Hands-on Tutorial

Web Conference 2021

https://theevann.github.io/webconf-pytorch-workshop/

Evann Courdier
PhD, Deep Learning for Computer Vision

What to expect ?

Tutorial format

- 4 hours
- Prerequisites
  - Python
  - Gradient Based Machine Learning
- Explanation Slides
- Interactive Notebooks

Tutorial page

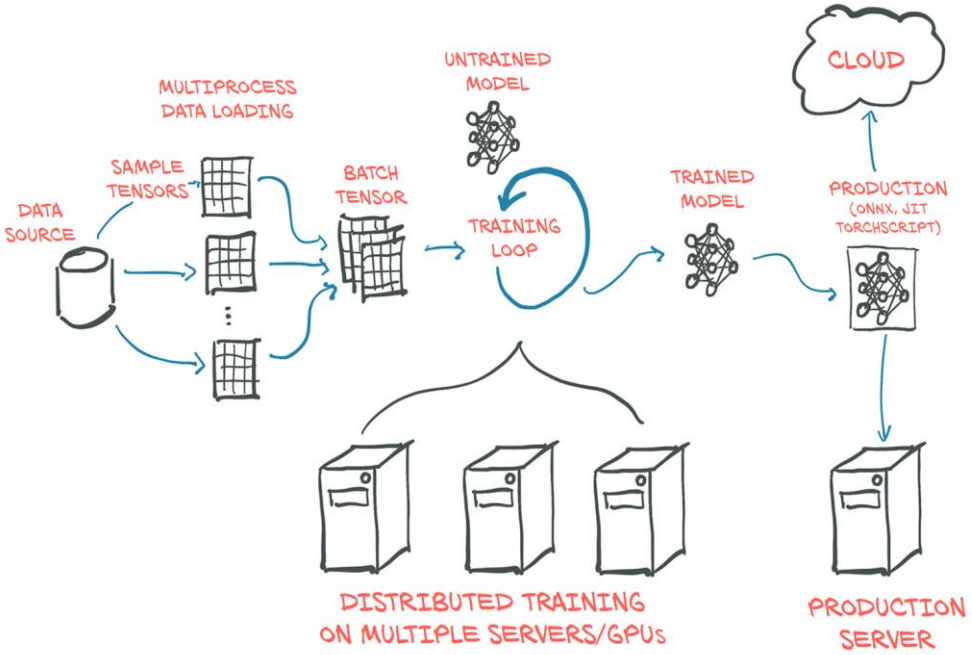
https://theevann.github.io/webconf-pytorch-workshop/

PyTorch

A library for scientific computing in Python, just like NumPy with:

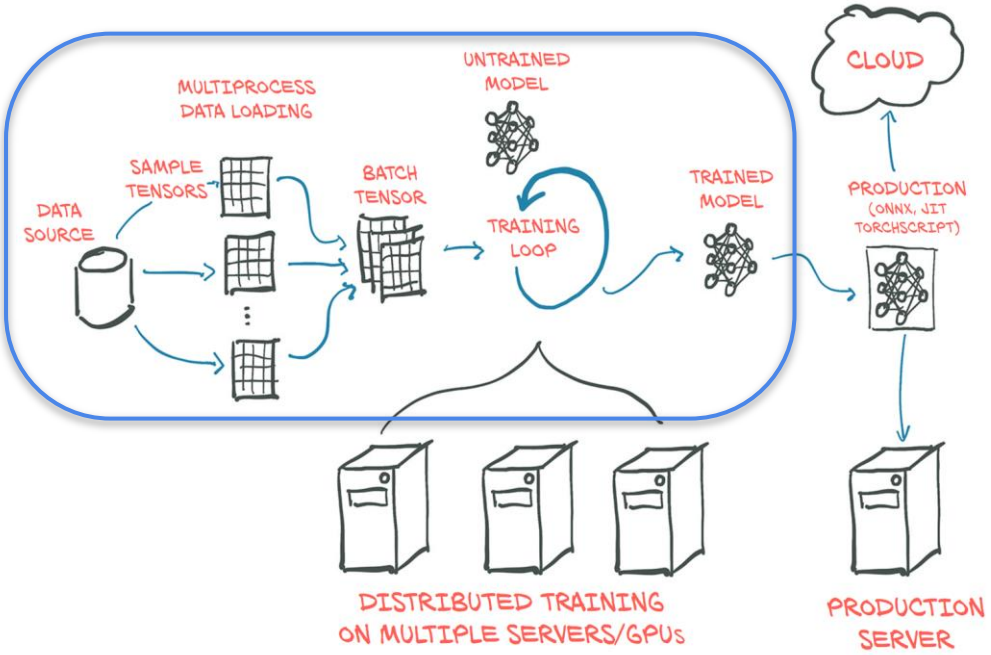
- GPU support
- Automatic differentiation & Optimization algorithms
- All necessary tools for Deep Learning

Deep Learning pipeline with PyTorch



All illustrations are taken from the Deep Learning with Pytorch e-book

Deep Learning pipeline with PyTorch



It’s all about Tensors

- Tensors are multi-dimensional arrays
- Very similar to NumPy for Tensor creation, indexing, masking

```
np.array([[1, 2, 3], [4, 5, 6]])

np.eye(2)

np.arange(1,5)

np.zeros(5)
```

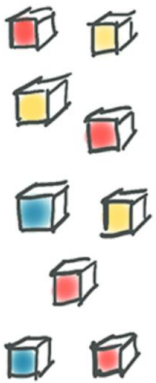
```
torch.tensor([[1, 2, 3], [4, 5, 6]])

torch.eye(2)

torch.arange(1,5)

torch.zeros(5)
```


jupyter

Tensors - Recap

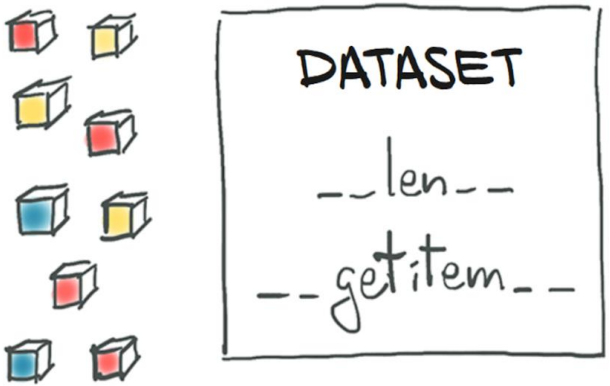
- Size and dimensions: `tensor.dim()` and `tensor.shape`
- Chain operations: `tensor.log().sum().exp()`
- In-place operations with underscore: `tensor.log_()`
- Reshape: `tensor.view(2,3)` and `tensor.view(-1, 3)`
- GPU ⇔ CPU: `torch.device` and `tensor.to(device)`

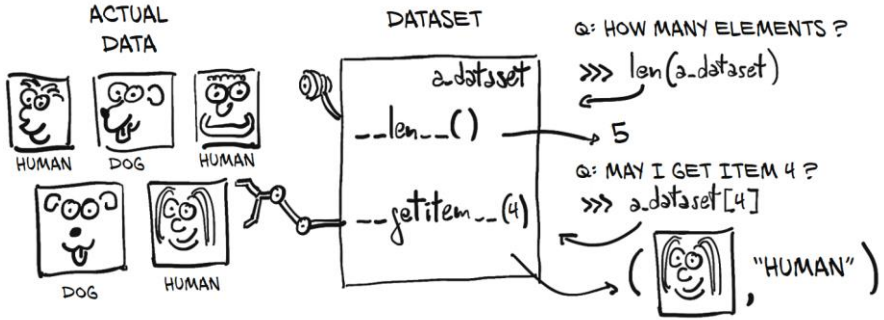
Working with data: Dataset & Dataloaders



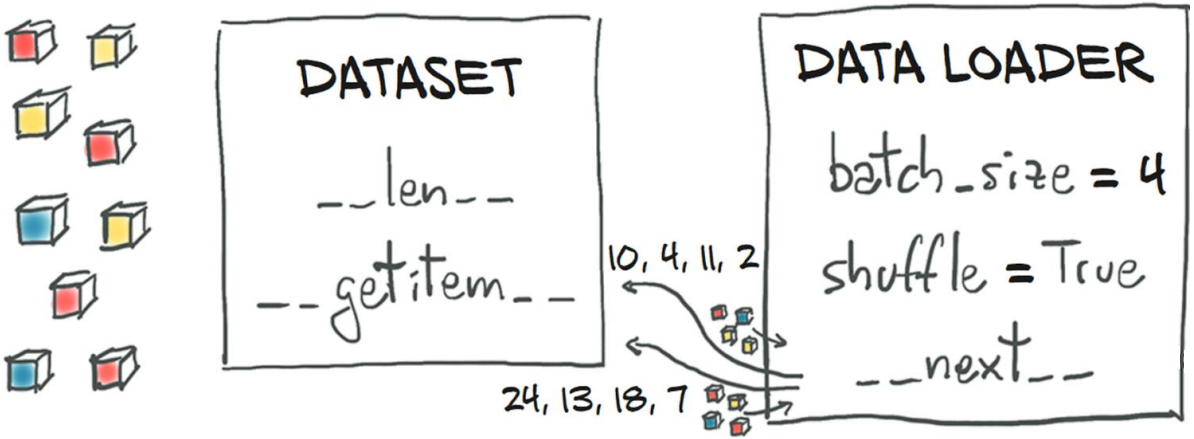
```
data/human-vs-dog
│
├──train
│       │
│       ├──human
│       │      ├── 20.jpg
│       │      ├── 104.jpg
│       │      ├── ...
│       │
│       └──dog
│              │   1.jpg
│              │   78.jpg
│              ├── ...
│
└── validation
       ├── human
       │      │   233.jpg
       │      │   12.jpg
       │      ├── ...
       │
       └──dog
              │   22.jpg
              │   77.jpg
              ├── ...
```

Working with data: Dataset & Dataloaders

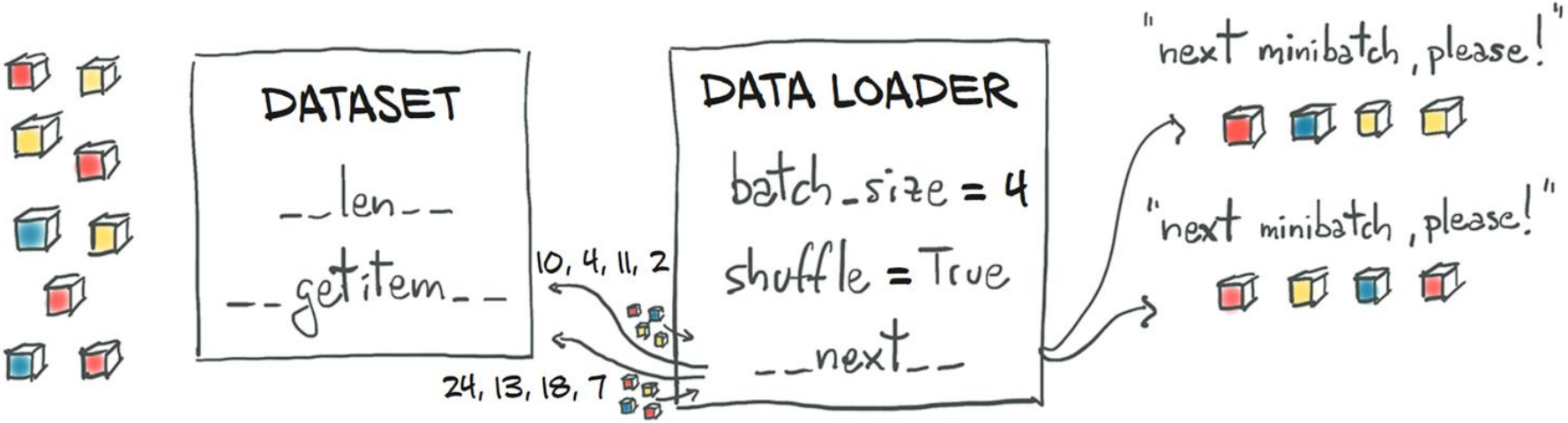




Working with data: Dataset & Dataloaders



Working with data: Dataset & Dataloaders




jupyter

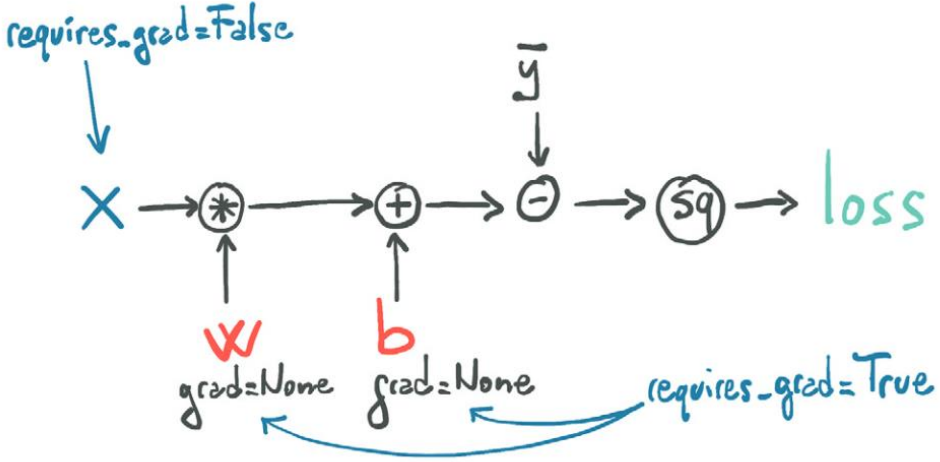
Computing gradients with Autograd

- Autograd: Automatic Differentiation package
- Each Tensor has a `requires_grad` boolean attribute
- Autograd creates a graph to record all operations during the computation
- Call `tensor.backward()` to compute all gradients automatically
- Gradients are accumulated into the `tensor.grad` attribute

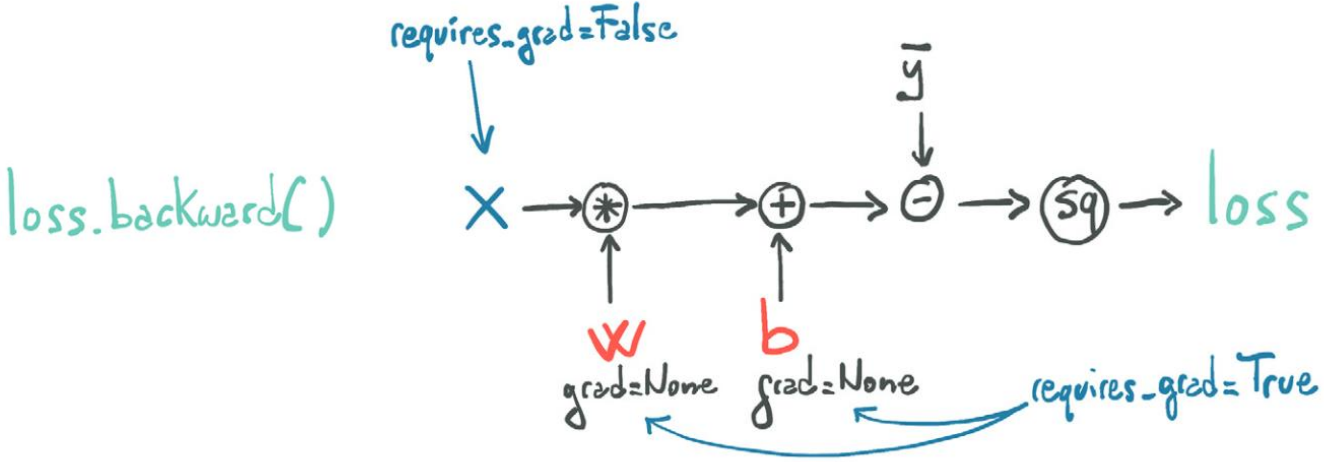
Computing gradients with Autograd

```
loss = (x * W + b - y) ** 2
```

Computing gradients with Autograd

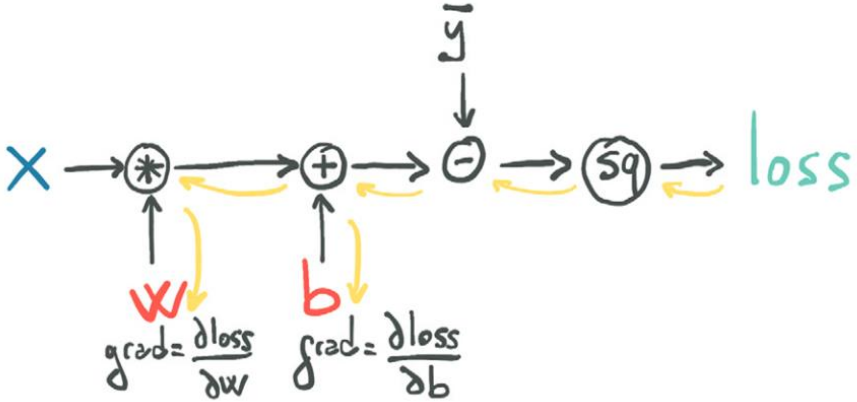


Computing gradients with Autograd



Computing gradients with Autograd

loss.backward()




jupyter

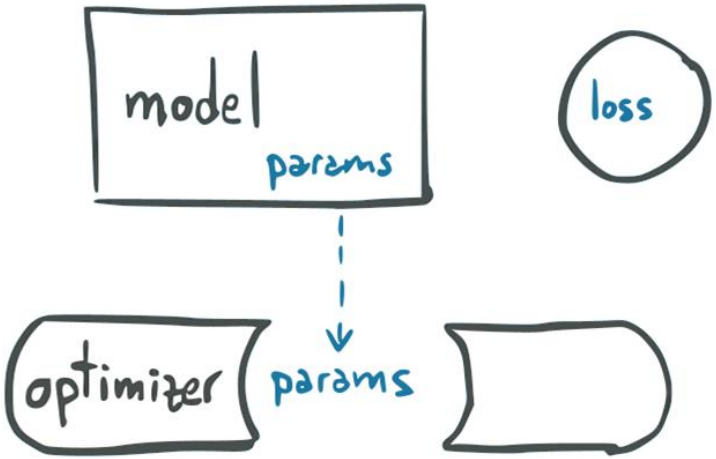
Optimizing parameters

- Use `torch.optim` submodule containing different optimizers
- The optimizer constructor takes a list of parameters
- A call to `optimizer.step()` updates the parameters

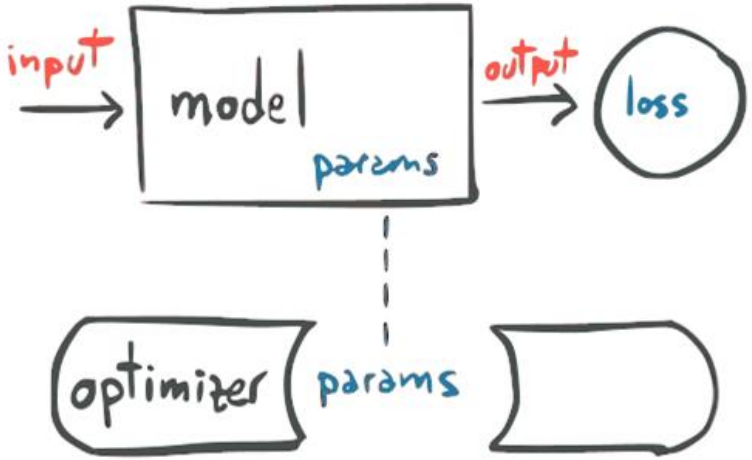
Instead of `model.zero_grad()`,

you can use `optimizer.zero_grad()`

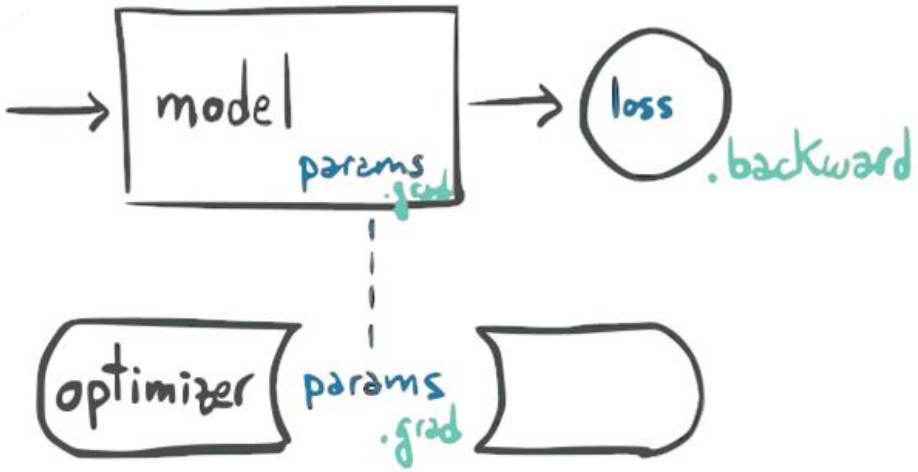
Optimizing parameters



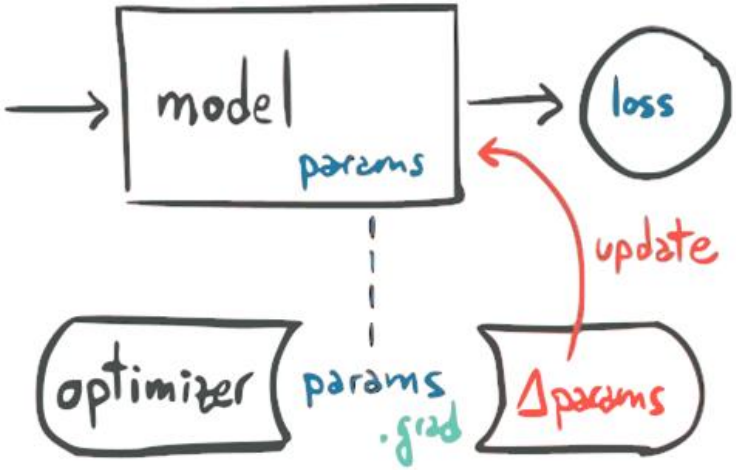
Optimizing parameters



Optimizing parameters



Optimizing parameters




jupyter

Modules - Overview

- Help building reusable model components
- Manage model parameters
- PyTorch provides lots of built-in modules

Modules - Managing Parameters

Modules help to:

- keep track of all parameters in your model.
- save/load your model
- reset all parameters gradients
- move all parameters to the gpu

Modules - torch.nn

Whole library dedicated to Neural Network, including:

- Linear / Convolution / Recurrent Layers
- Activation Functions (ReLU, Tanh, …)
- Loss Functions (MSE, CrossEntropy, ...)
- Pooling, Normalization, Dropout Layers

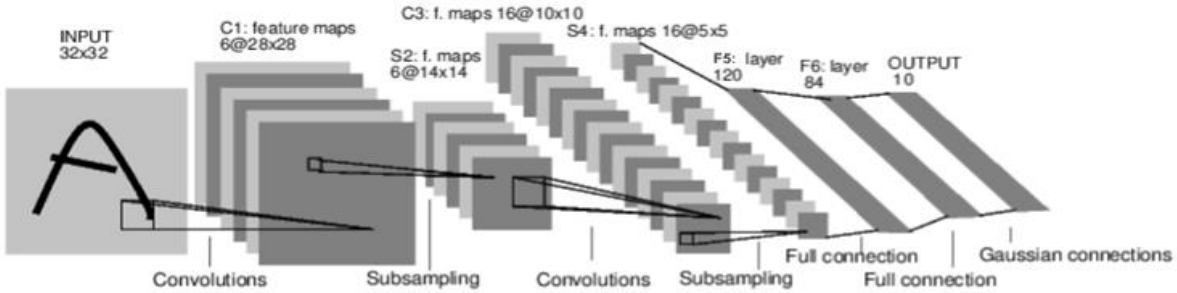
Modules - torch.nn.Module

- A class you inherit from to create a Module
- It needs to implement two methods:
  - The `__init__` function: What are the components of your model
  - The `forward` function: How these components are connected


jupyter

Walkthrough: Building and Training LeNet on MNIST

- Building LeNet5 with torch.nn.Module
- Loading MNIST dataset with TorchVision
- Training for multiple epochs with a custom train function

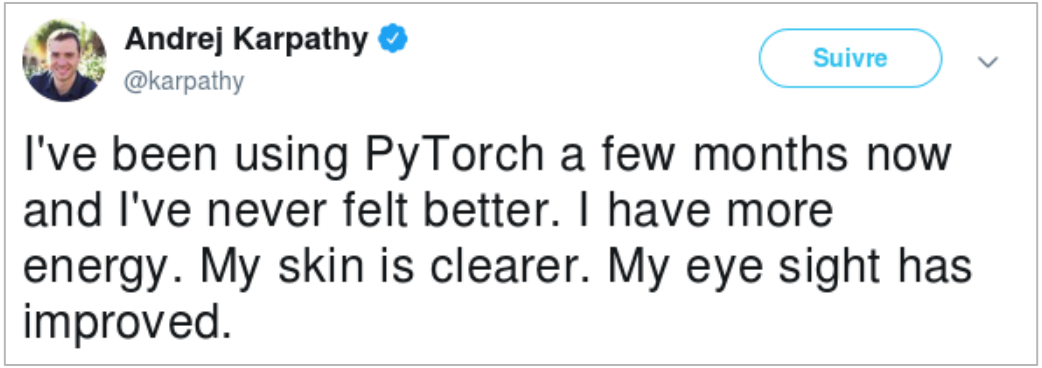



jupyter

What you've still to discover

- Torchvision, Torchtext, Torchaudio
- Multi-GPU Distributed Training
- Quantisation & Pruning
- High-Performance with TorchScript + JIT
- Going to Production with ONNX, TorchElastic and TorchServe
- Organising your code with PyTorch Lightning

Still wondering why PyTorch?



Other References & Material

- Our notebooks at github.com/theevann/webconf-pytorch-workshop/
- Deep-Learning with PyTorch e-book
- Official tutorials at pytorch.org/tutorials/
- A good tutorial on towardsdatascience at bit.ly/38VgfaT/
- Advanced EPFL Deep Learning Course at fleuret.org/ee559/

Hands-on Tutorial

Web Conference 2021