CIFAR-10

Dohyun Kim
dhkim1028@korea.ac.kr
Data Intelligence Lab, Korea University
2020.04.20.

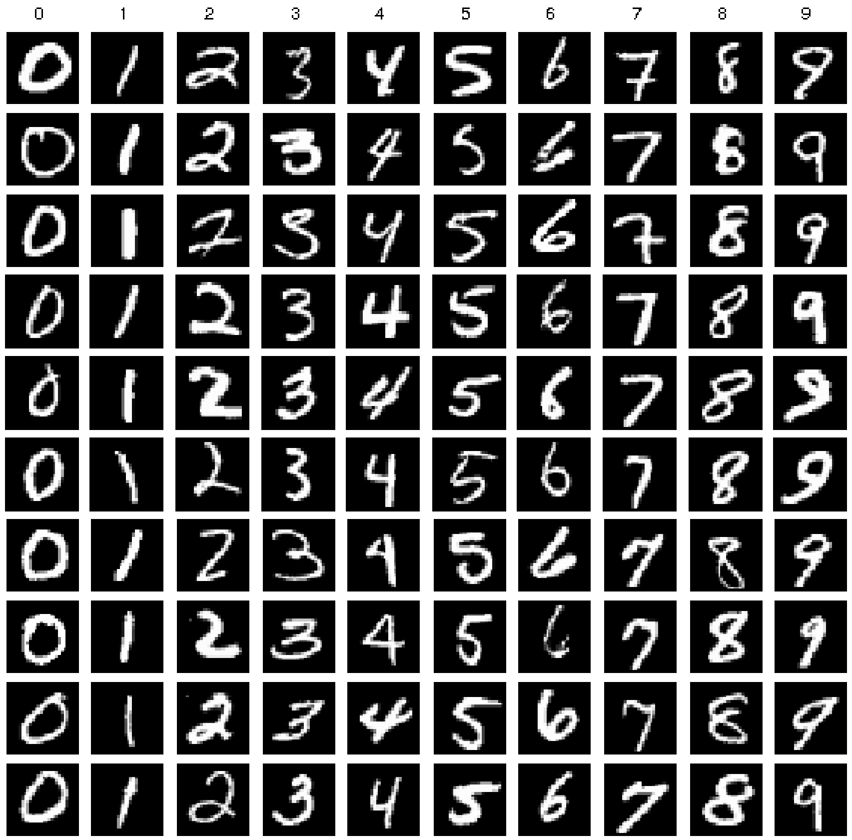
Class Lab – 기초 과제 일정

1. XOR (~4/05)

2. MNIST (~4/19)

3. CIFAR-10 (~5/03)

MNIST Review



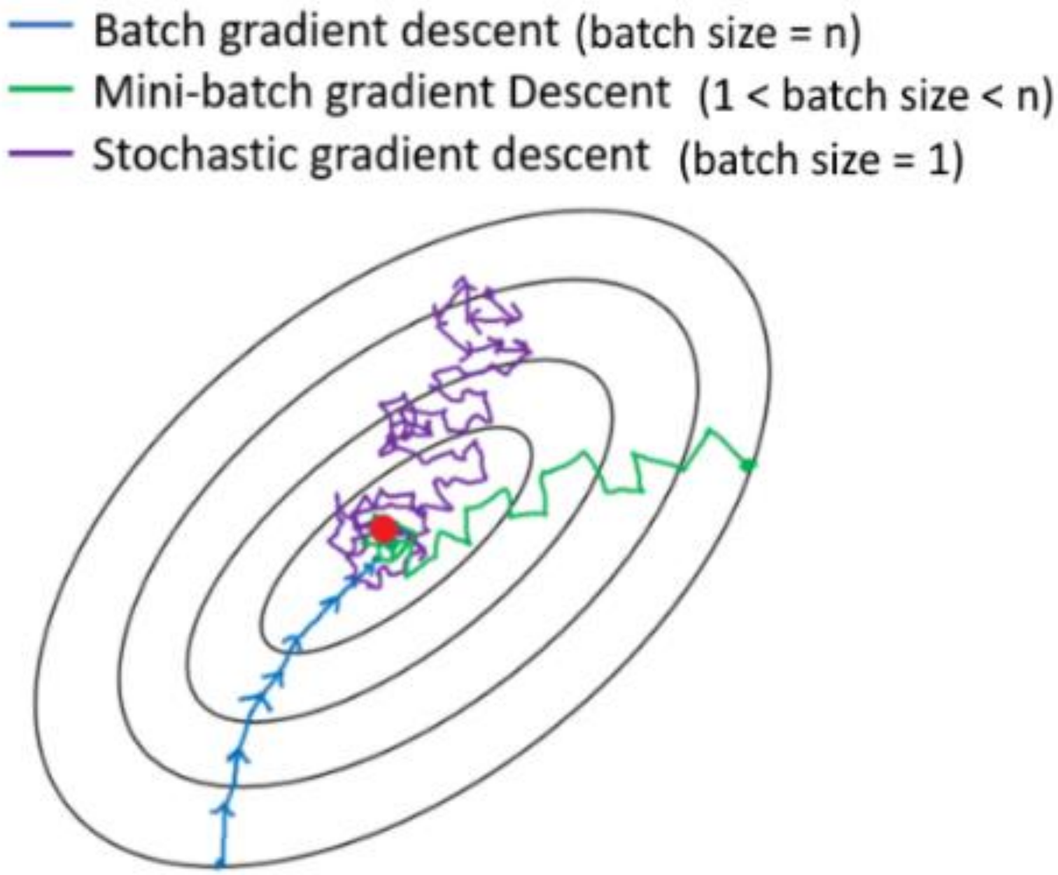
- The database contains 60,000 training images, 10,000 validation images, and 10,000 testing images with 10 classes.

- Shape of each data : [28, 28]

- Range : 0.0 to 1.0

MNIST Review

- **Mini-Batch**



- **Weight Initialization**

(1) Xavier Normal Initialization

$$W \sim N(0, Var(W))$$

$$Var(W) = \sqrt{\frac{2}{n_{in} + n_{out}}}$$

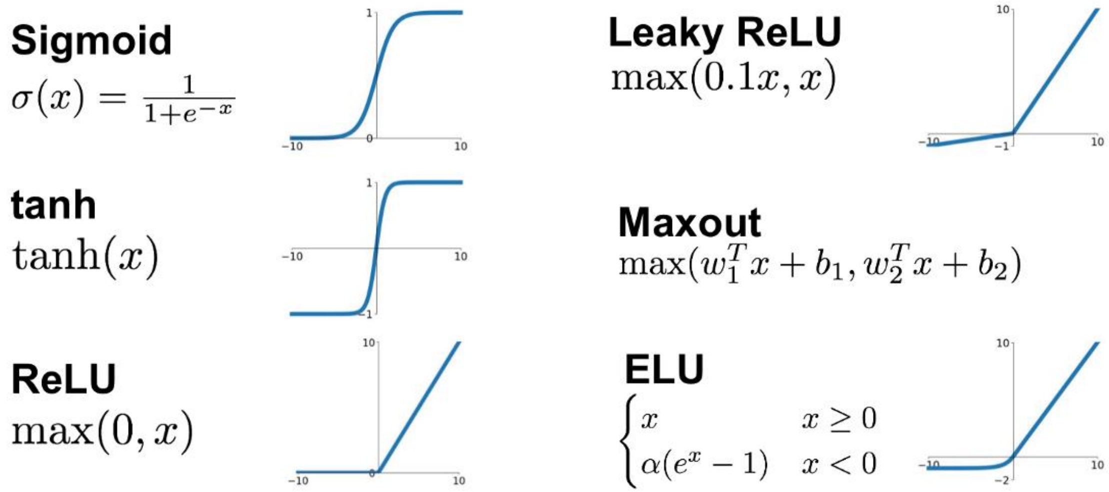
(2) He Normal Initialization

$$W \sim N(0, Var(W))$$

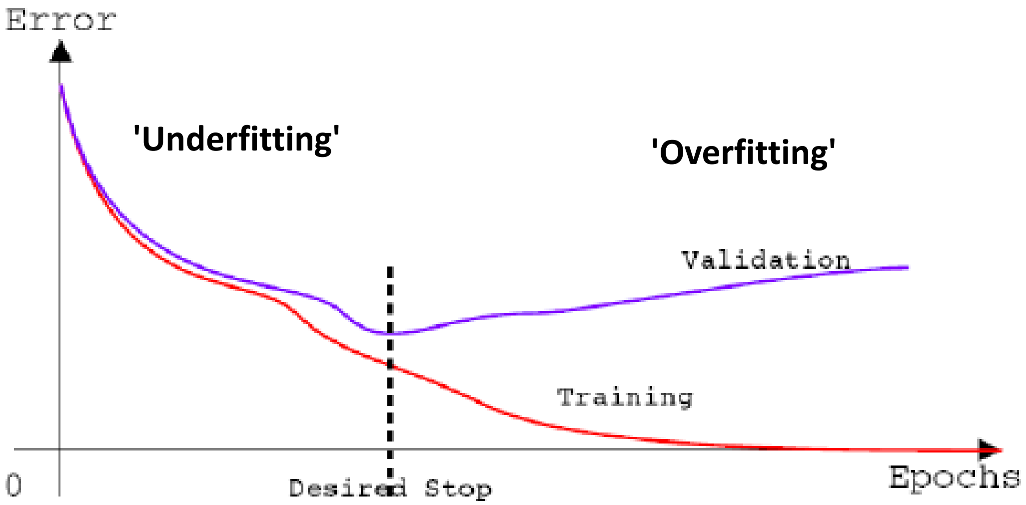
$$Var(W) = \sqrt{\frac{2}{n_{in}}}$$

MNIST Review

- **Activation Function**



- **Early Stopping**



MNIST Review

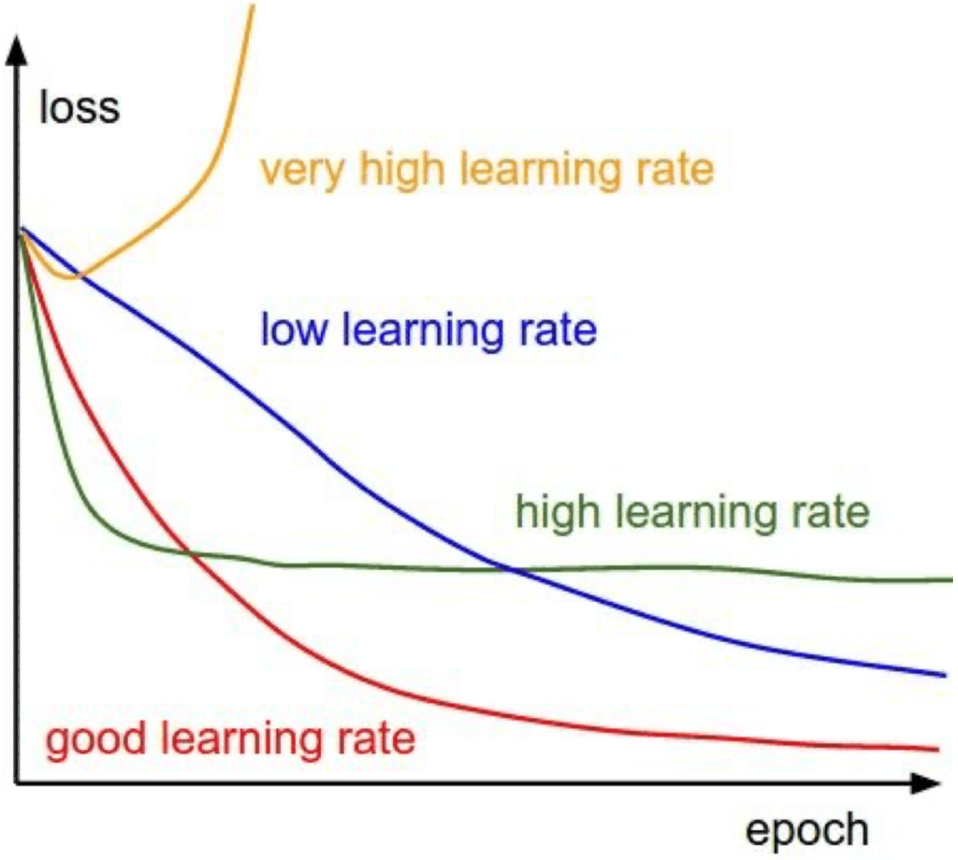
- **Parameter Norm Penalties**

Weight decay: $$E_t = \frac{1}{N_t}\sum_{n \in D_t} E_n + \frac{\lambda}{2}\underbrace{||w||^2}_{\text{L2-norm}}$$

$$w^{t+1} = w^t - \varepsilon\left(\frac{1}{N_t}\sum \nabla E_n + \lambda w^t\right)$$

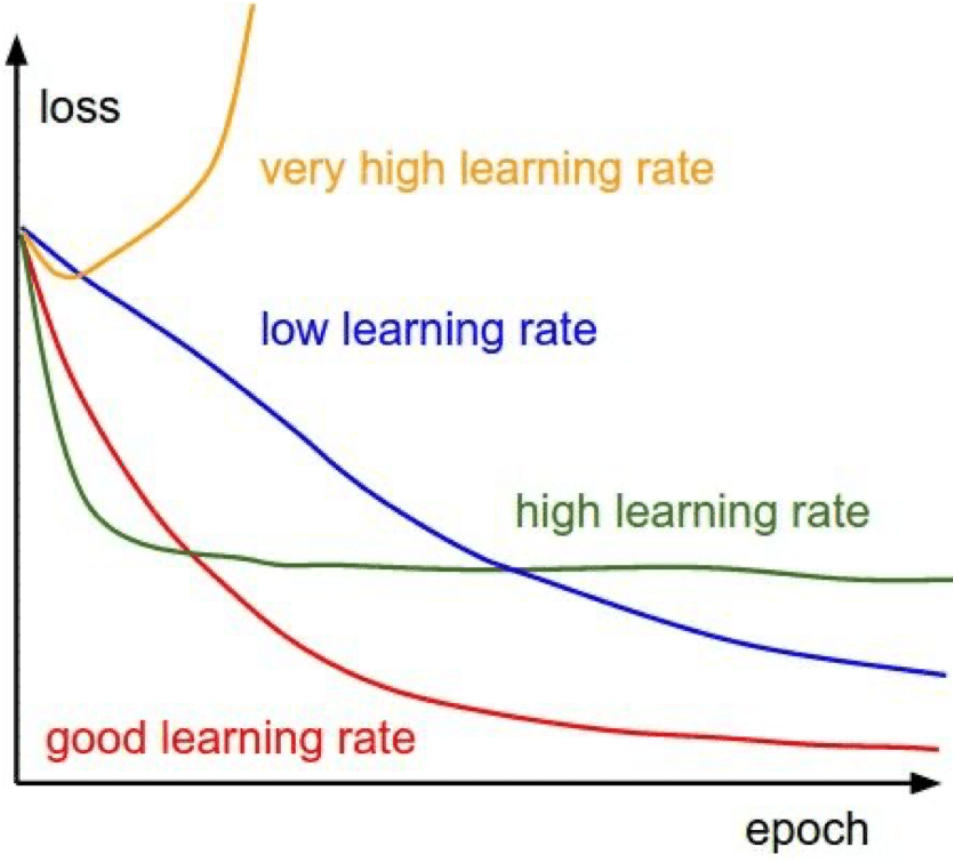
Weight restriction: $||w||^2 < c$

Learning Rate Decay



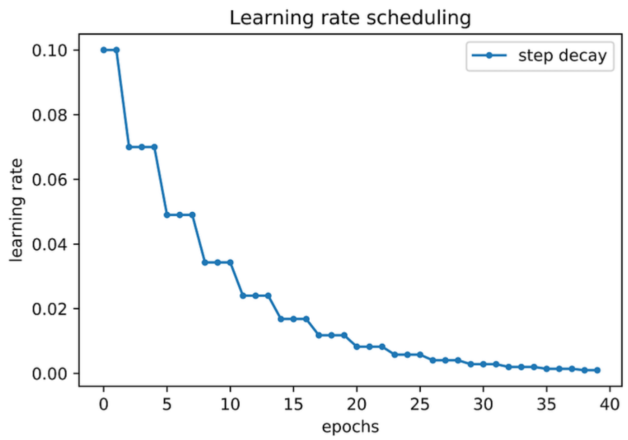
We should choose proper learning rate to find global optimum.

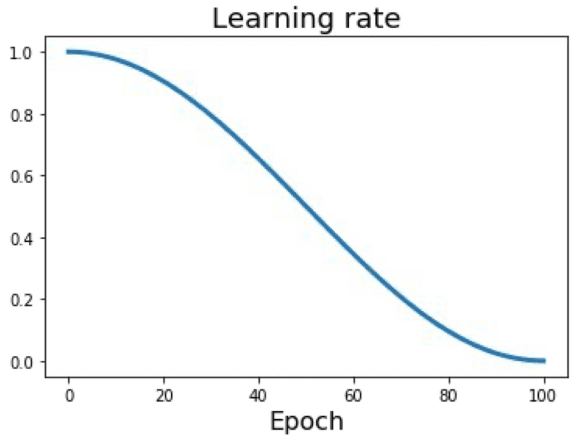
Choosing Proper Learning Rate

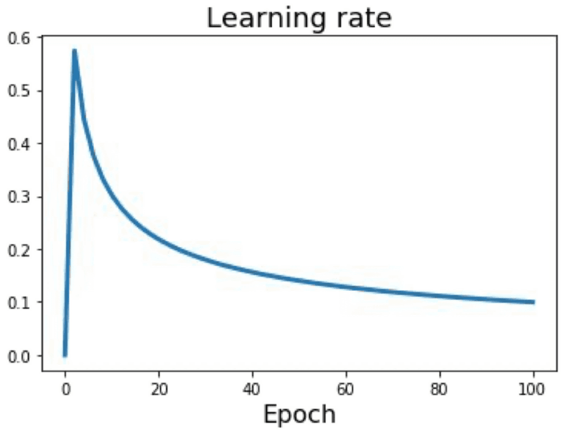


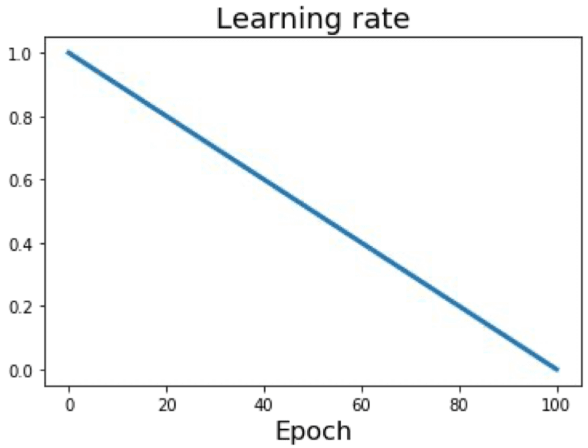
We should choose proper learning rate to find global optimum.

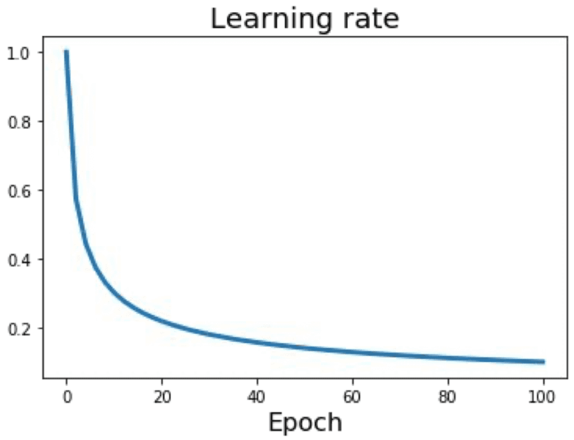
Learning Rate Scheduling
(Learning Rate Decay)



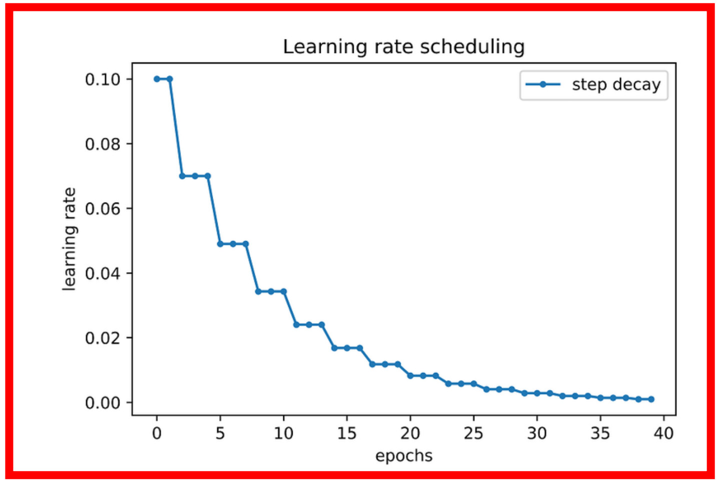


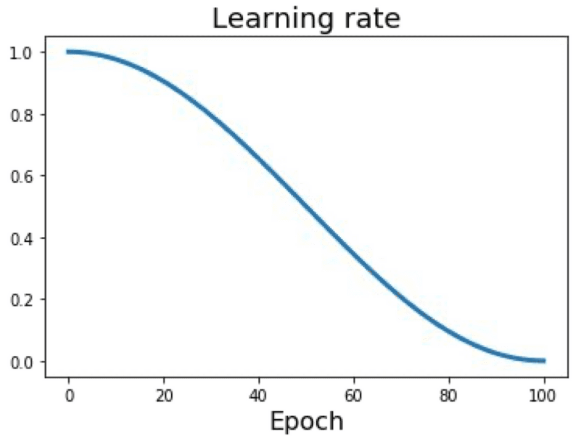


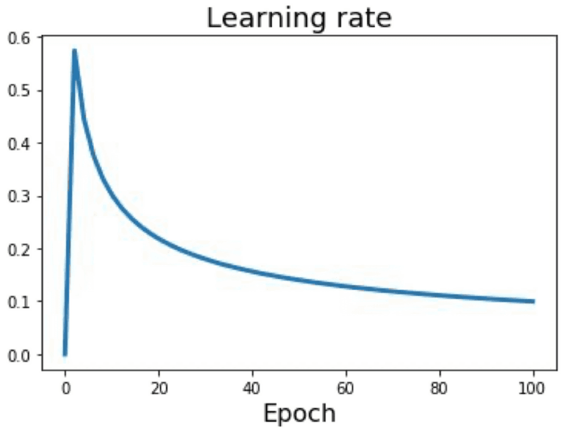


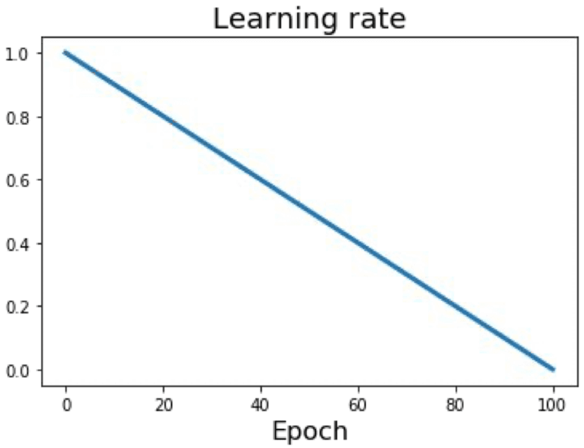


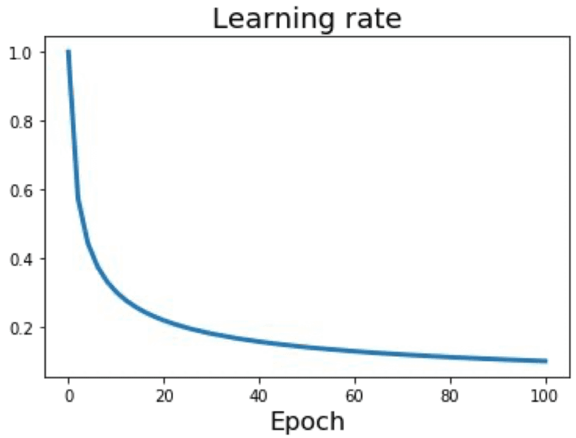
Learning Rate Scheduling
(Learning Rate Decay)





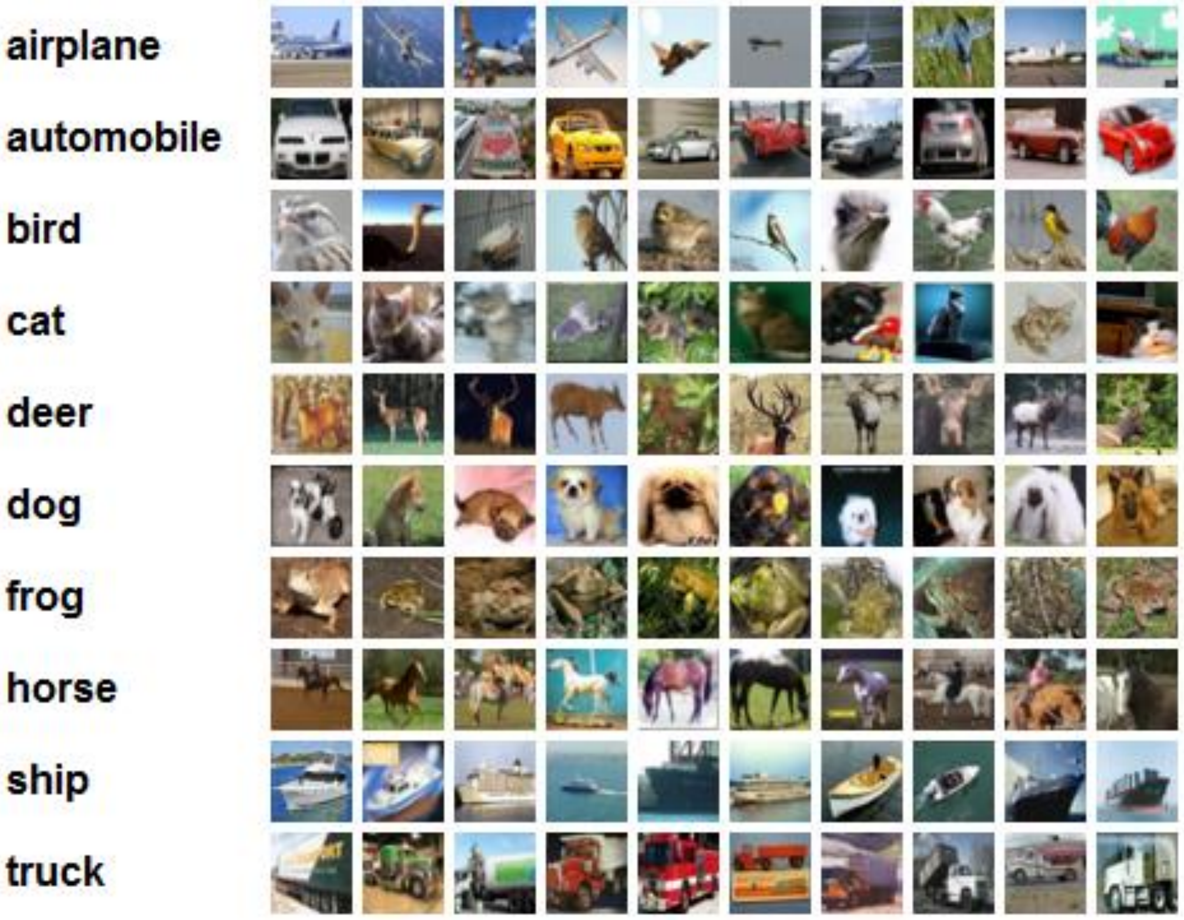






We use step decay method in training ResNet32.

Assignment #3 : CIFAR-10



Assignment #3 : CIFAR-10

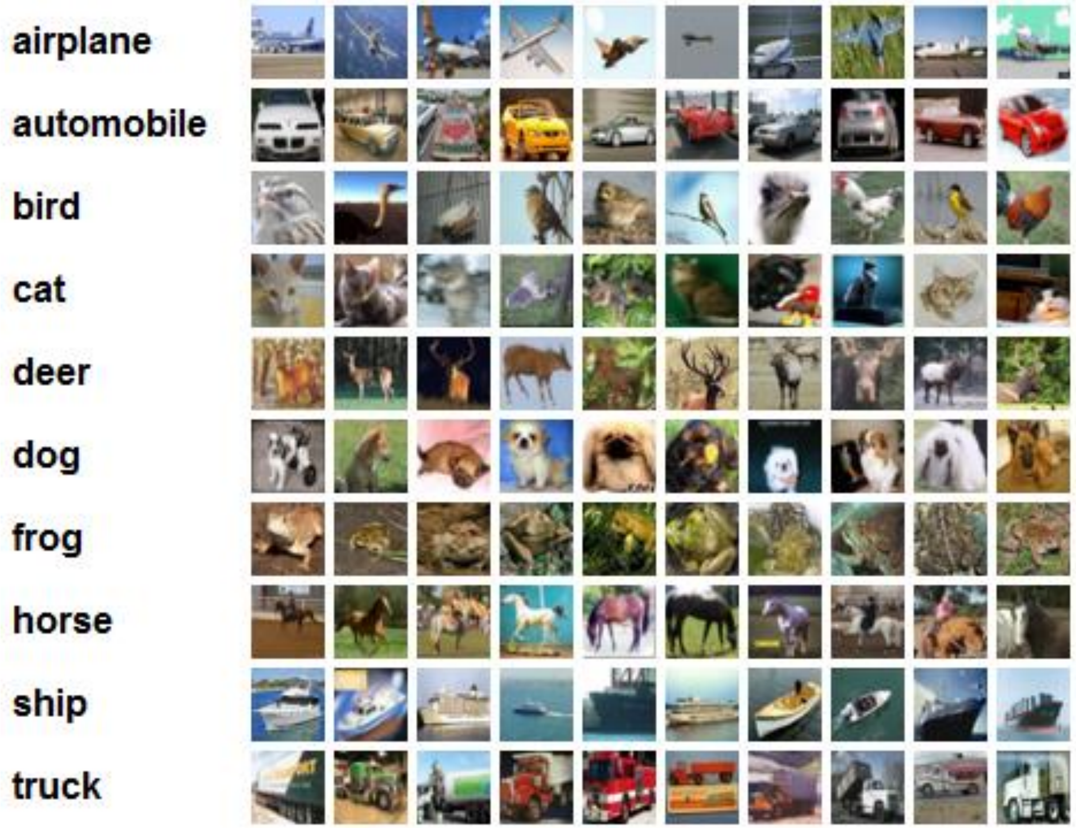
- **Introduction**

  - CIFAR-10 : **C**anadian **I**nstitute **F**or **A**dvanced **R**esearch

  - This is a collection of images that are commonly used to train machine learning and computer vision algorithms.

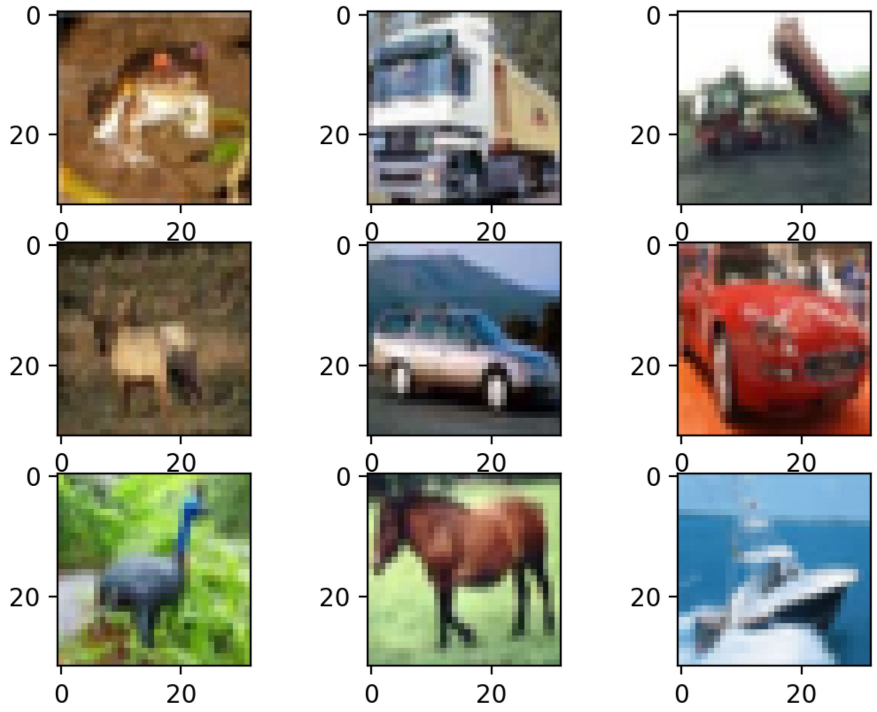
  - The database is also widely used for training and testing in the field of machine learning.

  - The database contains 50,000 32*32 training images, 10,000 validation images, and 10,000 testing images with 10 classes.



[1] https://www.cs.toronto.edu/~kriz/cifar.html

Assignment #3 : CIFAR-10



- Shape of each data : [3, 32, 32]

- Range : 0 to 255

- You can see the image of each data.
  (available in the assignment code)

Standardization

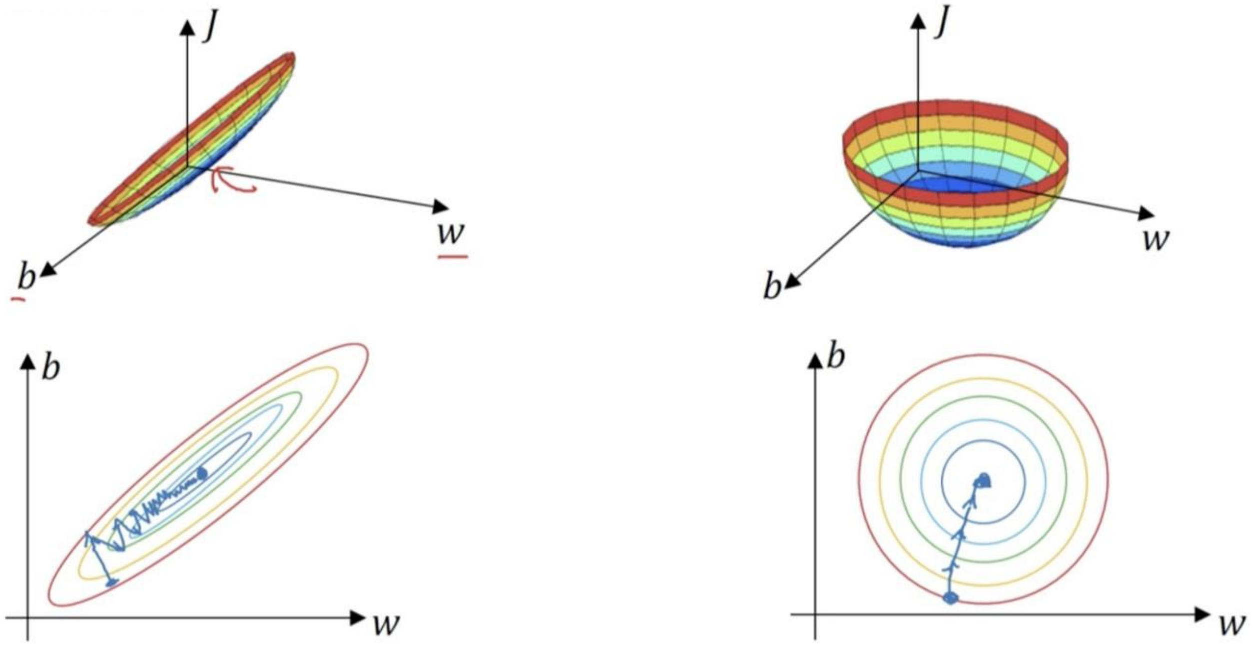
- Standardize the input data.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$x := x - \mu$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)^2$$

Unstandardized   Standardized



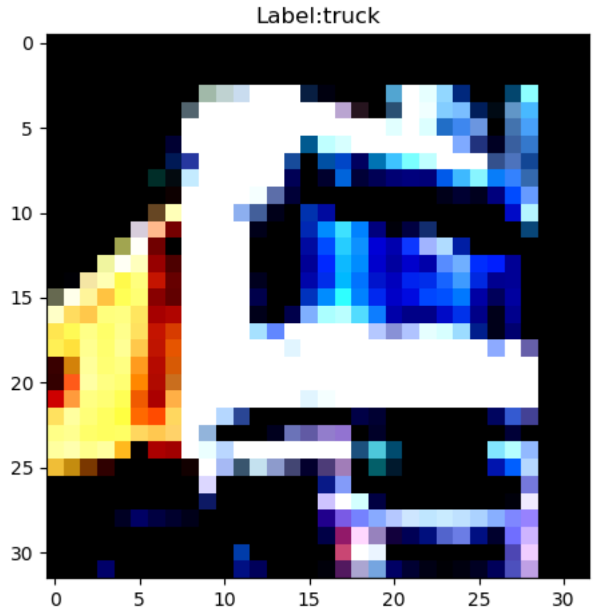
- We use standardization to apply gradient descent algorithm easily.

Standardization

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

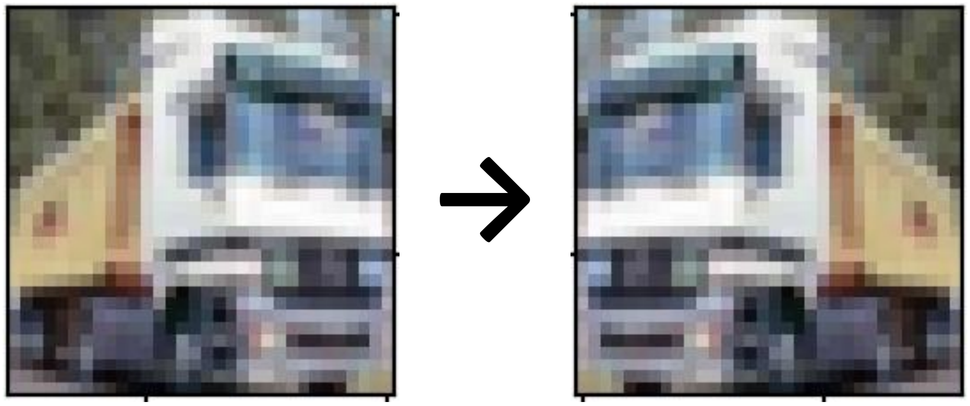
$$x := x - \mu$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)^2$$



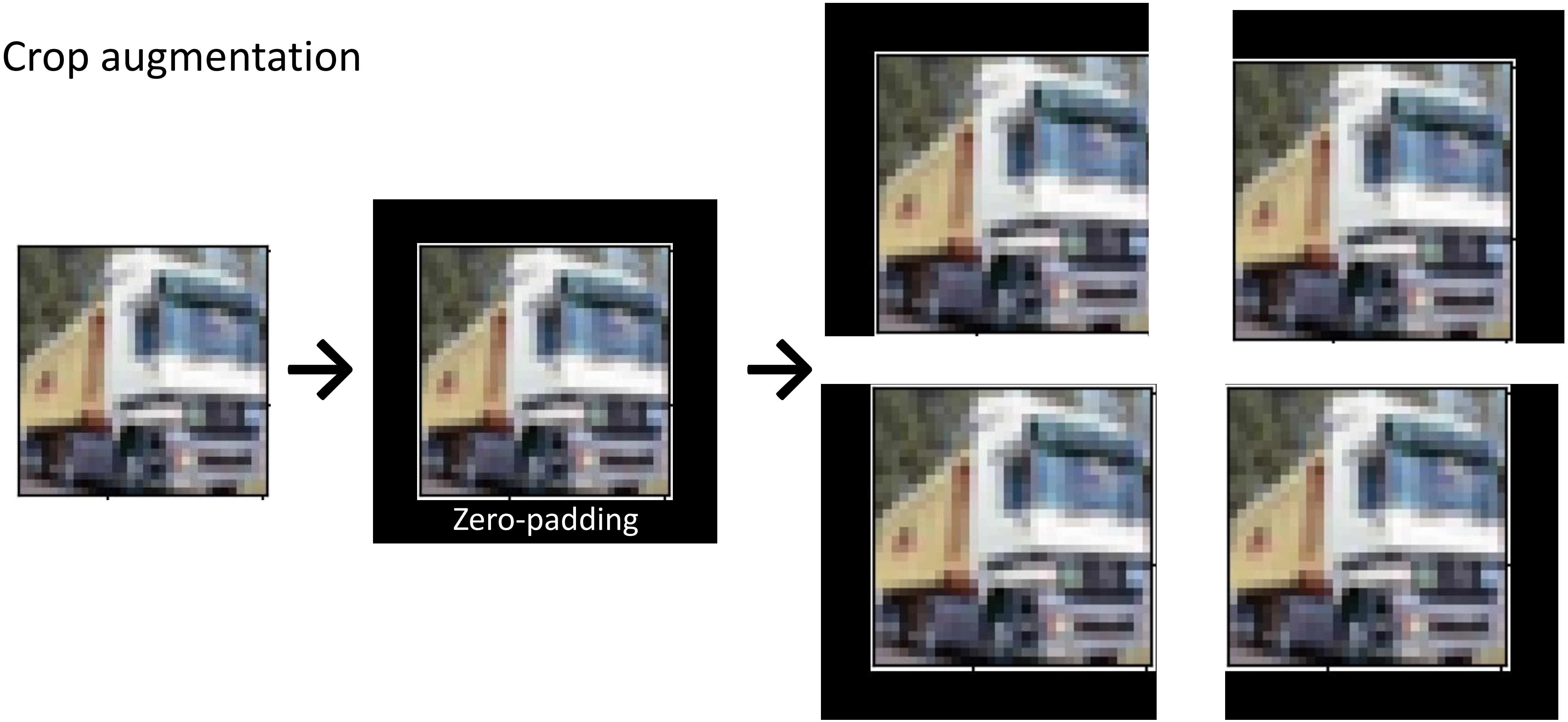
Data Augmentation

1) Flip augmentation

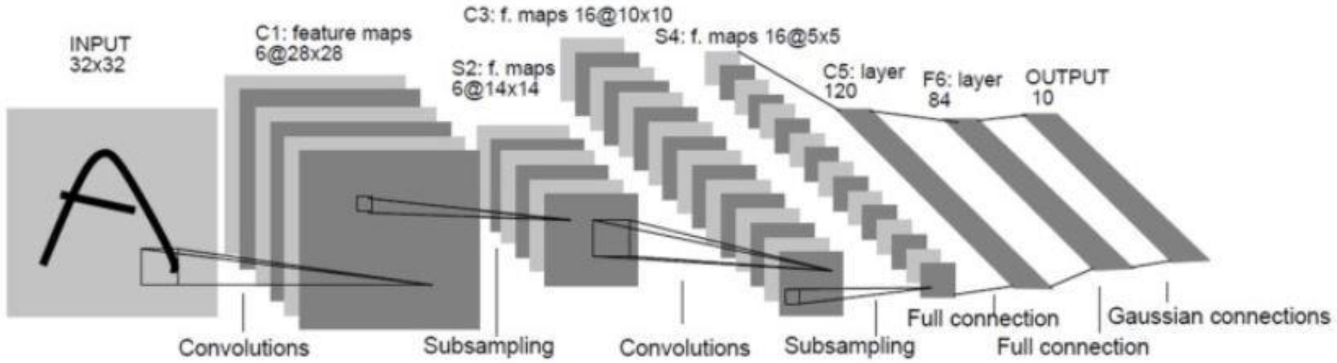


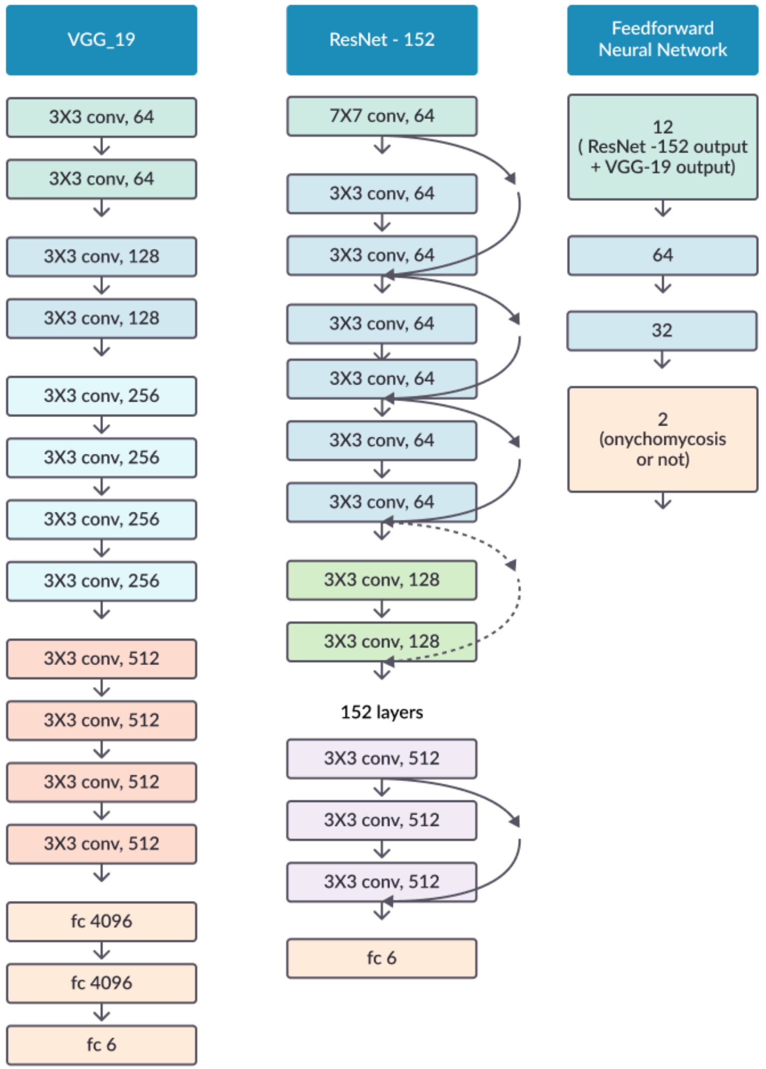
Data Augmentation

2) Crop augmentation

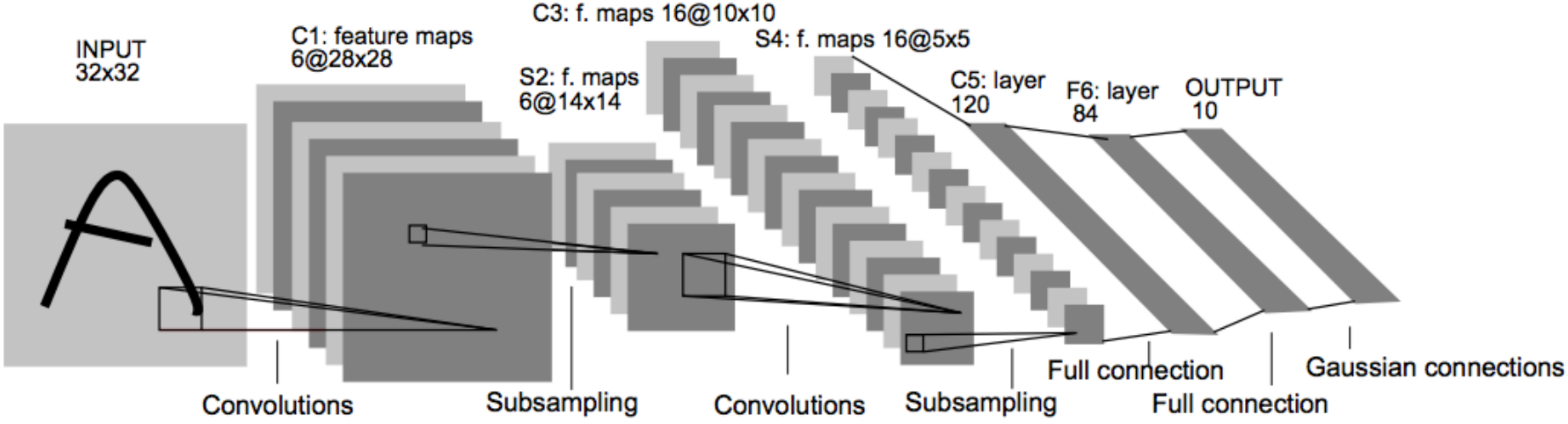


Convolutional Neural Network



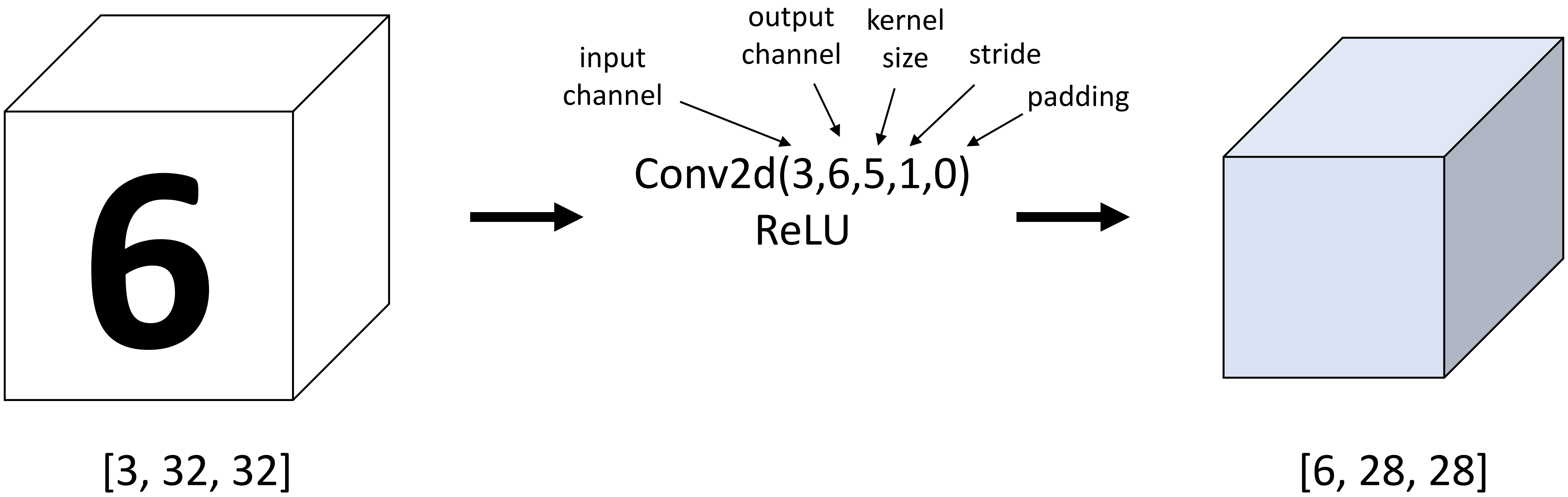


[1. LeNet-5[1]]

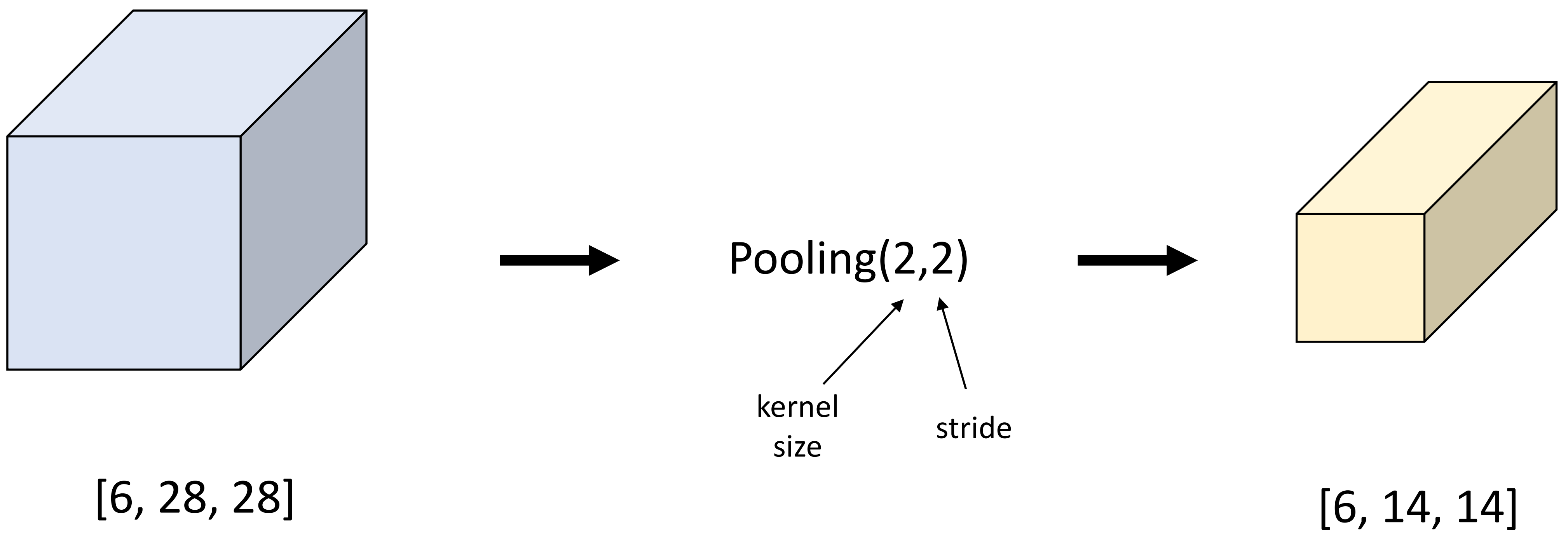


[1] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

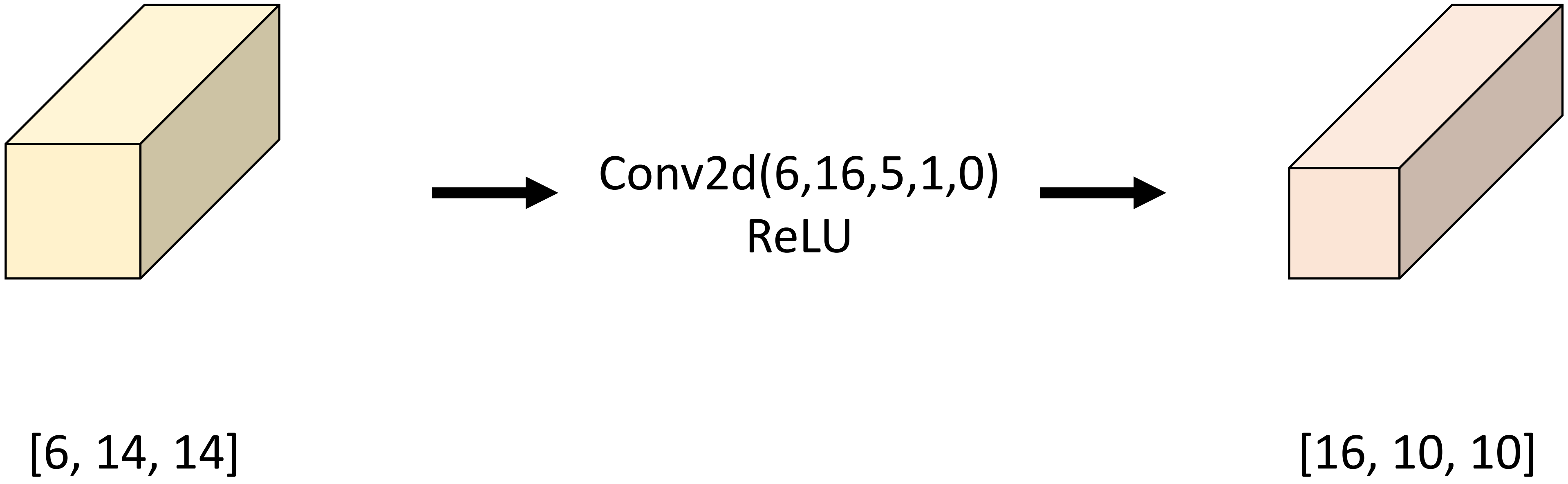
[1. LeNet-5]


6
input channel
output channel
kernel size
stride
padding
Conv2d(3,6,5,1,0)
ReLU
[3, 32, 32]
[6, 28, 28]

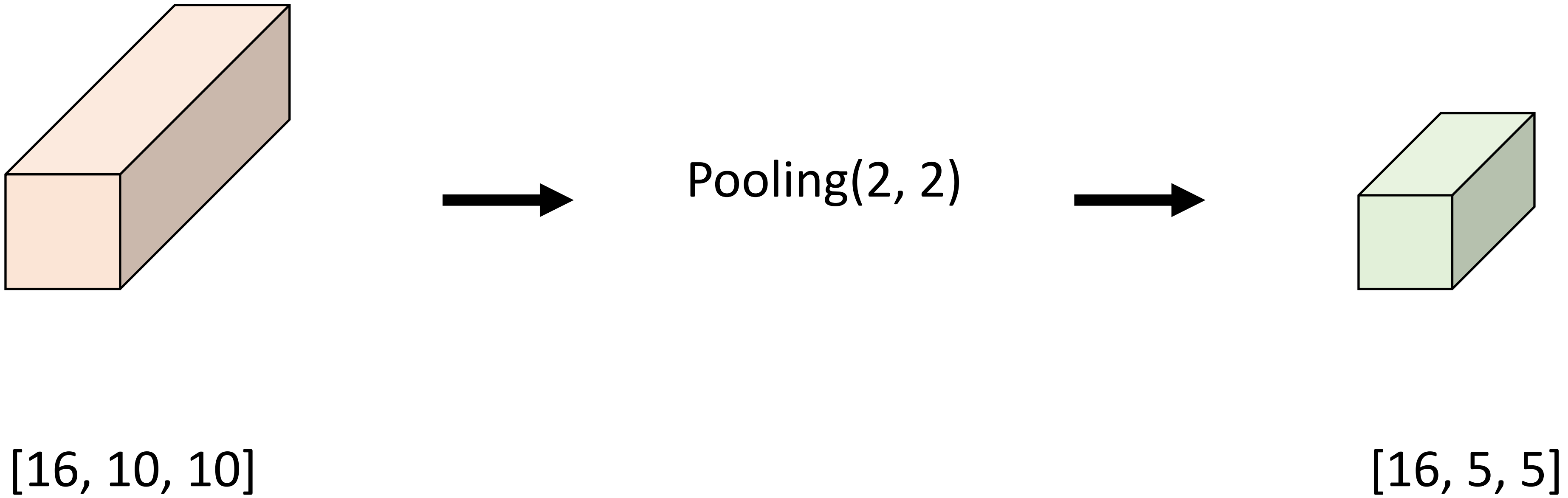
[1. LeNet-5]



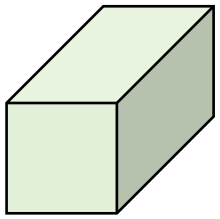
[1. LeNet-5]



[1. LeNet-5]



[1. LeNet-5]



[16, 5, 5]

Reshape([16,5,5]→[16*5*5])
Fully Connected(16*5*5,120)
ReLU
Fully Connected(120,84)
ReLU
Fully Connected(84,10)

[10]

Class size

[2. ResNet-32]

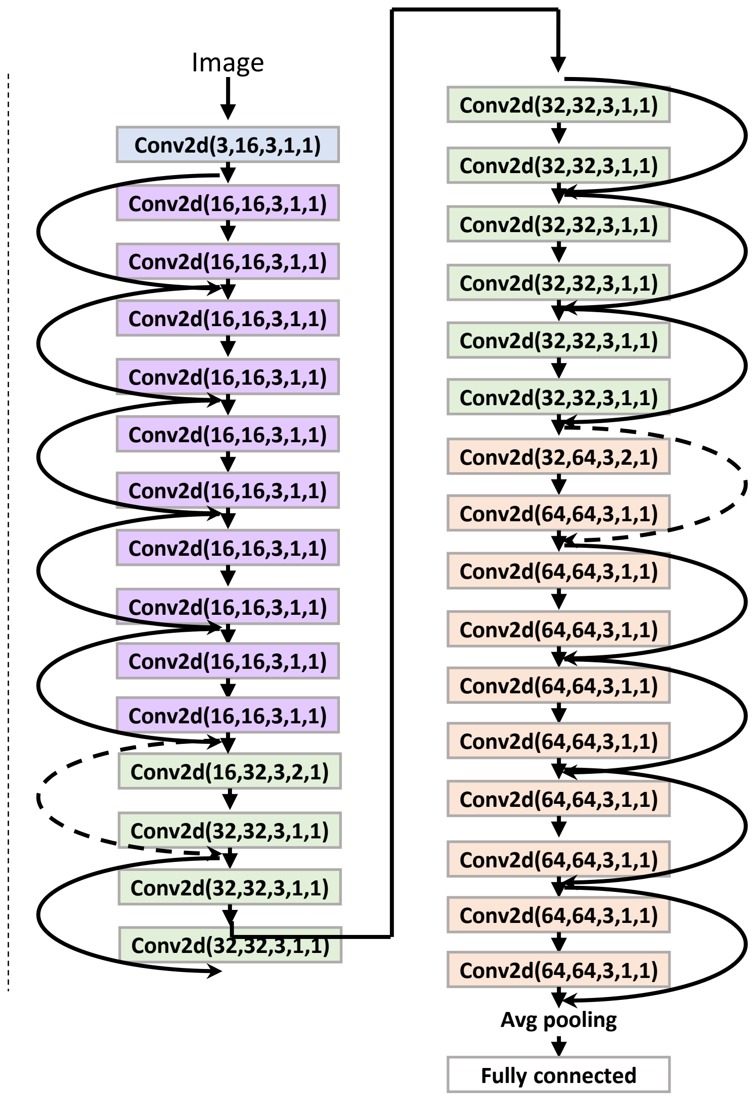


Image
Conv2d(3,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,32,3,2,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,64,3,2,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Avg pooling
Fully connected

[2. ResNet-32]

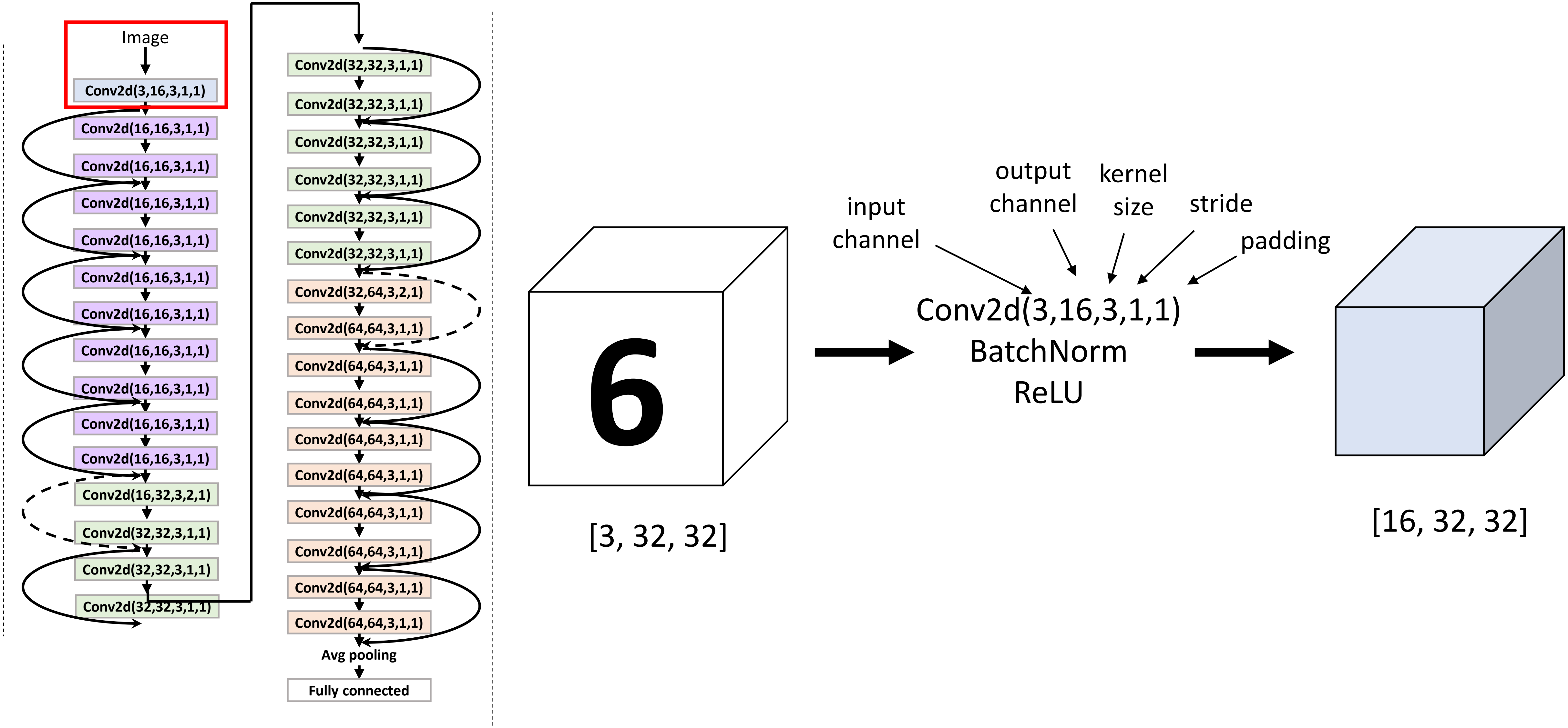


Image
Conv2d(3,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,32,3,2,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,64,3,2,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Avg pooling
Fully connected
6
[3, 32, 32]
input channel
output channel
kernel size
stride
padding
Conv2d(3,16,3,1,1)
BatchNorm
ReLU
[16, 32, 32]

[2. ResNet-32]

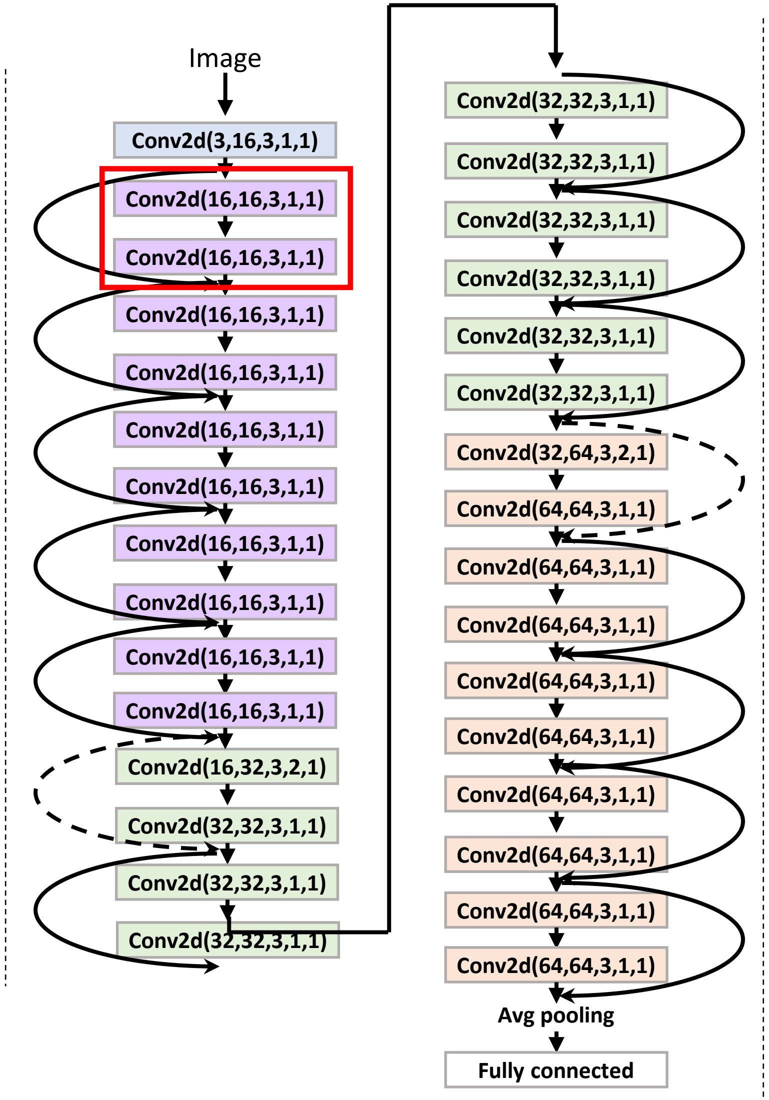
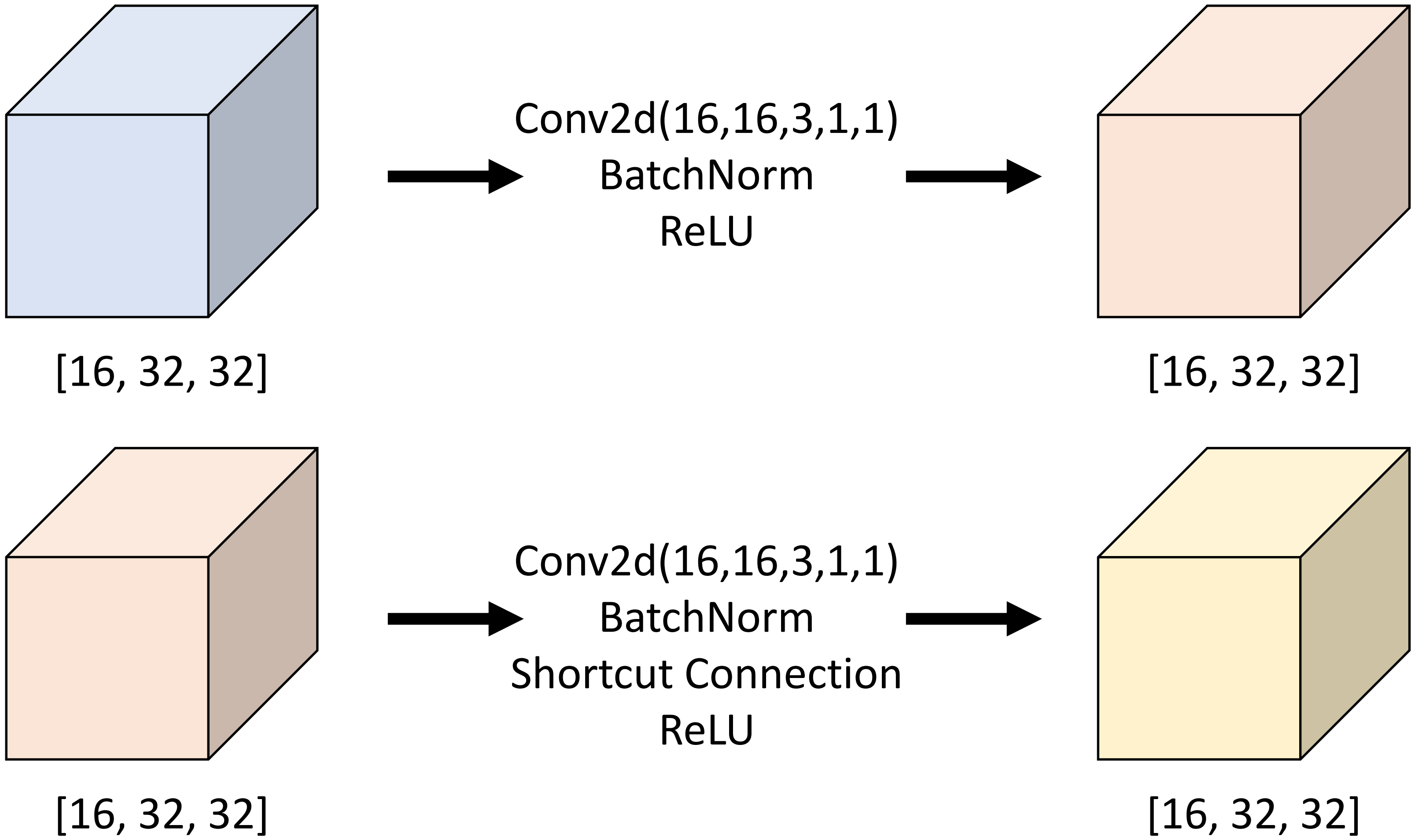
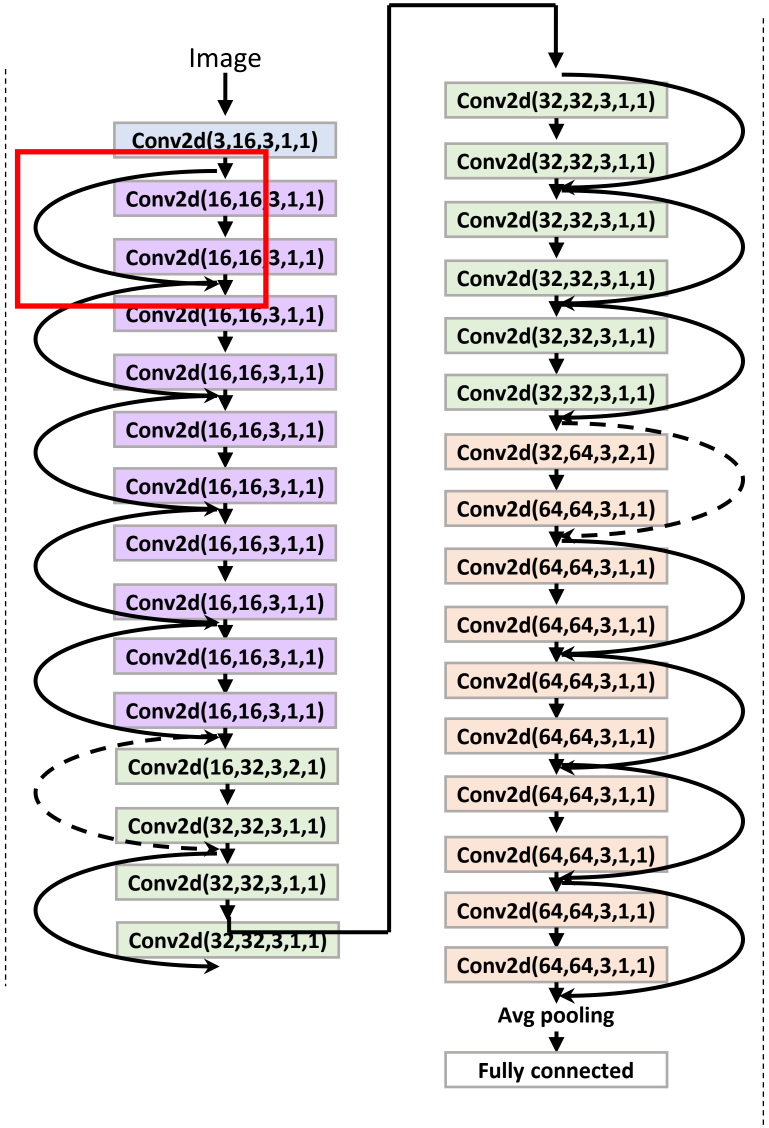


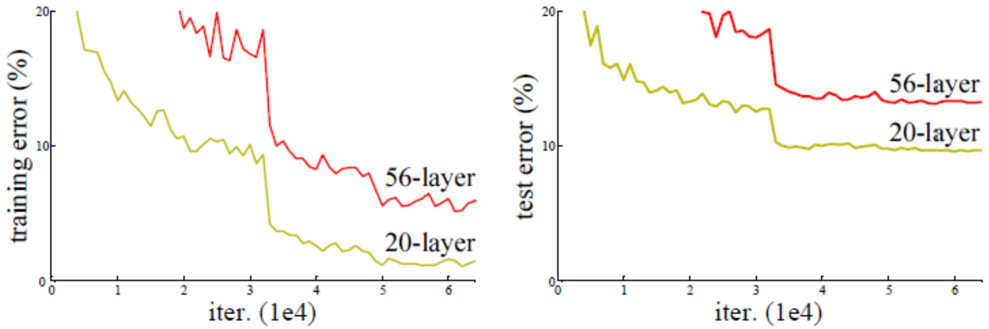
Image
Conv2d(3,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,32,3,2,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,64,3,2,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Avg pooling
Fully connected


Conv2d(16,16,3,1,1)
BatchNorm
ReLU
[16, 32, 32]
[16, 32, 32]
Conv2d(16,16,3,1,1)
BatchNorm
Shortcut Connection
ReLU
[16, 32, 32]
[16, 32, 32]

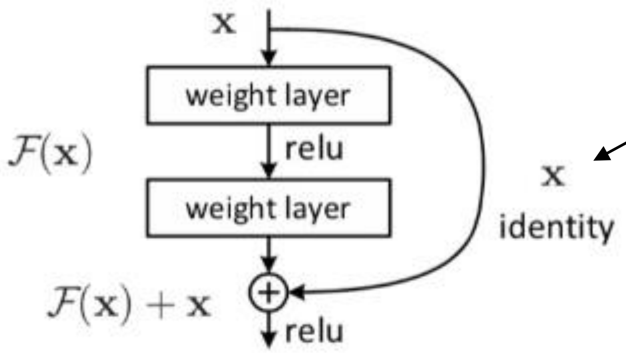
[2. ResNet-32]



- Deeper network does not gives higher accuracy,
  Due to Gradient Vanishing and degradation.



[Residual Learning]



Shortcut Connection

[2. ResNet-32]

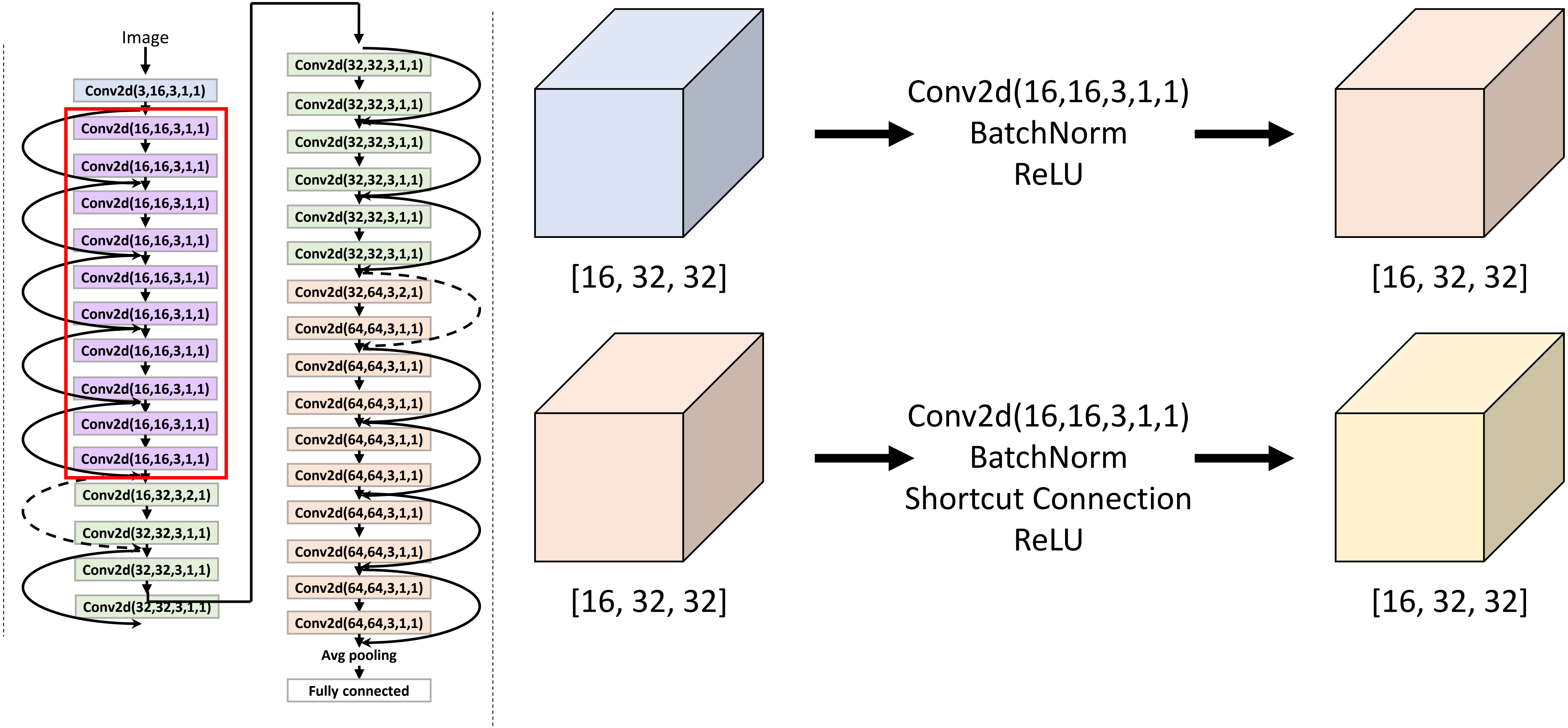


Image
Conv2d(3,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,32,3,2,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,64,3,2,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Avg pooling
Fully connected
Conv2d(16,16,3,1,1)
BatchNorm
ReLU
[16, 32, 32]
[16, 32, 32]
Conv2d(16,16,3,1,1)
BatchNorm
Shortcut Connection
ReLU
[16, 32, 32]
[16, 32, 32]

[2. ResNet-32]

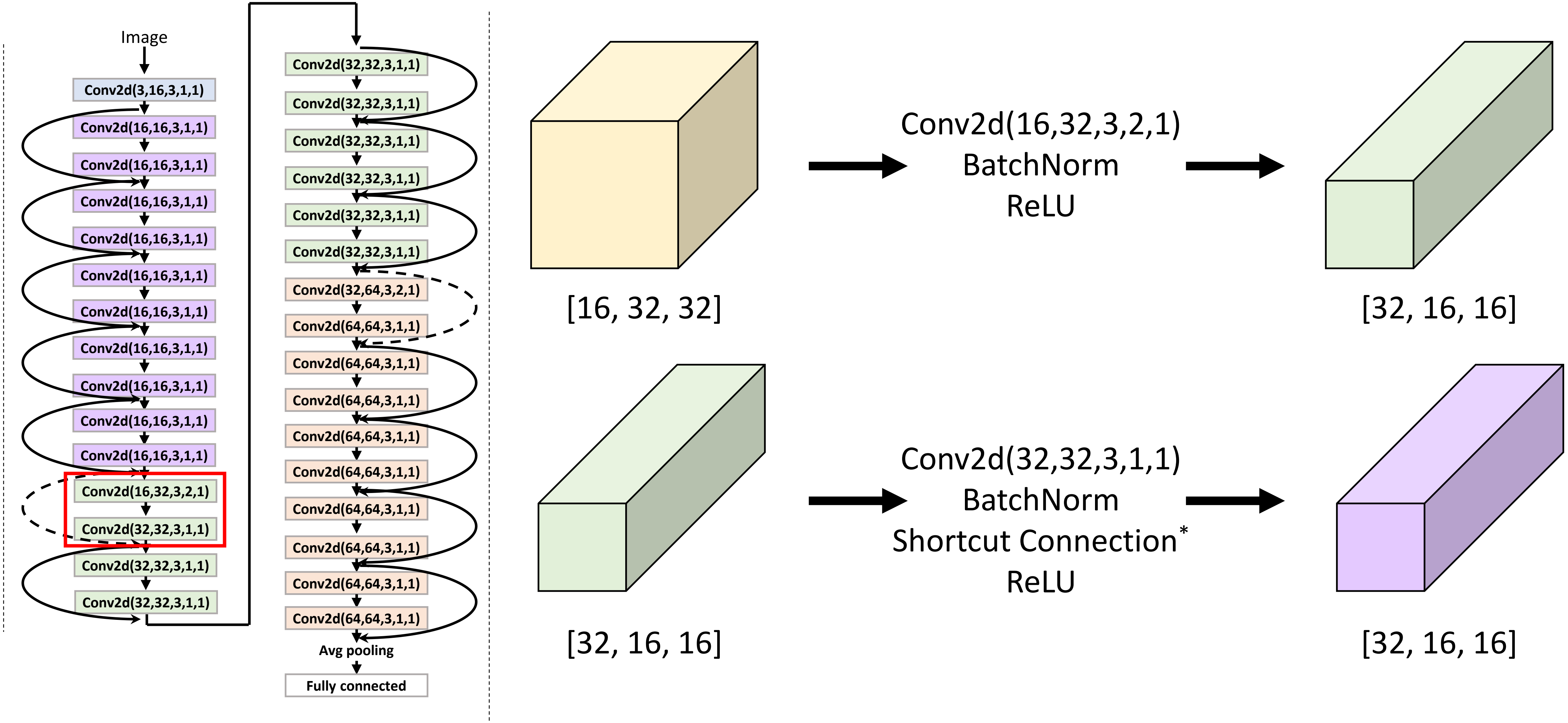
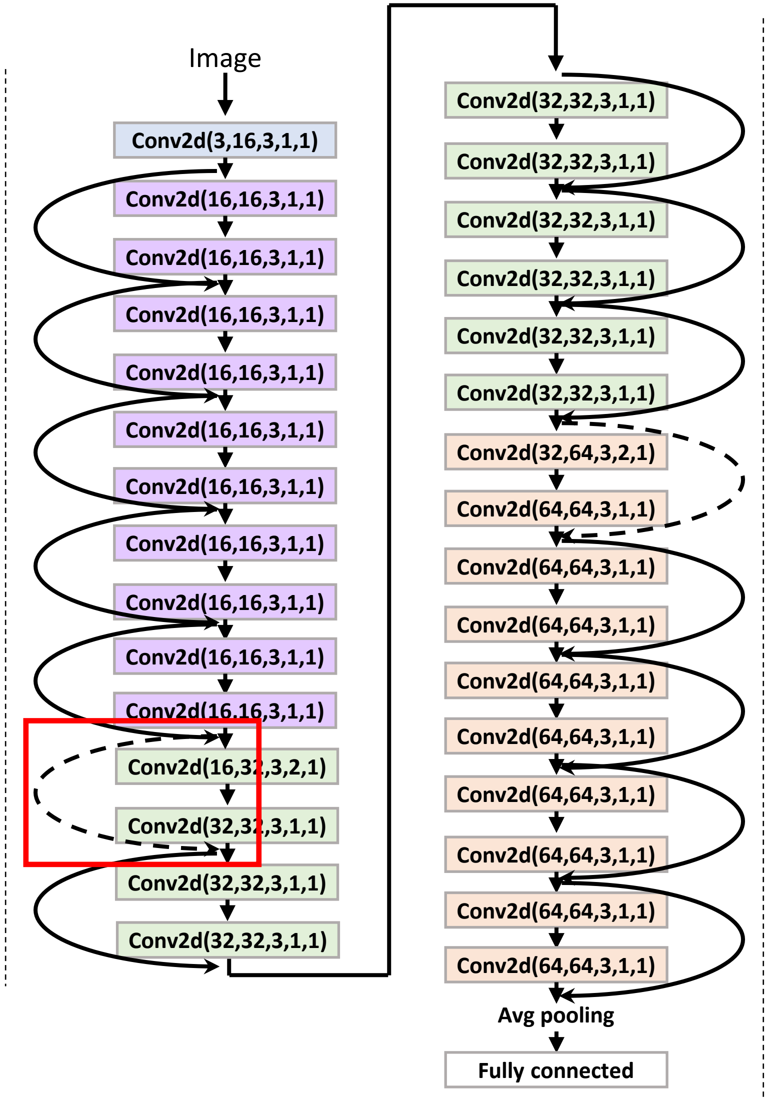
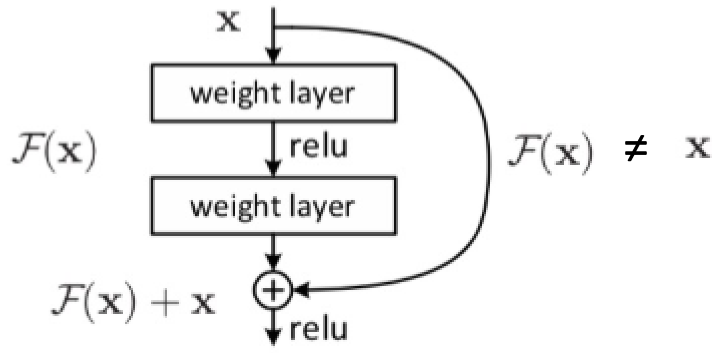


Image
Conv2d(3,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,32,3,2,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,64,3,2,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Avg pooling
Fully connected
[16, 32, 32]
Conv2d(16,32,3,2,1)
BatchNorm
ReLU
[32, 16, 16]
[32, 16, 16]
Conv2d(32,32,3,1,1)
BatchNorm
Shortcut Connection*
ReLU
[32, 16, 16]

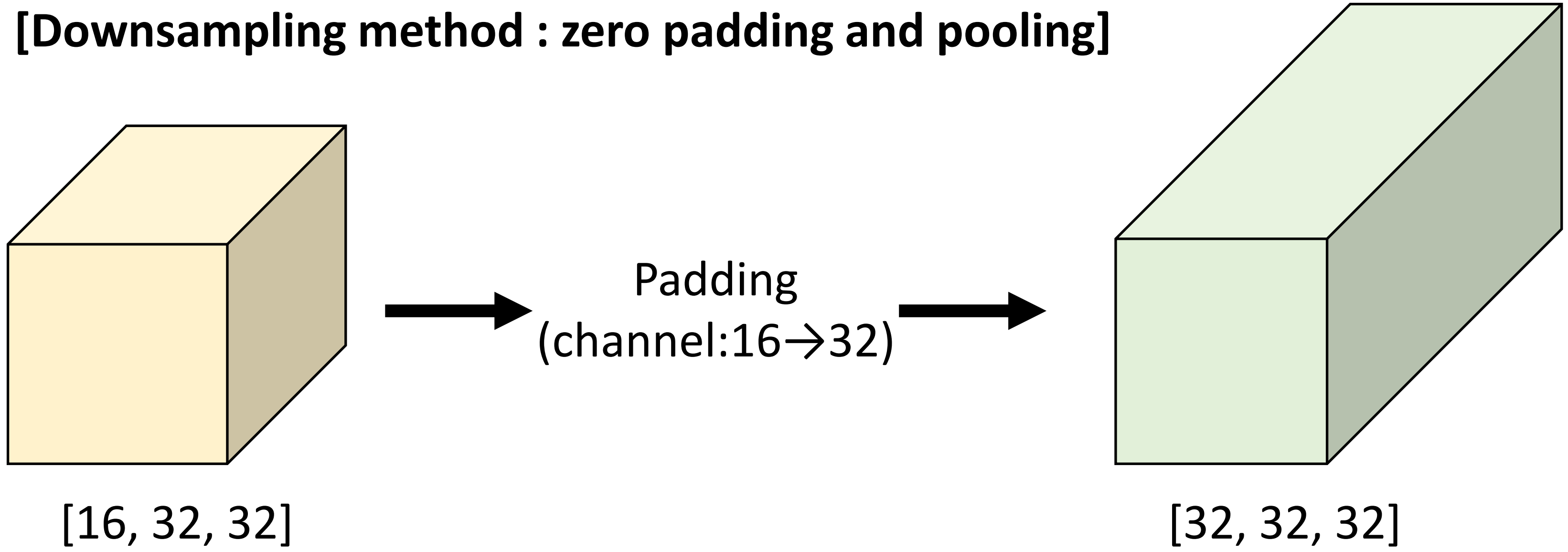
[2. ResNet-32]



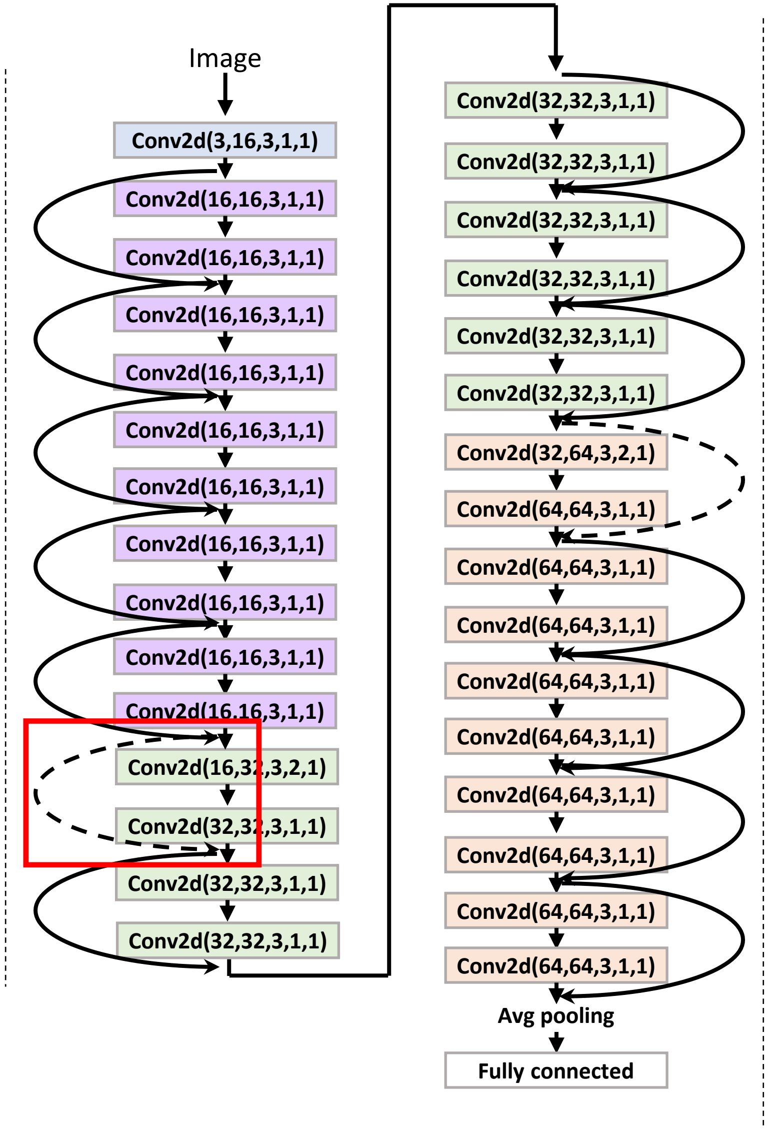
[Shortcut Connection with different shape]



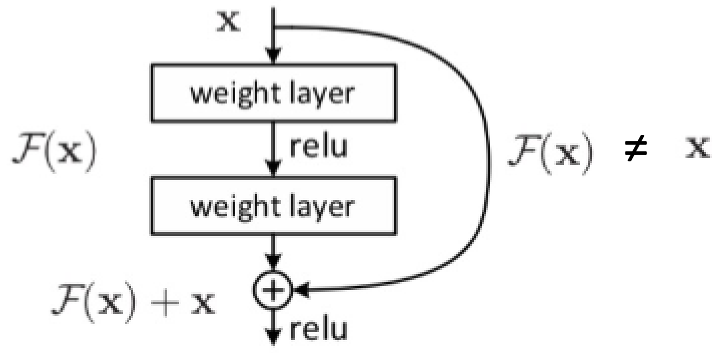
[Downsampling method : zero padding and pooling]



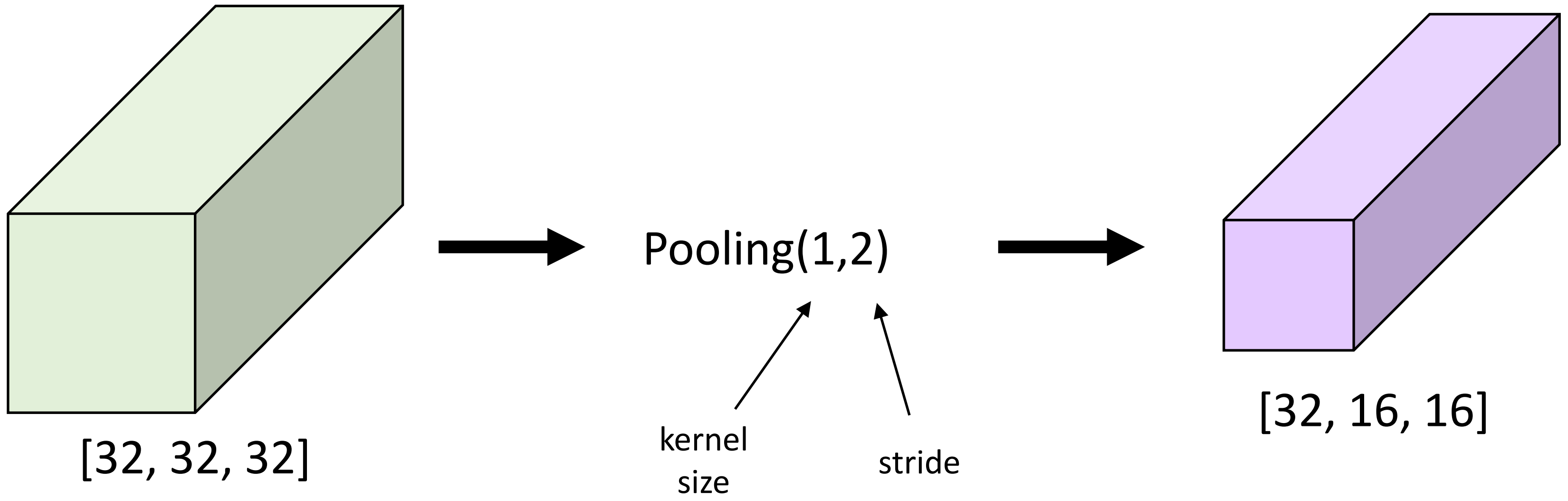
[2. ResNet-32]



[Shortcut Connection with different shape]



[Downsampling method : zero padding and pooling]



[2. ResNet-32]

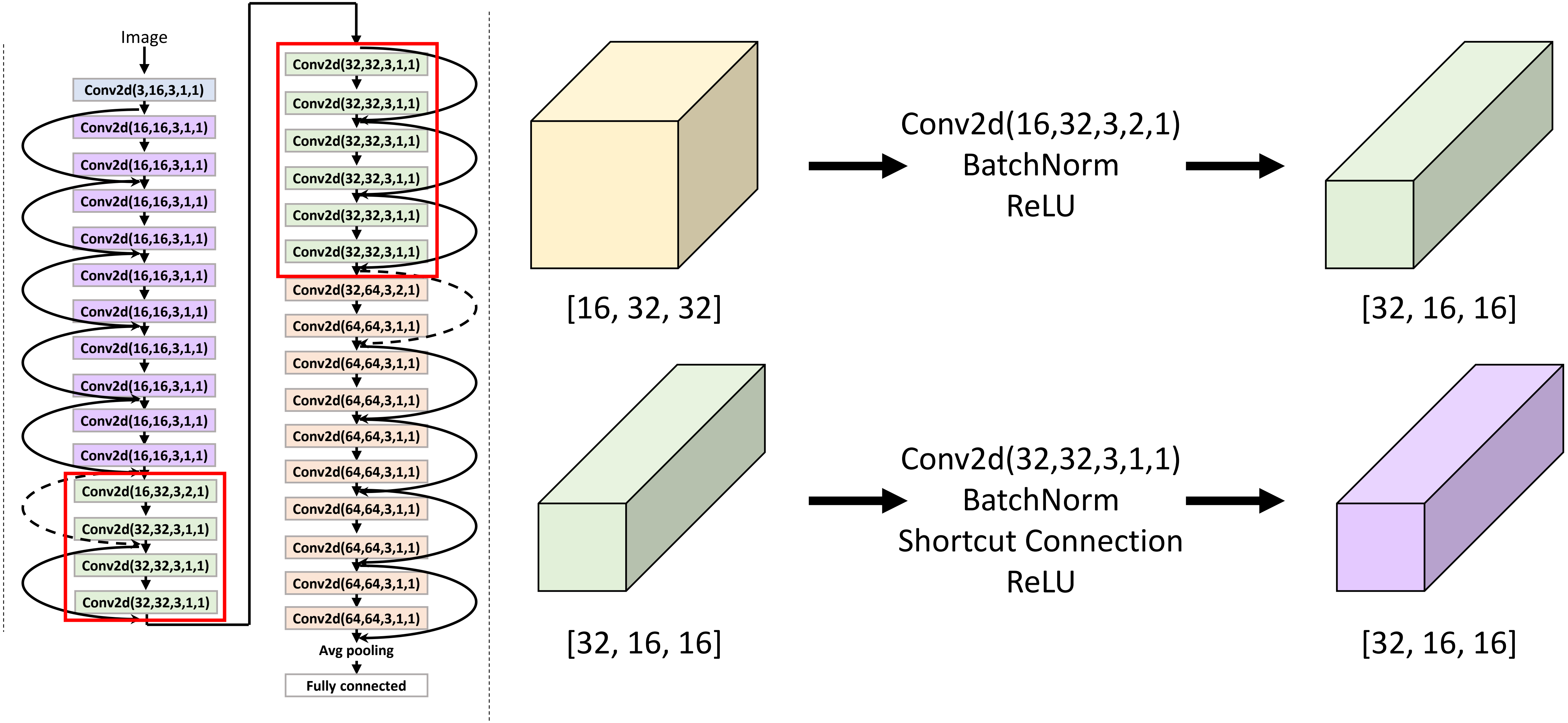


Image
Conv2d(3,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,32,3,2,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,64,3,2,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Avg pooling
Fully connected
Conv2d(16,32,3,2,1)
BatchNorm
ReLU
[16, 32, 32]
[32, 16, 16]
Conv2d(32,32,3,1,1)
BatchNorm
Shortcut Connection
ReLU
[32, 16, 16]
[32, 16, 16]

[2. ResNet-32]

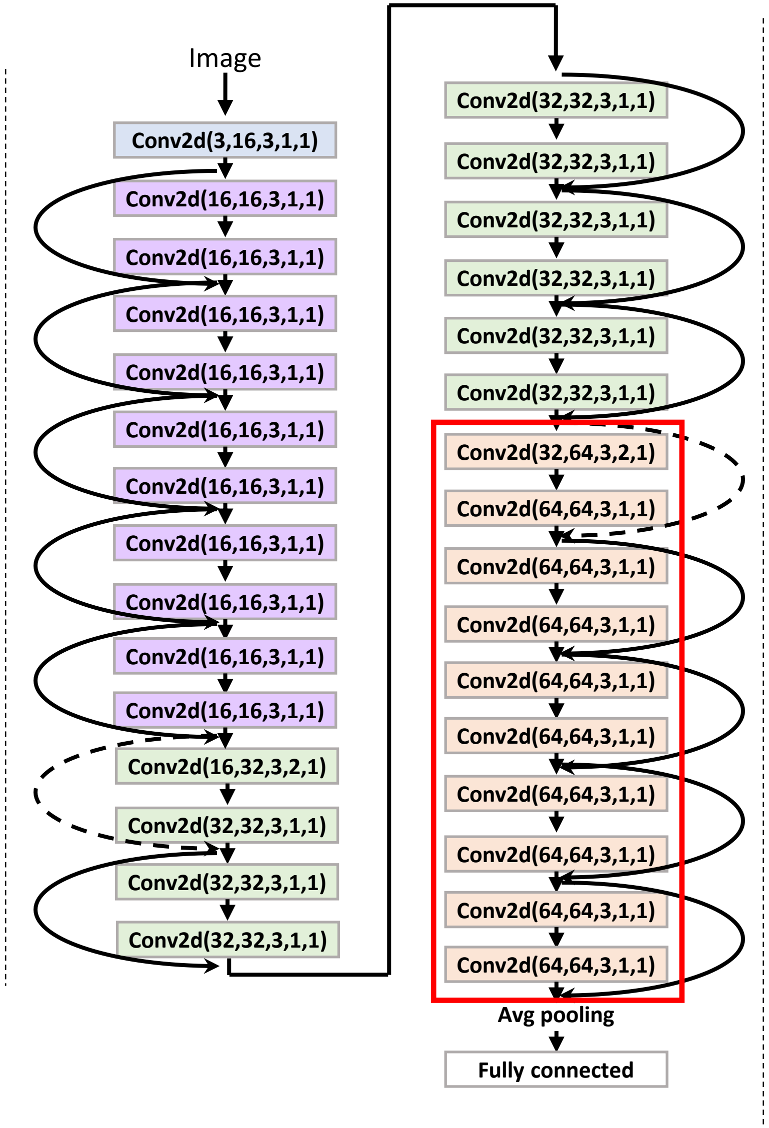
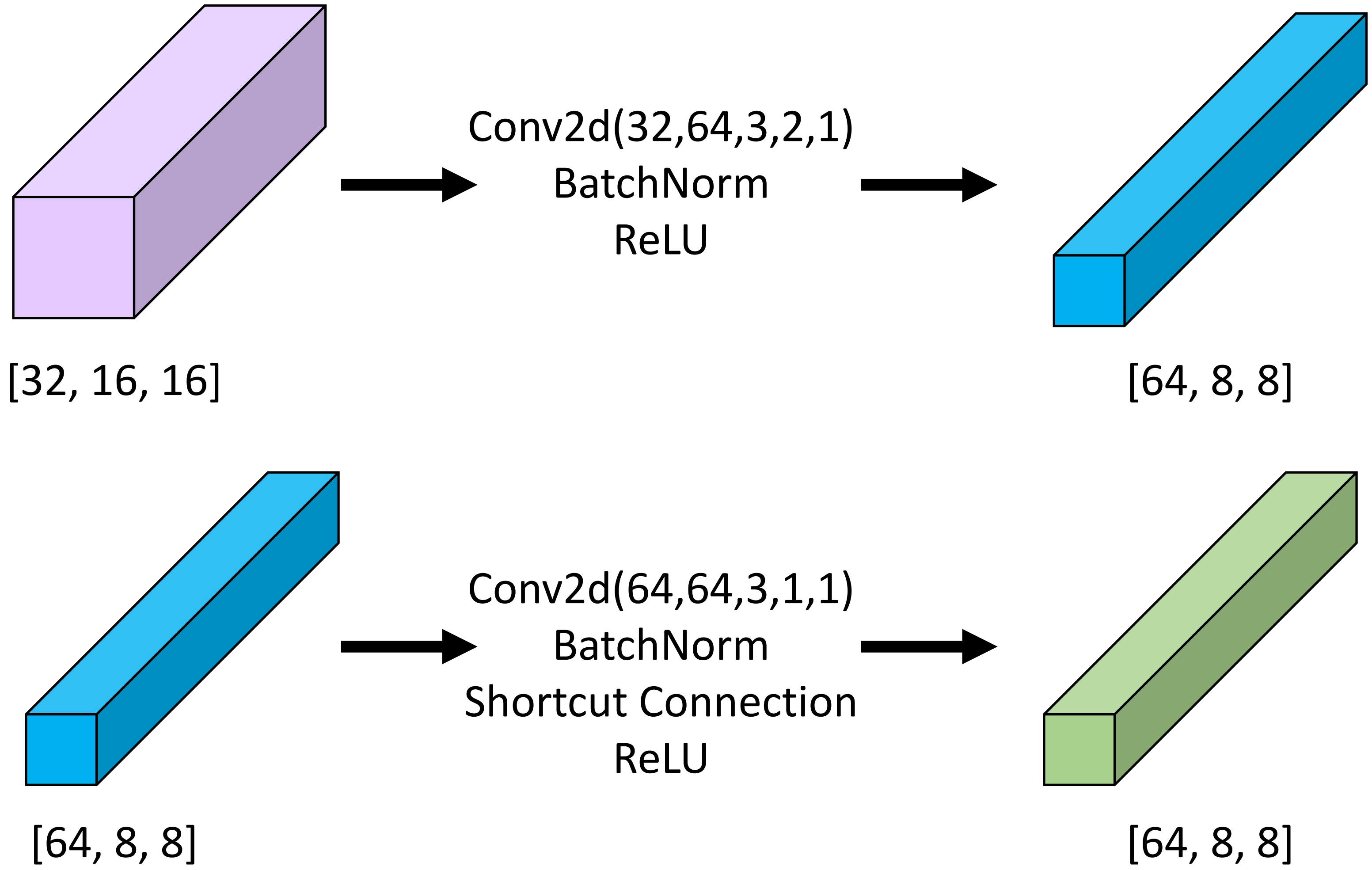


Image
Conv2d(3,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,32,3,2,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,64,3,2,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Avg pooling
Fully connected



Conv2d(32,64,3,2,1)
BatchNorm
ReLU
[32, 16, 16]
[64, 8, 8]
Conv2d(64,64,3,1,1)
BatchNorm
Shortcut Connection
ReLU
[64, 8, 8]
[64, 8, 8]

[2. ResNet-32]

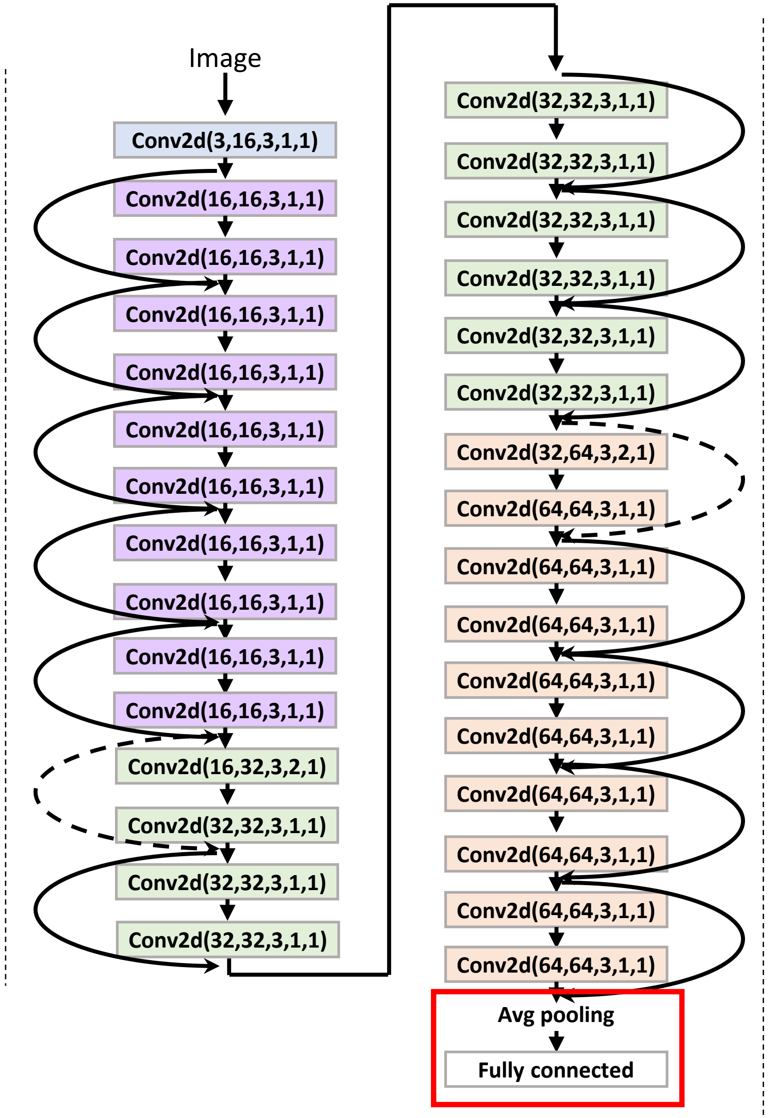
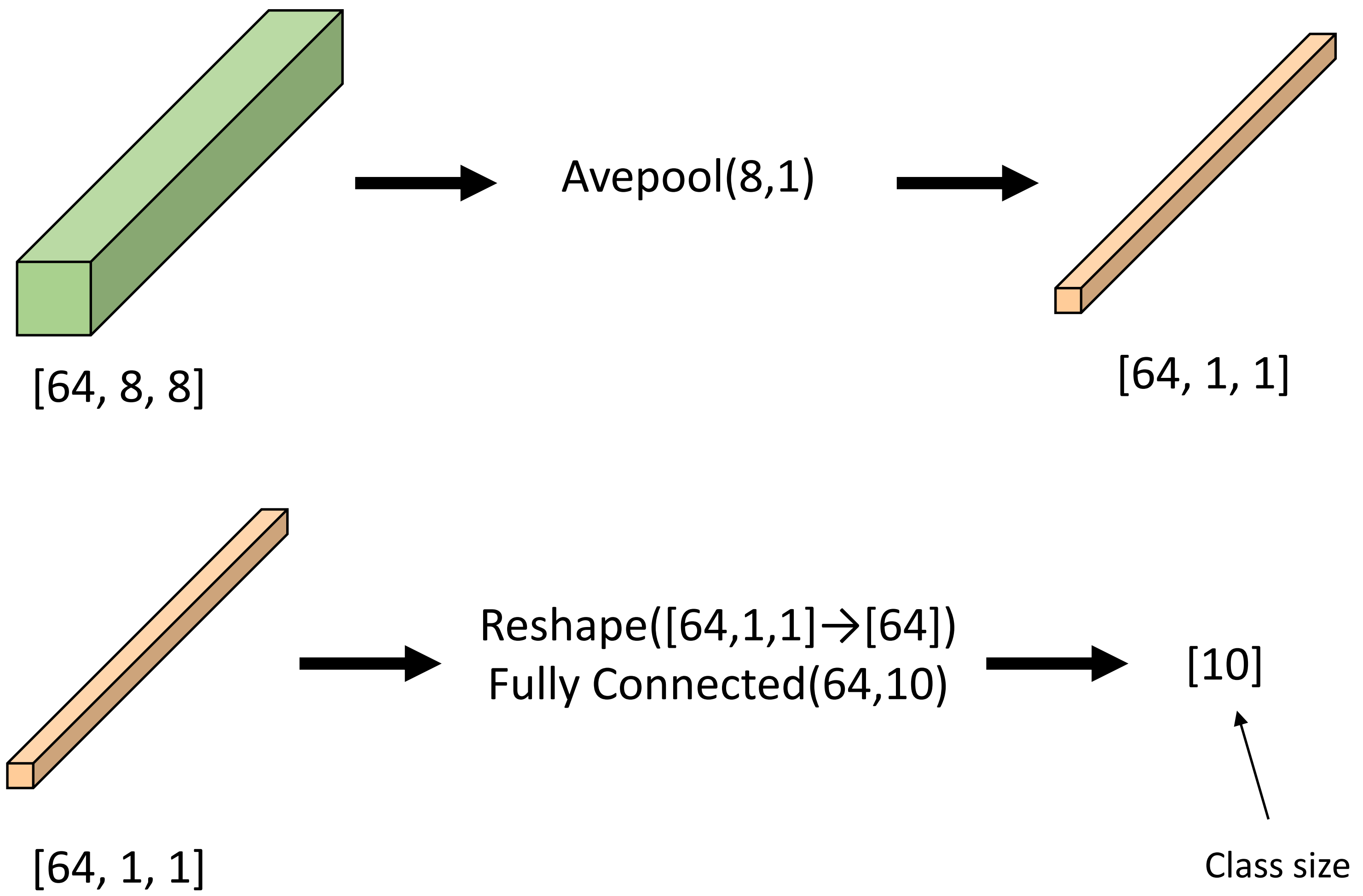


Image
Conv2d(3,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,16,3,1,1)
Conv2d(16,32,3,2,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,32,3,1,1)
Conv2d(32,64,3,2,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Conv2d(64,64,3,1,1)
Avg pooling
Fully connected



[64, 8, 8]
Avepool(8,1)
[64, 1, 1]
[64, 1, 1]
Reshape([64,1,1]→[64])
Fully Connected(64,10)
[10]
Class size

Code review

[Objective]

Your model should classifiy of the images into 10 classes.

[Classes]

classes = ('plane', 'car', 'bird', 'cat', 'deer',
'dog', 'frog', 'horse', 'ship', 'truck')

[Code structure]

- CIFAR10_configuration.py
- CIFAR10_evaluation.py
- CIFAR10_train.py
- LeNet5_model.py
- MLP_model.py
- ResNet_model.py

[CIFAR10_train.py]

```python
def data_load():
    # train data augmentation : 1) 데이터 좌우반전(2배). 2) size 4만큼 패딩 후 32의 크기로 random cropping
    transforms_train = transforms.Compose([
        transforms.RandomCrop(32, padding=4),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
    ])
    transforms_val = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
    ])

    # CIFAR10 dataset 다운로드
    train_data = dsets.CIFAR10(root='./dataset/', train=True, transform=transforms_train, download=True)
    val_data = dsets.CIFAR10(root="./dataset/", train=False, transform=transforms_val, download=True)

    return train_data, val_data
```

```python
if cfg.modelname == "MLP":
    ################################################################################################
    #                   TODO : 모델 학습을 위한 optimizer 정의                                        #
    ################################################################################################
    pass
    ################################################################################################
    #                                        END OF YOUR CODE                                      #
    ################################################################################################

elif cfg.modelname == "LeNet5":
    ################################################################################################
    #                   TODO : 모델 학습을 위한 optimizer 정의                                        #
    ################################################################################################
    pass
    ################################################################################################
    #                                        END OF YOUR CODE                                      #
    ################################################################################################
elif cfg.modelname == "ResNet32":
    optimizer = optim.SGD(model.parameters(), lr=cfg.lr, momentum=cfg.momentum, weight_decay=cfg.weight_decay)
    decay_epoch = [32000, 48000]
    step_lr_scheduler = lr_scheduler.MultiStepLR(optimizer,
                                                 milestones=decay_epoch, gamma=0.1)
```

Code review

[MLP_model.py]

```python
import torch.nn as nn


class MLP_model(nn.Module):
    def __init__(self):
        super().__init__()

        ################################################################################################
        #                          TODO : MLP 모델 생성 (구조는 실험해 보면서 결과가 좋은 것으로 사용할 것)          #
        ################################################################################################
        pass
        ################################################################################################
        #                                        END OF YOUR CODE                                      #
        ################################################################################################

    def forward(self, x):
        ################################################################################################
        #                          TODO : forward path 수행, 결과를 x에 저장                               #
        ################################################################################################
        pass
        ################################################################################################
        #                                        END OF YOUR CODE                                      #
        ################################################################################################
        return x
```

Code review

[LeNet5_model.py]

```python
import torch
import torch.nn as nn
import torch.nn.functional as F


class LeNet5_model(nn.Module):
    def __init__(self):
        super().__init__()
        ###############################################################################################################
        #                         TODO : LeNet5 모델 생성                                                             #
        ###############################################################################################################
        pass
        ################################################################################################################
        #                                              END OF YOUR CODE                                                #
        ################################################################################################################

    def forward(self, x):
        ###############################################################################################################
        #                         TODO : forward path 수행, 결과를 x에 저장                                           #
        ###############################################################################################################
        pass
        ################################################################################################################
        #                                              END OF YOUR CODE                                                #
        ################################################################################################################
        return x
```

Code review

[ResNet32_model.py]

You should find correct number or variable for X1~X10.

```python
def forward(self, x):
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.layers_2n(x)
    x = self.layers_4n(x)
    x = self.layers_6n(x)
    x = self.pool(x)
    x = x.view(x.size(0), -1)
    x = self.fc_out(x)
    return x
```

```python
class ResNet(nn.Module):

    def __init__(self, num_layers, block, num_classes=10):
        super().__init__()
        # input img : [3, 32, 32]
        self.num_layers = num_layers
        self.conv1 = nn.Conv2d(in_channels=X7, out_channels=X8,
                               kernel_size=X9, padding=X10, bias=False)
        self.bn1 = nn.BatchNorm2d(16)
        self.relu = nn.ReLU(inplace=True)

        # feature map size = [16,32,32]
        self.layers_2n = self.get_layers(block, 16, 16, stride=1)
        # feature map size = [32,16,16]
        self.layers_4n = self.get_layers(block, 16, 32, stride=2)
        # feature map size = [64,8,8]
        self.layers_6n = self.get_layers(block, 32, 64, stride=2)

        # output layers
        self.pool = nn.AvgPool2d(8, stride=1)
        self.fc_out = nn.Linear(64, num_classes)

        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                nn.init.kaiming_normal_(m.weight, mode='fan_out', nonlinearity='relu')
            if isinstance(m, nn.BatchNorm2d):
                nn.init.constant_(m.weight, 1)
                nn.init.constant_(m.bias, 0)

    def get_layers(self, block, in_channels, out_channels, stride):
        if stride == 2:
            down_sample = True
        else:
            down_sample = False

        layers_list = nn.ModuleList([block(in_channels, out_channels, stride, down_sample)])

        for _ in range(self.num_layers - 1):
            layers_list.append(block(out_channels, out_channels))

        return nn.Sequential(*layers_list)
```

Code review

[ResNet32_model.py]

You should find correct number or variable for X1~X10.

```python
class IdentityPadding(nn.Module):
    def __init__(self, in_channels, out_channels, stride):
        super().__init__()
        self.pooling = nn.MaxPool2d(kernel_size=1, stride=stride)
        self.add_channels = out_channels - in_channels

    def forward(self, x):
        x = F.pad(x, [0, 0, 0, 0, 0, self.add_channels])
        x = self.pooling(x)
        return x
```

```python
class ResidualBlock(nn.Module):
    def __init__(self, in_channels, out_channels, stride=1, down_sample=False):
        super().__init__()

        self.conv1 = nn.Conv2d(in_channels, out_channels,
                               kernel_size=X1, stride=X2,
                               padding=X3, bias=False)
        self.bn1 = nn.BatchNorm2d(out_channels)
        self.relu = nn.ReLU(inplace=True)

        self.conv2 = nn.Conv2d(out_channels, out_channels,
                               kernel_size=X4, stride=X5,
                               padding=X6, bias=False)
        self.bn2 = nn.BatchNorm2d(out_channels)
        self.stride = stride
        if down_sample:
            self.down_sample = IdentityPadding(in_channels, out_channels, stride)
        else:
            self.down_sample = None

    def forward(self, x):
        shortcut = x
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu(x)

        x = self.conv2(x)
        x = self.bn2(x)

        if self.down_sample is not None:
            shortcut = self.down_sample(shortcut)

        x += shortcut
        x = self.relu(x)
        return x
```

Evaluation Report

CIFAR-10 Evaluation Report													
	Model	GPU(O/X)	Batch_size	Activation function	Weight initialization	Optimizer	Learning rate	Momentum	Weight decay	LR decay	training time (m)	Early stopping epoch	Accuracy
Setting #1	MLP												
Setting #2	Lenet5												
Setting #3	ResNet32		128	ReLU	He_normal	SGD	0.1	0.9	O	O			

Validation dataset accuracy plot

Setting #1

Setting #2

Setting #3

[결과 정리]

Evaluation Report

Fill in these cells.

CIFAR-10 Evaluation Report

	Model	GPU(O/X)	Batch_size	Activation function	Weight initialization	Optimizer	Learning rate	Momentum	Weight decay	LR decay	training time (m)	Early stopping epoch	Accuracy
Setting #1	MLP												
Setting #2	Lenet5												
Setting #3	ResNet32		128	ReLU	He_normal	SGD	0.1	0.9	O	O			

Validation dataset accuracy plot

Setting #1 Setting #2 Setting #3

[결과 정리]

Evaluation Report

Plot an accuracy plot of the validation dataset for each setting.

CIFAR-10 Evaluation Report													
	Model	GPU(O/X)	Batch_size	Activation function	Weight initialization	Optimizer	Learning rate	Momentum	Weight decay	LR decay	training time (m)	Early stopping epoch	Accuracy
Setting #1	MLP												
Setting #2	Lenet5												
Setting #3	ResNet32		128	ReLU	He_normal	SGD	0.1	0.9	O	O			

Validation dataset accuracy plot

Setting #1 Setting #2 Setting #3

[결과 정리]

Evaluation Report

CIFAR-10 Evaluation Report													
	Model	GPU(O/X)	Batch_size	Activation function	Weight initialization	Optimizer	Learning rate	Momentum	Weight decay	LR decay	training time (m)	Early stopping epoch	Accuracy
Setting #1	MLP												
Setting #2	Lenet5												
Setting #3	ResNet32		128	ReLU	He_normal	SGD	0.1	0.9	O	O			

Validation dataset accuracy plot

Setting #1 | Setting #2 | Setting #3

Summarize the report of each experimental setting.

[결과 정리]

Assignment #3 : CIFAR-10

- **Objective**

Your model should classifiy of the images into 10 classes.

- **Requirements**

1. Implement multi-layer perceptron with Pytorch or Tensorflow.

(Basic Pytorch code is provided)

(Find the best hyperparameters condition)

2. Implement LeNet5 model.

(Find the best hyperparameters condition)

3. Find the correct X1~X10 values in ResNet32 model and complete the code.

4. Implement with 3 settings stated in the evaluation report, and report the result of each settings.

5. You should attach the plot of the validation dataset accuracy plot. (implemented in pytorch code)

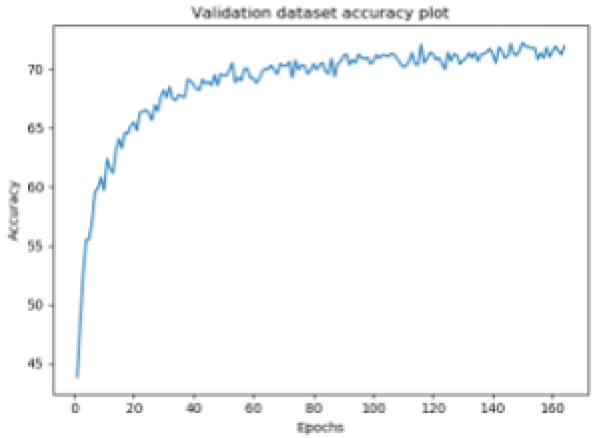
6. You should report the experimental results.

(all kinds of additional experiments are recommended)

model

"Truck"

[Validation dataset accuracy plot]



- **Evaluation Criteria**

Simplicity	How concisely did you write the code?
Performance	How well did the results of the code perform?
Brevity and Clarity	How concisely and clearly did you explain the results?

Assignment #3 : CIFAR-10

- Due to : **~ 5.3(Sun)**

- Submission : Online submission on blackboard

- Your submission should contain
  1) The whole code of your implementation
  2) The evaluation report

- You must implement the components yourself!
- File name : StudentID_Name.zip

Q & A

조교 김도현 : dhkim1028@korea.ac.kr