DSCoV
Pytorch Tutorial

Minju Jung

Prerequisites

- Basic knowledge of Python
- Basic concepts of Deep Learning
    - Models
        - Multi-layer perceptron (MLP), Convolutional neural networks (CNNs) and etc.
    - Non-linear activations
        - Sigmoid, Hyperbolic tangent, ReLU and etc.
    - Loss functions
        - L1, L2, and etc.
    - Optimizers
        - Stochastic gradient descent (SGD), Adam and etc.

Keras

Chainer

TensorFlow

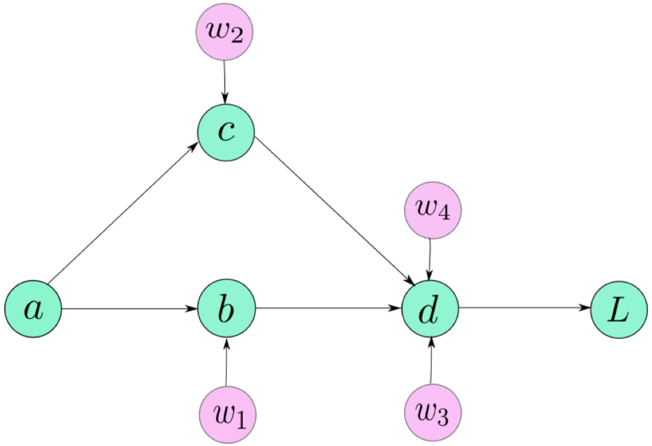
PYTORCH

Automatic differentiation

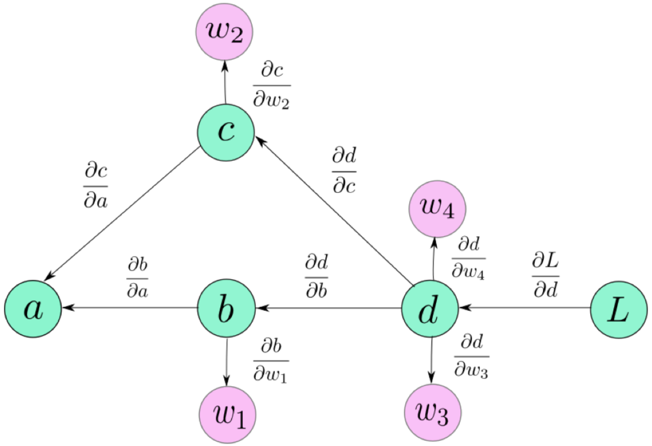
```
b = w1 * a
c = w2 * a
d = (w3 * b) + (w4 * c)
L = f(d)
```

Forward pass

Backward pass



Building a computational graph



"The computation graph is simply a data structure that allows you to efficiently apply the chain rule to compute gradients for all of your parameters."

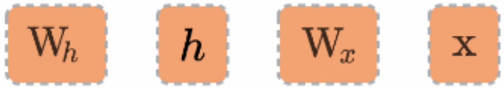
https://towardsdatascience.com/getting-started-with-pytorch-part-1-understanding-how-automatic-differentiation-works-5008282073ec

Dynamic graph

A graph is created on the fly

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))
```



https://blog.algorithmia.com/exploring-the-deep-learning-framework-pytorch

Static (define-then-run) vs dynamic (define-by-run) graphs

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: 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

Static (define-then-run) vs dynamic (define-by-run) graphs

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)
```

https://dev-jm.tistory.com/4

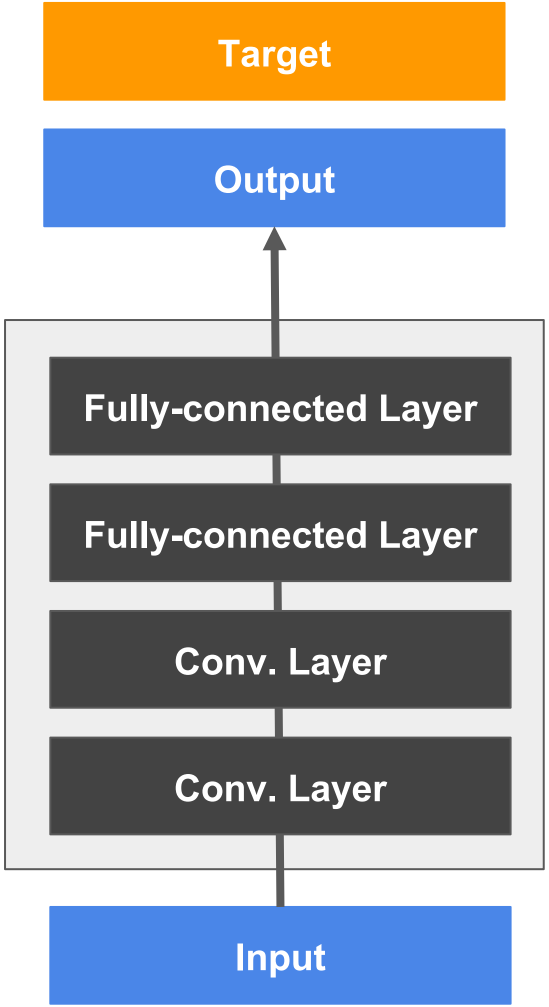
Contents of tutorial on Google Colab

- Basics of tensor
- Building a neural network model for image classification
  - Model
  - Optimizer
  - Loss
  - Dataset & DataLoader
  - Save & Load the pre-trained model

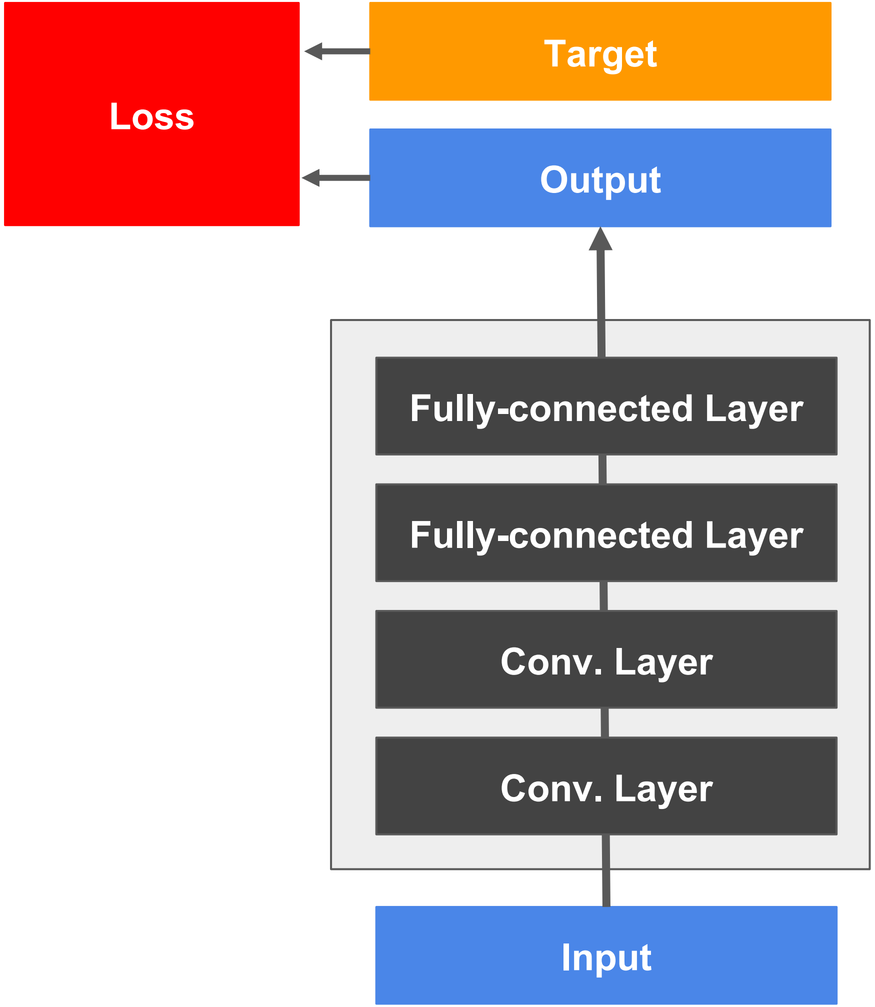
Appendix

An overview of deep learning

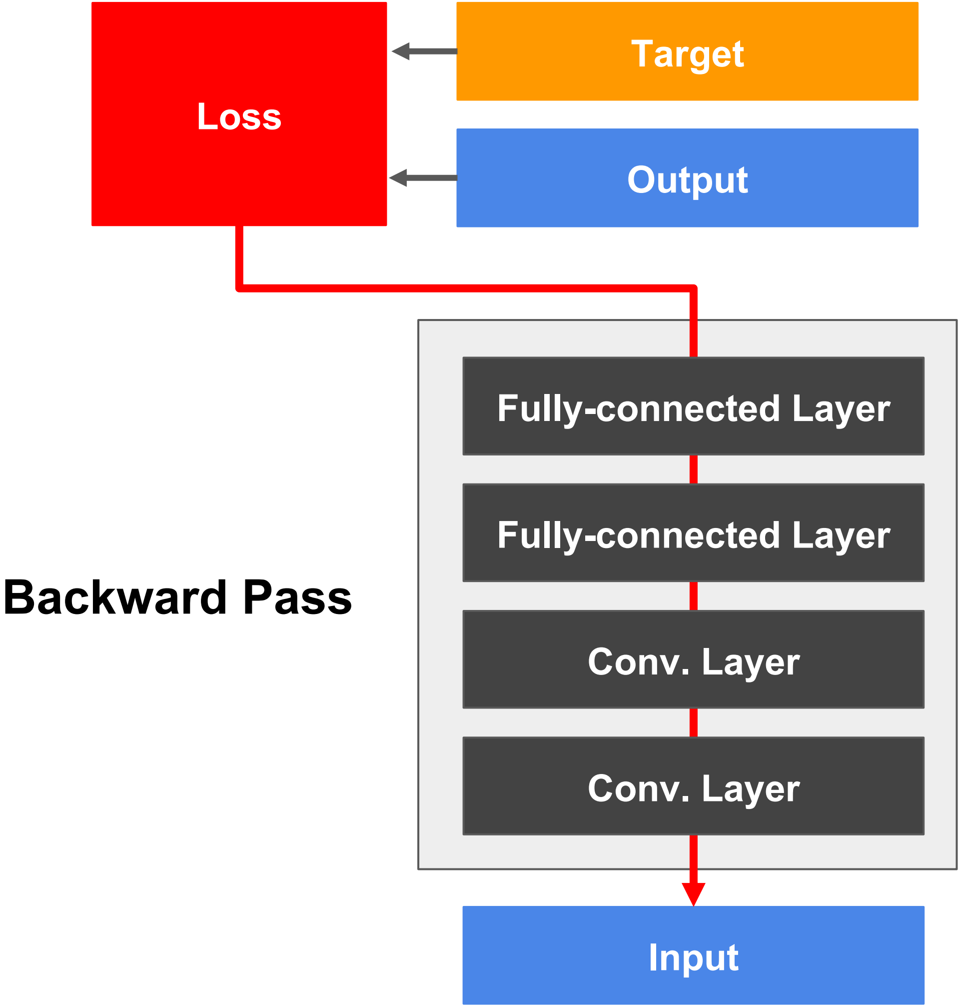
Forward Pass



An overview of deep learning

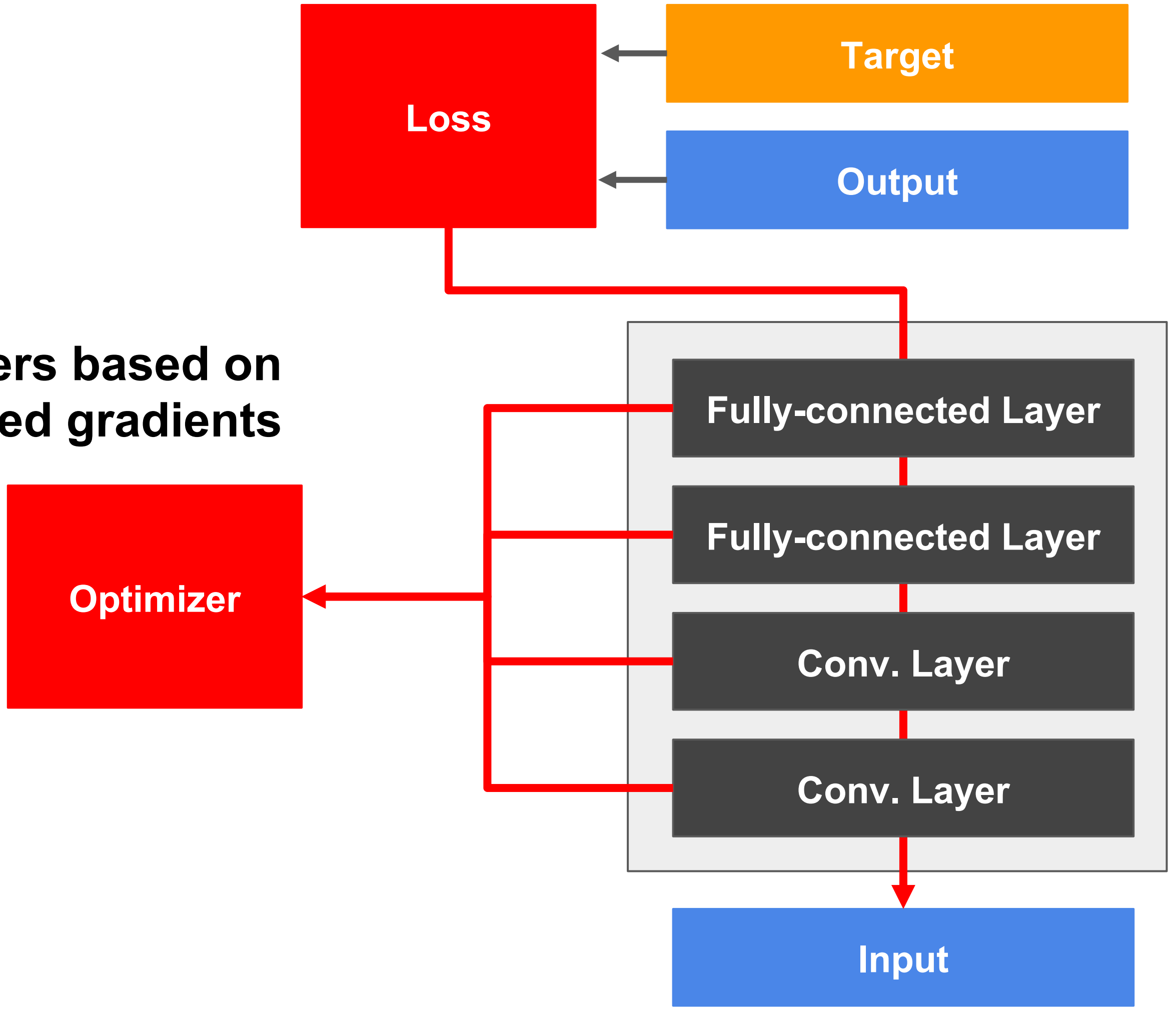


An overview of deep learning



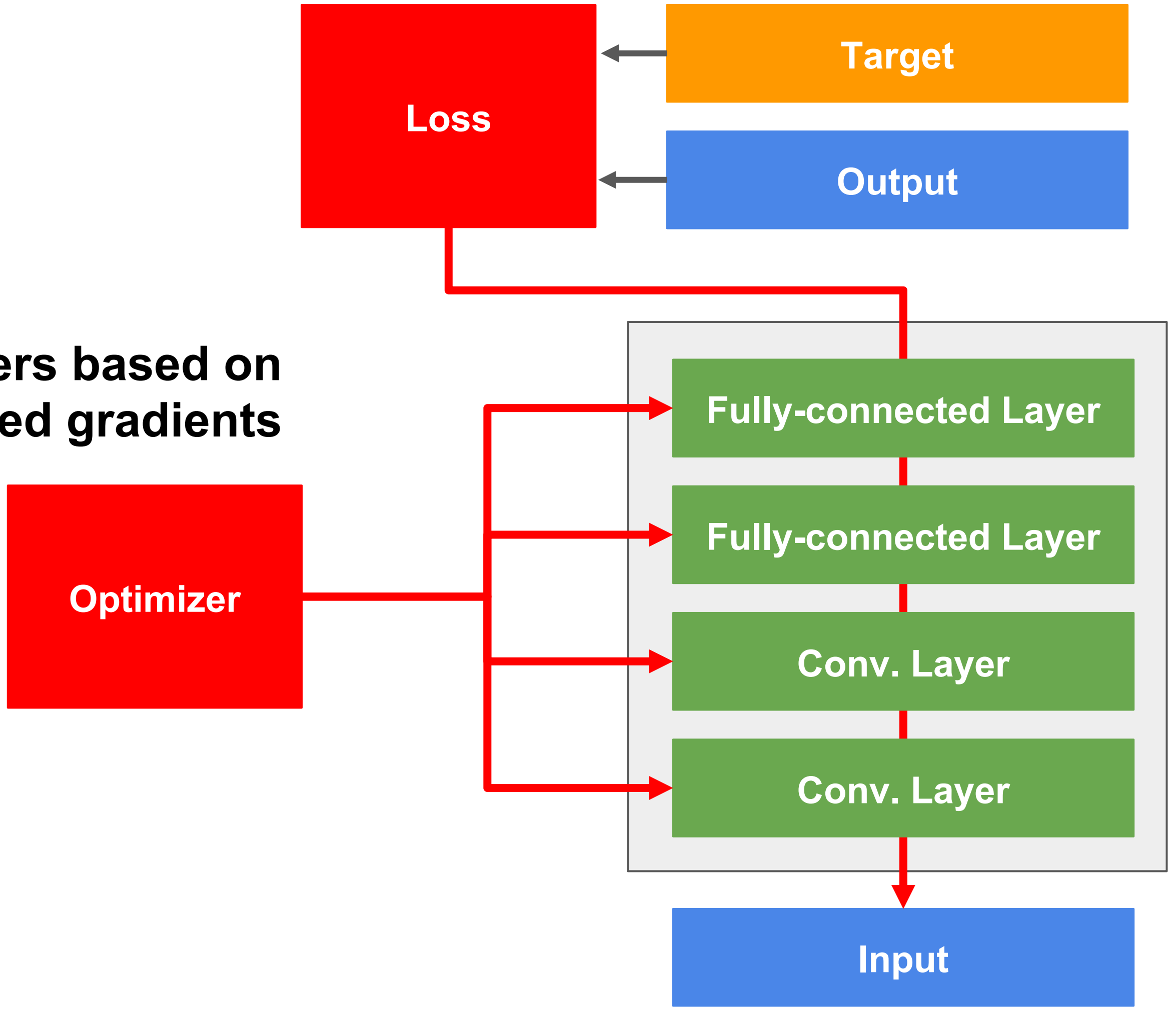
An overview of deep learning

Update the parameters based on the computed gradients

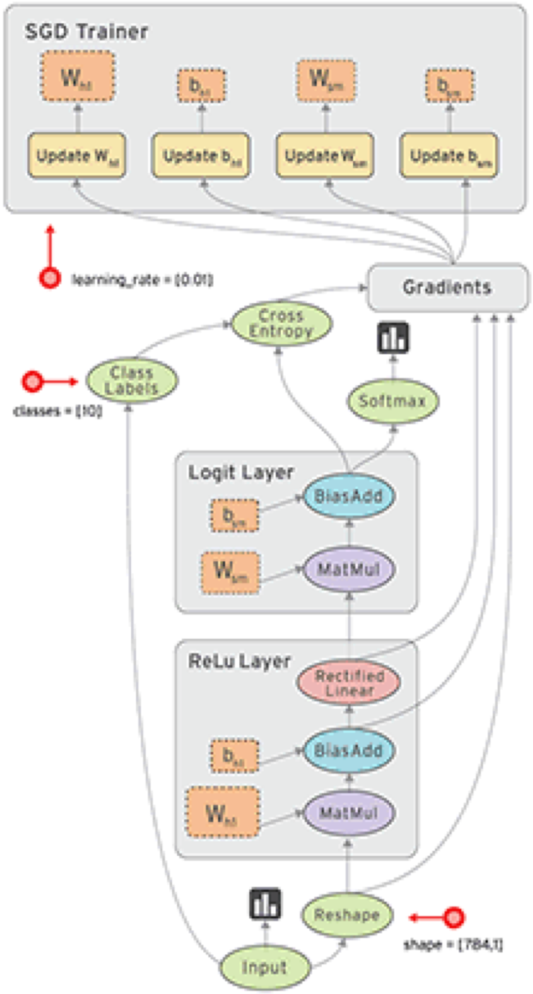


An overview of deep learning

Update the parameters based on the computed gradients



An overview of deep learning



CS231n: Convolutional Neural Networks for Visual Recognition:
http://cs231n.stanford.edu/