Introduction to PyTorch

Ahmed Hosny Abdel-Gawad

Senior AI/CV Engineer

What is PyTorch?

What is PyTorch?

PyTorch is a Python package that provides two high-level features:

- **Tensor** computation (like NumPy) with strong GPU acceleration
- Deep neural networks built on a **tape-based** autograd system

Usually, **PyTorch** is used either as:

- A replacement for NumPy to use the power of GPUs.
- A deep learning research platform that provides maximum flexibility and speed.

PyTorch
Tensor

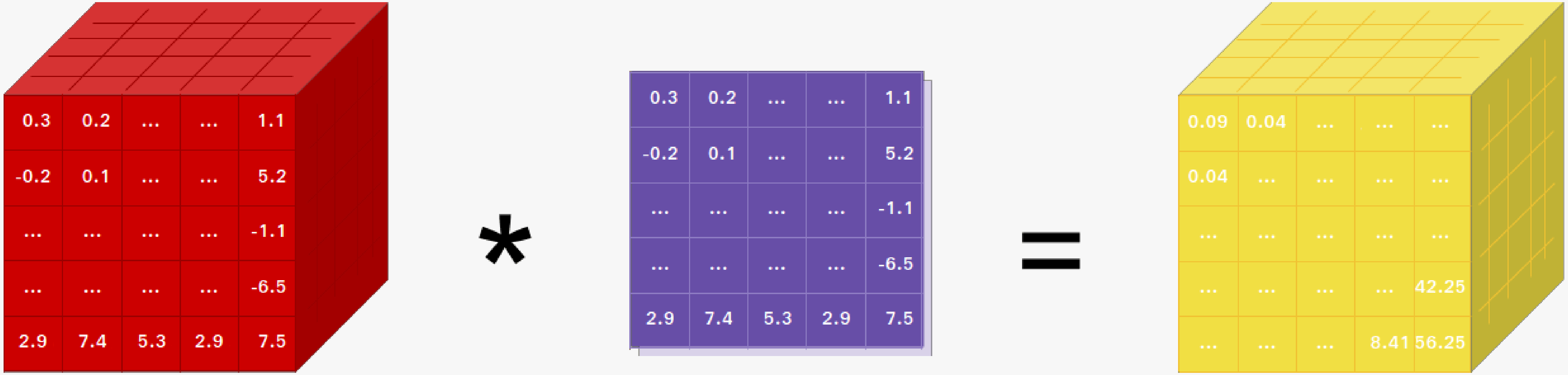
What is PyTorch?

At a granular level, **PyTorch** is a library that consists of the following components:

Component	Description
torch	A Tensor library like NumPy, with strong GPU support
torch.autograd	A tape-based automatic differentiation library that supports all differentiable Tensor operations in torch
torch.jit	A compilation stack (TorchScript) to create serializable and optimizable models from PyTorch code
torch.nn	A neural networks library deeply integrated with autograd designed for maximum flexibility
torch.multiprocessing	Python multiprocessing, but with magical memory sharing of torch Tensors across processes. Useful for data loading and Hogwild training
torch.utils	DataLoader and other utility functions for convenience

A GPU-Ready Tensor Library

If you use NumPy, then you have used Tensors (a.k.a. ndarray).



0.3	0.2	...	...	1.1
-0.2	0.1	...	...	5.2
...	...	...	...	-1.1
...	...	...	...	-6.5
2.9	7.4	5.3	2.9	7.5

*

0.3	0.2	...	...	1.1
-0.2	0.1	...	...	5.2
...	...	...	...	-1.1
...	...	...	...	-6.5
2.9	7.4	5.3	2.9	7.5

=

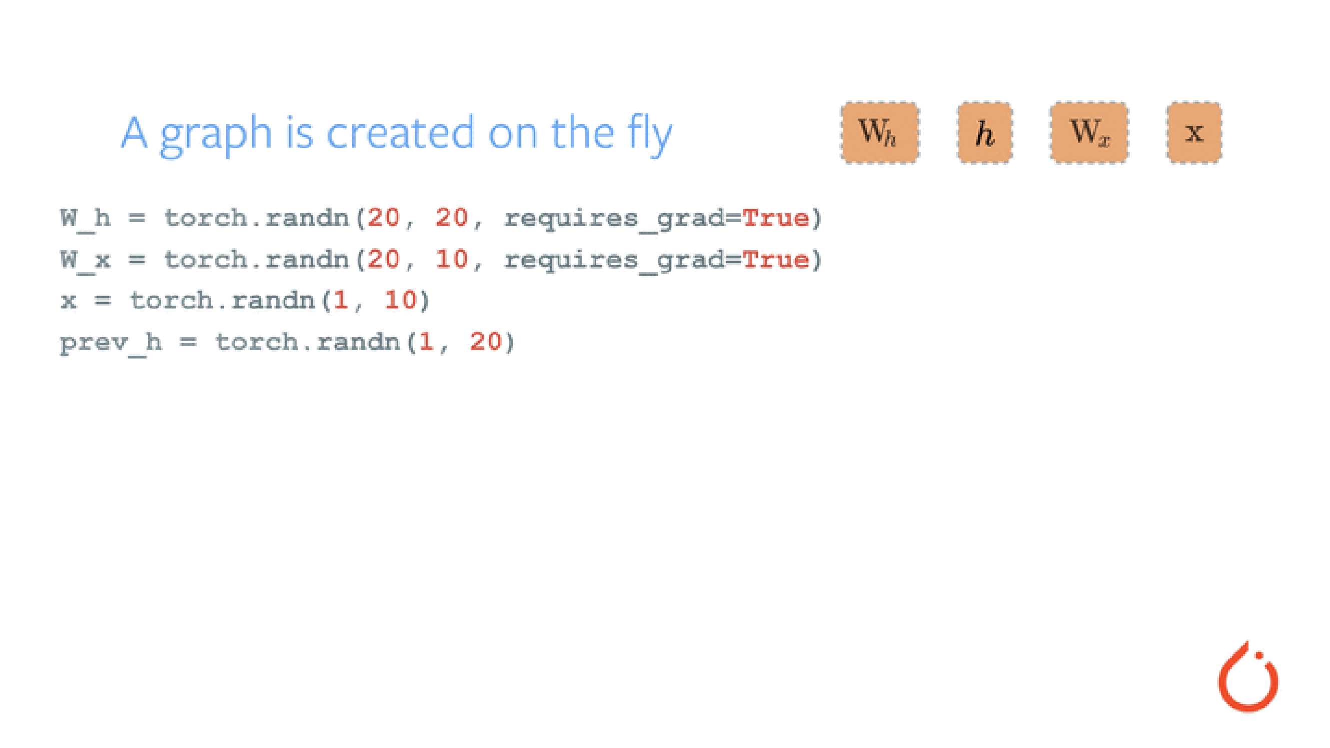
0.09	0.04	...	...	...
0.04	...	...	...	...
...	...	...	...	...
...	...	...	...	42.25
...	...	...	8.41	56.25

PyTorch provides Tensors that can live either on the **CPU** or the **GPU** and accelerates the computation by a huge amount.

PyTorch provide a wide variety of tensor routines to accelerate and fit the scientific computation needs such as slicing, indexing, math operations, linear algebra, reductions.

Dynamic Neural Networks: Tape-Based Autograd

PyTorch has a unique way of building neural networks: using and replaying a tape recorder.



A graph is created on the fly

```
W_h = torch.randn(20, 20, requires_grad=True)
W_x = torch.randn(20, 10, requires_grad=True)
x = torch.randn(1, 10)
prev_h = torch.randn(1, 20)
```

More...

PyTorch uses a technique called reverse-mode auto-differentiation, which allows to change the way the network behaves arbitrarily with zero lag or overhead.

Pytorch Fundamentals: Tensors

First we are going to deal with the basic building block of machine learning and deep learning, the tensor.

Topic	Contents
Introduction to tensors	Tensors are the basic building block of all of machine learning and deep learning.
Creating tensors	Tensors can represent almost any kind of data (images, words, tables of numbers).
Getting information from tensors	If you can put information into a tensor, you'll want to get it out too.
Manipulating tensors	Machine learning algorithms (like neural networks) involve manipulating tensors in many different ways such as adding, multiplying, combining.
Dealing with tensor shapes	One of the most common issues in machine learning is dealing with shape mismatches (trying to mixed wrong shaped tensors with other tensors).

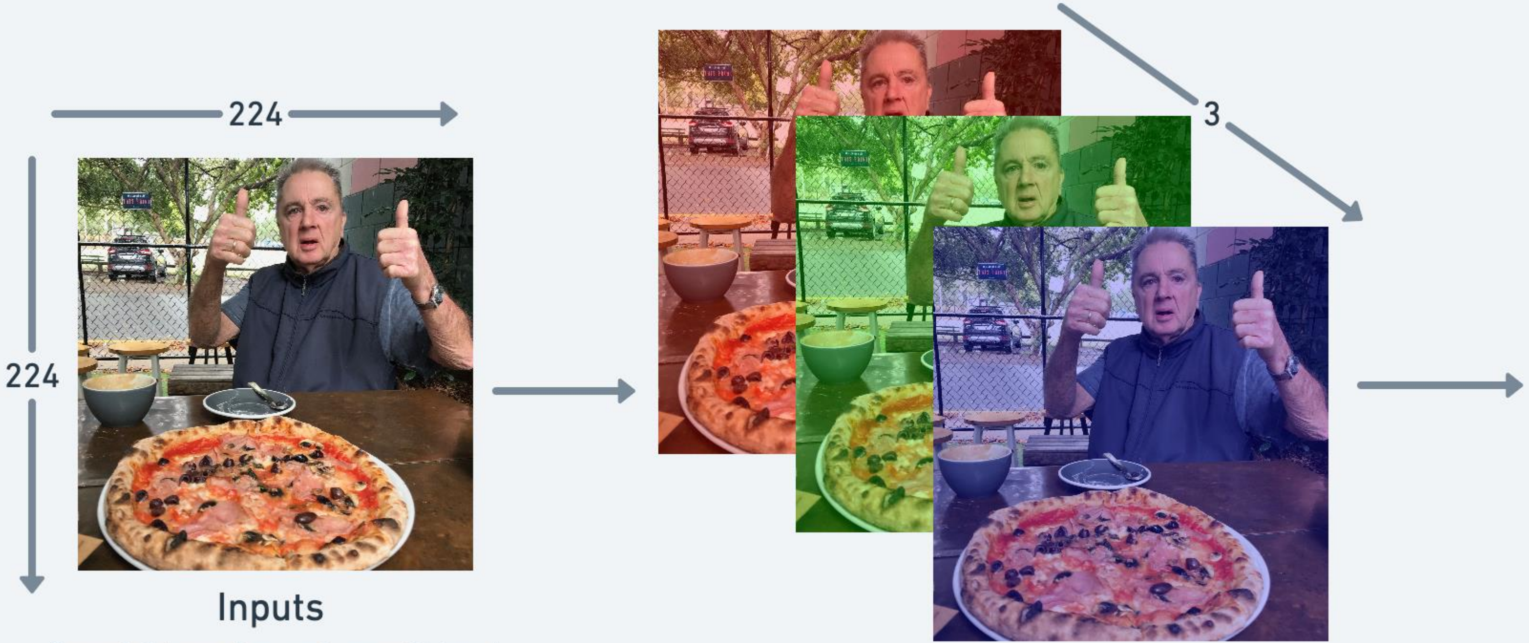
Pytorch Fundamentals: Tensors

First we are going to deal with the basic building block of machine learning and deep learning, the tensor.

Topic	Contents
Indexing on tensors	If you've indexed on a Python list or NumPy array, it's very similar with tensors, except they can have far more dimensions.
Mixing PyTorch tensors and NumPy	PyTorch plays with tensors (torch.Tensor), NumPy likes arrays (np.ndarray) sometimes you'll want to mix and match these.
Running tensors on GPU	GPUs (Graphics Processing Units) make your code faster, PyTorch makes it easy to run your code on GPUs.

What are Tensors?

Tensors are the fundamental building block of machine learning. Their job is to represent data in a numerical way.



Inputs
(could be almost anything)

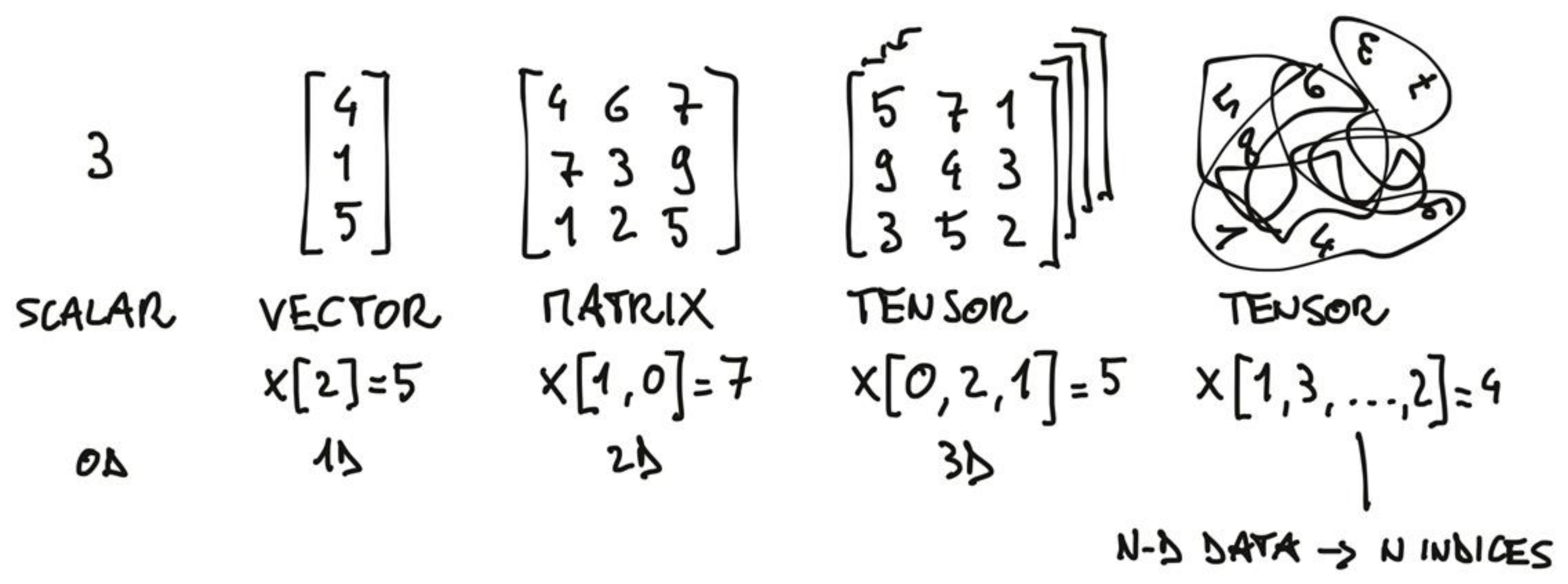
Processed inputs
(these get represented as a tensor, the processing will depend on the input)

[[[-0.01157, 0.02485, 0.02878...
-0.01271, 0.03971, 0.08827...
0.02680, 0.05589, -0.01068...
-0.00597, 0.00639, -0.01819...
...]]]

Tensor representation of image
with shape [3, 224, 224]

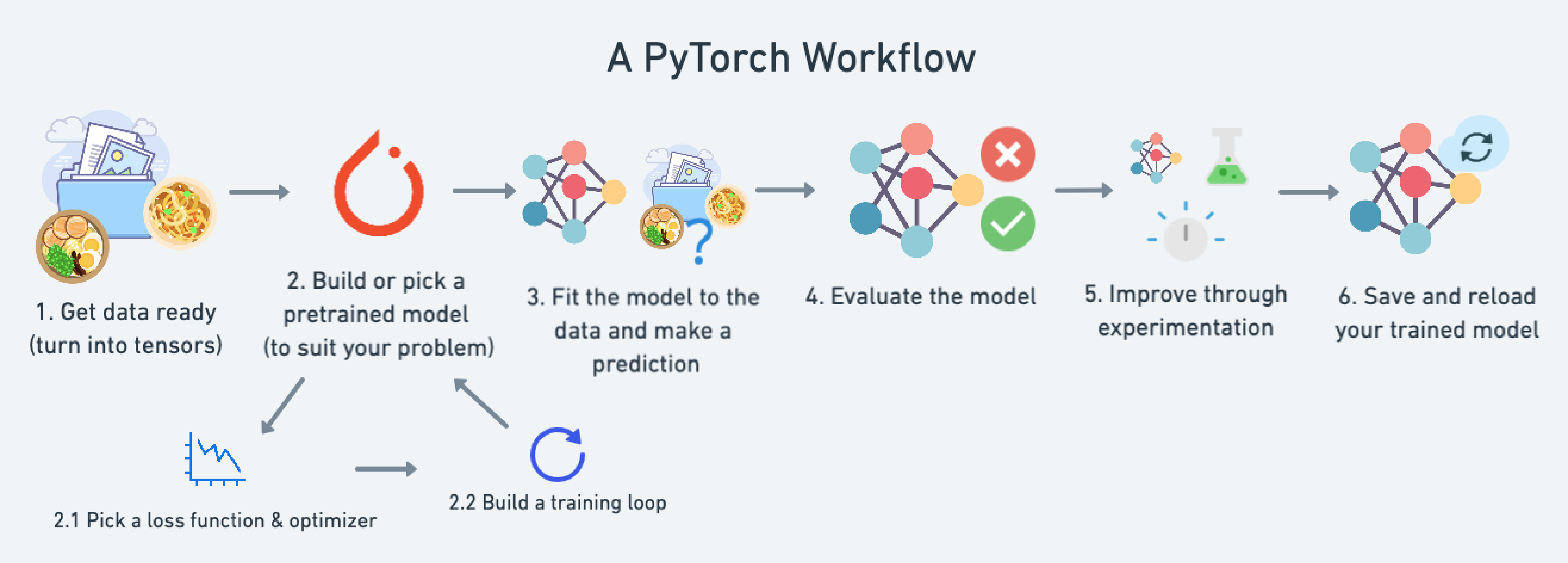
What are Tensors?

Tensors are the fundamental building block of machine learning. Their job is to represent data in a numerical way.



PyTorch Workflow

PyTorch Workflow


A PyTorch Workflow
1. Get data ready (turn into tensors)
2. Build or pick a pretrained model (to suit your problem)
3. Fit the model to the data and make a prediction
4. Evaluate the model
5. Improve through experimentation
6. Save and reload your trained model
2.1 Pick a loss function & optimizer
2.2 Build a training loop

For now, we'll use this workflow to predict a simple straight line but the workflow steps can be repeated and changed depending on the problem you're working on.

PyTorch Workflow

Topic	Contents
Getting data ready	Data can be almost anything but to get started we're going to create a simple straight line
Building a model	Here we'll create a model to learn patterns in the data, we'll also choose a loss function, optimizer and build a training loop.
Fitting the model to data (training)	We've got data and a model, now let's let the model (try to) find patterns in the (**training**) data.
Making predictions and evaluating a model (inference)	Our model's found patterns in the data, let's compare its findings to the actual (**testing**) data.
Saving and loading a model	You may want to use your model elsewhere, or come back to it later, here we'll cover that.

Thank You!