CEFET/RJ

Introduction to PyTorch

Machine Learning Course - PPCIC

Rafaela Castro

October/2020

About me

Master’s degree at CEFET-RJ | Data Architect at TRF2

Admission: May 2018 Conclusion: July 2020

Advisor: Eduardo Bezerra

Project: Apply convolutional neural networks to model spatiotemporal data

✉ rafaela.nascimento@eic.cefet-rj.br

in /in/rafaela00castro

/rafaela00castro

What is PyTorch?

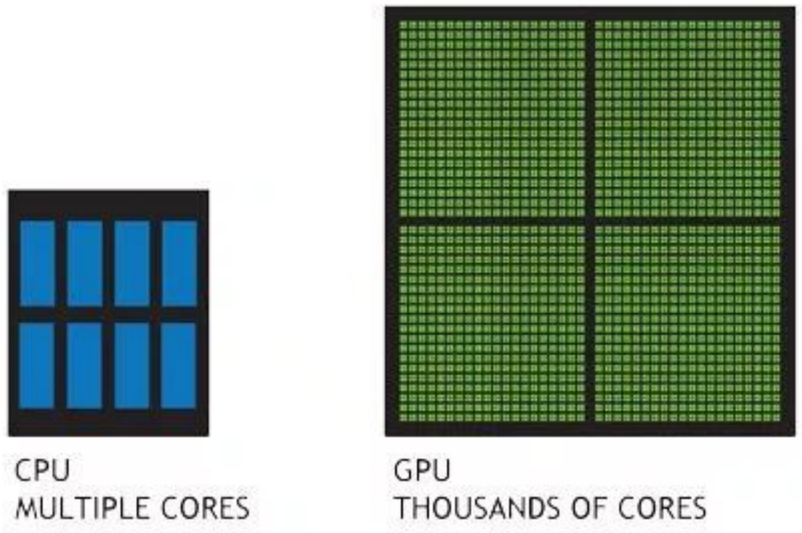
PyTorch is a **Python based** framework that is **similar to NumPy**, but with the added power of **GPUs**

https://www.analyticsvidhya.com/blog/2019/01/guide-pytorch-neural-networks-case-studies/

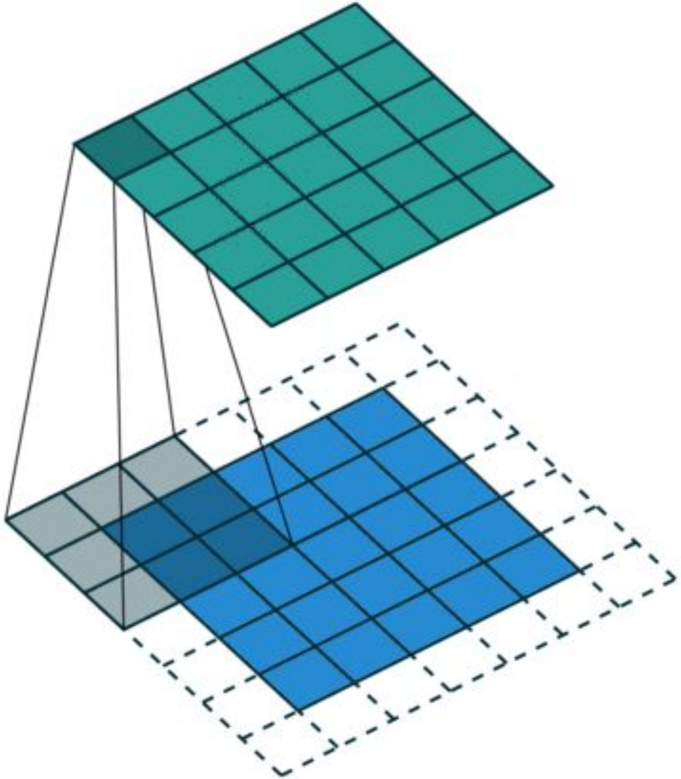
What is GPU?

Graphics Processing Unit (GPU)

- Originally used for 3D game rendering
- Designed for handling multiple tasks simultaneously
- Faster than the CPU when dealing with a huge amount of information



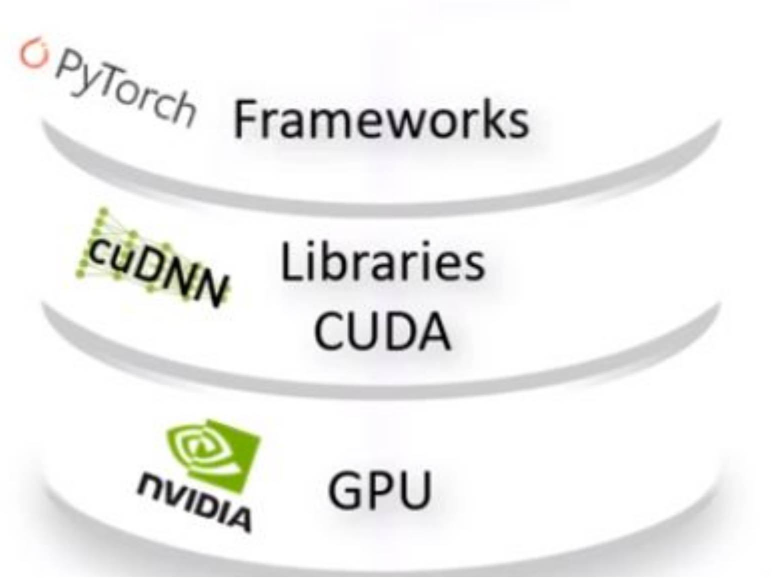
Example of task to run on GPU - Convolution operation



http://deeplizard.com/learn/video/6stDhEA0wFQ

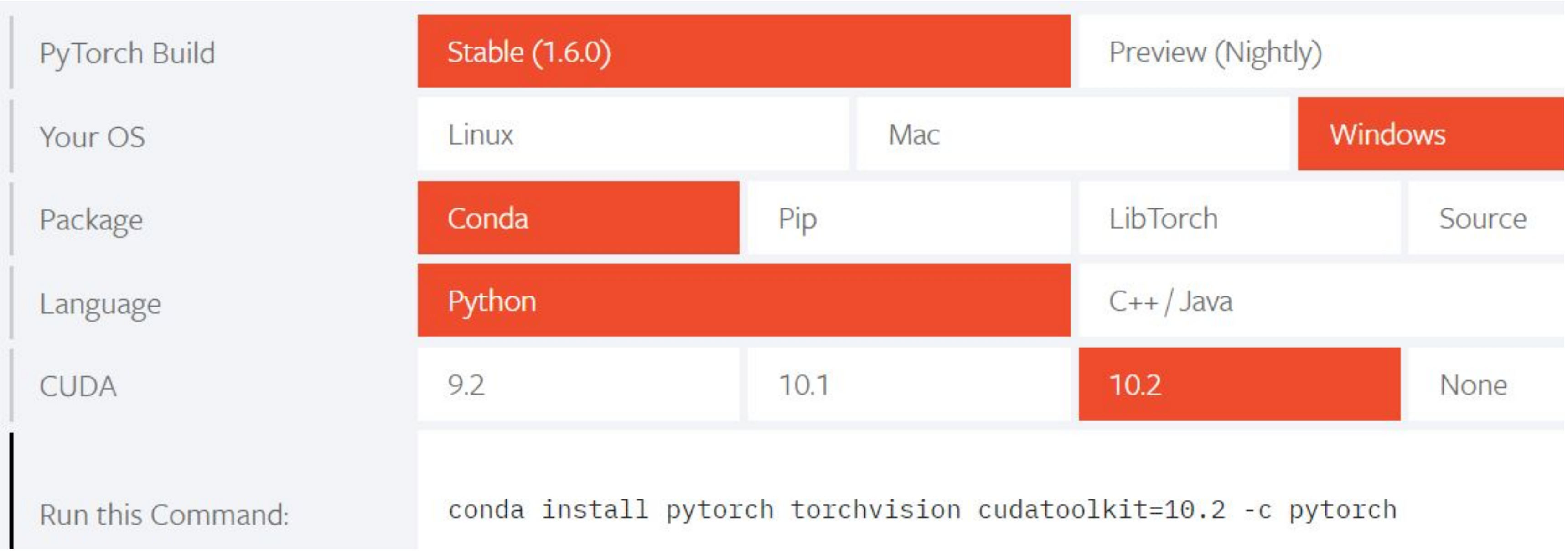
CUDA

- Software that pairs with GPU hardware making it easier for developers to build software that accelerates computations
- Nvidia hardware (GPU) and software (CUDA)



http://deeplizard.com/learn/video/6stDhEA0wFQ

PyTorch installation



PyTorch Build	[✓ Stable (1.6.0)]	Preview (Nightly)		
Your OS	Linux	Mac	[✓ Windows]	
Package	[✓ Conda]	Pip	LibTorch	Source
Language	[✓ Python]	C++ / Java		
CUDA	9.2	10.1	[✓ 10.2]	None

Run this Command:

```
conda install pytorch torchvision cudatoolkit=10.2 -c pytorch
```

https://pytorch.org/get-started/locally/

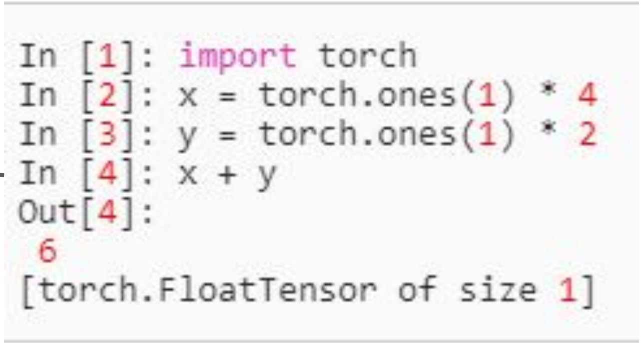
An overview of PyTorch

- Developed and maintained by Facebook (@soumithchintala)
- Released in 2016
- Uses an imperative programming (eager mode execution)
- Dynamic computation graphs
- “Many researchers use it because the API is **intuitive** and **easier** to learn, and get into **experimentation quickly**.”

https://blog.exxactcorp.com/pytorch-vs-tensorflow-in-2020-what-you-should-know-about-these-frameworks/

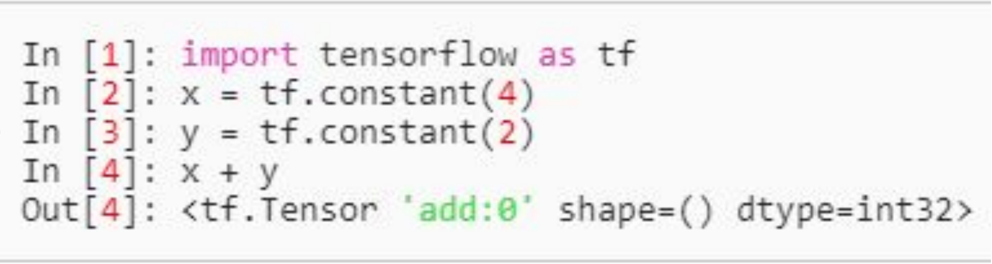
PyTorch 1.0 vs TensorFlow 1.x (old days)

PyTorch



```
In [1]: import torch
In [2]: x = torch.ones(1) * 4
In [3]: y = torch.ones(1) * 2
In [4]: x + y
Out[4]:
 6
[torch.FloatTensor of size 1]
```

TensorFlow

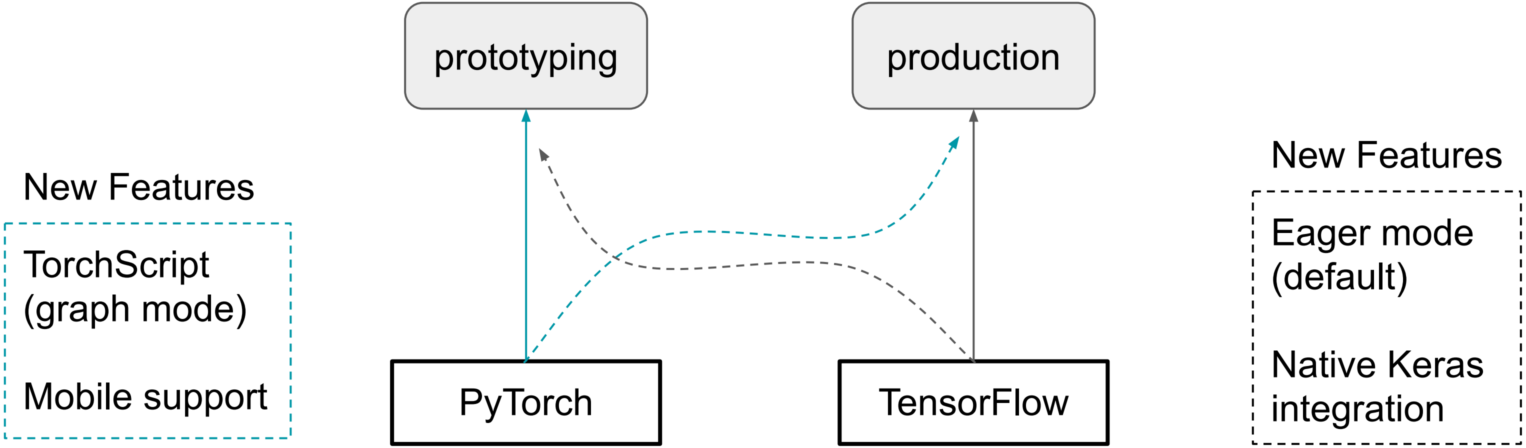


```
In [1]: import tensorflow as tf
In [2]: x = tf.constant(4)
In [3]: y = tf.constant(2)
In [4]: x + y
Out[4]: <tf.Tensor 'add:0' shape=() dtype=int32>
```

http://www.goldsborough.me/ml/ai/python/2018/02/04/20-17-20-a_promenade_of_pytorch/

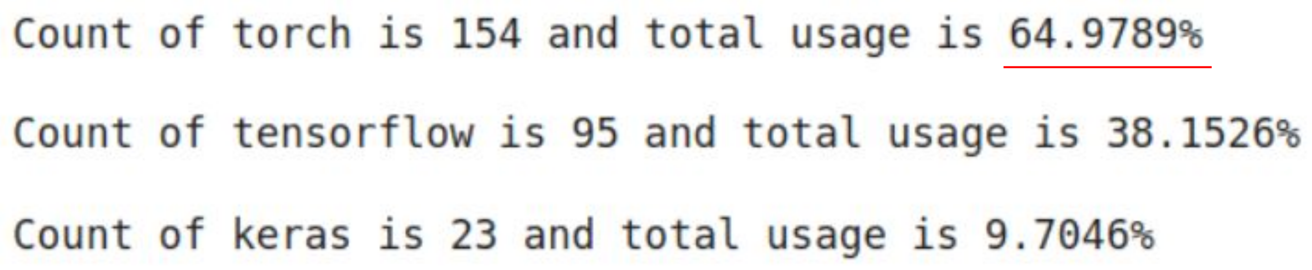
PyTorch 1.4+ vs TensorFlow 2.0+

- Each one moves trying to address their respective weaknesses.



Adoption in academic research

Main tools for deep learning at ICLR 2020 (International Conference on Learning Representations).



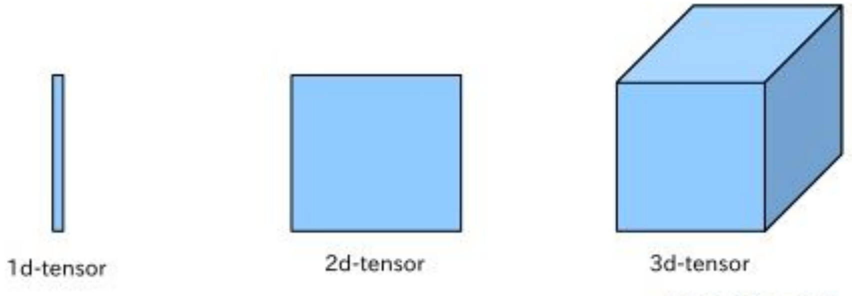
```
Count of torch is 154 and total usage is 64.9789%

Count of tensorflow is 95 and total usage is 38.1526%

Count of keras is 23 and total usage is 9.7046%
```

https://www.analyticsvidhya.com/blog/2020/05/key-takeaways-iclr-2020/

Key concepts of PyTorch

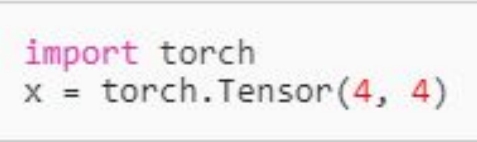
Tensor

- Data structure used by neural networks
- Similar to NumPy’s ndarray
- Generalization of scalars, vectors and matrices
    - A scalar is a 0 dimensional tensor
    - A vector is a 1 dimensional tensor
    - A matrix is a 2 dimensional tensor
    - A nd-array is an n dimensional tensor

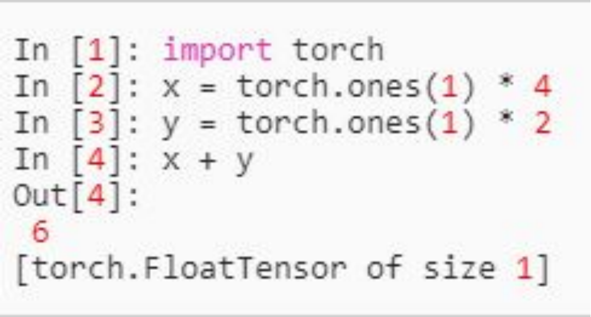


- PyTorch Tensors can utilize GPUs to accelerate computations

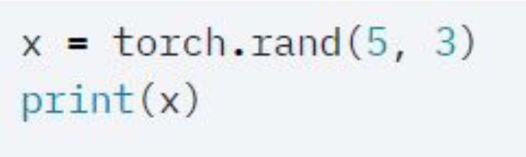
Tensor (2)



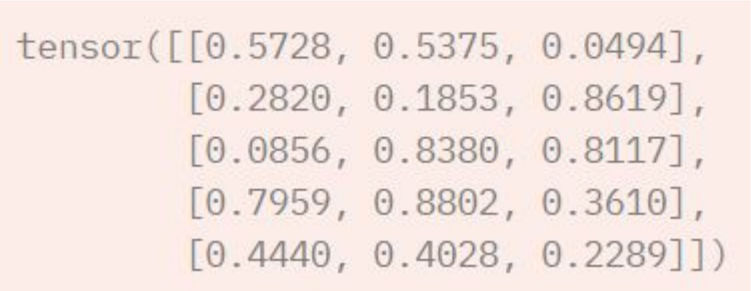
```
import torch
x = torch.Tensor(4, 4)
```



```
In [1]: import torch
In [2]: x = torch.ones(1) * 4
In [3]: y = torch.ones(1) * 2
In [4]: x + y
Out[4]:
 6
[torch.FloatTensor of size 1]
```



```
x = torch.rand(5, 3)
print(x)
```

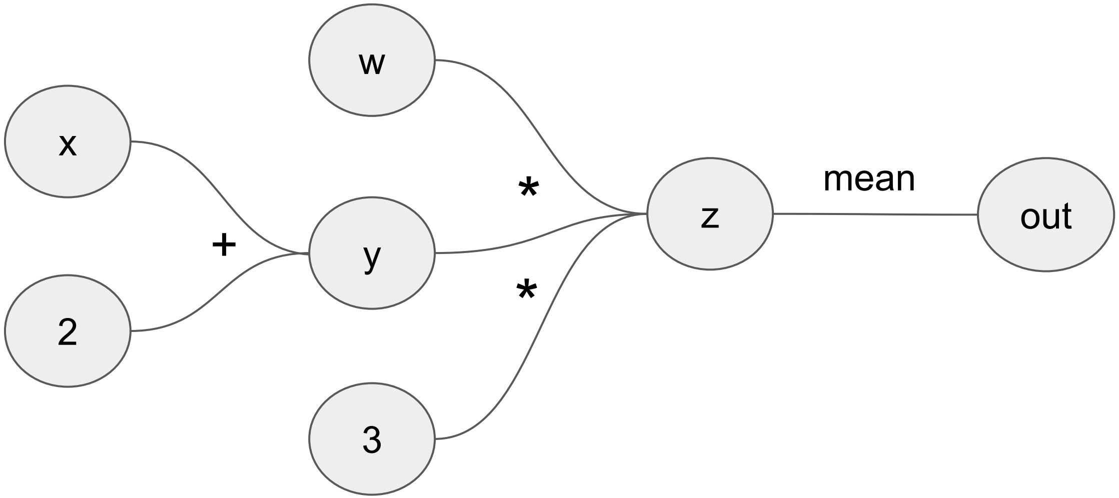


```
tensor([[0.5728, 0.5375, 0.0494],
        [0.2820, 0.1853, 0.8619],
        [0.0856, 0.8380, 0.8117],
        [0.7959, 0.8802, 0.3610],
        [0.4440, 0.4028, 0.2289]])
```

https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html#sphx-glr-beginner-blitz-tensor-tutorial-py

Computational graphs

- The graph is created on the fly!

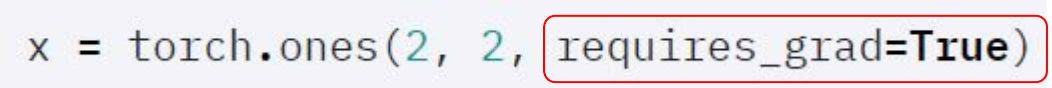


Autograd package

- Automates the computation of backward pass in neural networks
- How is it works?
  a. Track all operation using `.requires_grad` as `True`
  b. Operations on Tensors (forward pass) will define a dynamic computational graph
  c. The `.backward()` function automatically calculates all gradients

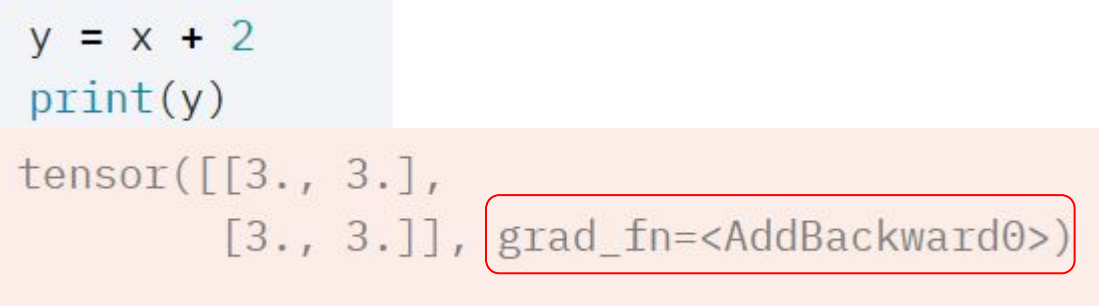
Autograd package (2)

a



```python
x = torch.ones(2, 2, requires_grad=True)
```

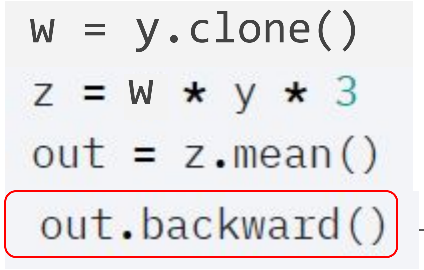
b



```python
y = x + 2
print(y)
```

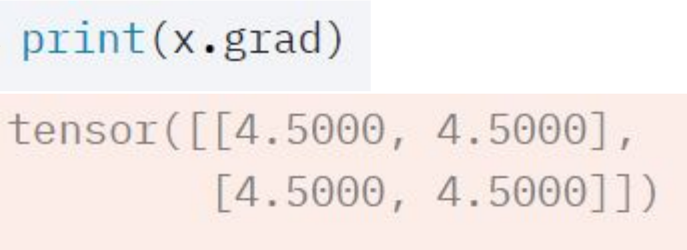
```
tensor([[3., 3.],
        [3., 3.]], grad_fn=<AddBackward0>)
```

c



```python
w = y.clone()
z = w * y * 3
out = z.mean()
out.backward()
```

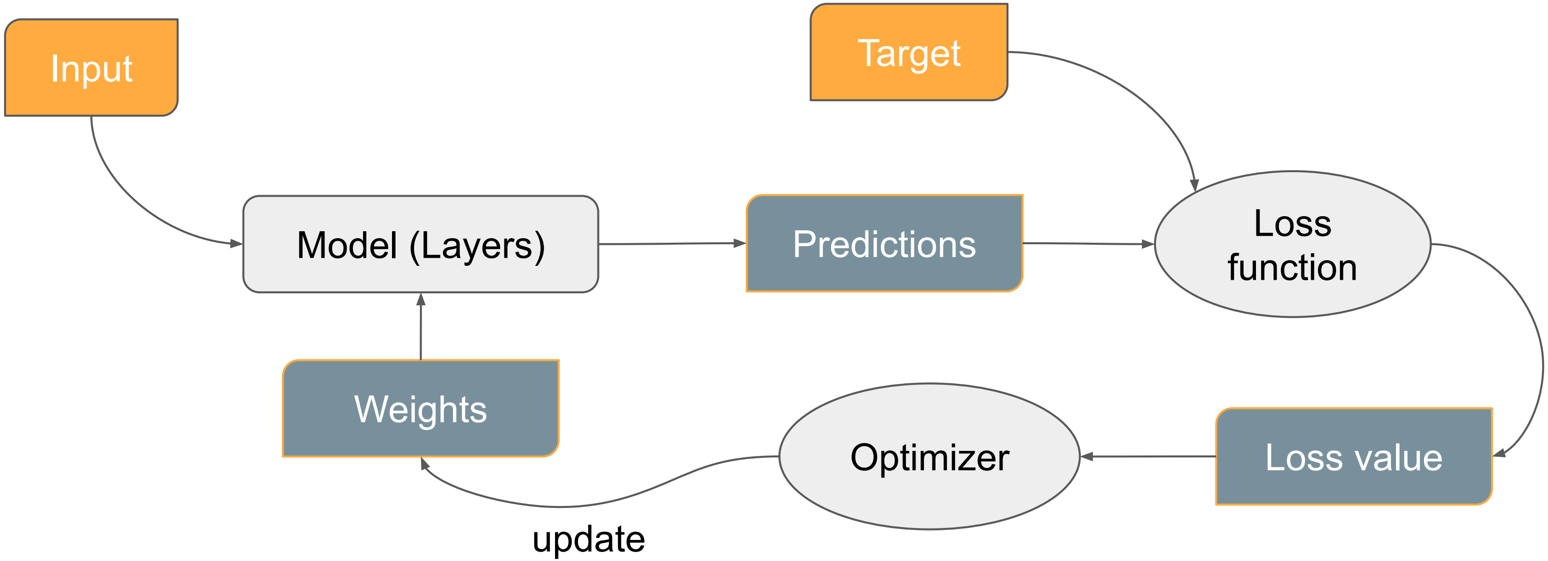
Print gradients d(out)/dx



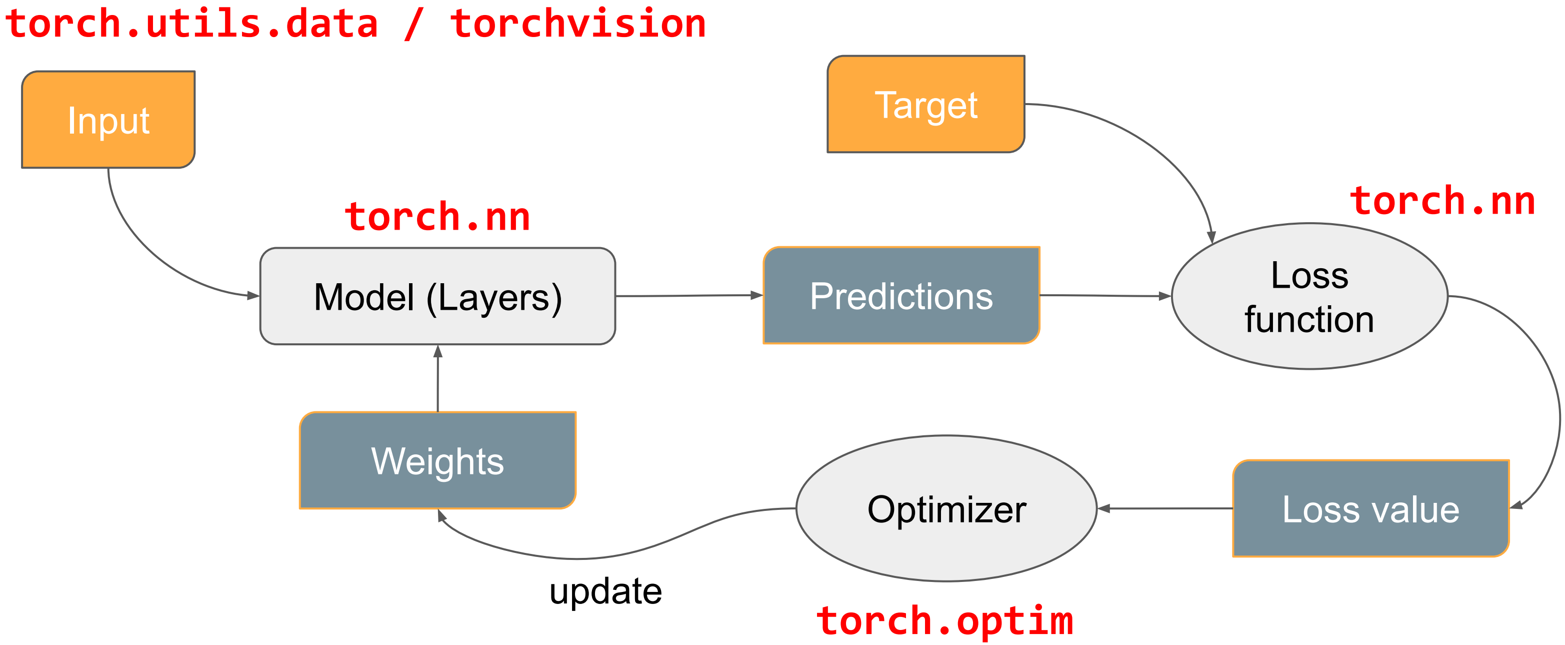
```python
print(x.grad)
```

```
tensor([[4.5000, 4.5000],
        [4.5000, 4.5000]])
```

Supervised learning



PyTorch packages



torch.utils.data / torchvision

Input

Target

torch.nn

Model (Layers)

Predictions

torch.nn

Loss function

Weights

Optimizer

Loss value

update

torch.optim

NN package

- Defines a set of modules, which are equivalent to neural network **layers**
  - `torch.nn.Module()` = base class for all neural network modules
  - `torch.nn.Linear()` = represents a linear (dense/fully-connected) layer
  - `torch.nn.Conv2d()` = applies 2D convolution

https://pytorch.org/docs/stable/nn.html

NN package (2)

- Defines a set of **activation functions** used when training neural networks
  - `torch.nn.ReLU()` = Rectified Linear Unit activation function
  - `torch.nn.Sigmoid()`
  - `torch.nn.Softmax()`

https://pytorch.org/docs/stable/nn.html#non-linear-activations-weighted-sum-nonlinearity

NN package (3)

- Defines a set of **loss functions** used when training neural networks
  - `torch.nn.MSELoss()` = a Mean Squared Error loss
  - `torch.nn.L1Loss()` = measures the mean absolute error (MAE)
  - `torch.nn.CrossEntropyLoss()` = useful when training a classification problem with *C* classes

https://pytorch.org/docs/stable/nn.html#loss-functions

NN package (4)

- Defines a set of **normalization** layers
  - `torch.nn.BatchNorm1d()` = normalizes the output of a previous layer

- Defines a set of **droupout** layers
  - `torch.nn.Dropout3d()` = randomly disables the neurons during the training phase

https://pytorch.org/docs/stable/nn.html#normalization-layers

https://pytorch.org/docs/stable/nn.html#dropout-layers

Optim package

- Implements **optimization algorithms** that will update the parameters based on the computed gradients
    - `torch.optim.SGD()` = implements Stochastic Gradient Descent
    - `torch.optim.Adam()` = implements adaptive moment estimation
    - `torch.optim.RMSprop()` = optimization algorithm first proposed by Geoffrey Hinton (link)
- L2 regularization is included in optimizers: `weight_decay` parameter = λ

https://pytorch.org/docs/stable/optim.html

Data package

- Provides tools to preparing the data
  - `torch.utils.data.Dataset()` = base class representing a dataset
  - `torch.utils.data.DataLoader()` = loads the data in batch and can shuffle them

- Traditional data pipeline for deep learning
  - Pre-processing the data on CPU (data augmentation, cropping, etc),
  - Then loading small batches of pre-processed data on the GPU

https://pytorch.org/docs/stable/data.html

Torchvision package

- Consists of tools for **computer vision**
- Datasets
  - `torchvision.datasets.MNIST()` = handwritten digits
- Models
  - `torchvision.models.alexnet()`
- Image transformations
  - `torchvision.transforms.RandomCrop()` = crop the image at a random location

https://pytorch.org/docs/stable/torchvision/

What is the difference between epoch, batch size and iteration?

Epoch consists of **one full cycle** through the dataset

Batch size is the **number of examples** used in one training iteration

Iterations is the **number of steps** needed to complete one epoch

Example

We have **1000 images** in training dataset.

batch size

Let’s **divide** the training dataset into parts each one with **250 images**

So, it will take **4 iterations** to complete **1 epoch**.

$$\frac{\texttt{Dataset size}}{\texttt{Batch size}} \texttt{ = ceil(number iterations) => 1 epoch}$$

References

- PyTorch Tutorials
  - PyTorch official site: https://pytorch.org/tutorials/
- Deep Learning with PyTorch
  - Videos and blog post: http://deeplizard.com/learn/video/v5cngxo4mIg
  - Book: https://pytorch.org/assets/deep-learning/Deep-Learning-with-PyTorch.pdf
  - Course by Facebook: https://www.udacity.com/course/deep-learning-pytorch--ud188
- High-level training APIs
  - Fast.ai: Library that simplifies training using PyTorch: https://docs.fast.ai
  - Skorch: Scikit-learn compatible library that wraps PyTorch: https://skorch.readthedocs.io/

Hands-on PyTorch