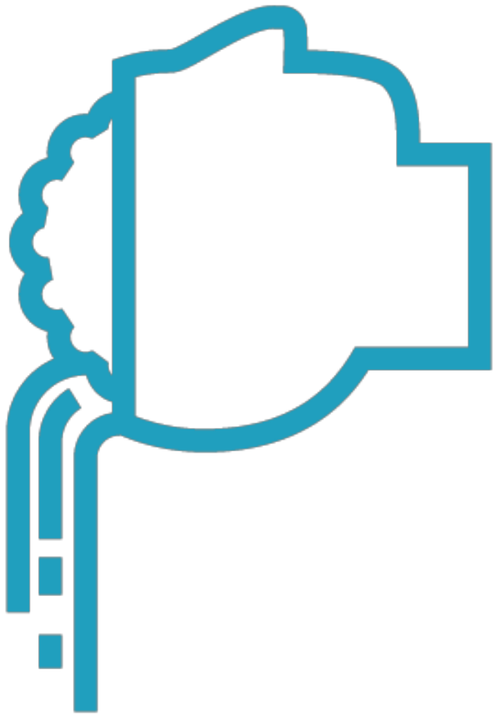


Developers can write CUDA-compliant code

Code must be understood by CUDA-aware framework (e.g. PyTorch)

If CUDA-enabled GPUs are available, speedup will automatically occur

CUDA and PyTorch



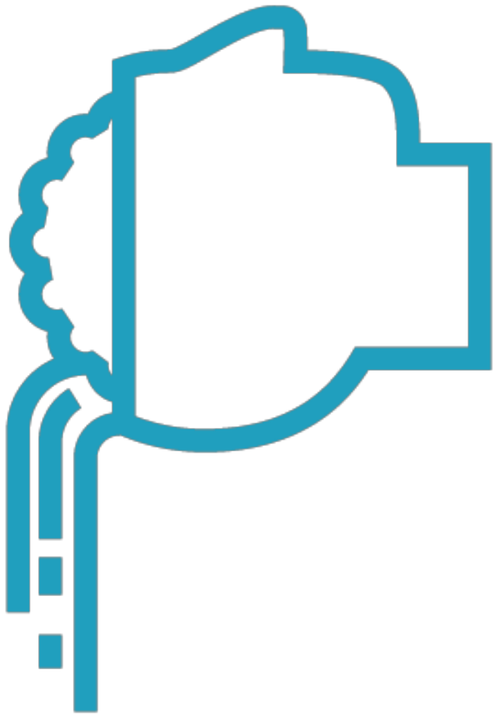
`torch.cuda` **for CUDA operations**

Special tensor types for CUDA e.g.
`torch.cuda.FloatTensor`

`torch.cuda.device` **to select GPU**

Tracks currently selected GPU and creates tensors on it

CUDA and PyTorch



Cross-GPU operations not allowed by default

- Exceptions: copy_(), to(), cuda(), and other methods with copy semantics

Ops on tensors across devices will cause errors

- Can mitigate this by enabling peer-to-peer memory access

Asynchronous Execution

GPU operations are asynchronous by default

Enqueued to particular device, executed later

Allows execution of many more computations in parallel

Asynchronous Execution

Asynchronous execution typically invisible to user

- FIFO order of queuing
- Automatic synchronization by PyTorch between devices

CUDA Streams

Linear sequence of operations for execution on a single device

By default, each device has a default stream

Operations within a stream are serialized by PyTorch (order is deterministic)

Order of execution across streams is not deterministic

Device-agnostic Code

Device-agnostic code explicitly handles GPU and CPU cases

Common pattern is to use argparse to read user arguments

Code can then be invoked with runtime flags to enable or disable CUDA

Demo

Exploring PyTorch installation options

Demo

Setting up a virtual machine instance on the Google Cloud

Demo

Installing PyTorch on Linux using Conda

Demo

Installing PyTorch on Linux using pip

Demo

Adding GPU support and installing CUDA

Demo

Installing PyTorch on a Linux VM with GPU and CUDA support using Conda

Demo

Installing PyTorch on a Linux VM with GPU and CUDA support using pip

Summary

Exploring PyTorch installation options

GPU support in PyTorch using CUDA

Installing PyTorch on CPU-only machines

Installing PyTorch on GPU-enabled machines