cs231n Lecture 8

Deep Learning Software

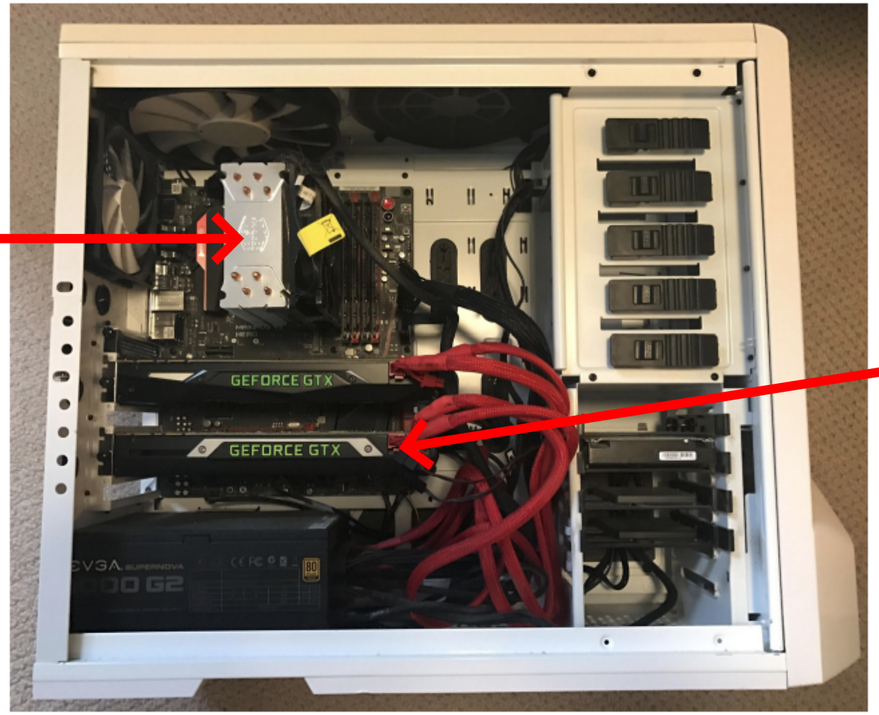
2176410 한현경

Contents

01. CPU vs. GPU

CPU

- Central Processing Unit



GPU

- Graphics Processing Unit
- Grapics Card
- takes up a lot of power
- NVIDIA vs. AMD?
    - 딥러닝에서는 주로 NVIDIA 사용

01. CPU vs. GPU

CPU vs. GPU in practice

	# Cores	Clock Speed	Memory	Price
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.4 GHz	Shared with system	$339
CPU (Intel Core i7-6950X)	10 (20 threads with hyperthreading)	3.5 GHz	Shared with system	$1723
GPU (NVIDIA Titan Xp)	3840	1.6 GHz	12 GB GDDR5X	$1200
GPU (NVIDIA GTX 1070)	1920	1.68 GHz	8 GB GDDR5	$399

CPU: Fewer cores, but each core is much faster and much more capable; great at sequential tasks

CPU

- core 수가 적지만 각각의 core가 독립적으로 작동
- multithreading을 사용하면 최대 20가지의 task 처리 가능
- 대부분의 메모리를 RAM에서 끌어 씀 (RAM: 8, 12, 16, 32GB)

GPU: More cores, but each core is much slower and "dumber"; great for parallel tasks

GPU

- core 수가 훨씬 많지만 각각의 clock speed 가 느림
- 독립적x. 많은 core가 하나의 task.
- 행렬곱 연산과 같은 병렬 처리에 유리
- 자체적으로 RAM을 가지고 있음 (RAM-GPU통신 bottleneck 문제)

01. CPU vs. GPU

Programming GPUs

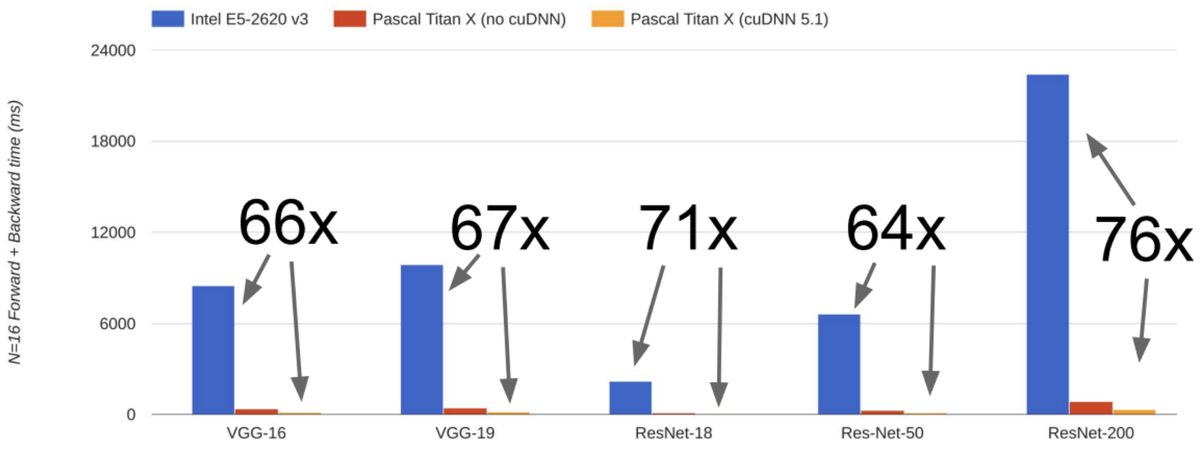
- CUDA(NVIDIA only)
    - Write C-like codes that runs directly on the GPU -> 어려움!
    - Higher-level APIs : **cuBLAS, cuFFT, cuDNN**, etc
    - **cuDNN** : NVIDIA 하드웨어에서 잘 돌아가는 라이브러리(convolution, forward/backward pass, batch normalization, recurrent networks, etc)를 NVIDIA에서 만들어서 공개한 것.

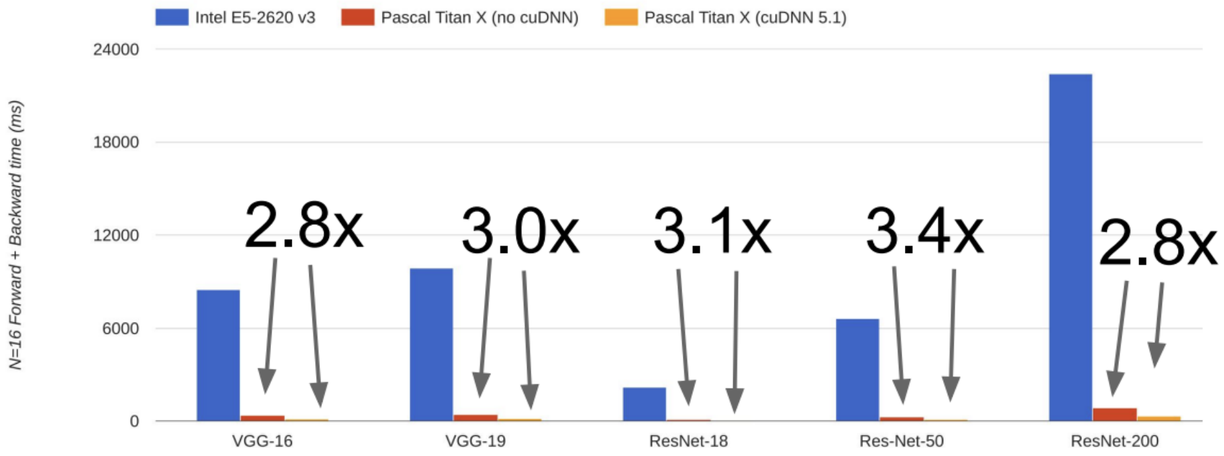
- OpenCL
    - CUDA와 비슷한데 AMD 하드웨어나 CPU에서도 돌아감.
    - usually slower

- Udacity

01. CPU vs. GPU

CPU vs. GPU in practice





속도에 대한 성능 비교:

GPU > CPU

cuDNN > unoptimized CUDA
(3x performance boost)

02. Deep Learning Frameworks

Today

A bit about these

Caffe
(UC Berkeley)

Caffe2
(Facebook)

Torch
(NYU / Facebook)

PyTorch
(Facebook)

Theano
(U Montreal)

TensorFlow
(Google)

Mostly these

Paddle
(Baidu)

CNTK
(Microsoft)

MXNet
(Amazon)

Developed by U Washington, CMU, MIT, Hong Kong U, etc but main framework of choice at AWS

And others...

02. Deep Learning Frameworks

왜 필요한가?

(1) Easily build computational graphs

(2) Easily compute gradients in computational graphs

(3) Run it efficiently on GPU(wrap cuDNN, cuBLAS, etc)

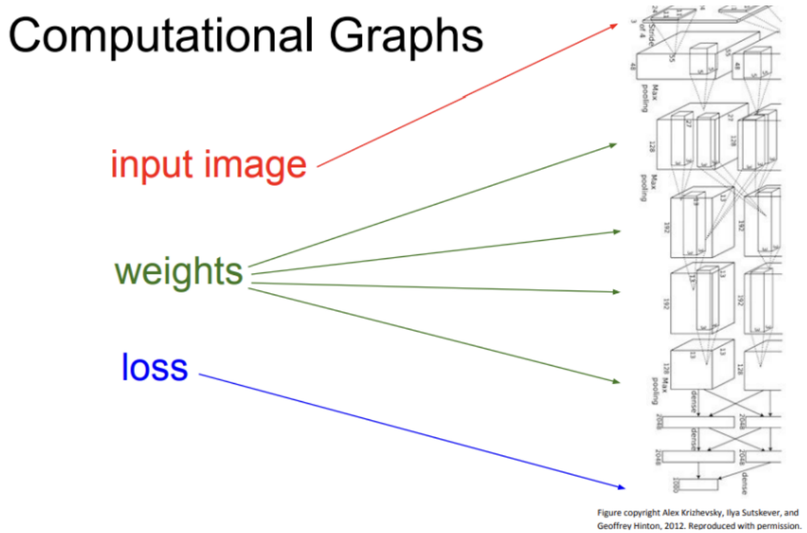


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

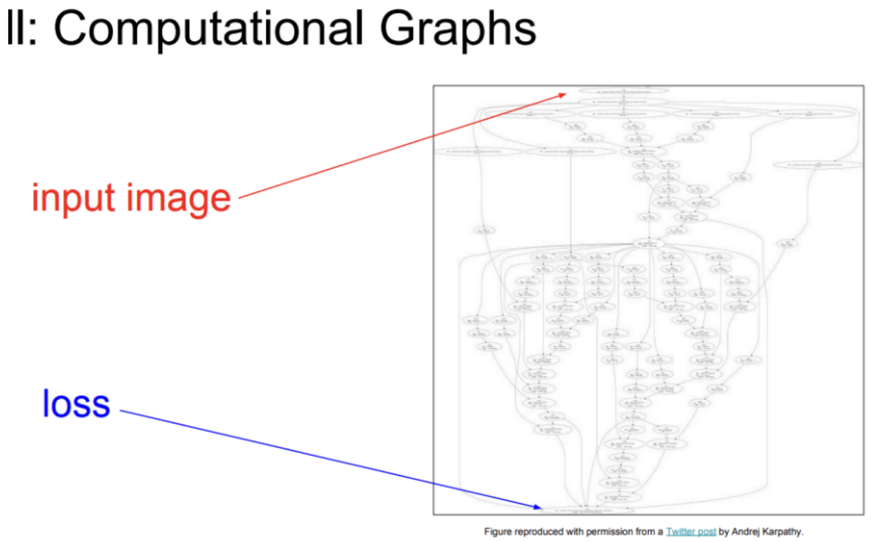
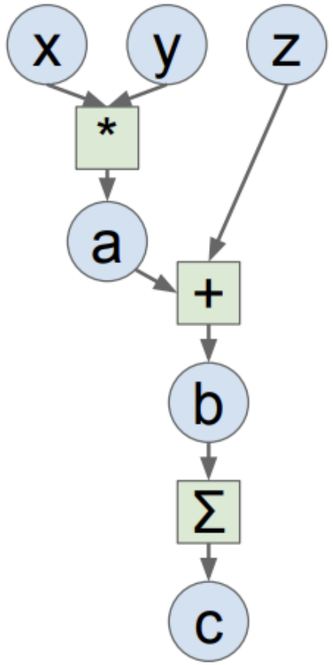


Figure reproduced with permission from a Twitter post by Andrej Karpathy.

02. Deep Learning Frameworks

Computational Graphs



Create forward computational graph

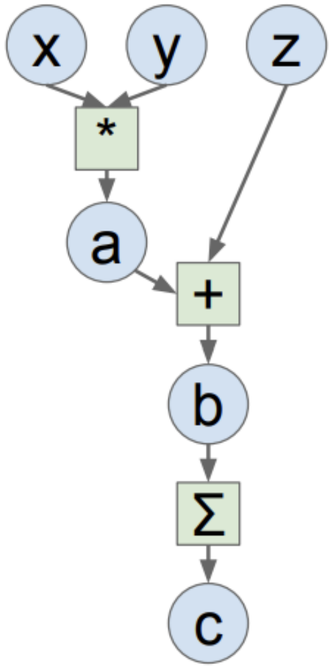
TensorFlow

```
# Basic computational graph
import numpy as np
np.random.seed(0)
import tensorflow as tf

N, D = 3, 4

x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)
z = tf.placeholder(tf.float32)

a = x * y
b = a + z
c = tf.reduce_sum(b)

grad_x, grad_y, grad_z = tf.gradients(c, [x, y, z])

with tf.Session() as sess:
    values = {
        x: np.random.randn(N, D),
        y: np.random.randn(N, D),
        z: np.random.randn(N, D),
    }
    out = sess.run([c, grad_x, grad_y, grad_z],
                   feed_dict=values)
    c_val, grad_x_val, grad_y_val, grad_z_val = out
```


(1) numpy와 유사한 forward pass 코드 작성

02. Deep Learning Frameworks

Computational Graphs



Ask TensorFlow to compute gradients

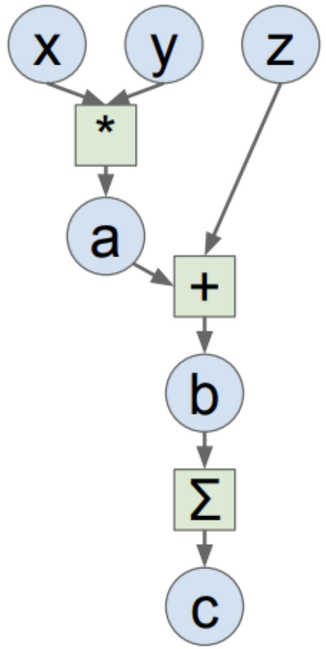
TensorFlow

```
# Basic computational graph
import numpy as np
np.random.seed(0)
import tensorflow as tf

N, D = 3, 4

x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)
z = tf.placeholder(tf.float32)

a = x * y
b = a + z
c = tf.reduce_sum(b)

grad_x, grad_y, grad_z = tf.gradients(c, [x, y, z])

with tf.Session() as sess:
    values = {
        x: np.random.randn(N, D),
        y: np.random.randn(N, D),
        z: np.random.randn(N, D),
    }
    out = sess.run([c, grad_x, grad_y, grad_z],
                   feed_dict=values)
    c_val, grad_x_val, grad_y_val, grad_z_val = out
```


(2) Tensorflow가 gradient를 직접 계산해줌.
(엔지니어가 backward pass를 작성하지 않아도 됨.)

02. Deep Learning Frameworks

Computational Graphs



Tell TensorFlow to run on **GPU**

TensorFlow

```
import numpy as np
np.random.seed(0)
import tensorflow as tf

N, D = 3000, 4000

with tf.device('/gpu:0'):
    x = tf.placeholder(tf.float32)
    y = tf.placeholder(tf.float32)
    z = tf.placeholder(tf.float32)

    a = x * y
    b = a + z
    c = tf.reduce_sum(b)

grad_x, grad_y, grad_z = tf.gradients(c, [x, y, z])

with tf.Session() as sess:
    values = {
        x: np.random.randn(N, D),
        y: np.random.randn(N, D),
        z: np.random.randn(N, D),
    }
    out = sess.run([c, grad_x, grad_y, grad_z],
                   feed_dict=values)
    c_val, grad_x_val, grad_y_val, grad_z_val = out
```


(3) 코드 한 줄로 CPU/GPU에서 실행하게 만들 수 있음.

02. Deep Learning Frameworks

Numpy

```
import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)

grad_c = 1.0
grad_b = grad_c * np.ones((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a * y
grad_y = grad_a * x
```

TensorFlow

```
import numpy as np
np.random.seed(0)
import tensorflow as tf

N, D = 3, 4

with tf.device('/gpu:0'):
    x = tf.placeholder(tf.float32)
    y = tf.placeholder(tf.float32)
    z = tf.placeholder(tf.float32)

    a = x * y
    b = a + z
    c = tf.reduce_sum(b)

grad_x, grad_y, grad_z = tf.gradients(c, [x, y, z])

with tf.Session() as sess:
    values = {
        x: np.random.randn(N, D),
        y: np.random.randn(N, D),
        z: np.random.randn(N, D),
    }
    out = sess.run([c, grad_x, grad_y, grad_z],
                   feed_dict=values)
    c_val, grad_x_val, grad_y_val, grad_z_val = out
```

PyTorch

```
import torch
from torch.autograd import Variable

N, D = 3, 4

x = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)
y = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)
z = Variable(torch.randn(N, D).cuda(),
             requires_grad=True)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()

print(x.grad.data)
print(y.grad.data)
print(z.grad.data)
```

PyTorch도
거의 비슷함!

numpy와 비슷하고, gradient도 알아서 계산해주고, GPU에서도 돌아간다!

02. Deep Learning Frameworks

Tensorflow

```
import numpy as np
import tensorflow as tf
```

(Assume imports at the top of each snipppet)

TensorFlow: Neural Net

(1) computational graph 정의

First **define** computational graph

(2) data를 넣고 graph 실행

Then **run** the graph many times

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.placeholder(tf.float32, shape=(D, H))
w2 = tf.placeholder(tf.float32, shape=(H, D))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))

grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])
```

```
with tf.Session() as sess:
    values = {x: np.random.randn(N, D),
              w1: np.random.randn(D, H),
              w2: np.random.randn(H, D),
              y: np.random.randn(N, D),}
    out = sess.run([loss, grad_w1, grad_w2],
                   feed_dict=values)
    loss_val, grad_w1_val, grad_w2_val = out
```

02. Deep Learning Frameworks

Tensorflow

TensorFlow: Neural Net

input값(data) 받을 자리 만들기

Create **placeholders** for input x, weights w1 and w2, and targets y

x와 w1의 행렬곱 연산 수행(tf.matmul)
+ ReLU 함수 구현(tf.maximum)

loss계산, w1와 w2에 대한 gradient를 tensorflow가 알아서 계산

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.placeholder(tf.float32, shape=(D, H))
w2 = tf.placeholder(tf.float32, shape=(H, D))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))

grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])

with tf.Session() as sess:
    values = {x: np.random.randn(N, D),
              w1: np.random.randn(D, H),
              w2: np.random.randn(H, D),
              y: np.random.randn(N, D),}
    out = sess.run([loss, grad_w1, grad_w2],
                   feed_dict=values)
    loss_val, grad_w1_val, grad_w2_val = out
```

02. Deep Learning Frameworks

Tensorflow

TensorFlow:
Neural Net

input값(data) 받을 자리 만들기

Create **placeholders** for input x, weights w1 and w2, and targets y

x와 w1의 행렬곱 연산 수행(tf.matmul)
+ ReLU 함수 구현(tf.maximum)

loss계산, w1와 w2에 대한 gradient를 tensorflow가 알아서 계산

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.placeholder(tf.float32, shape=(D, H))
w2 = tf.placeholder(tf.float32, shape=(H, D))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))

grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])

with tf.Session() as sess:
    values = {x: np.random.randn(N, D),
              w1: np.random.randn(D, H),
              w2: np.random.randn(H, D),
              y: np.random.randn(N, D),}
    out = sess.run([loss, grad_w1, grad_w2],
                   feed_dict=values)
    loss_val, grad_w1_val, grad_w2_val = out
```

just 그래프 "구성".
아직 실행하지 않은 상태!

02. Deep Learning Frameworks

Tensorflow

TensorFlow: Neural Net

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.placeholder(tf.float32, shape=(D, H))
w2 = tf.placeholder(tf.float32, shape=(H, D))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))

grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])

with tf.Session() as sess:
    values = {x: np.random.randn(N, D),
              w1: np.random.randn(D, H),
              w2: np.random.randn(H, D),
              y: np.random.randn(N, D),}
    out = sess.run([loss, grad_w1, grad_w2],
                   feed_dict=values)
    loss_val, grad_w1_val, grad_w2_val = out
```

그래프를 실행시키는 session에 진입

Now done building our graph, so we enter a **session** so we can actually run the graph

그래프에 들어갈 실제 값 생성 (numpy array)

원하는 출력값(numpy array): loss, grad_w1, grad_w2
feed_dict를 통해 실제 값 전달

02. Deep Learning Frameworks

Tensorflow

TensorFlow: Neural Net

Problem: copying weights between CPU / GPU each step

Train the network: Run the graph over and over, use gradient to update weights

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.placeholder(tf.float32, shape=(D, H))
w2 = tf.placeholder(tf.float32, shape=(H, D))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))
grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])

with tf.Session() as sess:
    values = {x: np.random.randn(N, D),
              w1: np.random.randn(D, H),
              w2: np.random.randn(H, D),
              y: np.random.randn(N, D),}
    learning_rate = 1e-5
    for t in range(50):
        out = sess.run([loss, grad_w1, grad_w2],
                       feed_dict=values)
        loss_val, grad_w1_val, grad_w2_val = out
        values[w1] -= learning_rate * grad_w1_val
        values[w2] -= learning_rate * grad_w2_val
```


반복적으로 session.run을 호출해 loss와 grad를 계산

가중치를 업데이트하기 위해 그레디언트를 계산해서 수동으로 Gradient descent에 반영

반복문을 사용하고자 하는 경우

02. Deep Learning Frameworks

Tensorflow

TensorFlow: Optimizer

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.Variable(tf.random_normal((D, H)))
w2 = tf.Variable(tf.random_normal((H, D)))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff * diff, axis=1))

optimizer = tf.train.GradientDescentOptimizer(1e-5)
updates = optimizer.minimize(loss)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    values = {x: np.random.randn(N, D),
              y: np.random.randn(N, D),}
    losses = []
    for t in range(50):
        loss_val, _ = sess.run([loss, updates],
                               feed_dict=values)
```

Can use an **optimizer** to compute gradients and update weights

Remember to execute the output of the optimizer!

Fei-Fei Li & Justin Johnson & Serena Yeung Lecture 8 -5757 April 27, 2017

optimizer에 최적화 알고리즘이 내장됨.
명령어에 learning rate을 지정해주면 내부적으로 계산 후 update명령어에 의해 반복문이 돌 때마다 update

02. Deep Learning Frameworks

Tensorflow

TensorFlow: Loss

Use predefined common lossees

L2 loss를 직접 구하는 코드를 짜지 않아도 됨!

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.Variable(tf.random_normal((D, H)))
w2 = tf.Variable(tf.random_normal((H, D)))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
loss = tf.losses.mean_squared_error(y_pred, y)

optimizer = tf.train.GradientDescentOptimizer(1e-3)
updates = optimizer.minimize(loss)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    values = {x: np.random.randn(N, D),
              y: np.random.randn(N, D),}
    for t in range(50):
        loss_val, _ = sess.run([loss, updates],
                               feed_dict=values)
```


마찬가지로 loss도 직접 연산 알고리즘을 만들지 않고
tensor operations 사용

02. Deep Learning Frameworks

Keras: High-level Wrapper

딥러닝 아키텍처에서의
대부분의 세부사항을
대신 다뤄줌.

Tensorflow 추상화 패키지

Keras: High-Level Wrapper

Define model object as a sequence of layers

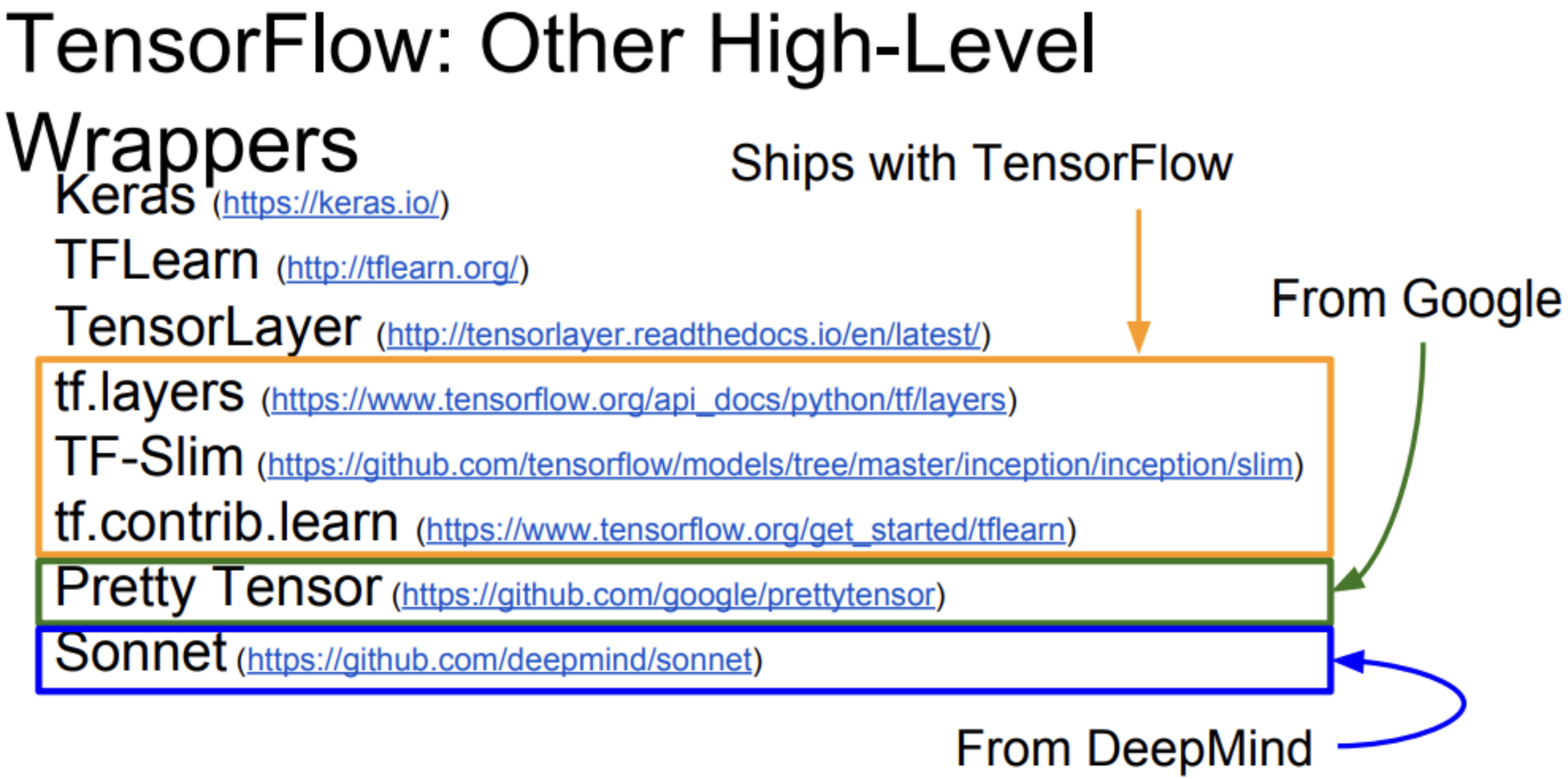
optimizer 사용

모델 / loss function specify

코드 한 줄로 train

```
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.optimizers import SGD

N, D, H = 64, 1000, 100

model = Sequential()
model.add(Dense(input_dim=D, output_dim=H))
model.add(Activation('relu'))
model.add(Dense(input_dim=H, output_dim=D))

optimizer = SGD(lr=1e0)
model.compile(loss='mean_squared_error',
              optimizer=optimizer)

x = np.random.randn(N, D)
y = np.random.randn(N, D)
history = model.fit(x, y, nb_epoch=50,
                    batch_size=N, verbose=0)
```

02. Deep Learning Frameworks



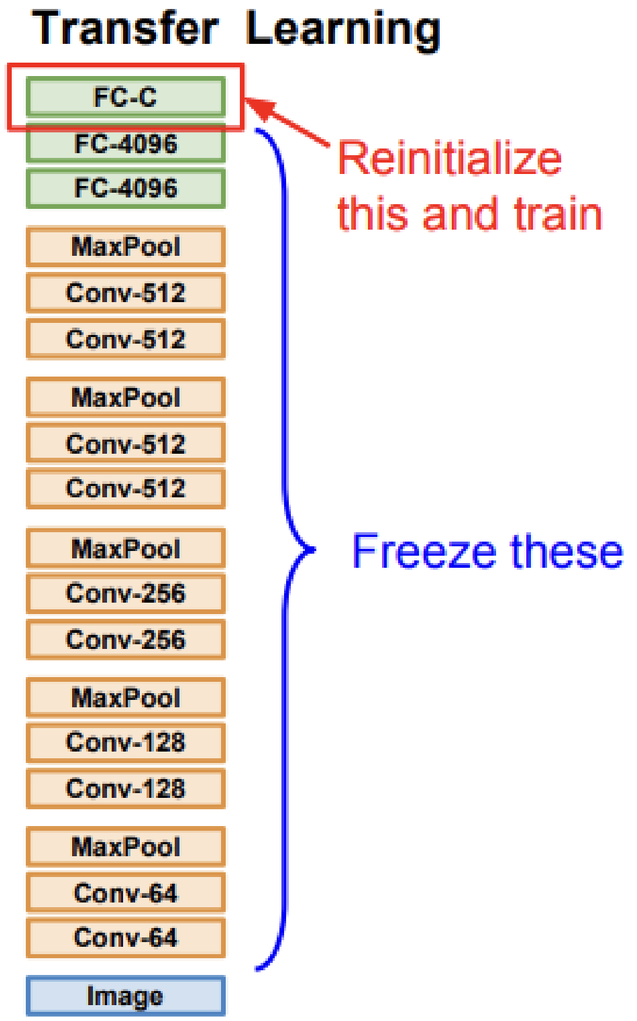
02. Deep Learning Frameworks

Pre-trained models

Examples

Classify images

```
from resnet50 import ResNet50
from keras.preprocessing import image
from imagenet_utils import preprocess_input, decode_predictions

model = ResNet50(weights='imagenet')

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
print('Predicted:', decode_predictions(preds))
# print: [[u'n02504458', u'African_elephant']]
```



02. Deep Learning Frameworks

Theano

Side Note: Theano

TensorFlow is similar in many ways to **Theano** (earlier framework from Montreal)

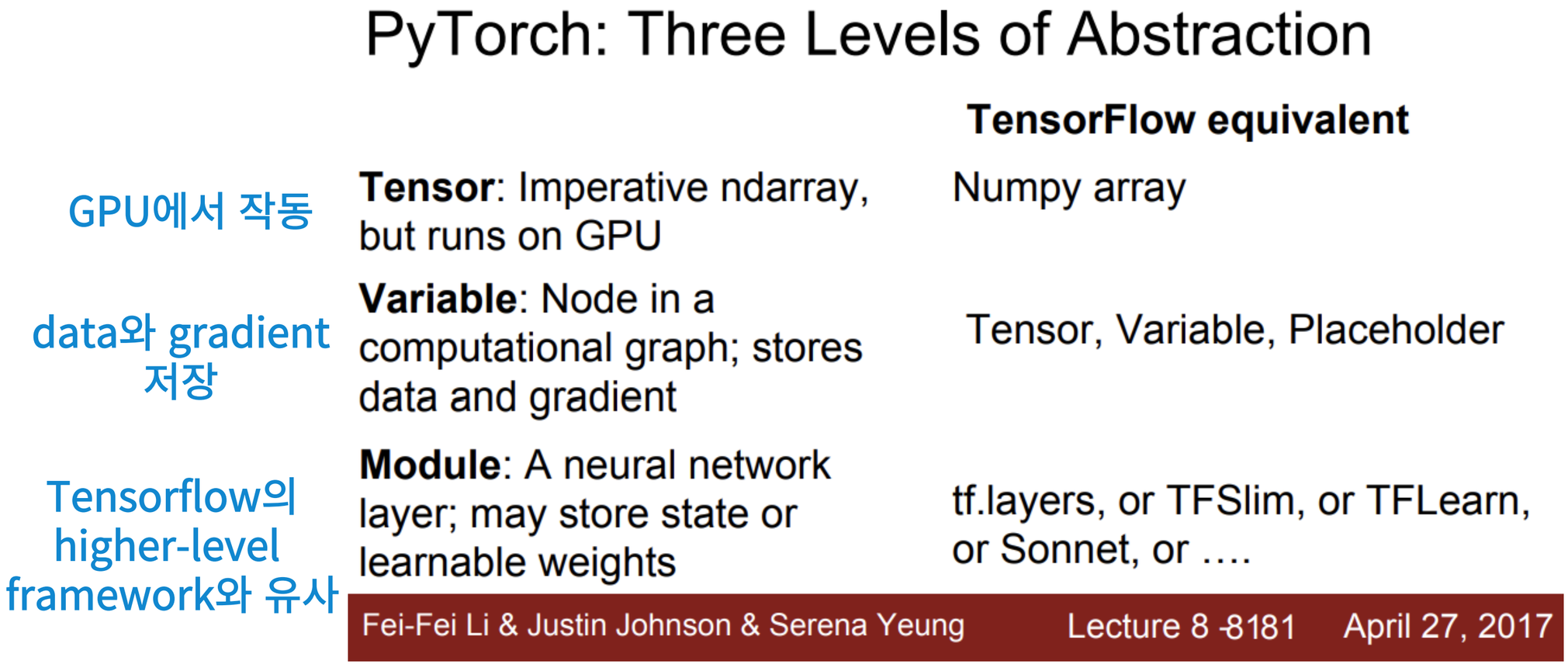
```
import theano
import theano.tensor as T

# Batch size, input dim, hidden dim, num classes
N, D, H, C = 64, 1000, 100, 10

x = T.matrix('x')
y = T.vector('y', dtype='int64')
w1 = T.matrix('w1')
w2 = T.matrix('w2')

# Forward pass: Compute scores
a = x.dot(w1)
a_relu = T.nnet.relu(a)
scores = a_relu.dot(w2)

# Forward pass: compute softmax loss
probs = T.nnet.softmax(scores)
loss = T.nnet.categorical_crossentropy(probs, y).mean()

# Backward pass: compute gradients
dw1, dw2 = T.grad(loss, [w1, w2])

f = theano.function(
      inputs=[x, y, w1, w2],
      outputs=[loss, scores, dw1, dw2],
    )
```

Fei-Fei Li & Justin Johnson & Serena Yeung Lecture 8 - 72 April 27, 2017

Tensorflow는 Theano에서 영감을 받았다! (코드도 비슷함)

02. Deep Learning Frameworks

PyTorch의 추상화



02. Deep Learning Frameworks

PyTorch: Tensors

tensor를 사용해 input값 받기
Create random tensors for data and weights

forward pass

backward pass

gradient와 learning rate을 통해 가중치 계산

```
import torch

dtype = torch.FloatTensor

N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in).type(dtype)
y = torch.randn(N, D_out).type(dtype)
w1 = torch.randn(D_in, H).type(dtype)
w2 = torch.randn(H, D_out).type(dtype)

learning_rate = 1e-6
for t in range(500):
    h = x.mm(w1)
    h_relu = h.clamp(min=0)
    y_pred = h_relu.mm(w2)
    loss = (y_pred - y).pow(2).sum()

    grad_y_pred = 2.0 * (y_pred - y)
    grad_w2 = h_relu.t().mm(grad_y_pred)
    grad_h_relu = grad_y_pred.mm(w2.t())
    grad_h = grad_h_relu.clone()
    grad_h[h < 0] = 0
    grad_w1 = x.t().mm(grad_h)

    w1 -= learning_rate * grad_w1
    w2 -= learning_rate * grad_w2
```

02. Deep Learning Frameworks

자동으로 gradient를 계산하는 함수

but 필요한 대부분의 연산들은 이미 구현되었을 것이기 때문에 autograd을 직접 사용할 일은 잘 없다!

PyTorch: Autograd

We will not want gradients (of loss) with respect to data

Do want gradients with respect to weights

```
import torch
from torch.autograd import Variable

N, D_in, H, D_out = 64, 1000, 100, 10
x = Variable(torch.randn(N, D_in), requires_grad=False)
y = Variable(torch.randn(N, D_out), requires_grad=False)
w1 = Variable(torch.randn(D_in, H), requires_grad=True)
w2 = Variable(torch.randn(H, D_out), requires_grad=True)

learning_rate = 1e-6
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    if w1.grad: w1.grad.data.zero_()
    if w2.grad: w2.grad.data.zero_()
    loss.backward()

    w1.data -= learning_rate * w1.grad.data
    w2.data -= learning_rate * w2.grad.data
```

02. Deep Learning Frameworks

PyTorch: optim

Use an **optimizer** for different update rules

```
import torch
from torch.autograd import Variable

N, D_in, H, D_out = 64, 1000, 100, 10

x = Variable(torch.randn(N, D_in))
y = Variable(torch.randn(N, D_out), requires_grad=False)

model = torch.nn.Sequential(
          torch.nn.Linear(D_in, H),
          torch.nn.ReLU(),
          torch.nn.Linear(H, D_out))
loss_fn = torch.nn.MSELoss(size_average=False)

learning_rate = 1e-4
optimizer = torch.optim.Adam(model.parameters(),
                             lr=learning_rate)
for t in range(500):
    y_pred = model(x)
    loss = loss_fn(y_pred, y)

    optimizer.zero_grad()
    loss.backward()

    optimizer.step()
```


optimizer 사용해 model의 parameter update

02. Deep Learning Frameworks

PyTorch: nn

Define our model as a sequence of layers

nn also defines common loss functions

```
import torch
from torch.autograd import Variable

N, D_in, H, D_out = 64, 1000, 100, 10
x = Variable(torch.randn(N, D_in))
y = Variable(torch.randn(N, D_out), requires_grad=False)

model = torch.nn.Sequential(
          torch.nn.Linear(D_in, H),
          torch.nn.ReLU(),
          torch.nn.Linear(H, D_out))
loss_fn = torch.nn.MSELoss(size_average=False)

learning_rate = 1e-4
for t in range(500):
    y_pred = model(x)
    loss = loss_fn(y_pred, y)

    model.zero_grad()
    loss.backward()

    for param in model.parameters():
        param.data -= learning_rate * param.grad.data
```


neural net에 사용되는 high-level wrapper
(Tensorflow의 Keras equivalent)

02. Deep Learning Frameworks

PyTorch: nn
Define new Modules

Define our whole model
as a single Module

```
import torch
from torch.autograd import Variable

class TwoLayerNet(torch.nn.Module):
    def __init__(self, D_in, H, D_out):
        super(TwoLayerNet, self).__init__()
        self.linear1 = torch.nn.Linear(D_in, H)
        self.linear2 = torch.nn.Linear(H, D_out)

    def forward(self, x):
        h_relu = self.linear1(x).clamp(min=0)
        y_pred = self.linear2(h_relu)
        return y_pred

N, D_in, H, D_out = 64, 1000, 100, 10

x = Variable(torch.randn(N, D_in))
y = Variable(torch.randn(N, D_out), requires_grad=False)

model = TwoLayerNet(D_in, H, D_out)

criterion = torch.nn.MSELoss(size_average=False)
optimizer = torch.optim.SGD(model.parameters(), lr=1e-4)
for t in range(500):
    y_pred = model(x)
    loss = criterion(y_pred, y)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

forward pass

loss.backward를 호출하면 매 반복시 마다
그래디언트가 자동으로 계산

tensorflow에는 high-level wrapper가 여러 개 소개되었지만 pytorch는 nn뿐!(2017기준...)->그치만 아주 잘 작동함

02. Deep Learning Frameworks

PyTorch: DataLoaders

A **DataLoader** wraps a **Dataset** and provides minibatching, shuffling, multithreading, for you

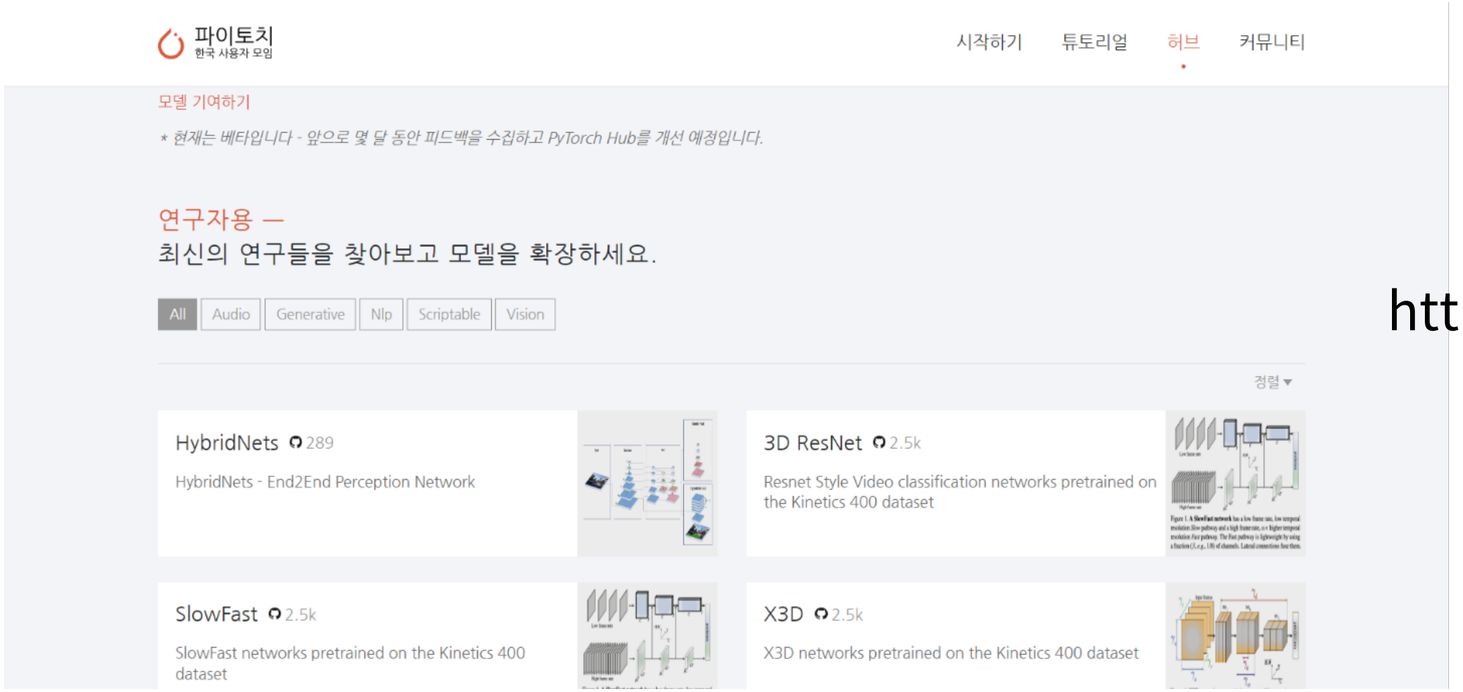
When you need to load custom data, just write your own Dataset class

```
import torch
from torch.autograd import Variable
from torch.utils.data import TensorDataset, DataLoader

N, D_in, H, D_out = 64, 1000, 100, 10

x = torch.randn(N, D_in)
y = torch.randn(N, D_out)

loader = DataLoader(TensorDataset(x, y), batch_size=8)

model = TwoLayerNet(D_in, H, D_out)

criterion = torch.nn.MSELoss(size_average=False)
optimizer = torch.optim.SGD(model.parameters(), lr=1e-4)
for epoch in range(10):
    for x_batch, y_batch in loader:
        x_var, y_var = Variable(x), Variable(y)
        y_pred = model(x_var)
        loss = criterion(y_pred, y_var)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```


데이터를 이용하고자 할 때 데이터를 어떤 방식으로 읽을 것이지를 명시하는 dataset class만 작성하면data loader로 wrapping 시켜서 학습함

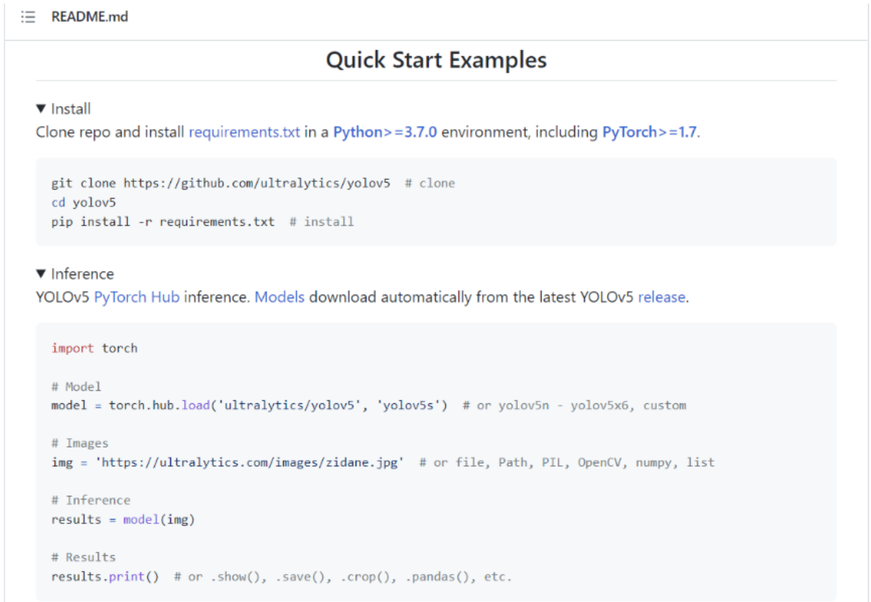
02. Deep Learning Frameworks

Pretrained models



https://pytorch.kr/hub/

https://pytorch.kr/hub/ultralytics_yolov5/



02. Deep Learning Frameworks

Pretrained models

```
!pip install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu113
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/, https://download.pytorch.org/whl/cu113
Requirement already satisfied: torch in /usr/local/lib/python3.7/dist-packages (1.12.0+cu113)
Requirement already satisfied: torchvision in /usr/local/lib/python3.7/dist-packages (0.13.0+cu113)
Requirement already satisfied: torchaudio in /usr/local/lib/python3.7/dist-packages (0.12.0+cu113)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from torch) (4.1.1)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.7/dist-packages (from torchvision) (7.1.2)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from torchvision) (2.23.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from torchvision) (1.21.6)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->torchvision) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->torchvision) (2022.6.15)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->torchvision) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->torchvision) (2.10)
```

```
!git clone https://github.com/ultralytics/yolov5
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 12631, done.
remote: Counting objects: 100% (56/56), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 12631 (delta 26), reused 31 (delta 13), pack-reused 12575
Receiving objects: 100% (12631/12631), 12.34 MiB | 19.68 MiB/s, done.
Resolving deltas: 100% (8681/8681), done.
```

```
!cd yolov5
!pip install -r requirements.txt
```

```
#install dependencies
import torch #import pytorch
from matplotlib import pyplot as plt #import matplotlib
import numpy as np  #import numpy(helps data transformation)
import cv2 #import opencv
```

```
#load model
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting PyYAML>=5.3.1
  Downloading PyYAML-6.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (596 kB)
Installing collected packages: PyYAML
  Attempting uninstall: PyYAML
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed PyYAML-6.0
```

```
from google.colab import files
uploaded = files.upload()
```

파일 선택 선택된 파일 없음 Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving friends.jpg to friends.jpg
```

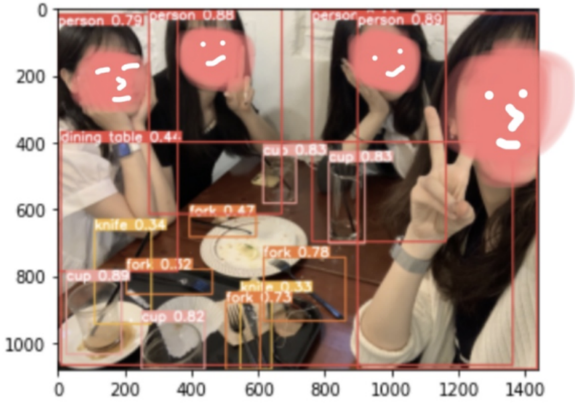
```
img = "friends.jpg"
```

```
results = model(img)
results.print()
```

```
image 1/1: 1080x1440 4 persons, 4 cups, 4 forks, 2 knifes, 1 dining table
Speed: 89.7ms pre-process, 456.8ms inference, 1.4ms NMS per image at shape (1, 3, 480, 640)
```

```
#rendering
%matplotlib inline
plt.imshow(np.squeeze(results.render()))  #imshow 함수로 render
plt.show()
```


```
#rendering
%matplotlib inline
plt.imshow(np.squeeze(results.render()))  #imshow 함수로 render
plt.show()
```

02. Deep Learning Frameworks

Torch vs PyTorch

Torch	**PyTorch**
(-) Lua	(+) Python
(-) No autograd	(+) Autograd
(+) More stable	(-) Newer, still changing
(+) Lots of existing code	(-) Less existing code
(0) Fast	(0) Fast

Conclusion: Probably use PyTorch for new projects

Torch로부터 발전해옴.. PyTorch를 쓰는 것이 더 좋다!

02. Deep Learning Frameworks

Tensorflow는 computational graph를
한 번 만들어 iterate

Static vs Dynamic Graphs

Tensorflow: static

TensorFlow: Build graph once, then run many times (**static**)

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.Variable(tf.random_normal((D, H)))
w2 = tf.Variable(tf.random_normal((H, D)))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))
grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])

learning_rate = 1e-5
new_w1 = w1.assign(w1 - learning_rate * grad_w1)
new_w2 = w2.assign(w2 - learning_rate * grad_w2)
updates = tf.group(new_w1, new_w2)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    values = {x: np.random.randn(N, D),
              y: np.random.randn(N, D),}
    losses = []
    for t in range(50):
        loss_val, _ = sess.run([loss, updates],
                               feed_dict=values)
```

Build graph

Run each iteration

PyTorch: Dynamic

PyTorch: Each forward pass defines a new graph (**dynamic**)

```
import torch
from torch.autograd import Variable

N, D_in, H, D_out = 64, 1000, 100, 10
x = Variable(torch.randn(N, D_in), requires_grad=False)
y = Variable(torch.randn(N, D_out), requires_grad=False)
w1 = Variable(torch.randn(D_in, H), requires_grad=True)
w2 = Variable(torch.randn(H, D_out), requires_grad=True)

learning_rate = 1e-6
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    if w1.grad: w1.grad.data.zero_()
    if w2.grad: w2.grad.data.zero_()
    loss.backward()

    w1.data -= learning_rate * w1.grad.data
    w2.data -= learning_rate * w2.grad.data
```

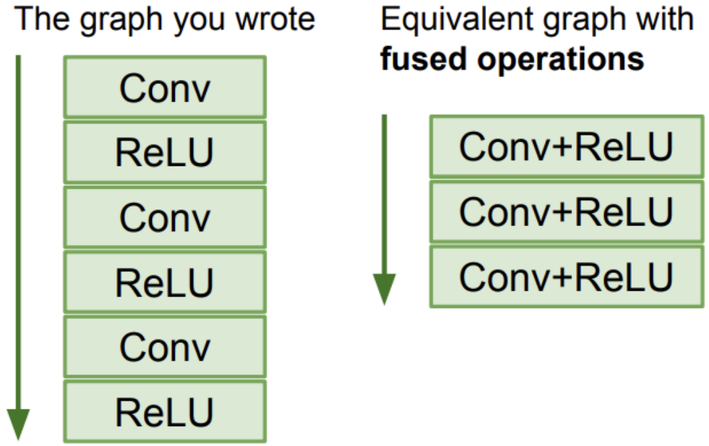
New graph each iteration

PyTorch는 forward pass 할 때 마다 매번
새로운 computational graph를 만듦

02. Deep Learning Frameworks

Static vs Dynamic: Optimization

With static graphs, framework can **optimize** the graph for you before it runs!



Static vs Dynamic: Serialization

Static

Once graph is built, can **serialize** it and run it without the code that built the graph!

Dynamic

Graph building and execution are intertwined, so always need to keep code around

02. Deep Learning Frameworks

Static vs Dynamic: Conditional

$$y = \begin{cases} w1 * x & \text{if } z > 0 \\ w2 * x & \text{otherwise} \end{cases}$$

PyTorch: Normal Python

```
N, D, H = 3, 4, 5

x = Variable(torch.randn(N, D))
w1 = Variable(torch.randn(D, H))
w2 = Variable(torch.randn(D, H))

z = 10
if z > 0:
    y = x.mm(w1)
else:
    y = x.mm(w2)
```

TensorFlow: Special TF control flow operator!

```
N, D, H = 3, 4, 5
x = tf.placeholder(tf.float32, shape=(N, D))
z = tf.placeholder(tf.float32, shape=None)
w1 = tf.placeholder(tf.float32, shape=(D, H))
w2 = tf.placeholder(tf.float32, shape=(D, H))

def f1(): return tf.matmul(x, w1)
def f2(): return tf.matmul(x, w2)
y = tf.cond(tf.less(z, 0), f1, f2)

with tf.Session() as sess:
    values = {
        x: np.random.randn(N, D),
        z: 10,
        w1: np.random.randn(D, H),
        w2: np.random.randn(D, H),
    }
    y_val = sess.run(y, feed_dict=values)
```

02. Deep Learning Frameworks

Static vs Dynamic: Loops

$$y_t = (y_{t-1} + x_t) * w$$

PyTorch: Normal Python

```
T, D = 3, 4
y0 = Variable(torch.randn(D))
x = Variable(torch.randn(T, D))
w = Variable(torch.randn(D))

y = [y0]
for t in range(T):
    prev_y = y[-1]
    next_y = (prev_y + x[t]) * w
    y.append(next_y)
```

TensorFlow: Special TF control flow

```
T, N, D = 3, 4, 5
x = tf.placeholder(tf.float32, shape=(T, D))
y0 = tf.placeholder(tf.float32, shape=(D,))
w = tf.placeholder(tf.float32, shape=(D,))

def f(prev_y, cur_x):
    return (prev_y + cur_x) * w

y = tf.foldl(f, x, y0)

with tf.Session() as sess:
    values = {
        x: np.random.randn(T, D),
        y0: np.random.randn(D),
        w: np.random.randn(D),
    }
    y_val = sess.run(y, feed_dict=values)
```

02. Deep Learning Frameworks

Caffe

Caffe: Training / Finetuning

No need to write code!

1. Convert data (run a script)
2. Define net (edit prototxt)
3. Define solver (edit prototxt)
4. Train (with pretrained weights) (run a script)

```
name: "LogisticRegressionNet"
layers {
  top: "data"
  top: "label"
  name: "data"
  type: HDF5_DATA
  hdf5_data_param {
    source: "examples/hdf5_classification/data/train.txt"
    batch_size: 10
  }
  include {
    phase: TRAIN
  }
}
layers {
  bottom: "data"
  top: "fc1"
  name: "fc1"
  type: INNER_PRODUCT
  blobs_lr: 1
  blobs_lr: 2
  weight_decay: 1
  weight_decay: 0
```

```
  inner_product_param {
    num_output: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layers {
  bottom: "fc1"
  bottom: "label"
  top: "loss"
  name: "loss"
  type: SOFTMAX_LOSS
}
```

02. Deep Learning Frameworks

기업이 사용하는 Framework

Google:
TensorFlow

"One framework to rule them all"

Facebook:
PyTorch +Caffe2

Research

Production

감사합니다!