

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>

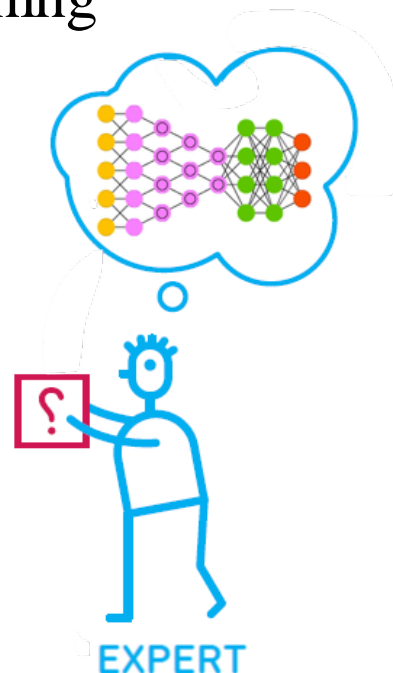


Goals

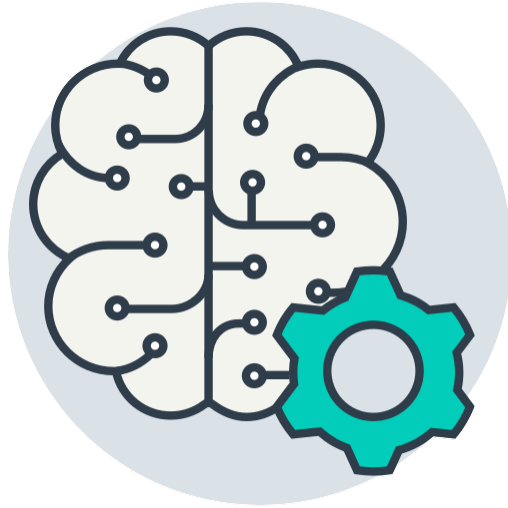
- Basic understanding of machine learning/deep learning
- PyTorch implementation skills

- Zero to All!

- Basic algebra + probability
- Basic python



What is ML?



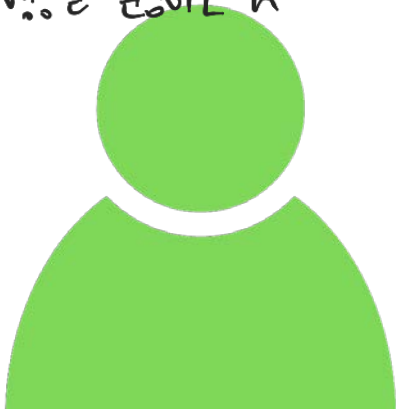
What is Human Intelligence?



What is Human Intelligence?

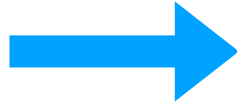
What to eat for lunch?

다양한 정보를 센서를 통해 감지하고
처리해서 행동을 결정하는 것



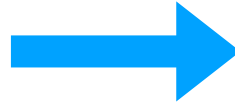
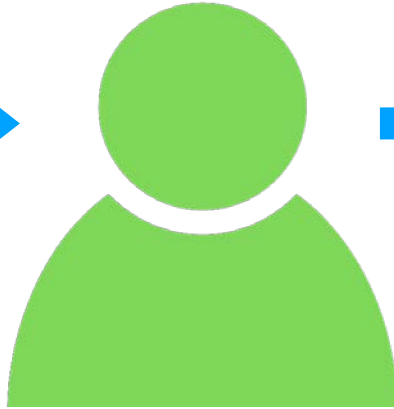
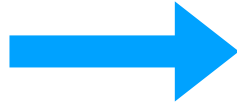
What is Human Intelligence?

What to eat for lunch?



What is Human Intelligence?

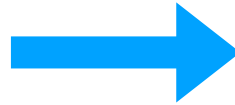
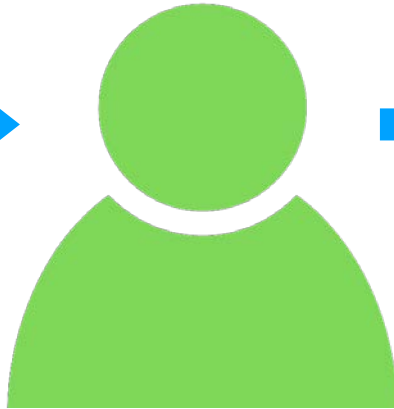
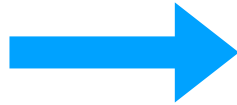
What to eat for lunch?



Infer

What is Human Intelligence?

What to dress?



Infer

What is Human Intelligence?

What is this picture?

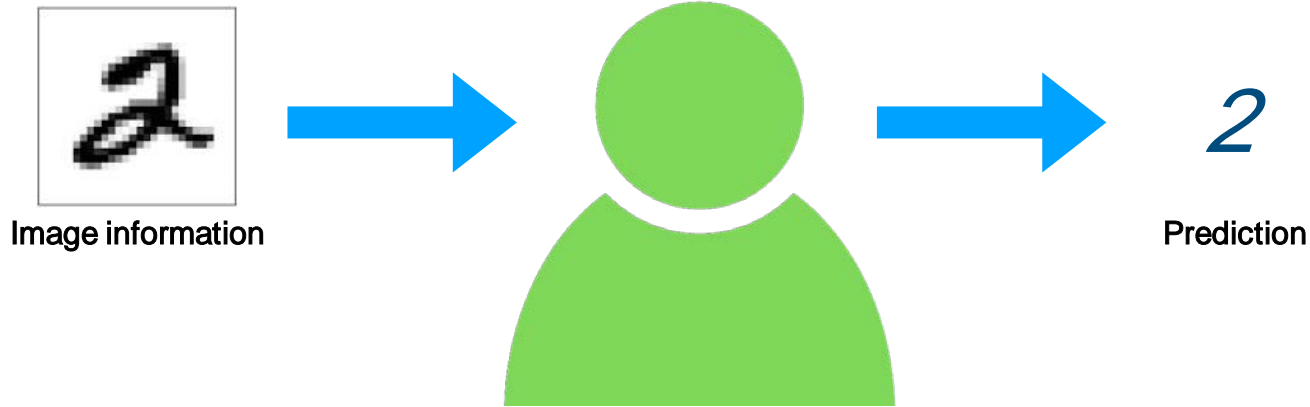


Image information



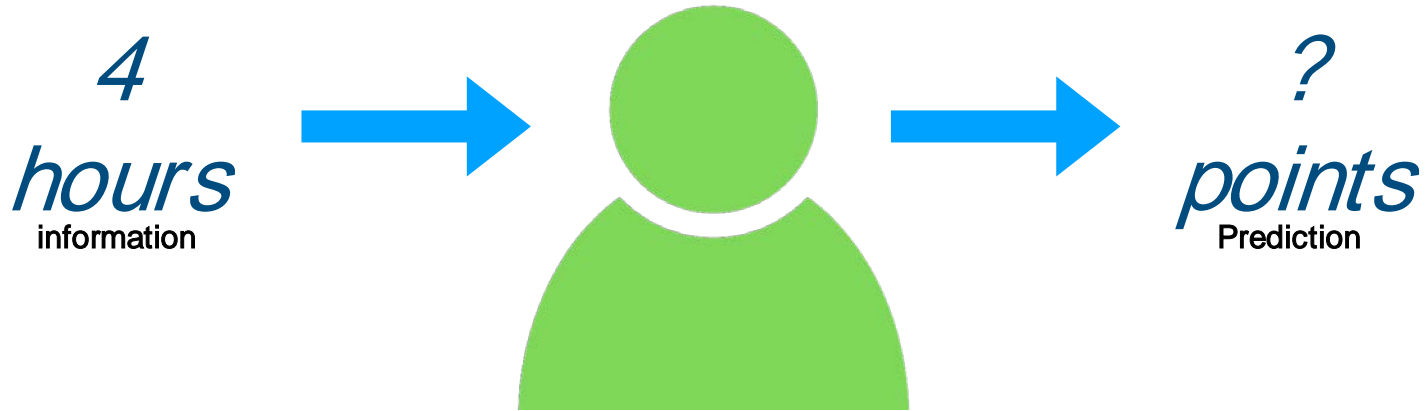
What is Human Intelligence?

What is this number?



What is Human Intelligence?

What would be the grade if I study 4 hours?



Machine Learning

What to dress?

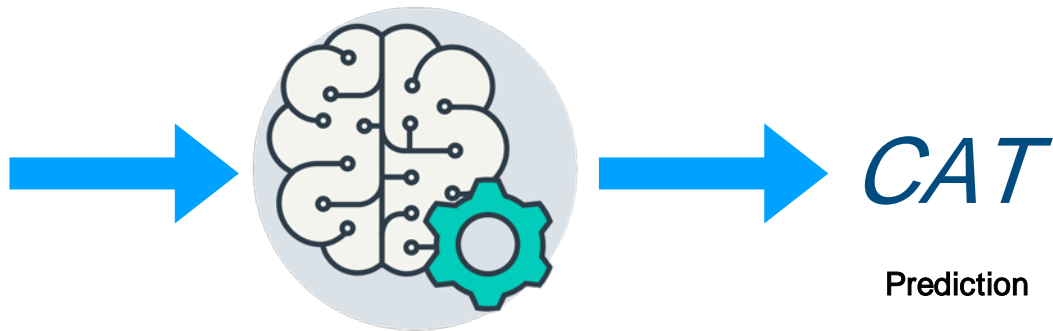


Machine Learning

What is this picture?

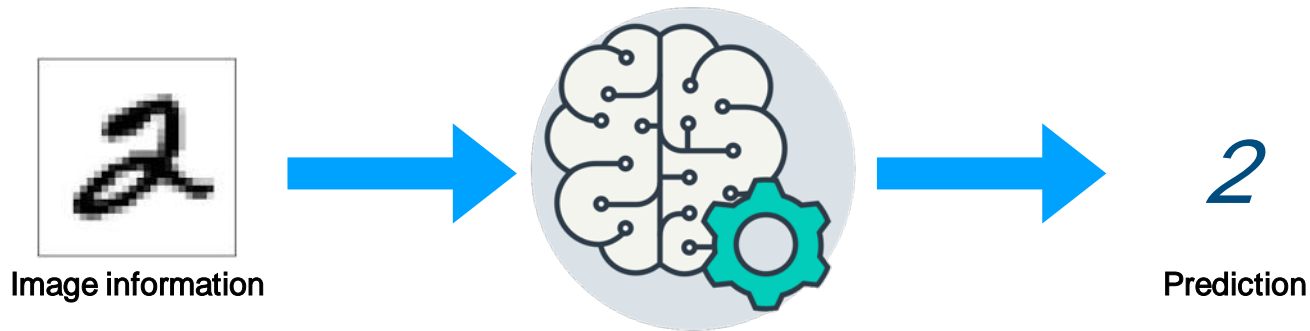


Image information



Machine Learning

What is this number?



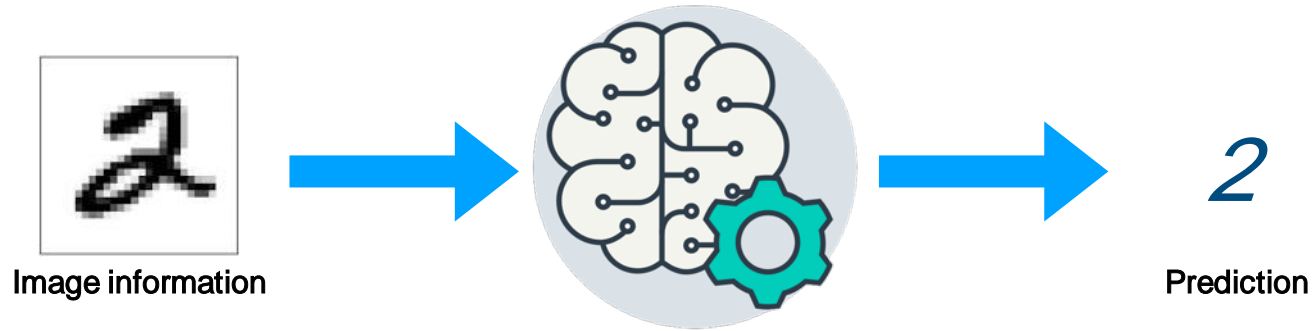
Machine Learning

What would be the grade if I study 4 hours?



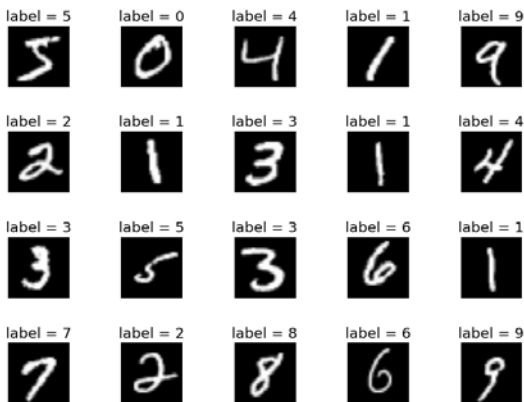
Machine Learning

Machine needs lots of training



Machine Learning

Machine needs lots of training



*Labeled
dataset*



training

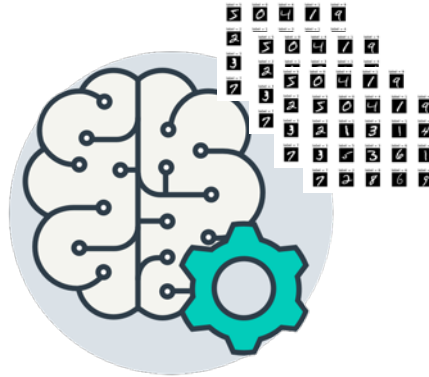


Model

이런 식으로
입력 → 레이블
||
입력에 대한 목표값

Machine Learning

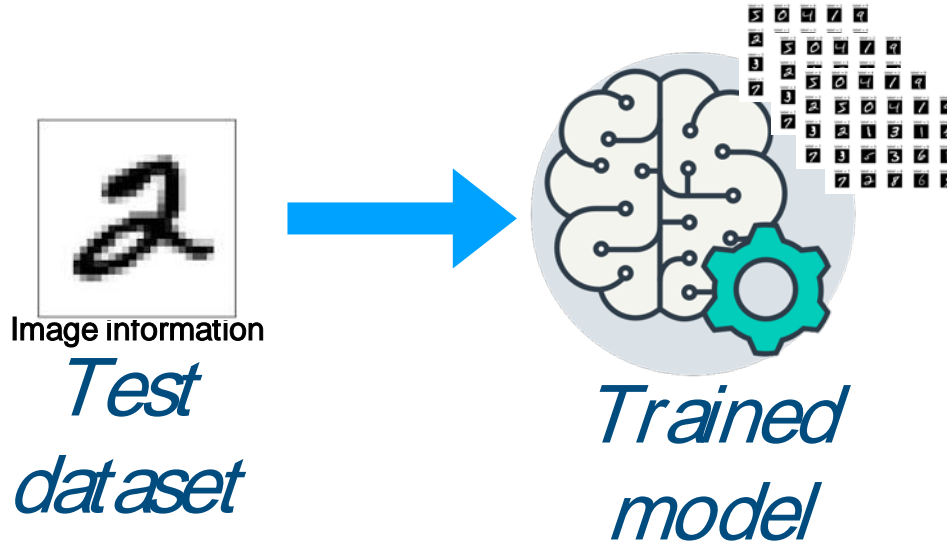
Predict (test) with trained model



*Trained
model*

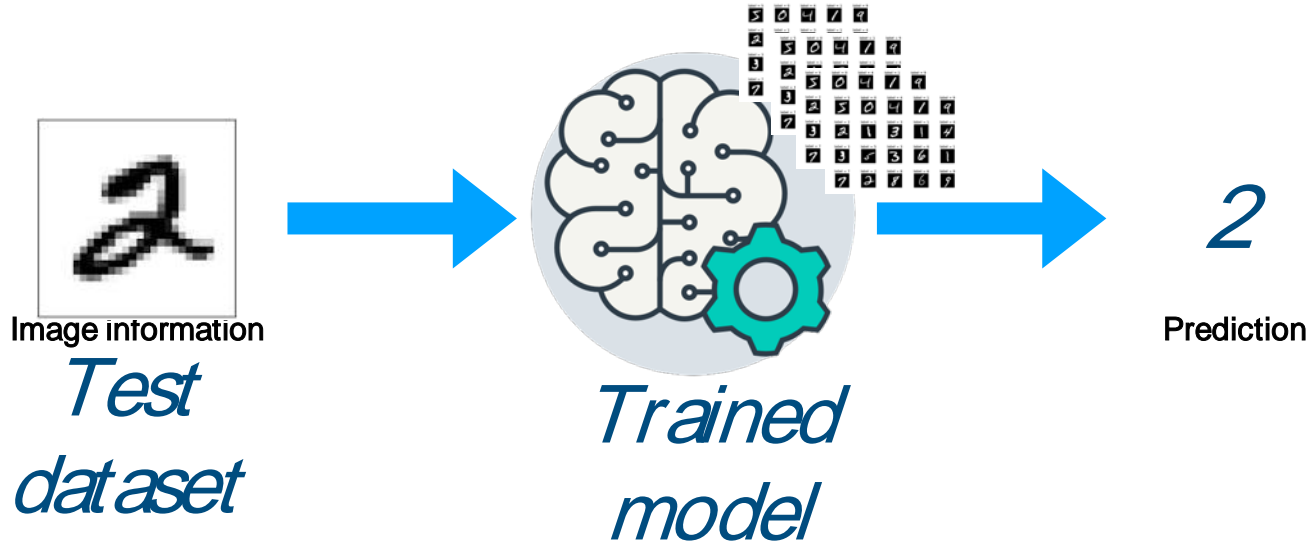
Machine Learning

Predict (test) with trained model



Machine Learning

Predict (test) with trained model



Machine Learning

What would be the grade if I study 4 hours?

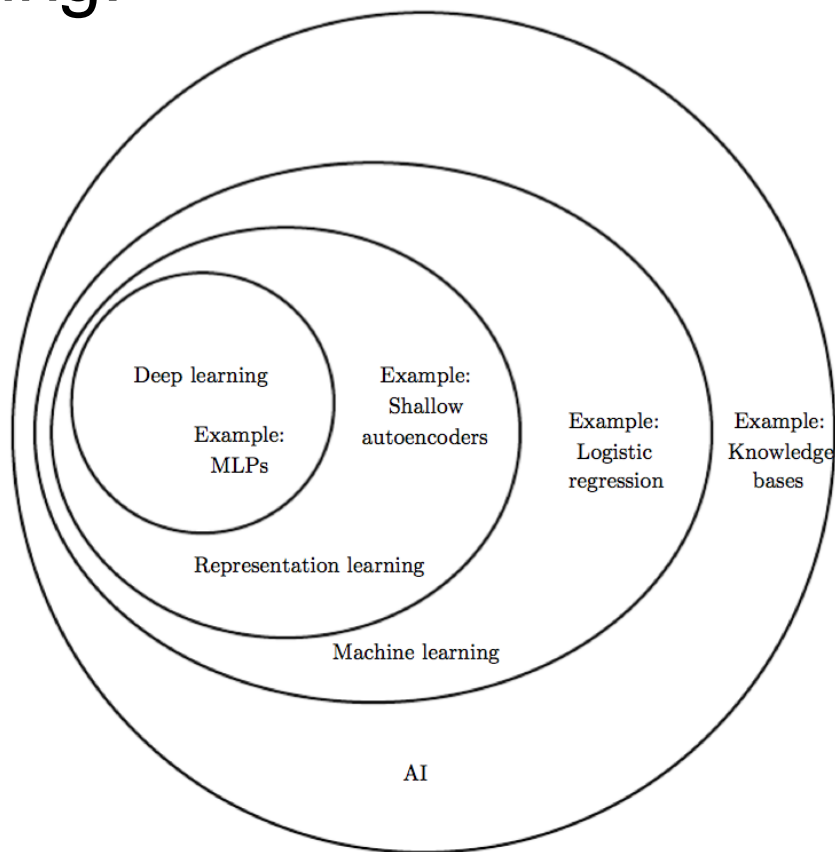


Hours (x)	Points (y)
1	2
2	4
3	6
4	?

Training dataset 훈련 집합

Test dataset 테스트 집합

Deep Learning?



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



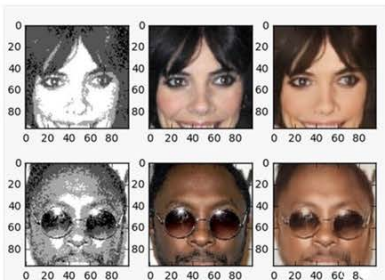
Artistic Style Transfer

Applying The Style Of An Artwork To A Photo



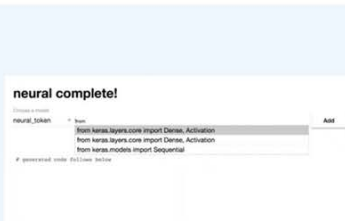
Predicting Cardiac Abnormalities

Using Phonocardiogram (PCG) Data



Creating Photorealistic Images

Improving Images From A Gameboy Camera

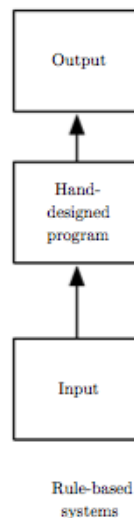


Code Autocompletion

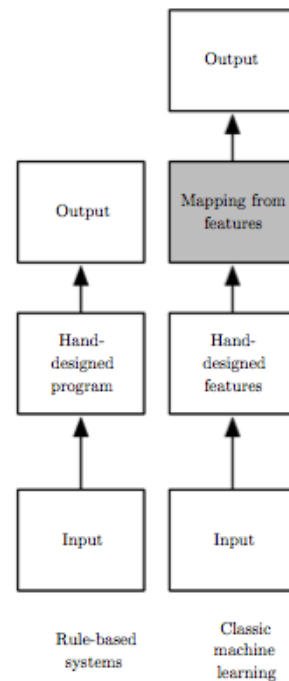
NN That Autocompletes NN Code

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

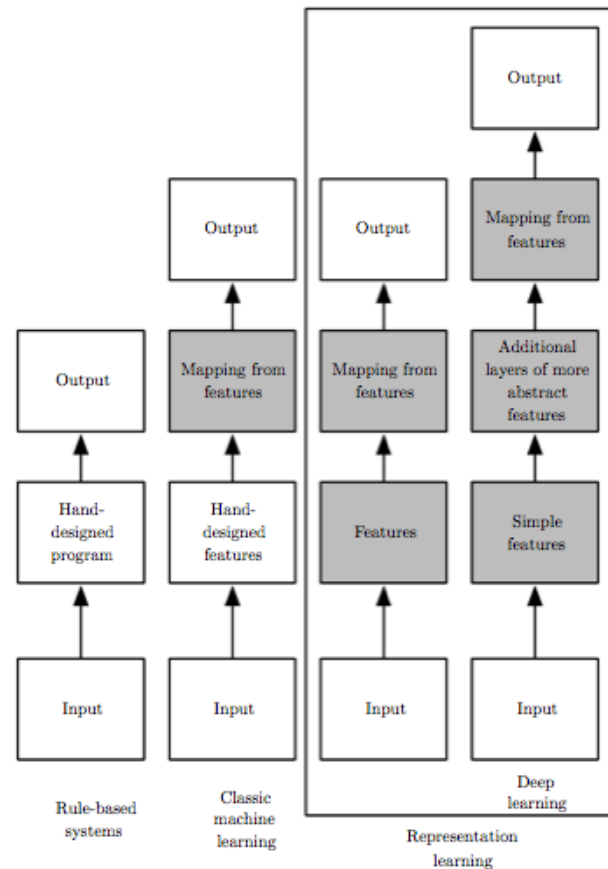
Why We Care as Developer?



Why We Care as Developer?

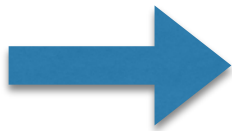
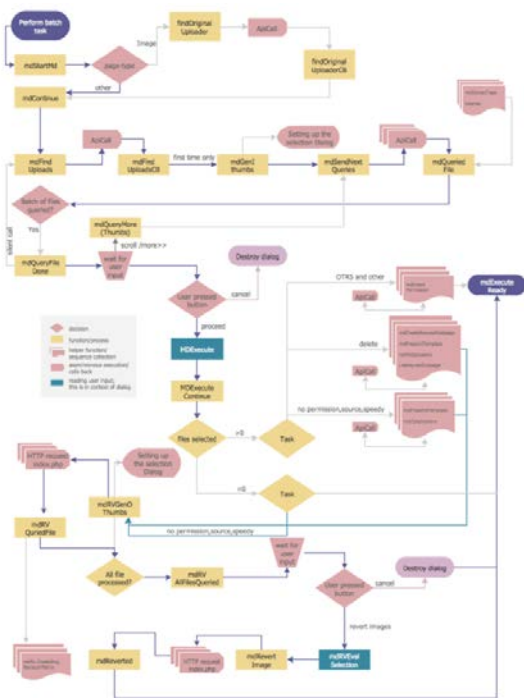


Why We Care as Developer?



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

Gradient 자동 계산

Why PYTORCH

- More Pythonic (imperative)
 - Flexible
 - Intuitive and cleaner code
 - Easy to debug
- More Neural Networkic → 다른 구조처럼 코딩 가능
 - 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
```


Google Colab



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

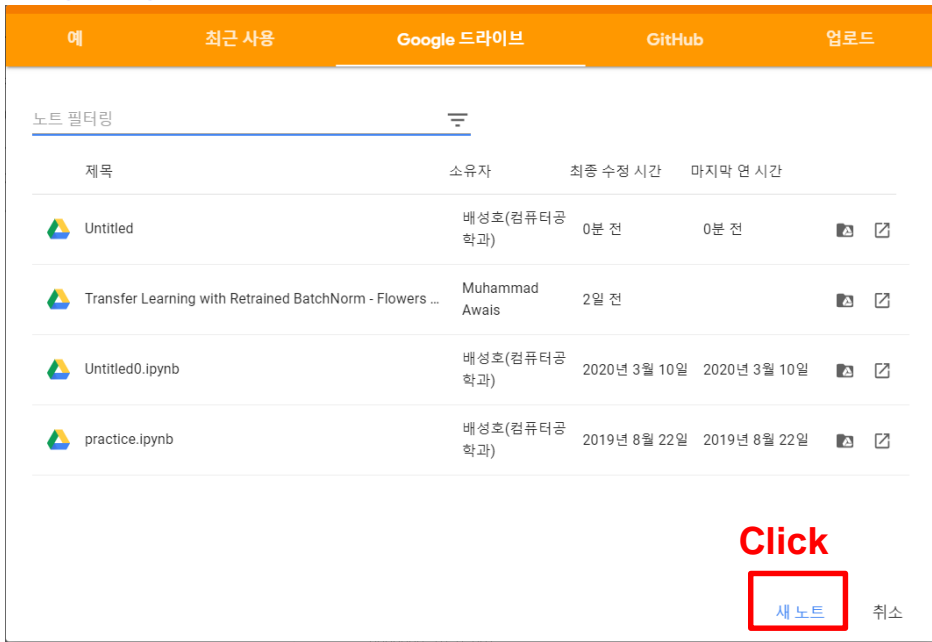
- Notebook based on Jupyter/iPython
- Free TPU (Tensor Processing Unit) *supported*
- Python 2.7 and Python 3.6 *supported*
- Tensorflow, Scikit-learn, Matplotlib, PyTorch, etc. *installed*
- Bash command *supported*
- Google Drive, GitHub Repo, GitHub Gist, etc. *interlocked*

Google Colab



How to use:

- 1. Enter <https://colab.research.google.com/>
- 2. Click (New Note)
- Note: 12 hours duration

A screenshot of the Google Colab web interface. At the top, there are tabs for '예' (Example), '최근 사용' (Recent), 'Google 드라이브' (Google Drive), 'GitHub', and '업로드' (Upload). Below the tabs is a table of notebooks. The table has columns for '제목' (Title), '소유자' (Owner), '최종 수정 시간' (Last modified), and '마지막 연 시간' (Last sync). The notebooks listed are 'Untitled', 'Transfer Learning with Retrained BatchNorm - Flowers ...', 'Untitled0.ipynb', and 'practice.ipynb'. At the bottom right, there is a red box containing the text '새 노트' (New Note) and a '취소' (Cancel) button. Above the red box, the word 'Click' is written in red.

제목	소유자	최종 수정 시간	마지막 연 시간
Untitled	배성호(컴퓨터공학과)	0분 전	0분 전
Transfer Learning with Retrained BatchNorm - Flowers ...	Muhammad Awais	2일 전	
Untitled0.ipynb	배성호(컴퓨터공학과)	2020년 3월 10일	2020년 3월 10일
practice.ipynb	배성호(컴퓨터공학과)	2019년 8월 22일	2019년 8월 22일

Click

새 노트 취소

Google Colab



How to use:

- 3. Change the runtime-type



TPUs in Colab

File Edit View Insert Runtime Tools Help

Table of contents

<> | TPU in Colab

License

Enabling and testing the

Input data

Model

Training

Predictions

Save and re-loading our

Next steps

+ Section

- Run all Ctrl+F9
- Run before Ctrl+F8
- Run the focused cell Ctrl+Enter
- Run selection Ctrl+Shift+Enter
- Run after Ctrl+F10
- Interrupt execution Ctrl+M
- Restart runtime Ctrl+M
- Restart and run all
- Factory reset runtime
- Change runtime type**
- Manage sessions
- View runtime logs

Copy to Drive



ll work through training
d TPUs. Our model will t
ion, rose, sunflower, or t
framework, new to TPUs

Notebook settings

Hardware accelerator

- None
- GPU
- TPU



Output when saving this notebook

CANCEL

SAVE



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
```

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]])
```


Mathematical operations

- (_) as postfix: in-place operation

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

# add two tensors
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)
```

$y = y + x$

```
↳ 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

Notebook settings

Hardware accelerator
GPU ?

To get the most out of Colab, avoid using a GPU unless you need one. [Learn more](#)

☐ Omit code cell output when saving this notebook

CANCEL SAVE

```
▶ print(torch.cuda.device_count())
```

1

Notebook settings

Hardware accelerator
None ?

☐ Omit code cell output when saving this notebook

CANCEL SAVE

```
[10] print(torch.cuda.device_count())
```

0

Google Colab

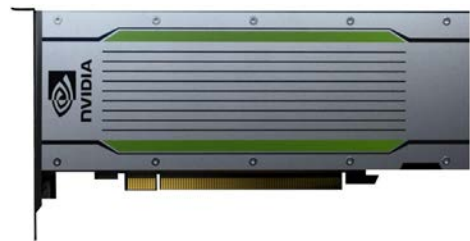
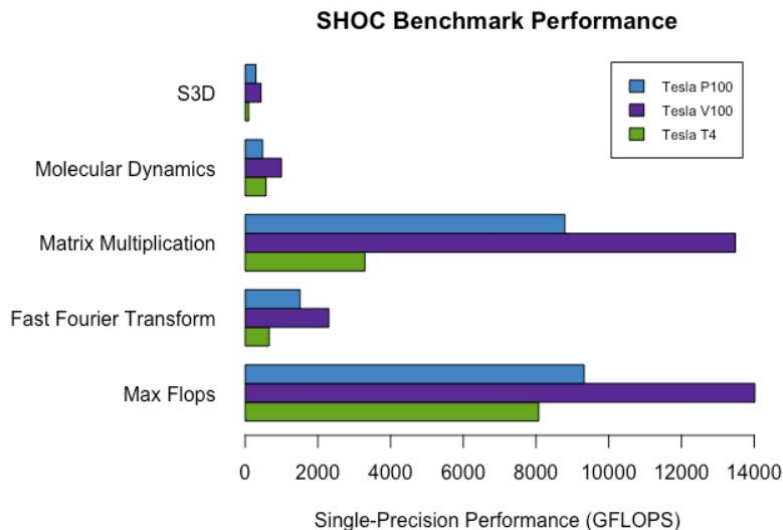


CUDA Support

- You may get P100 or T4

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

↳ Tesla P100-PCI-E-16GB



T4

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 [ndarray](#) 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