Deep Learning Toolkit *(PyTorch & Timm)*

Rowel Atienza, PhD

University of the Philippines

github.com/roatienza

2022

Outline

Environment, Code Editor

Python

Tensor libraries – numpy, einsum, einops

PyTorch, Timm

Huggingface (HF), Gradio, Streamlit

HF Accelerator, GitHub

Machines – Colab, DeepNote, Kaggle, SageMaker

Other tools

PyTorch

https://pytorch.org/

https://github.com/yunjey/pytorch-tutorial

Why PyTorch?

Easy to build, train, validate and debug models

Available implementation and pre-trained weights of state-of-the-art (SOTA) models

Huge community of users

Production-ready

Install and Test

```
pip install torch torchvision torchaudio
```

Activate python3

```
>>> import torch
>>> print(torch.__version__)
1.10.2
```

Introducing PyTorch for Deep Learning

`torch.Tensor`

Model Inference

Tensor

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

Tensor – PyTorch Data Structure

Numpy data structure: `ndarray`

```
>>> a = np.ones((1,2))
>>> type(a)
<class 'numpy.ndarray’>
```

PyTorch data structure: `Tensor`

```
>>> b = torch.ones((1,2))
>>> type(b)
<class 'torch.Tensor'>
```

Numpy `ndarray` vs PyTorch `Tensor`

Data Structure	CPU	AI Accelerator		
		GPU	TPU	IPU
Numpy `ndarray`	✓	✗	✗	✗
PyTorch `Tensor`	✓	✓	✓	✓

Tensor operations/attributes

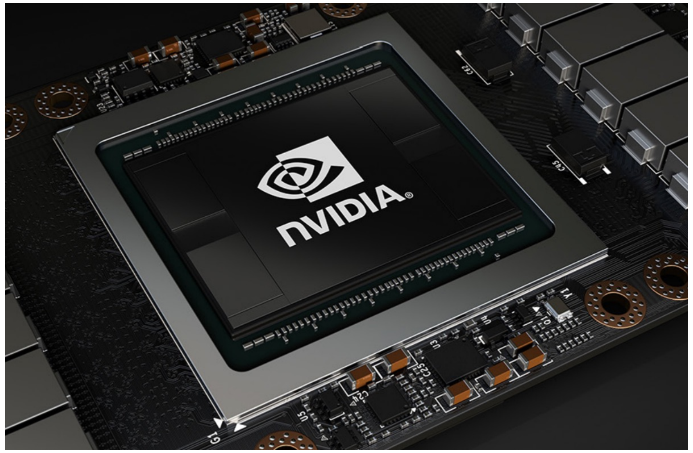
```
Initialize:
a = torch.tensor(
                 [[2., 2.],
                  [4., 4.]])
x = torch.tensor(
                 [[1.], [2.]])
Size/shape:
>>> a.size()
torch.Size([2, 2])
>>> a.shape
torch.Size([2, 2])
>>> a.dtype
torch.float32
```

Multiply:

```
>>> a @ x
tensor([[ 6.],
         [12.]])
>>> torch.matmul(a,x)
tensor([[ 6.],
         [12.]])
>>> einsum(
     'i j, j k -> i k’,
      a, x)
tensor([[ 6.],
         [12.]])
```

Available Devices for PyTorch

```
device = "cuda" if torch.cuda.is_available() else "cpu"
print(f"Using {device} device")
```

Laptop	GPU server
`Using cpu device`	`Using cuda device`
	

Tensor in GPU

```
a = torch.tensor([[2., 2.], [4., 4.]])
a.device
```

Laptop	GPU server
`device(type='cpu')`	`device(type='cpu')`

```
a = a.to(device)
```

Laptop	GPU server
`device(type='cpu')`	`device(type='cuda', index=0)`

Tensor in GPU and Back to CPU

Laptop	GPU server
>>> a tensor([[2., 2.], [4., 4.]])	>>> a tensor([[2., 2.], [4., 4.]], device='cuda:0')

Laptop – Back to CPU (No Change)	GPU server – Back to CPU

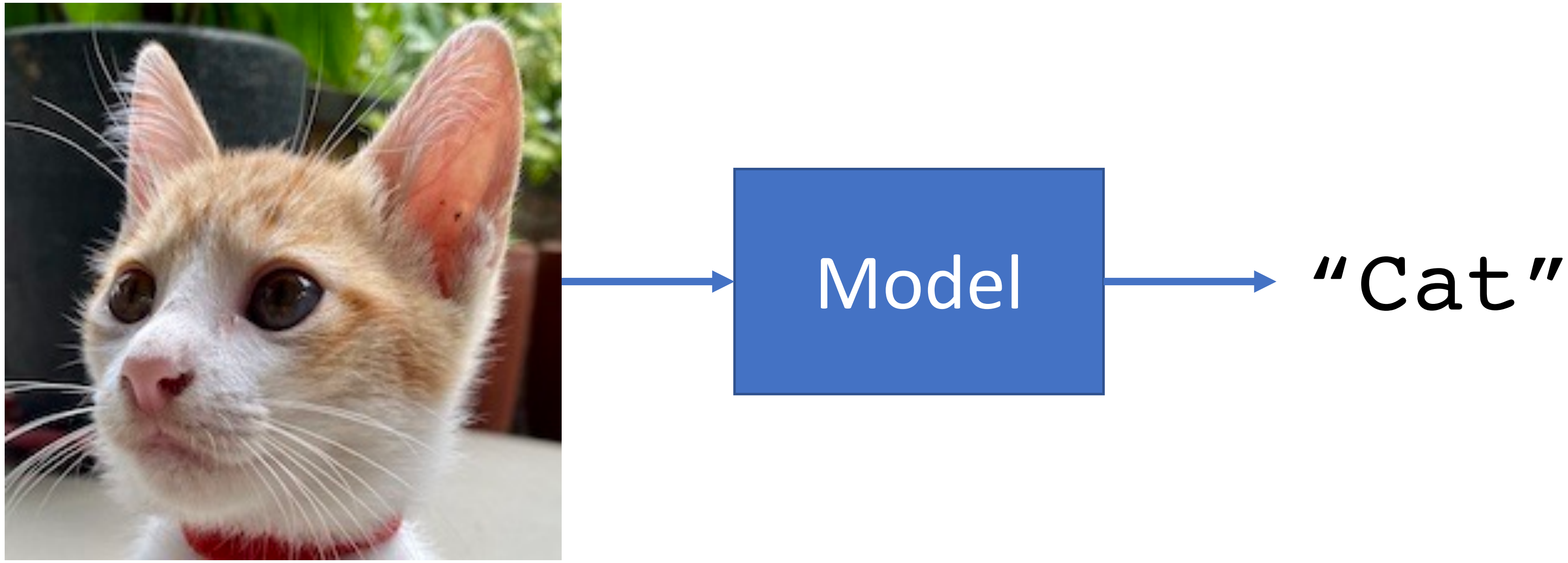
>>> a = a.cpu()
>>> a
tensor([[2., 2.],
        [4., 4.]])
	>>> a = a.cpu()
	>>> a
	tensor([[2., 2.],
	        [4., 4.]])

Numpy to PyTorch to Numpy

Data Structure	Device	Code
`np.ndarray`	CPU	`a = np.array([ [1., 2.], [2., 4.] ])`
`torch.Tensor`	CPU	`a = torch.from_numpy(a)`
`torch.Tensor`	GPU	`a = a.cuda()`
`torch.Tensor`	CPU	`a = a.cpu()`
`np.ndarray`	CPU	`a = a.numpy()`

Model Inference

Model Inference



Input Data → Trained Model → Output Prediction

Input

Can be any type of data

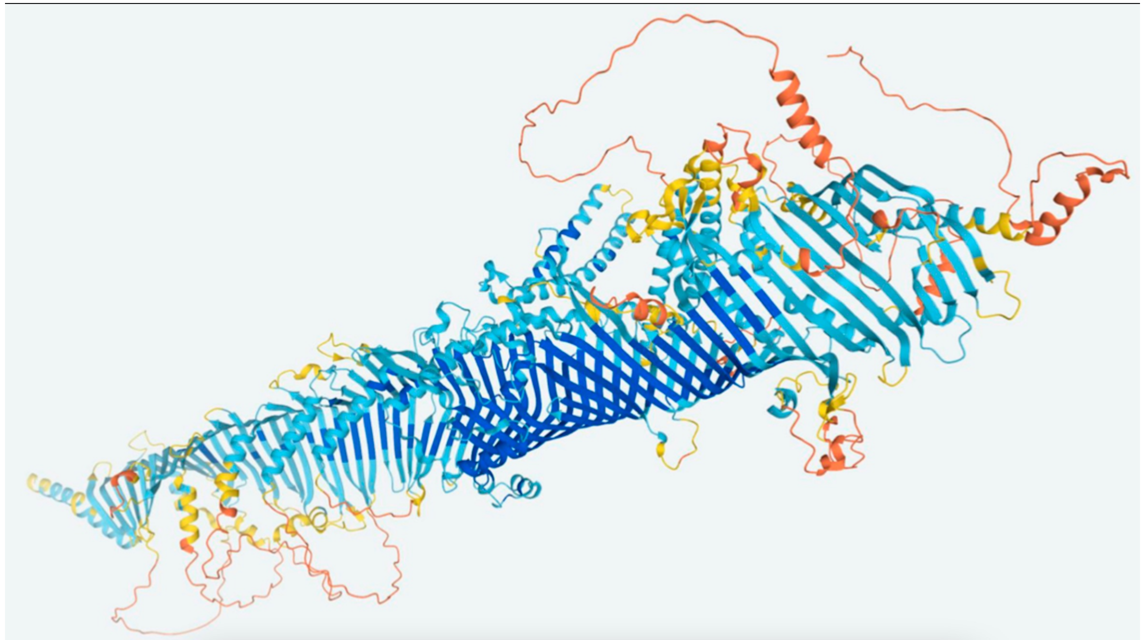
Vision: image, video

Waveforms: speech, music

3D: point cloud

Text: character, word, phoneme

Other forms: radar, multi-spectral, protein structure, etc



Protein structure of a fruitfly [Science.org 2021]

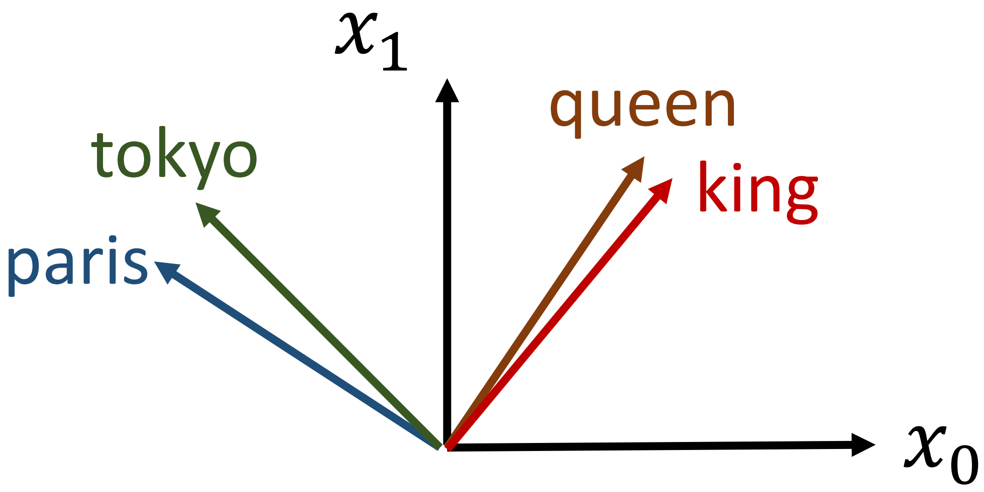
Loading Image Data

```
from PIL import Image
img = Image.open(“wonder_cat.jpg”)

# Visualize the data
# in Jupyter
display(img)
```

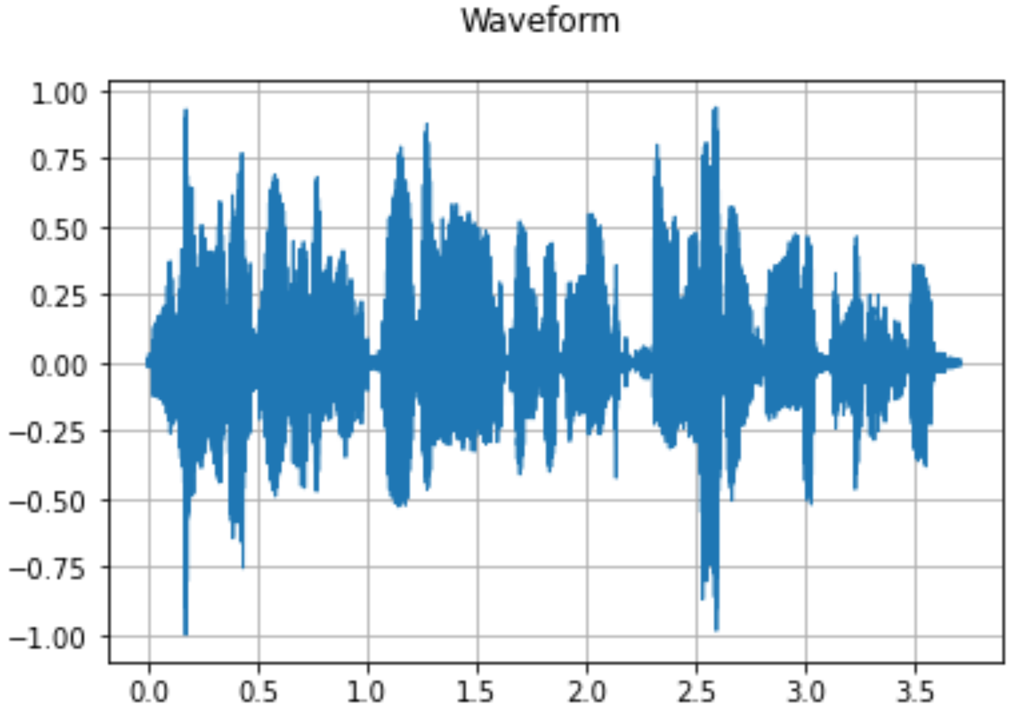
Loading Text Data

```
words = {"hello": 0, "world": 1}
embed_len = len(words)
embed_dim = 4
embed = torch.nn.Embedding(embed_len, embed_dim)
lookup = torch.tensor([words["hello"]], dtype=torch.long)
embed(lookup) # tensor([[-0.3745,  0.1376, -0.3058,  1.0258]])
```



Loading Audio/Speech Data

```
import librosa
wav, sample_rate = librosa.load(“data/ljspeech.wav”)
plot_waveform(wav, sample_rate)
```



Specialized PyTorch Libraries

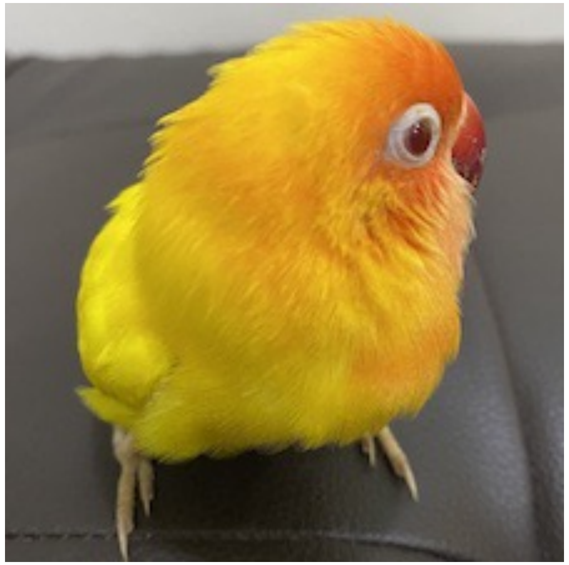
`torchvision` - package consists of popular datasets, model architectures, and common image transformations for computer vision.

`torchaudio` - library for audio and signal processing with PyTorch. It provides I/O, signal and data processing functions, datasets, model implementations and application components.

Other libraries – `torchtext`, `torchrec`

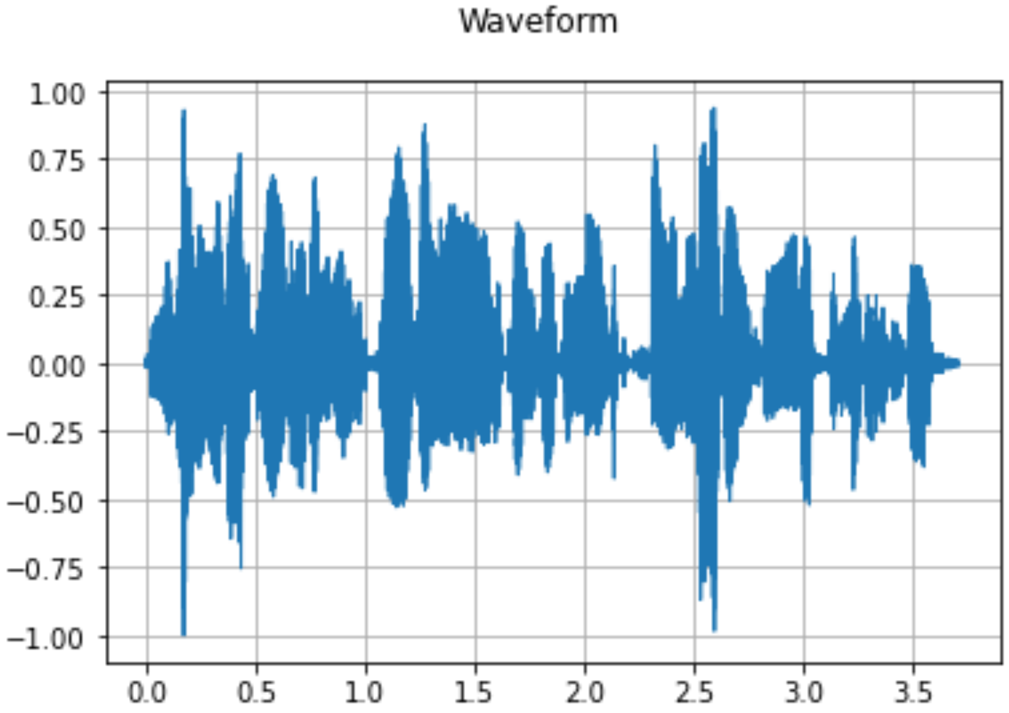
TorchVision

```
import torchvision
img = torchvision.io.read_image("data/birdie2.jpg")
img = torchvision.transforms.ToPILImage()(img)
display(img)
```



TorchAudio

```
import torchaudio
wav, sample_rate = torchaudio.load (“data/ljspeech.wav”)
plot_waveform(wav, sample_rate)
```



Loading Pre-trained Model from `torchvision`

```
resnet = torchvision.models.resnet18(pretrained=True)
```

Other pretrained models available:

AlexNet, SqueezeNet, VGG, EfficientNet, MobileNet, RegNet, ViT, ConvNeXt, etc.

See: https://pytorch.org/vision/master/models.html

Input Data Preparation for Model Ingestion

Simple transform:

```
from PIL import Image
import torchvision.transforms as transforms

img = Image.open(“wonder_cat.jpg”)
img = transforms.ToTensor()(img)
```

img

Input Data Preparation for Model Ingestion

Better:

```
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                  std=[0.229, 0.224, 0.225])
transform = transforms.Compose([
                      transforms.Resize(256),
                      transforms.CenterCrop(224),
                      transforms.ToTensor(),
                      normalize,])

# PIL image undergoes transforms.
img = transform(img)
```

img

Output: Model Prediction

Model must be in evaluation model: `resnet.eval()`

Ensure that there is a batch dim. If none, add:

```
img = rearrange(img, 'c h w -> 1 c h w’)
```

Do the inference in no gradient tracking context:

```
with torch.no_grad():
    pred = resnet(img)
```

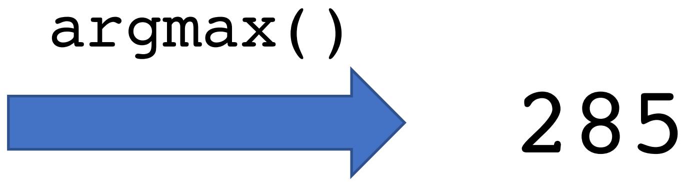
Finally, get the index of the maximum probability:

```
pred = torch.argmax(pred, dim=1)
```

What is `argmax()` of `pred` ?

`pred`

Index	Unnormalized Probabilities
0	1.7247
1	2.2064
…	
284	7.4005
285	11.4601
286	6.6287
…	
999	3.2967



Human Readable Labels

For ImageNet1k, each index corresponds to a text label:

```
{0: 'tench, Tinca tinca',
 1: 'goldfish, Carassius auratus',
 2: 'great white shark, white shark,
 3: 'tiger shark, Galeocerdo cuvieri',
 …
 998: 'ear, spike, capitulum',
 999: 'toilet tissue, toilet paper}
```

Human Readable Label

For example, pred has a value of 285. This value corresponds to:

```
…
 283: 'Persian cat',
 284: 'Siamese cat, Siamese',
 285: 'Egyptian cat',
 286: 'cougar, puma, catamount, mountain lion,
painter, panther, Felis concolor',
 287: 'lynx, catamount',
 288: 'leopard, Panthera pardus’,
…
```

TIMM: pyTorch IMage Models

https://rwightman.github.io/pytorch-image-models/

Why timm?

From the doc:

`timm` is a deep-learning library created by Ross Wightman and is a collection of SOTA computer vision models, layers, utilities, optimizers, schedulers, data-loaders, augmentations and also training/validating scripts with ability to reproduce ImageNet training results.

In short:

`timm` extends PyTorch by implementing many deep learning SOTA models, optimization, regularization and other useful algorithms.

Install and Use

Install:

```
    pip install timm
```

Use it like `torchvision`:

```
if use_timm:
  resnet = timm.create_model('resnet18', pretrained=True)
else:
  resnet = torchvision.models.resnet18(pretrained=True)
```

Code demo is next

https://github.com/roatienza/Deep-Learning-Experiments/blob/master/versions/2022/tools/python/pytorch_demo.ipynb