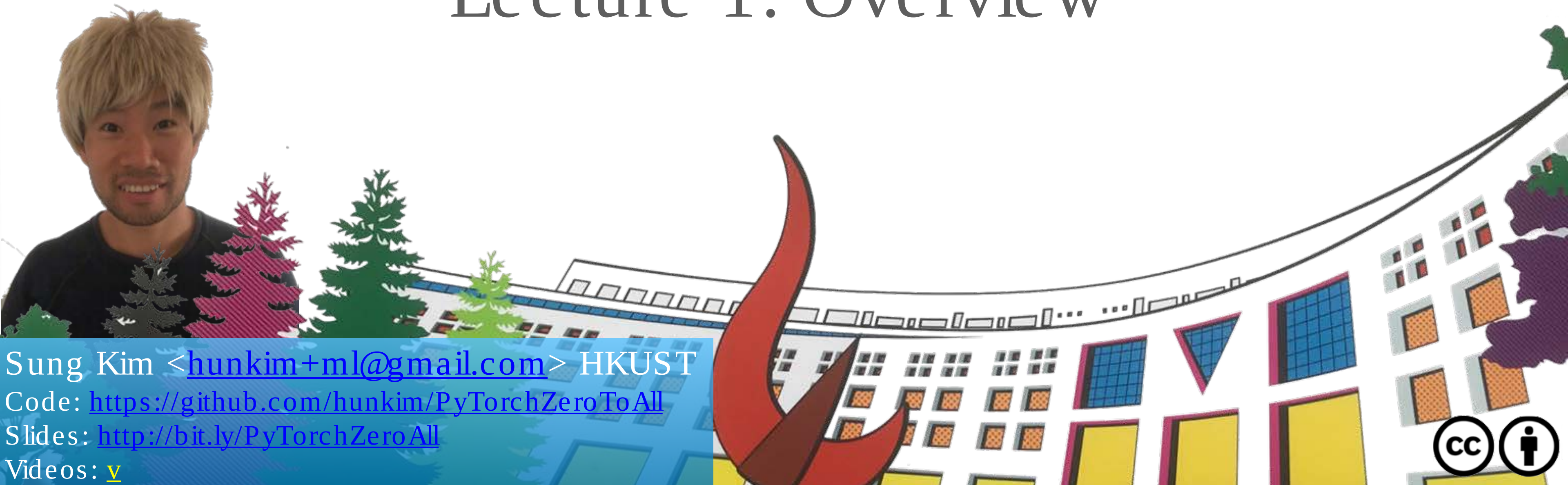
ML/DL for Everyone with PYTORCH

Lecture 1: Overview



Sung Kim <hunkim+ml@gmail.com> HKUST

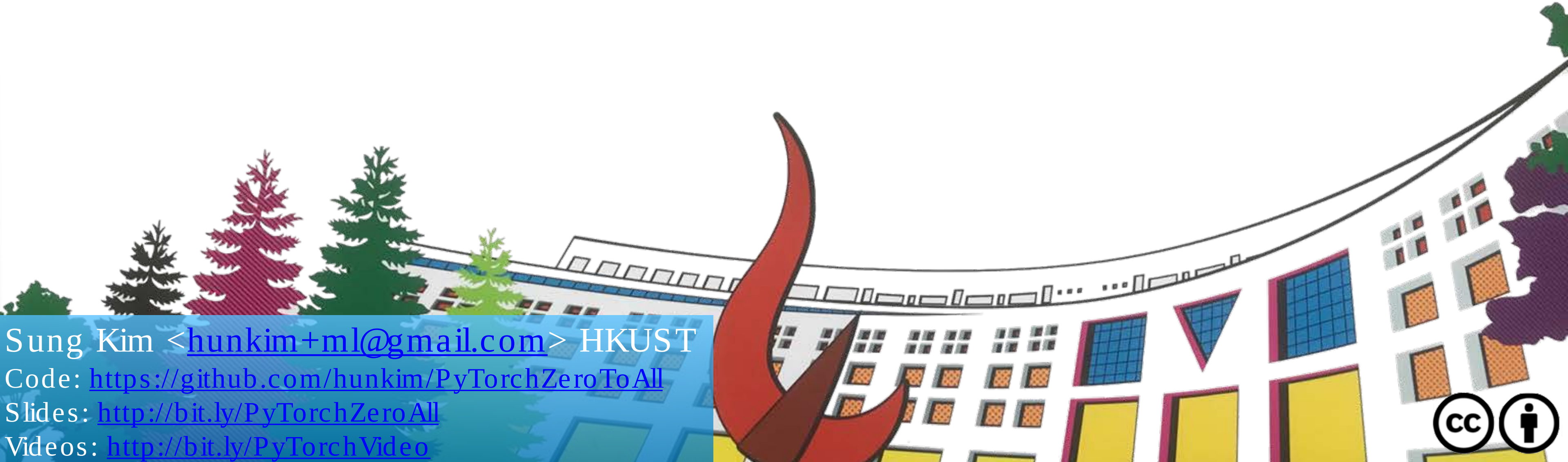
Code: https://github.com/hunkim/PyTorchZeroToAll

Slides: http://bit.ly/PyTorchZeroAll

Videos: v

ML/DL for Everyone with PYTORCH

Lecture 1: Overview

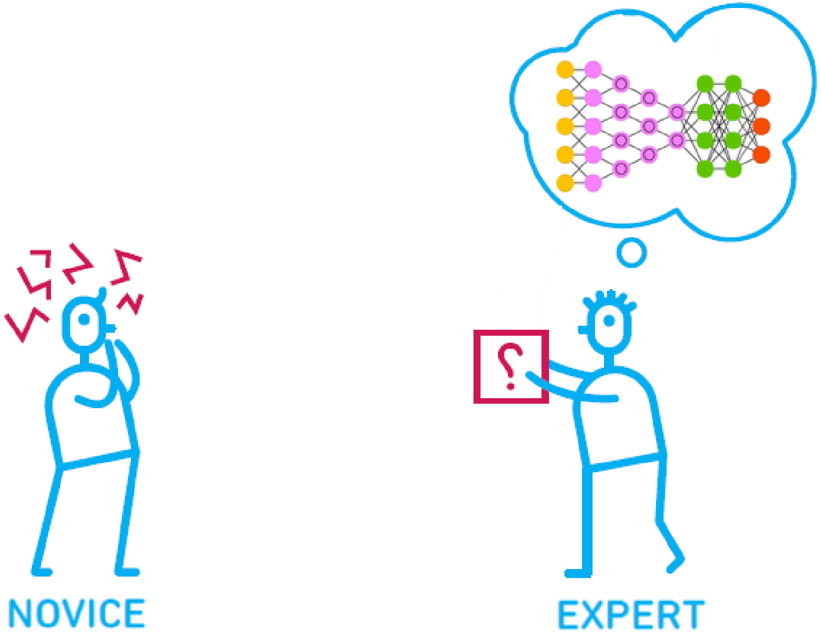


Sung Kim <hunkim+ml@gmail.com> HKUST
Code: https://github.com/hunkim/PyTorchZeroToAll
Slides: http://bit.ly/PyTorchZeroAll
Videos: http://bit.ly/PyTorchVideo

Goals

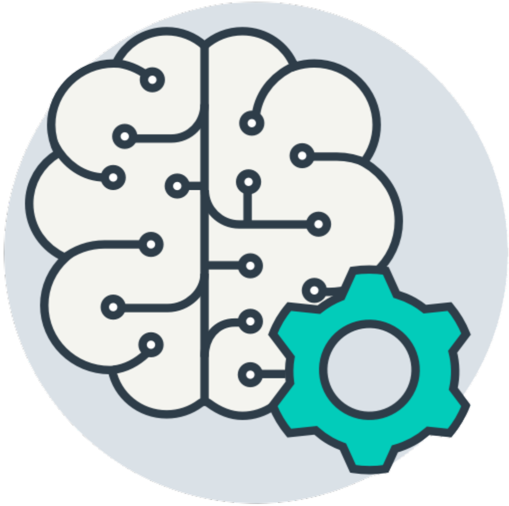
- Basic understanding of machine learning/deep learning
- PyTorch implementation skills

- Zero to All!
  - Basic algebra + probability
  - Basic python



Adapted the Illustration by Oliver Caviglioli

What is ML?



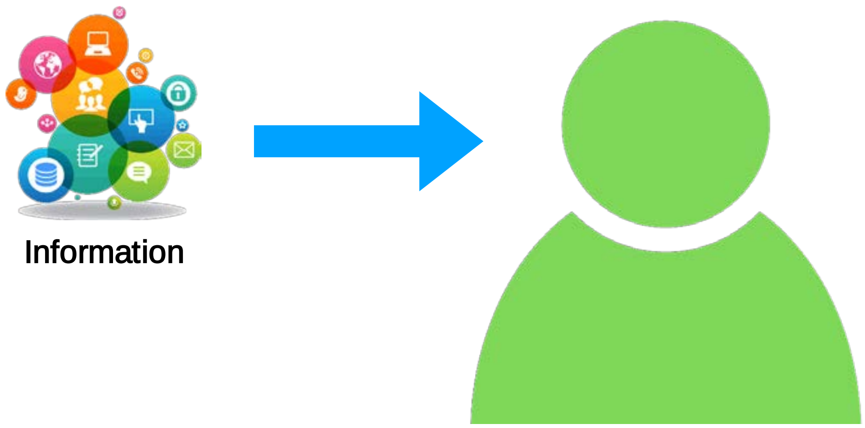
What is Human Intelligence?

What is Human Intelligence?

What to eat for lunch?

What is Human Intelligence?

What to eat for lunch?

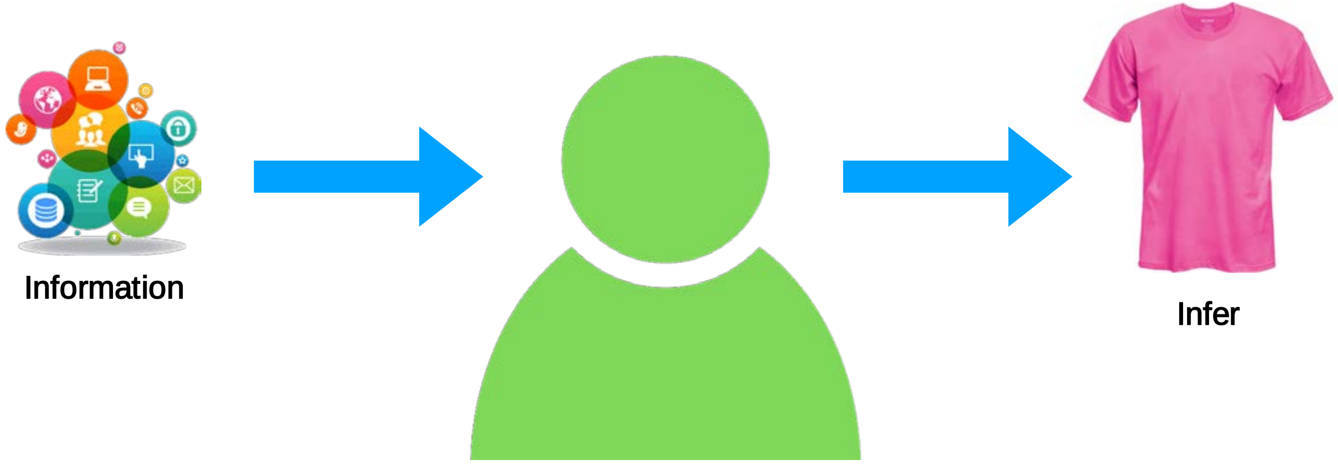


What is Human Intelligence?

What to eat for lunch?

What is Human Intelligence?

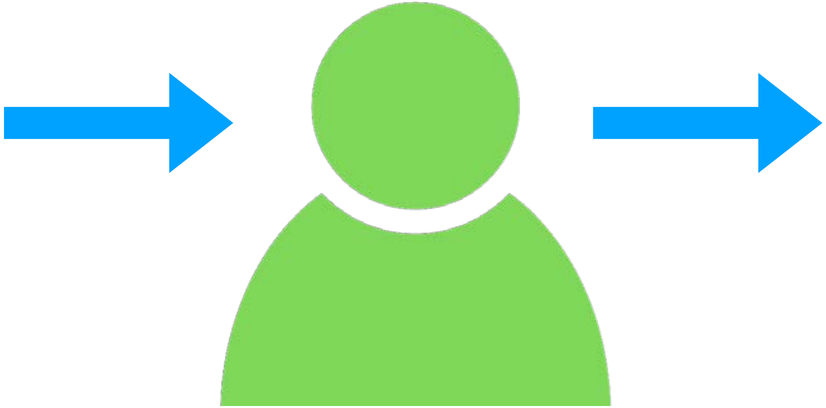
What to dress?



What is Human Intelligence?

What is this picture?

Image information

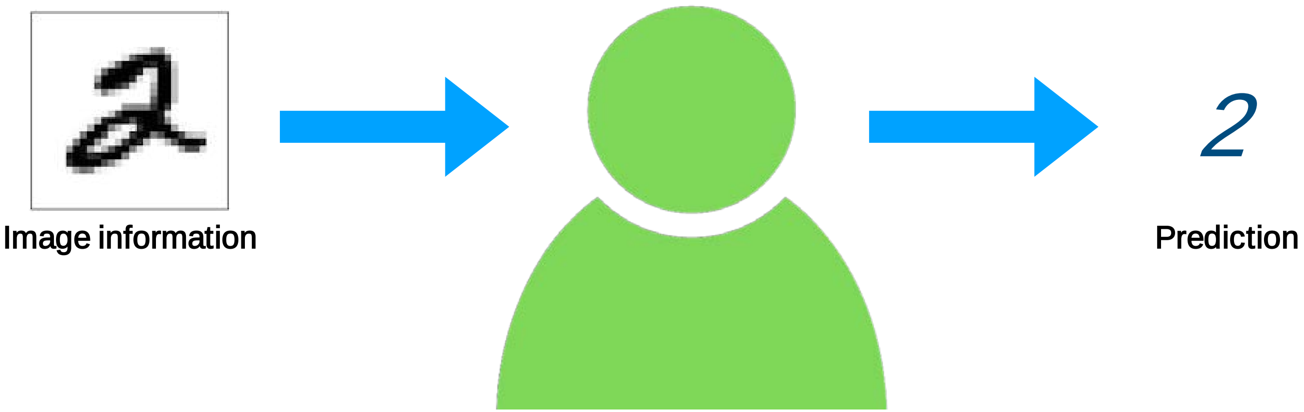


CAT

Prediction

What is Human Intelligence?

What is this number?

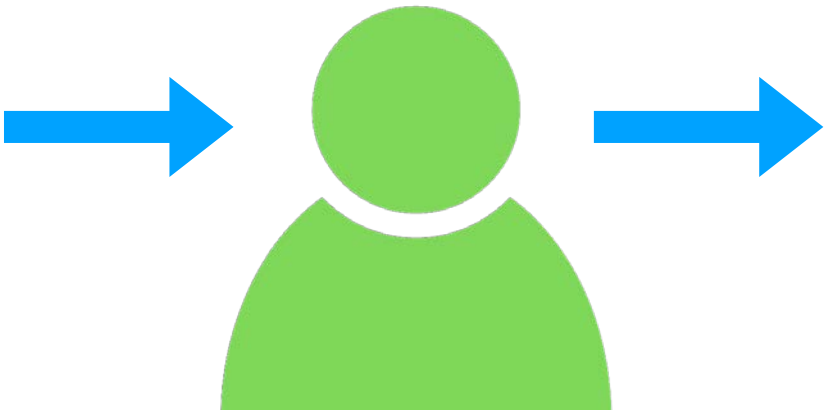


What is Human Intelligence?

What would be the grade if I study 4 hours?

4 hours

information



? points

Prediction

Machine Learning

What to dress?

Machine Learning

What is this picture?

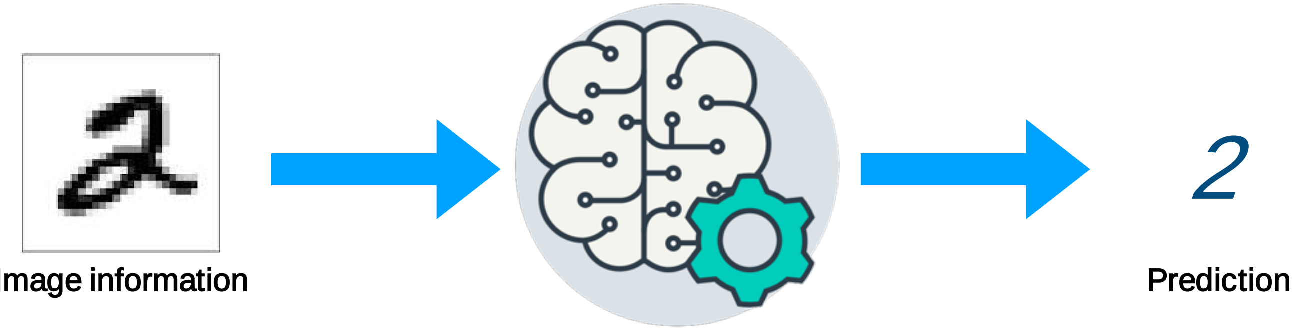
Image information

CAT

Prediction

Machine Learning

What is this number?



Machine Learning

What would be the grade if I study 4 hours?

4 hours

information

? points

Prediction

Machine Learning

Machine needs lots of training

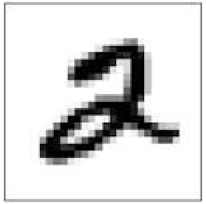


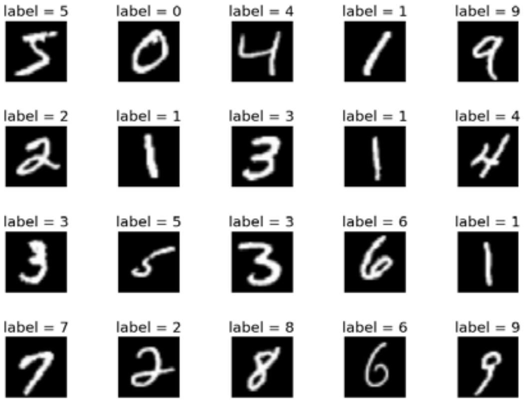
Image information

2

Prediction

Machine Learning

Machine needs lots of training



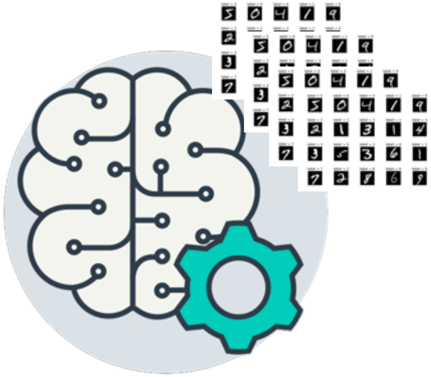
Labeled dataset

training

Model

Machine Learning

Predict (test) with trained model



Trained model

Machine Learning

Predict (test) with trained model

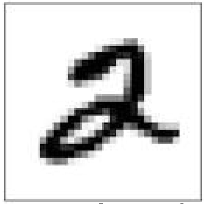
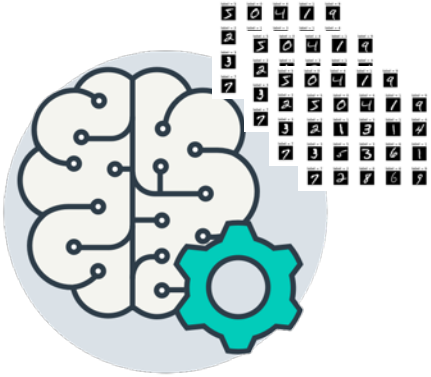


Image information

Test dataset



Trained model

Machine Learning

Predict (test) with trained model

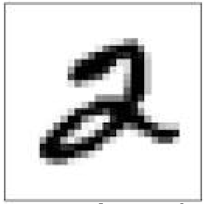
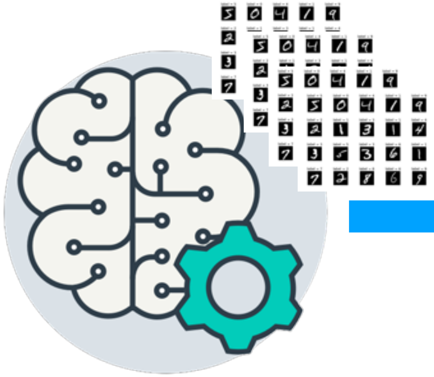


Image information

Test dataset



Trained model

2

Prediction

Machine Learning

What would be the grade if I study 4 hours?

4 hours

? points

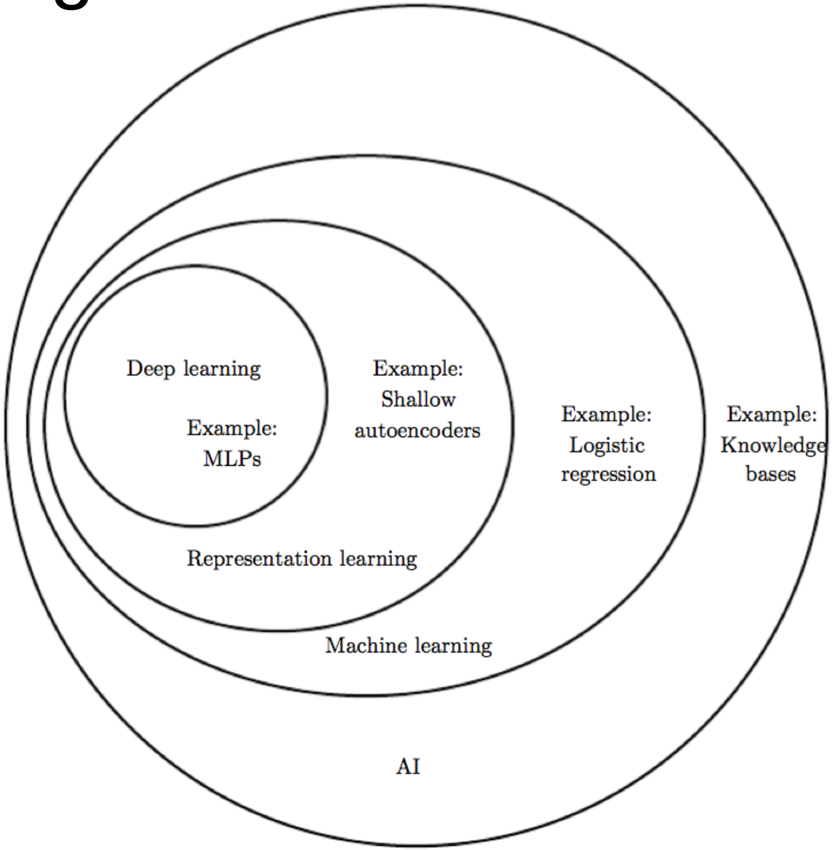
Prediction

Hours (x)	Points (y)
1	2
2	4
3	6
4	?

Training dataset

Test dataset

Deep Learning?



Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville

Why We Care?

Q: What color is court? A: Blue

Visual Question Answering

Answering Questions Based On Images

Generating New Words

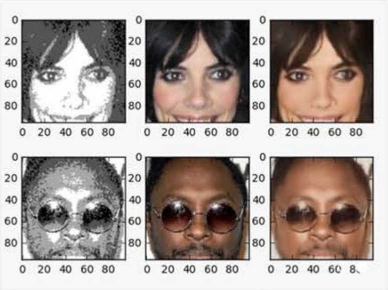
Word Generator Using RNNs

Artistic Style Transfer

Applying The Style Of An Artwork To A Photo

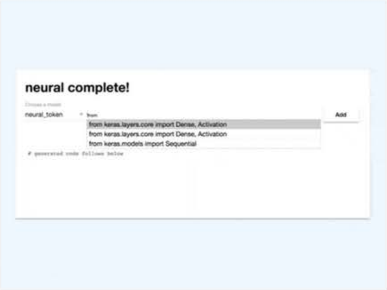
Predicting Cardiac Abnormalities

Using Phonocardiogram (PCG) Data



Creating Photorealistic Images

Improving Images From A Gameboy Camera

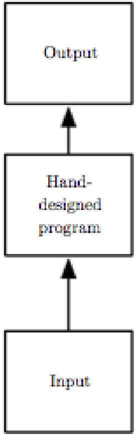


Code Autocompletion

NN That Autocompletes NN Code

More demos at https://ml-showcase.com/

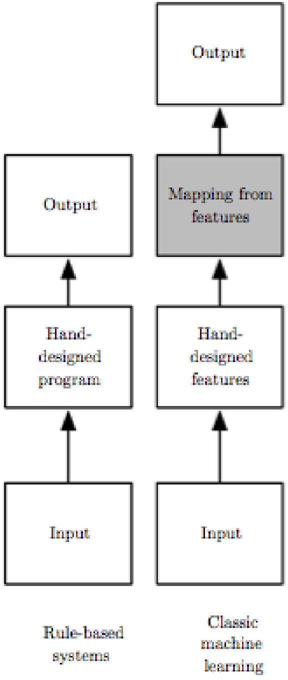
Why We Care as Developer?



Rule-based
systems

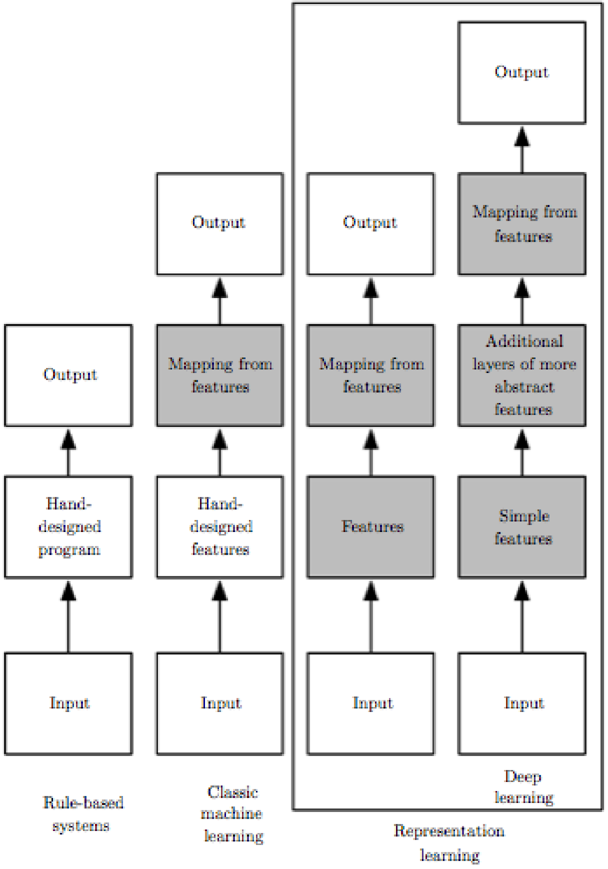
Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville

Why We Care as Developer?



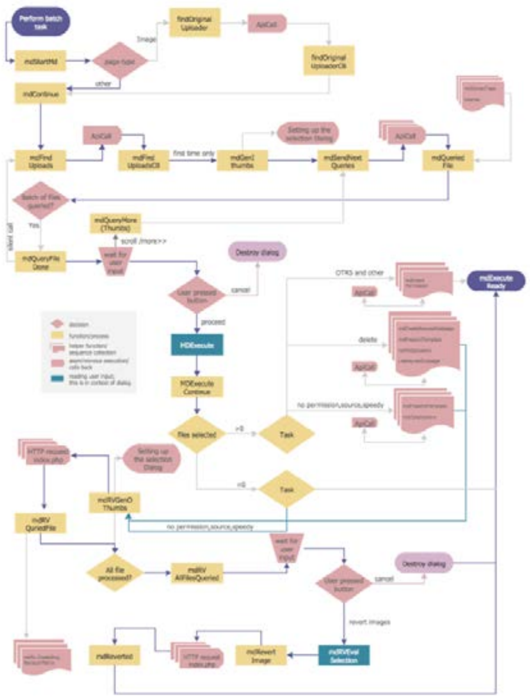
Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville

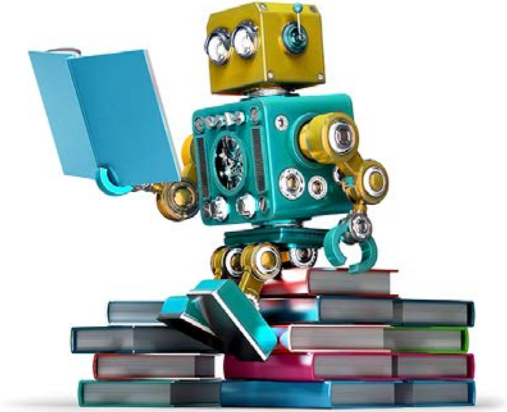
Why We Care as Developer?



Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville

Rule based VS representation learning





Good News

- Deep Learning is not too difficult (yet)
    - Basic algebra + probability + python
    - Less than one year study
- Many frameworks
- Unlimited study resources

- And most of all, it’s really fun

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

http://pytorch.org/about/

Why PYTORCH

- More Pythonic (imperative)
  - Flexible
  - Intuitive and cleaner code
  - Easy to debug
- More Neural Networkic
  - Write code as the network works
  - forward/backward

Install PYTORCH

Get Started.

Select your preferences, then run the PyTorch install command.

Please ensure that you are on the latest pip and numpy packages.
Anaconda is our recommended package manager

OS	Linux	OSX	
Package Manager	conda	pip	Source
Python	2.7	3.5	3.6
CUDA	7.5	8.0	None

Run this command:

```
pip3 install http://download.pytorch.org/whl/torch-0.2.0.post3-cp36-cp36m-macosx_10_7_x86_64.whl
pip3 install torchvision
# OSX Binaries dont support CUDA, install from source if CUDA is needed
```

http://pytorch.org

Google Colab

Google Colab is a platform helping machine learning education and research: https://colab.research.google.com/

- Notebook based on Jupyter/iPython
- Free TPU (Tensor Processing Unit) *supported*
- Python 2.7 and Python 3.6 *supported*
- Tensorflow, Scikit-learn, Matplotlib, PyTorch, etc. *installed*
- Bash command *supported*
- Google Drive, GitHub Repo, GitHub Gist, etc. *interlocked*

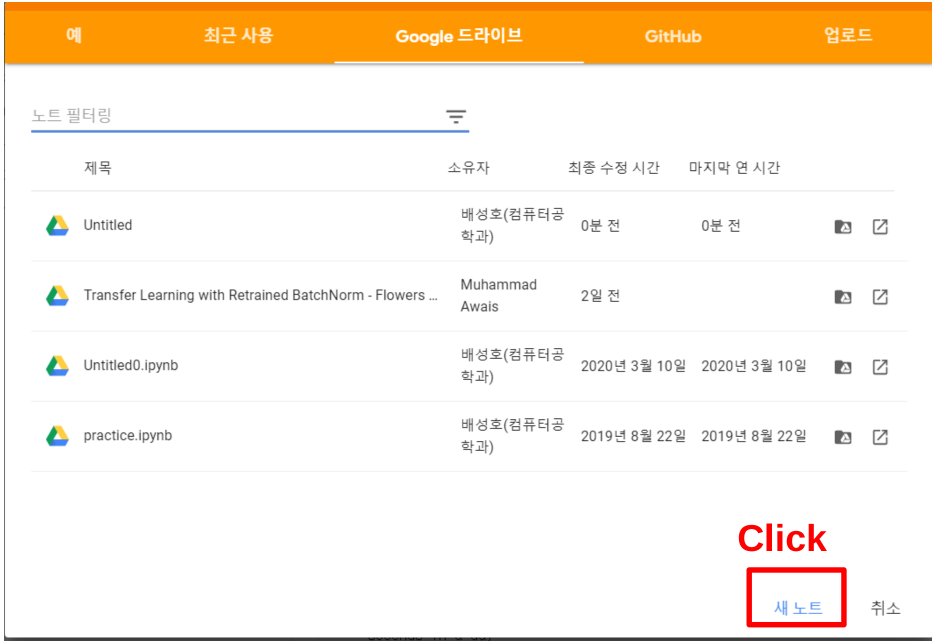
Google Colab

How to use:

- 1. Enter https://colab.research.google.com/

- 2. Click (New Note)

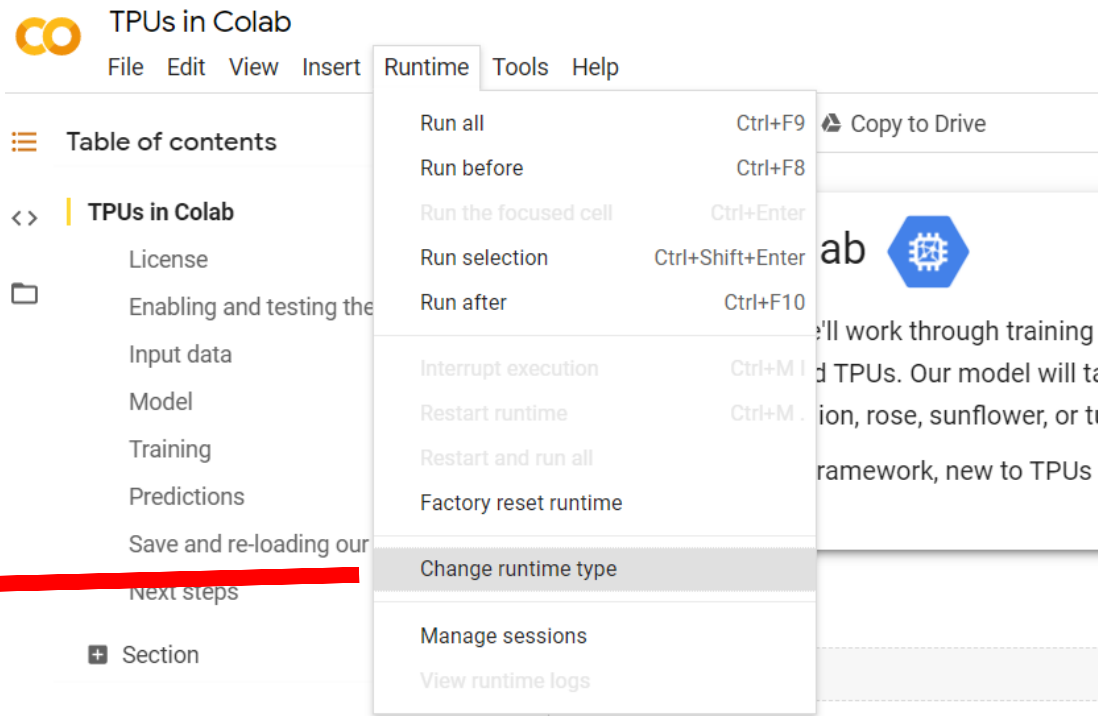
- Note: 12 hours duration

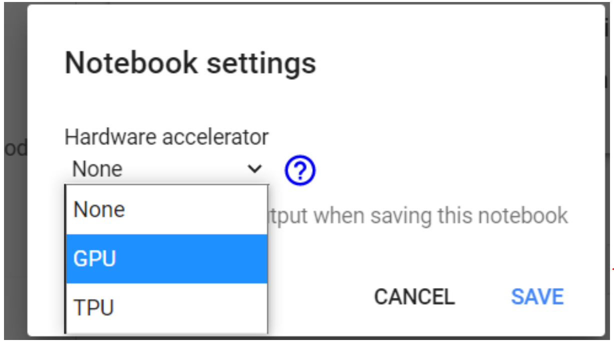


Google Colab

How to use:

- 3. Change the runtile-type





Google Colab

A Pytorch Tensor is conceptually identical to an n-dimensional numpy array.

- Unlike numpy, PyTorch Tensor can utilize GPUs to accelerate their numeric computations

- First, import PyTorch library that pre-installed

```
[24]  import torch
      import numpy as np
      import matplotlib.pyplot as plt
```

[Reference] https://www.marktechpost.com/2019/06/09/getting-started-with-pytorch-in-google-collab-with-free-gpu/

Google Colab

The default tensor type in PyTorch is a float defined as torch.FloatTensor

- We can create tensors:

```
[4]
    # creating a tensor of 3 rows and 2 columns consisting of ones
    x = torch.ones(3, 2)
    print(x)

    # creating a tensor of 3 rows and 2 columns consisting of zeros
    x = torch.zeros(3, 2)
    print(x)
```

```
tensor([[1., 1.],
        [1., 1.],
        [1., 1.]])
tensor([[0., 0.],
        [0., 0.],
        [0., 0.]])
```

Google Colab

Creating a tensor by random initialization

```
[9]  # To increase the reproducibility, we often set the random seed
     # to a specific value first
     torch.manual_seed(2)
     #generating tensor randomly
     x = torch.rand(3, 2)
     print(x)

     # generating tensor randomly from normal distribution
     x = torch.randn(3, 3)
     print(x)
```

```
tensor([[0.6147, 0.3810],
        [0.6371, 0.4745],
        [0.7136, 0.6190]])
tensor([[-2.1409, -0.5534, -0.5000],
        [-0.0815, -0.1633,  1.5277],
        [-0.4023,  0.0972, -0.5682]])
```

Google Colab

Slicing of Tensors (same as slicing ndarrays in Numpy)

```
[11] ## create a tensor
     x = torch.tensor([ [1,2],
                        [3, 4],
                        [5, 6] ] )
     print(x[:, 1]) # slice every row and only the last column
     print(x[0, :]) # slice every column in the first row
     y = x[1, 1] # take the element in the first row
                 # and first column and create a another tensor
     print(y)
```

```
tensor([2, 4, 6])
tensor([1, 2])
tensor(4)
```

Google Colab

Reshape Tensor

- .view reshapes the tensor to a shape with input dimensions
- -1: indicates that the shape will be inferred from previous dimensions

```
[15] x = torch.tensor( [[1, 2],
                        [3, 4],
                        [5,6 ]]) # (3 rows and 2 columns)
     y = x.view(2, 3) # reshape to 2 rows and 3 columns
     print(y)
     y = x.view(6,-1) #y shape will be 6x1
     print(y)
```

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

Google Colab

Mathematical operations

- (_) as postfix: in-place operation

```
[19] # create two tensors
     x = torch.ones([3, 2])
     y = torch.ones([3, 2])

     # add two tensros
     z = x + y # method 1
     z = torch.add(x,y) # method 2
     print(z)

     # subtract two tensors
     z = x - y # method 1
     torch.sub(x,y) # method 2
     print(z)
     y.add_(x) # tensor y added with x and result will be stored in y
     print(y)
```

```
tensor([[2., 2.],
        [2., 2.],
        [2., 2.]])
tensor([[0., 0.],
        [0., 0.],
        [0., 0.]])
tensor([[2., 2.],
        [2., 2.],
        [2., 2.]])
```

Google Colab

PyTorch to Numpy bridge

- .numpy: converting method from tensor to numpy

```
[22] x = torch.linspace(0, 1, steps = 5) # creating a tensor using linspace
     x_np = x.numpy() # convert tensor to numpy
     print(x)
     print(type(x), type(x_np)) # check the types
```

```
tensor([0.0000, 0.2500, 0.5000, 0.7500, 1.0000])
<class 'torch.Tensor'> <class 'numpy.ndarray'>
```

Google Colab

PyTorch to Numpy bridge

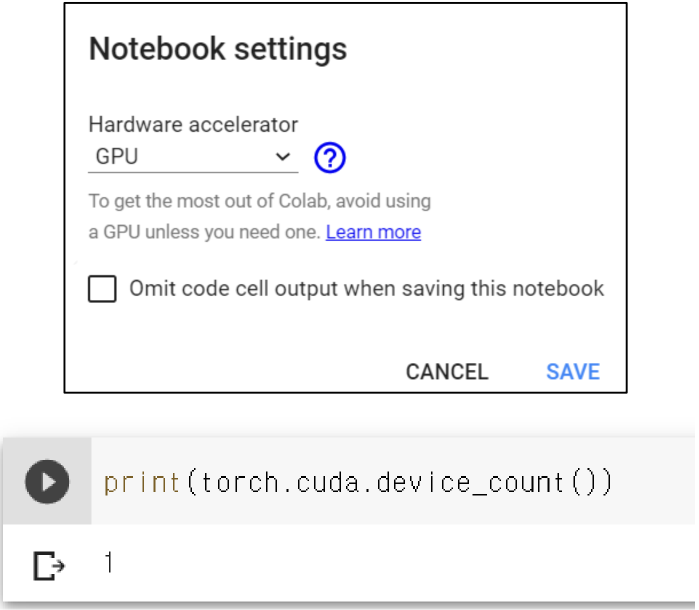
- .from_numpy: converting method from tensor to numpy

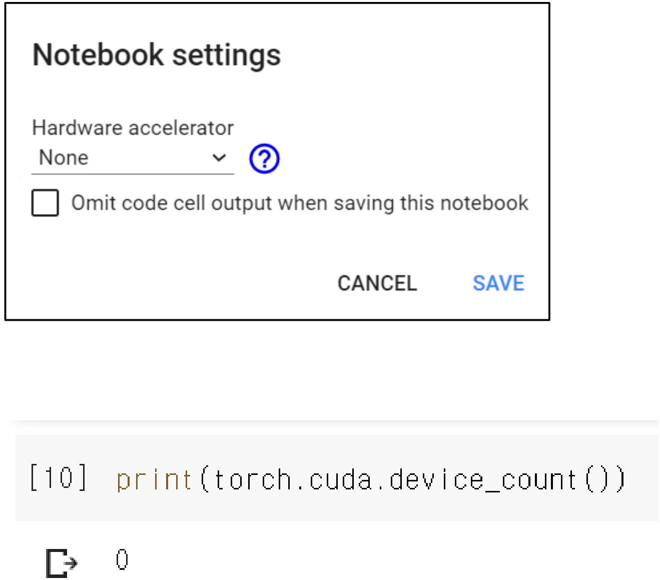
```
[33] a = np.random.randn(5) # generate a random numpy array
     a_pt = torch.from_numpy(a) # convert numpy array to a tensor
     print(a)
     print(type(a), type(a_pt))

[-0.4078755   0.12263927 -0.62535906  2.04968161  0.74182697]
<class 'numpy.ndarray'> <class 'torch.Tensor'>
```

Google Colab

CUDA Support

- .from_numpy: converting method from tensor to numpy



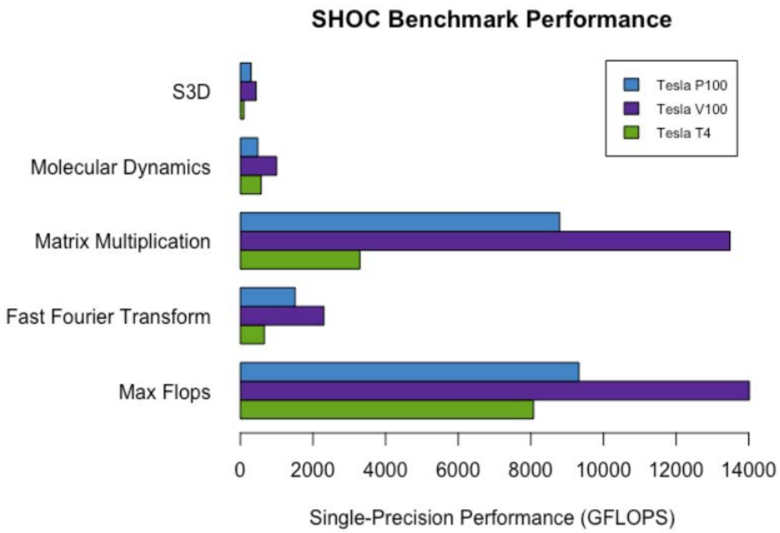


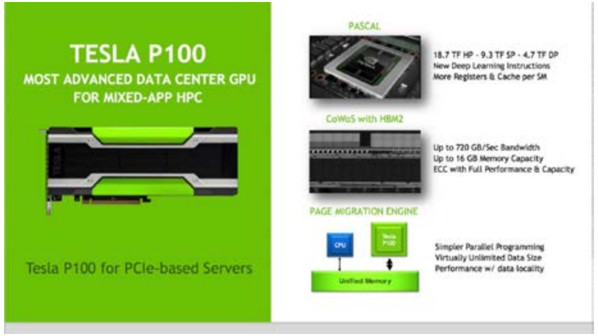
Google Colab

CUDA Support
- You may get P100 or T4

```
[46] print(torch.cuda.get_device_name(0))
```

```
Tesla P100-PCIE-16GB
```





T4

Google Colab

CUDA Support

- the selected GPU device can be changed with torch.cuda.device context manager

```
[47] # Assign cuda GPU located at location '0' to a variable
     cuda0 = torch.device('cuda:0')
     #performing the addition on GPU
     a = torch.ones(3, 2, device=cuda0) # creating a tensor 'a' on GPU
     b = torch.ones(3, 2, device=cuda0) # create a tensor 'b' on GPU
     c = a + b
     print(c)
```

```
tensor([[2., 2.],
        [2., 2.],
        [2., 2.]], device='cuda:0')
```

Google Colab

CUDA Support

- .cpu() : moves the results from GPU RAM to CPU RAM

```
[48] # move the result to CPU
     c = c.cpu()
     print(c)

tensor([[2., 2.],
        [2., 2.],
        [2., 2.]])
```

Google Colab

Automatic Differentiation (with autograd package)

- First, we create a tensor with requires_grad parameter set to True because we want to track all the operations performing on that tensor

```
[50] # create a tensor with requires_grad = True
     x = torch.ones([3,2], requires_grad = True)
     print(x)

     tensor([[1., 1.],
             [1., 1.],
             [1., 1.]], requires_grad=True)
```

Google Colab

Automatic Differentiation (with autograd package)

- grad_fn : a tensor type supporting to take derivative automatically

```
[51] y = x + 5 # tensor addition
     print(y) # check the result

     z = y*y + 1
     print(z)
     t = torch.sum(z) # add all the values in z
     print(t)
```

```
tensor([[6., 6.],
        [6., 6.],
        [6., 6.]], grad_fn=<AddBackward0>)
tensor([[37., 37.],
        [37., 37.],
        [37., 37.]], grad_fn=<AddBackward0>)
tensor(222., grad_fn=<SumBackward0>)
```

Google Colab

Automatic Differentiation (with autograd package)

- .backward(): performs a back-propagation (a way of updating the parameters in deep neural networks)
- param.grad : the value of partial derivative of output with respect to param

```
[52] t.backward() # perform backpropagation but pytorch will not print any output
     print(x.grad) # d(t)/dx = 2y + 1 at x = 1 and y = 6, where y = x + 5
                   # where t = sum(y*y), and y = x + 5

tensor([[12., 12.],
        [12., 12.],
        [12., 12.]])
```

Google Colab

Conclusion in Google Colab Section

- Google Colab & PyTorch is easy, powerful and for free
- PyTorch is based on tensor which has similar usage to narray in Numpy
- Autograd in PyTorch is one of the most powerful tool that allows to take derivative of variables automatically

Next Topics

- Linear, Logistic, softmax models
- DNN: Deep Neural Net
- CNN: Convolutional Neural Net
- Write everything in PyTorch

WHAT
NEXT

Lecture 2:
Linear Model