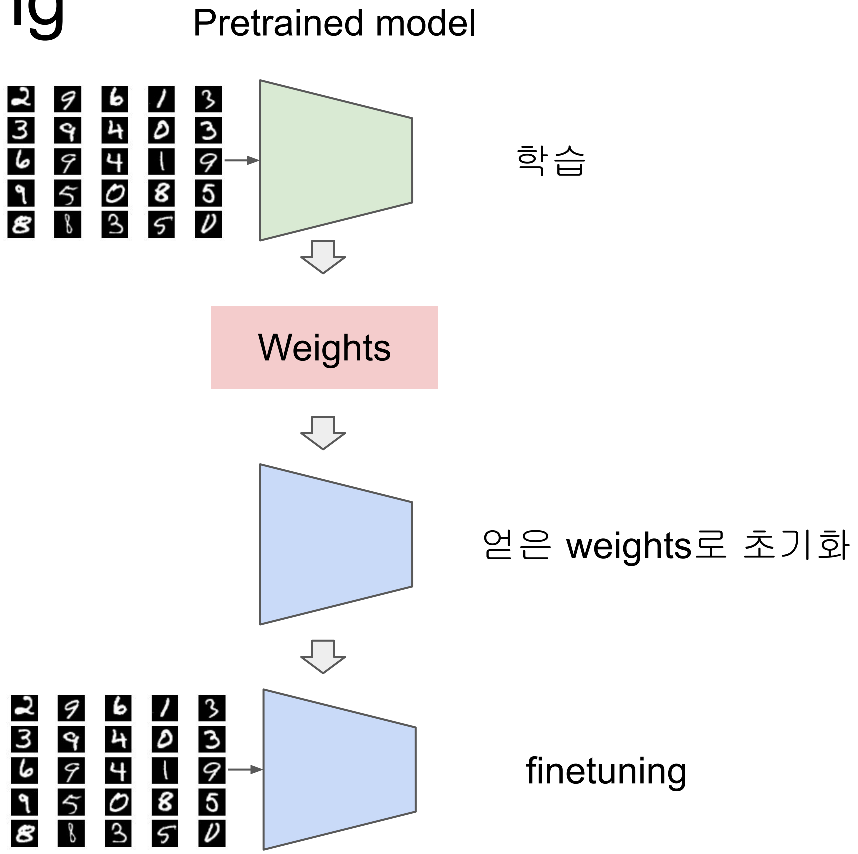
딥러닝 이해하기

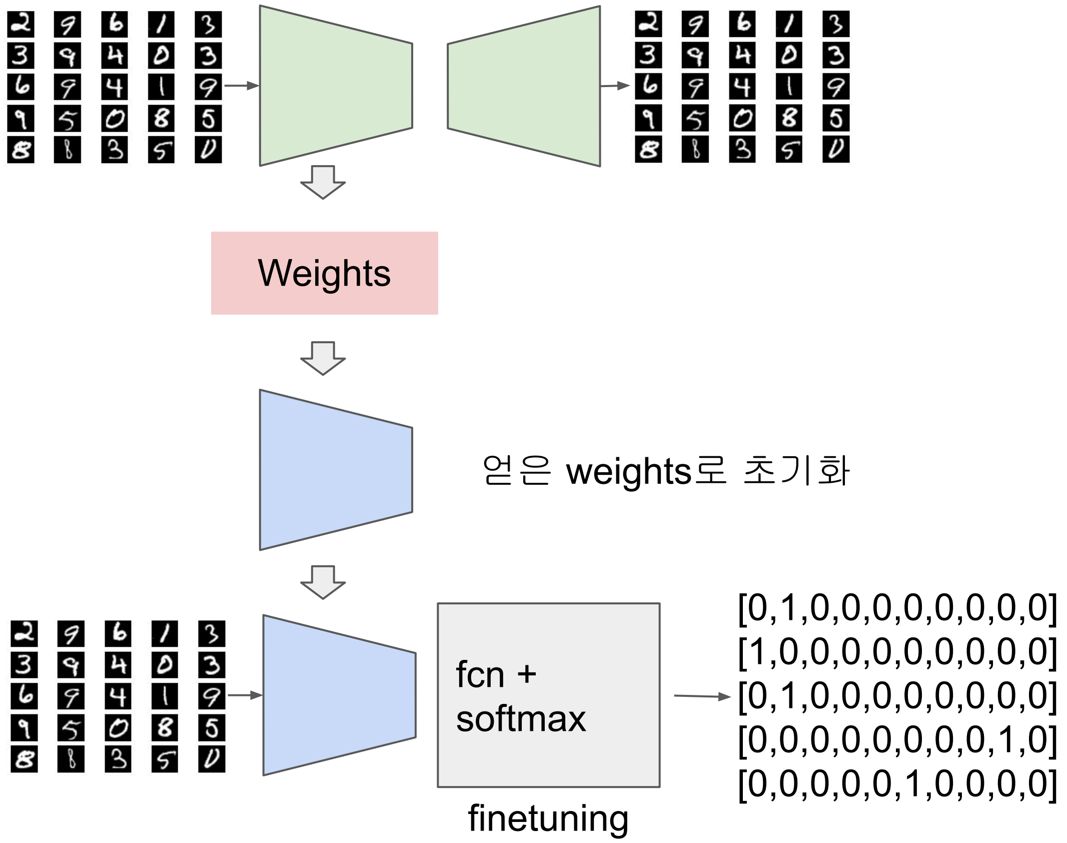
leejeyeol92@gmail.com

Transfer Learning

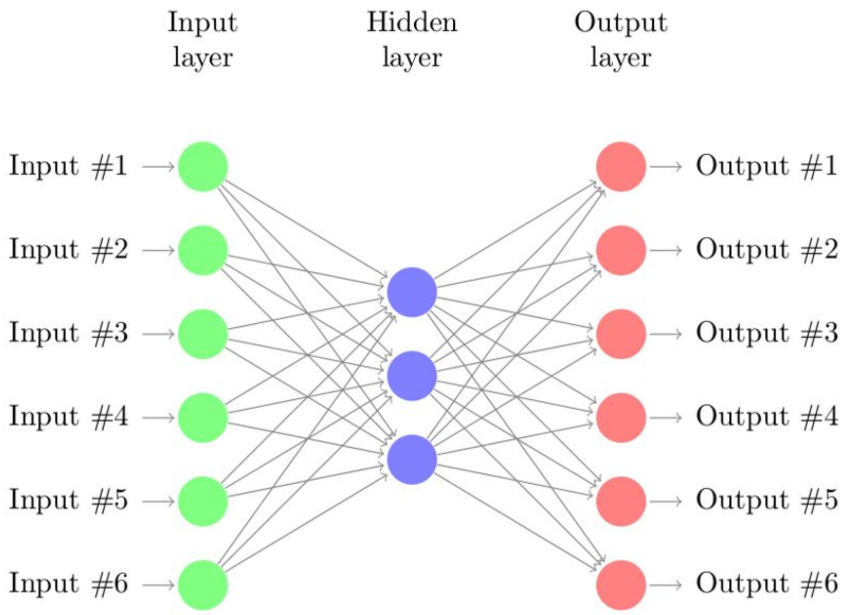
Transfer learning



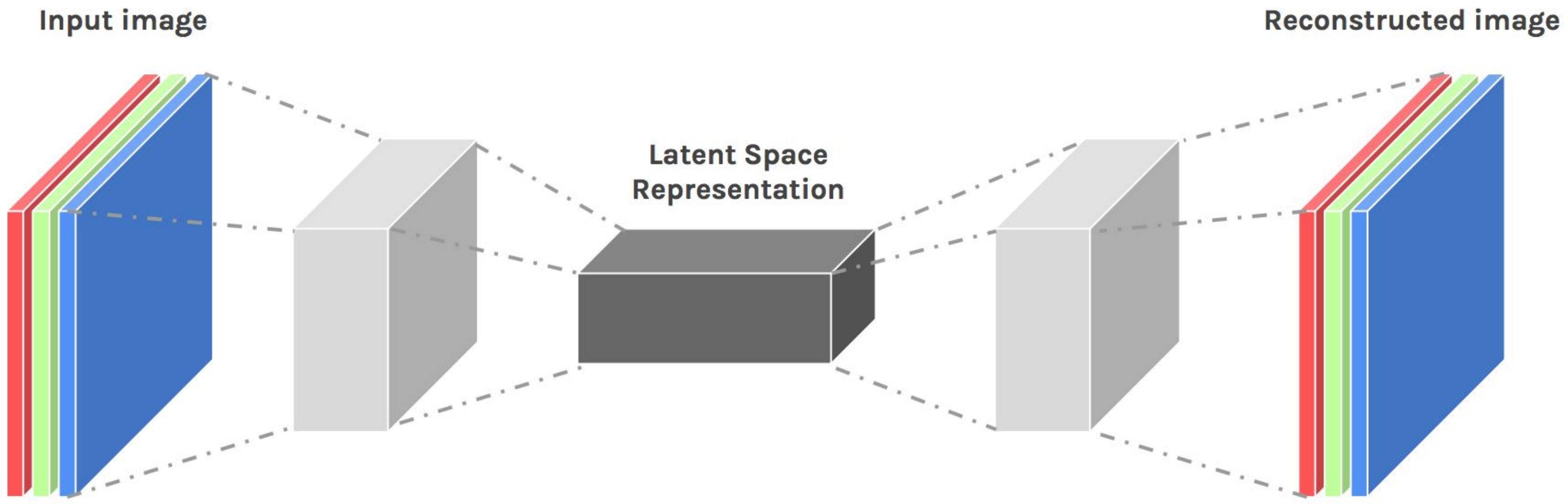
Transfer learning



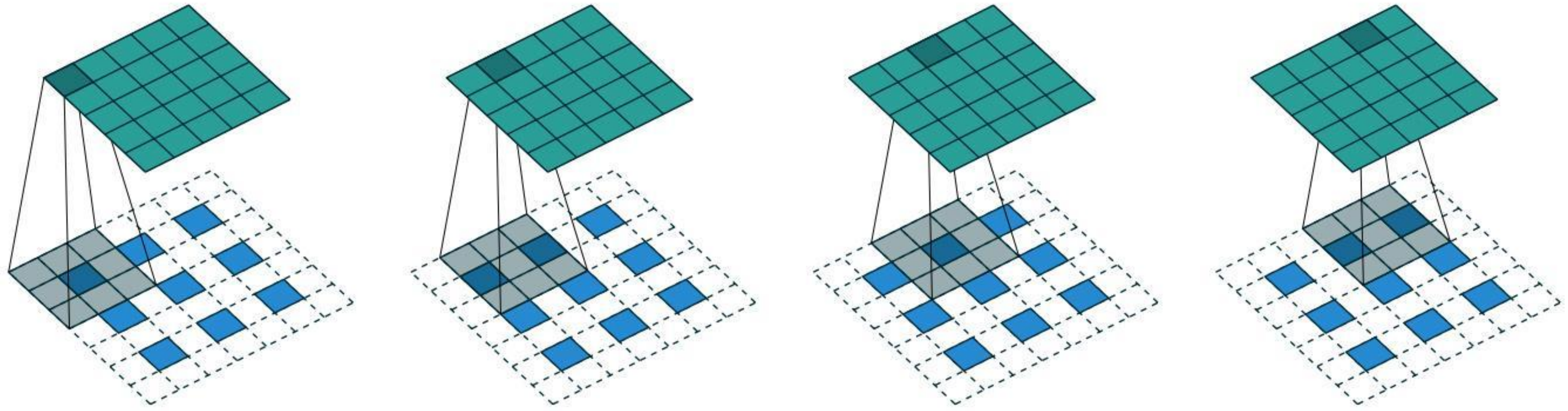
Autoencoder



Autoencoder

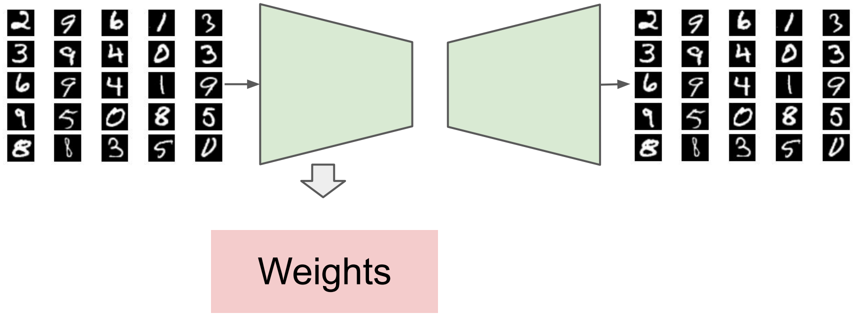


Transposed convolution



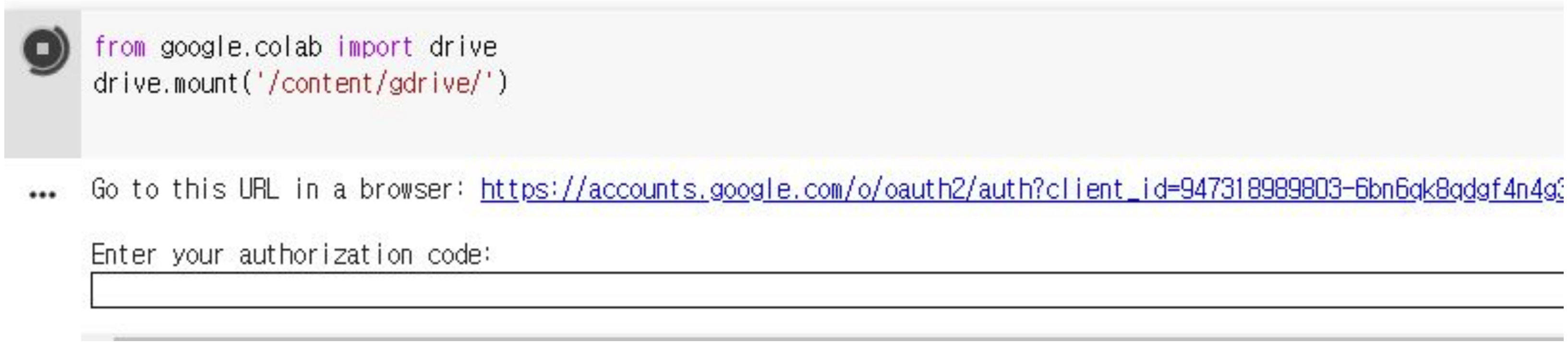
실습 12

CNN Autoencoder 만들고 weight 저장하기



실습 12

colab + 구글드라이브



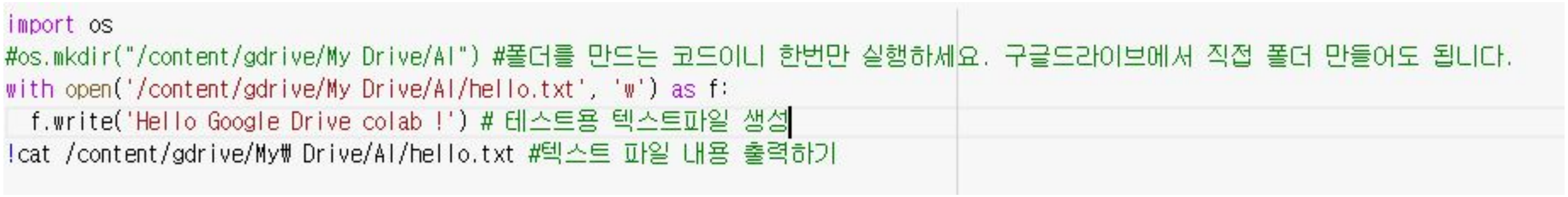
```
from google.colab import drive
drive.mount('/content/gdrive/')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3

Enter your authorization code:
```

링크 클릭 후 로그인.
로그인 이후 나오는 코드를 아래쪽 빈칸에 입력.
`Mounted at /content/gdrive/` 라고 나오면 성공

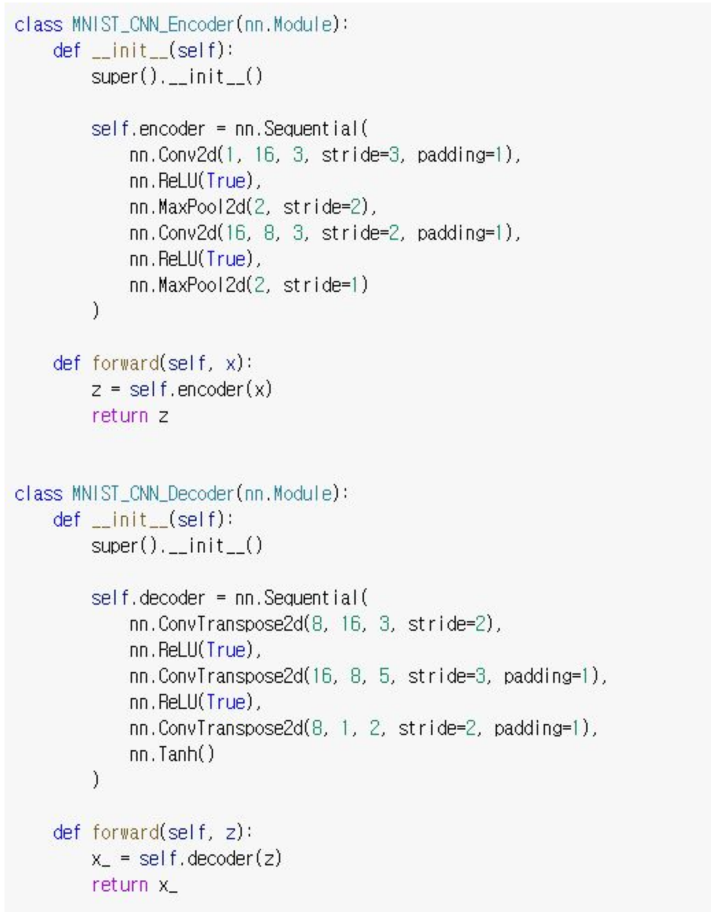
실습 12

colab + 구글드라이브



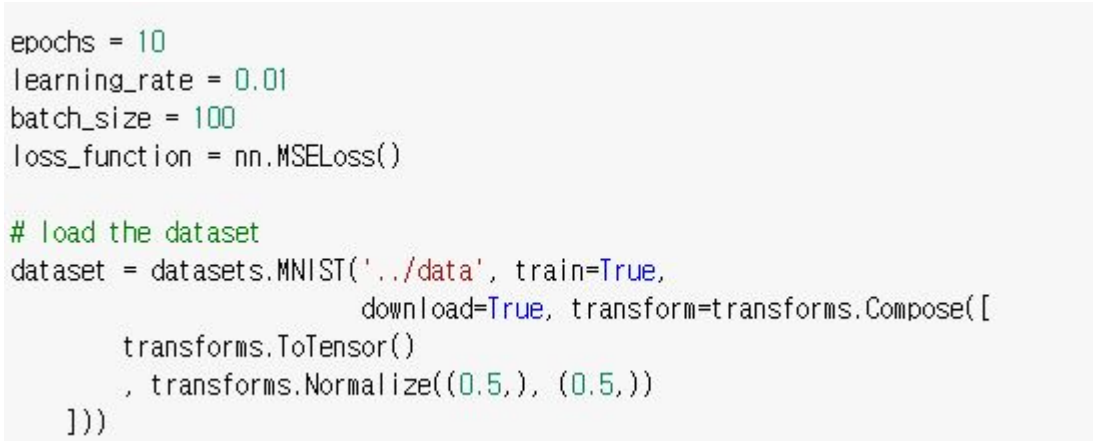
```
import os
#os.mkdir("/content/gdrive/My Drive/AI") #폴더를 만드는 코드이니 한번만 실행하세요. 구글드라이브에서 직접 폴더 만들어도 됩니다.
with open('/content/gdrive/My Drive/AI/hello.txt', 'w') as f:
  f.write('Hello Google Drive colab !') # 테스트용 텍스트파일 생성
!cat /content/gdrive/My\ Drive/AI/hello.txt #텍스트 파일 내용 출력하기
```

실습 12



```python
class MNIST_CNN_Encoder(nn.Module):
    def __init__(self):
        super().__init__()

        self.encoder = nn.Sequential(
            nn.Conv2d(1, 16, 3, stride=3, padding=1),
            nn.ReLU(True),
            nn.MaxPool2d(2, stride=2),
            nn.Conv2d(16, 8, 3, stride=2, padding=1),
            nn.ReLU(True),
            nn.MaxPool2d(2, stride=1)
        )

    def forward(self, x):
        z = self.encoder(x)
        return z


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

        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(8, 16, 3, stride=2),
            nn.ReLU(True),
            nn.ConvTranspose2d(16, 8, 5, stride=3, padding=1),
            nn.ReLU(True),
            nn.ConvTranspose2d(8, 1, 2, stride=2, padding=1),
            nn.Tanh()
        )

    def forward(self, z):
        x_ = self.decoder(z)
        return x_
```

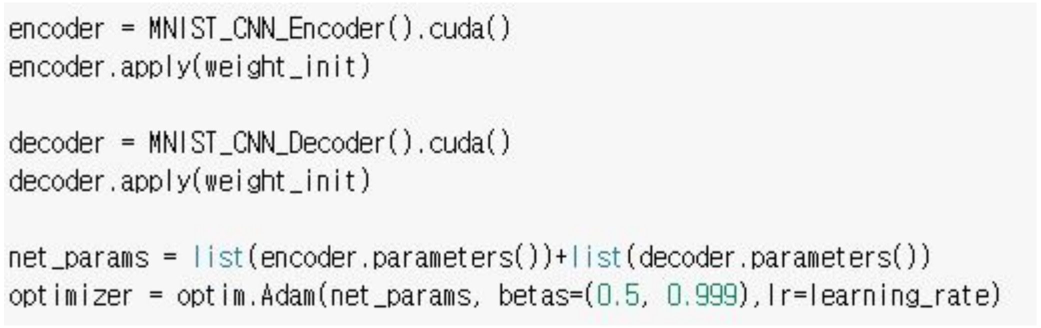
실습 12



```
epochs = 10
learning_rate = 0.01
batch_size = 100
loss_function = nn.MSELoss()

# load the dataset
dataset = datasets.MNIST('../data', train=True,
                         download=True, transform=transforms.Compose([
        transforms.ToTensor()
        , transforms.Normalize((0.5,), (0.5,))
    ]))
```

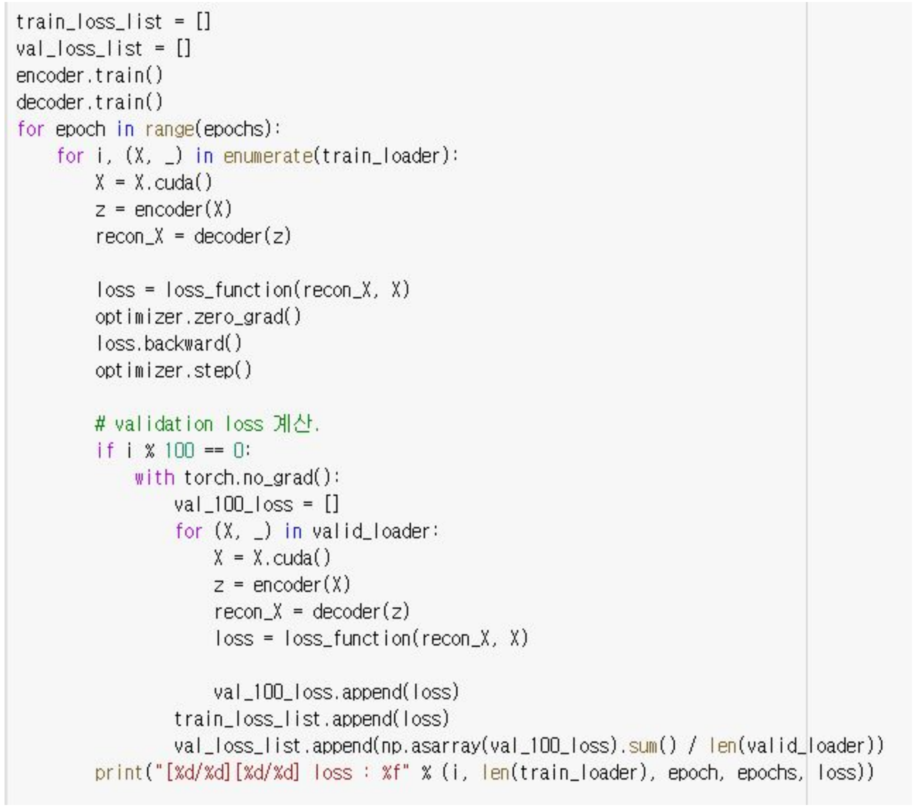
-1~1 scaling으로 normalize하였으며 test loader도 이렇게 만들어줍시다.

실습 12



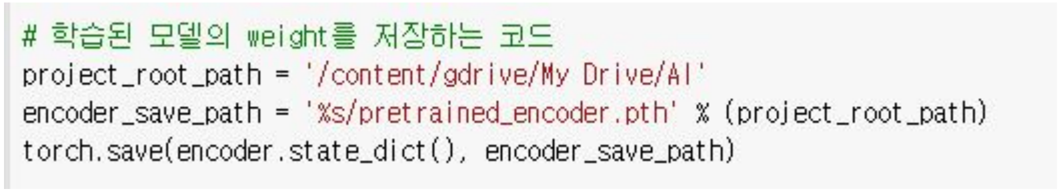
```
encoder = MNIST_CNN_Encoder().cuda()
encoder.apply(weight_init)

decoder = MNIST_CNN_Decoder().cuda()
decoder.apply(weight_init)

net_params = list(encoder.parameters())+list(decoder.parameters())
optimizer = optim.Adam(net_params, betas=(0.5, 0.999),lr=learning_rate)
```

실습 12



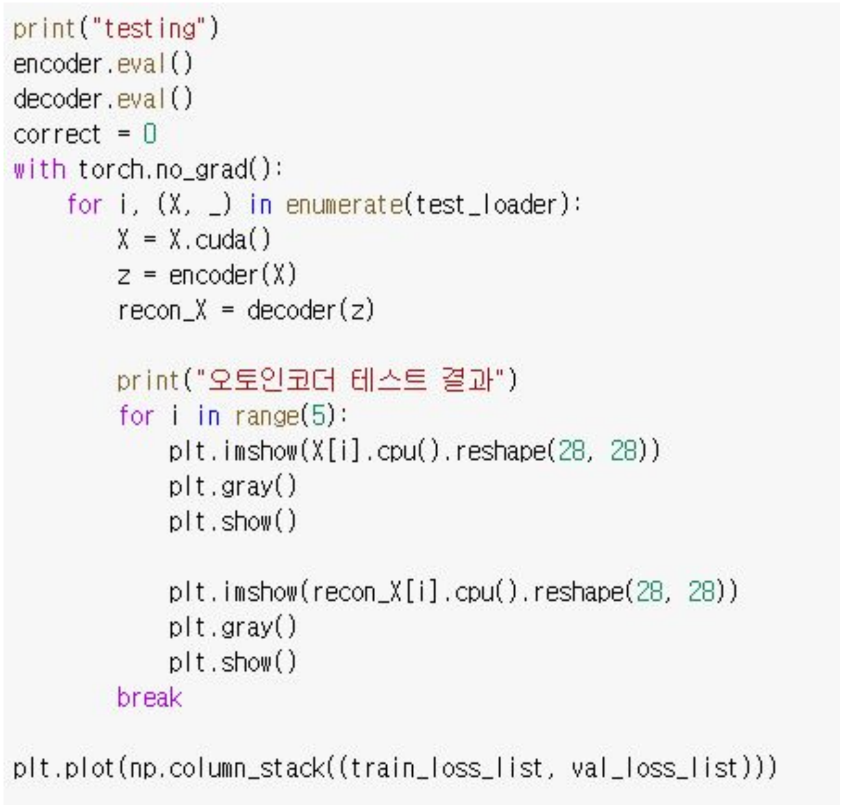
```
train_loss_list = []
val_loss_list = []
encoder.train()
decoder.train()
for epoch in range(epochs):
    for i, (X, _) in enumerate(train_loader):
        X = X.cuda()
        z = encoder(X)
        recon_X = decoder(z)

        loss = loss_function(recon_X, X)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # validation loss 계산.
        if i % 100 == 0:
            with torch.no_grad():
                val_100_loss = []
                for (X, _) in valid_loader:
                    X = X.cuda()
                    z = encoder(X)
                    recon_X = decoder(z)
                    loss = loss_function(recon_X, X)

                    val_100_loss.append(loss)
                train_loss_list.append(loss)
                val_loss_list.append(np.asarray(val_100_loss).sum() / len(valid_loader))
        print("[%d/%d][%d/%d] loss : %f" % (i, len(train_loader), epoch, epochs, loss))
```

실습 12

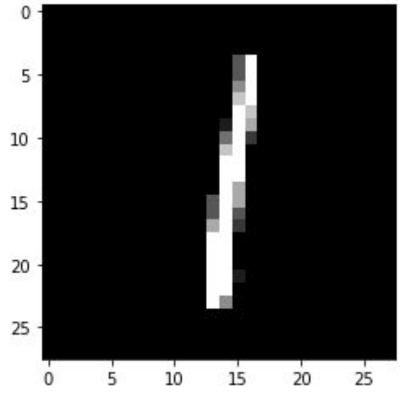


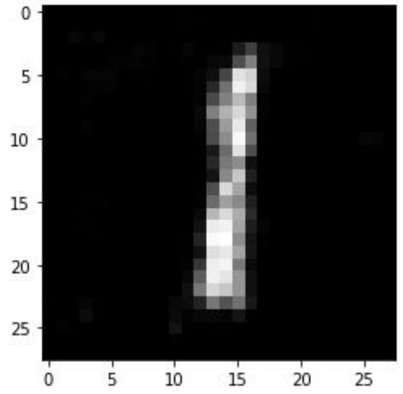
```
# 학습된 모델의 weight를 저장하는 코드
project_root_path = '/content/gdrive/My Drive/AI'
encoder_save_path = '%s/pretrained_encoder.pth' % (project_root_path)
torch.save(encoder.state_dict(), encoder_save_path)
```

실습 12



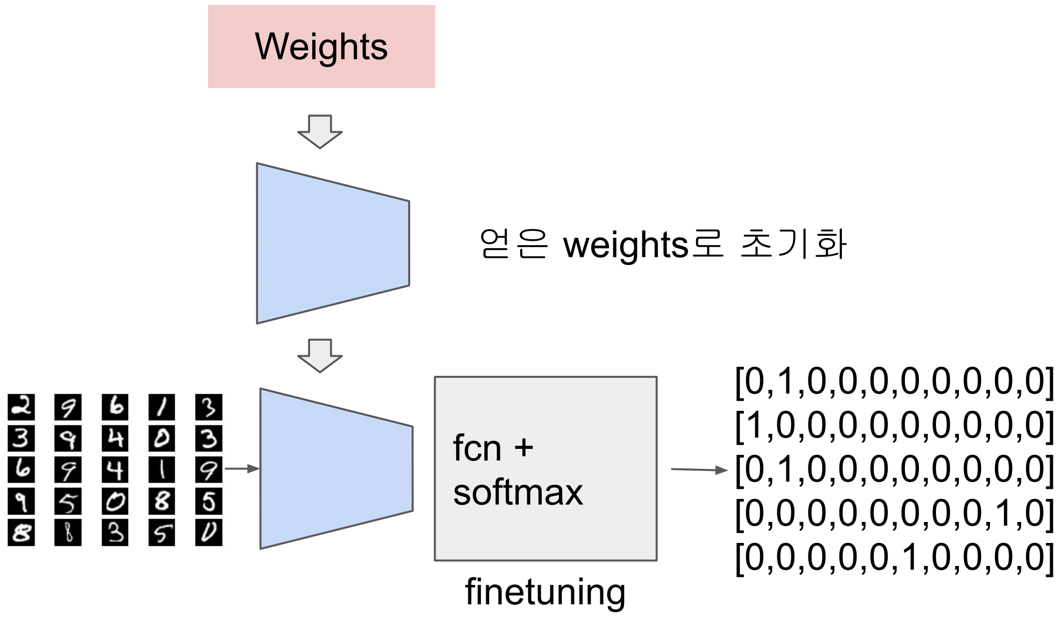
```
print("testing")
encoder.eval()
decoder.eval()
correct = 0
with torch.no_grad():
    for i, (X, _) in enumerate(test_loader):
        X = X.cuda()
        z = encoder(X)
        recon_X = decoder(z)

        print("오토인코더 테스트 결과")
        for i in range(5):
            plt.imshow(X[i].cpu().reshape(28, 28))
            plt.gray()
            plt.show()

            plt.imshow(recon_X[i].cpu().reshape(28, 28))
            plt.gray()
            plt.show()
        break

plt.plot(np.column_stack((train_loss_list, val_loss_list)))
```



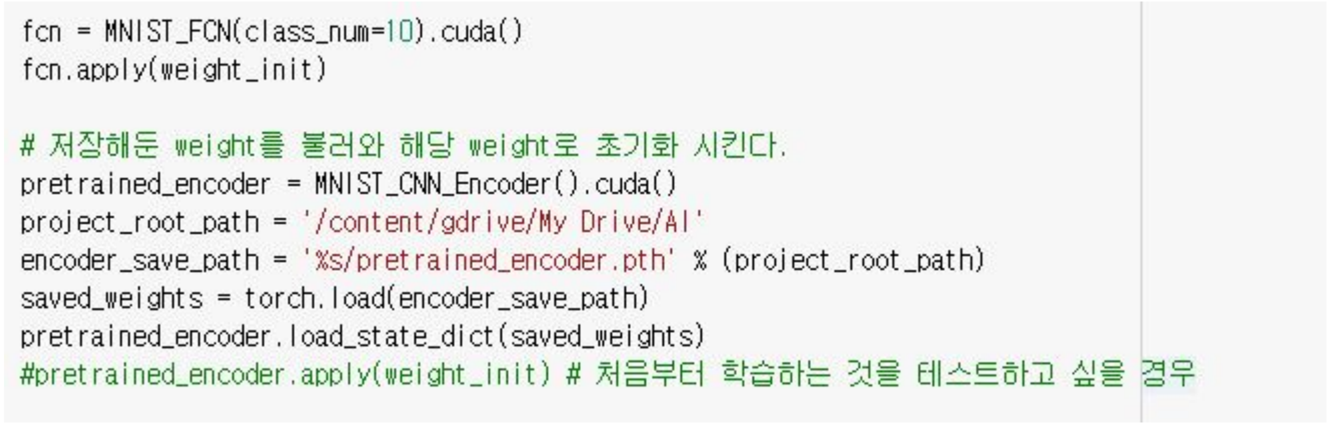


실습 12

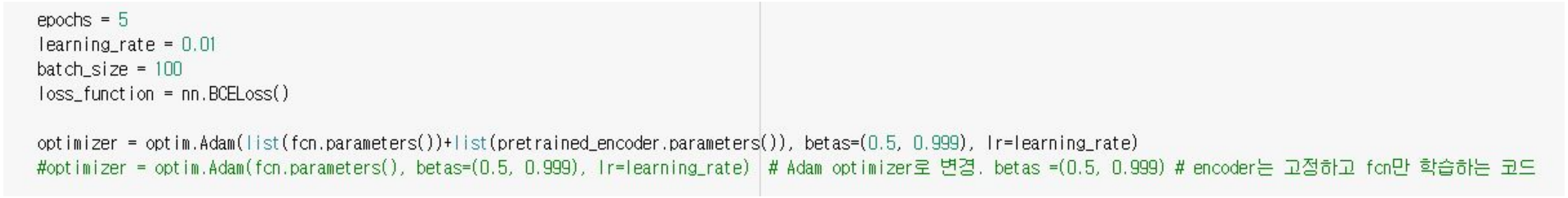
weight 저장된 weight 이용하여 모델 초기화하고 finetuning 하기



실습 12

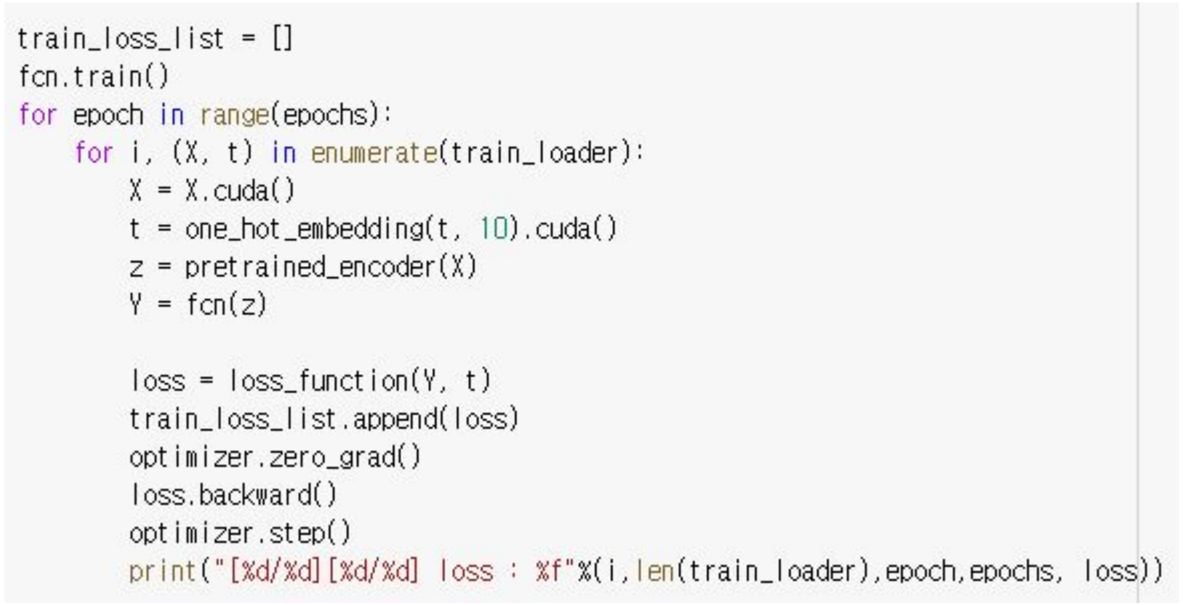


```
fcn = MNIST_FCN(class_num=10).cuda()
fcn.apply(weight_init)

# 저장해둔 weight를 불러와 해당 weight로 초기화 시킨다.
pretrained_encoder = MNIST_CNN_Encoder().cuda()
project_root_path = '/content/gdrive/My Drive/AI'
encoder_save_path = '%s/pretrained_encoder.pth' % (project_root_path)
saved_weights = torch.load(encoder_save_path)
pretrained_encoder.load_state_dict(saved_weights)
#pretrained_encoder.apply(weight_init) # 처음부터 학습하는 것을 테스트하고 싶을 경우
```



```
epochs = 5
learning_rate = 0.01
batch_size = 100
loss_function = nn.BCELoss()

optimizer = optim.Adam(list(fcn.parameters())+list(pretrained_encoder.parameters()), betas=(0.5, 0.999), lr=learning_rate)
#optimizer = optim.Adam(fcn.parameters(), betas=(0.5, 0.999), lr=learning_rate)  # Adam optimizer로 변경. betas =(0.5, 0.999) # encoder는 고정하고 fcn만 학습하는 코드
```

실습 12

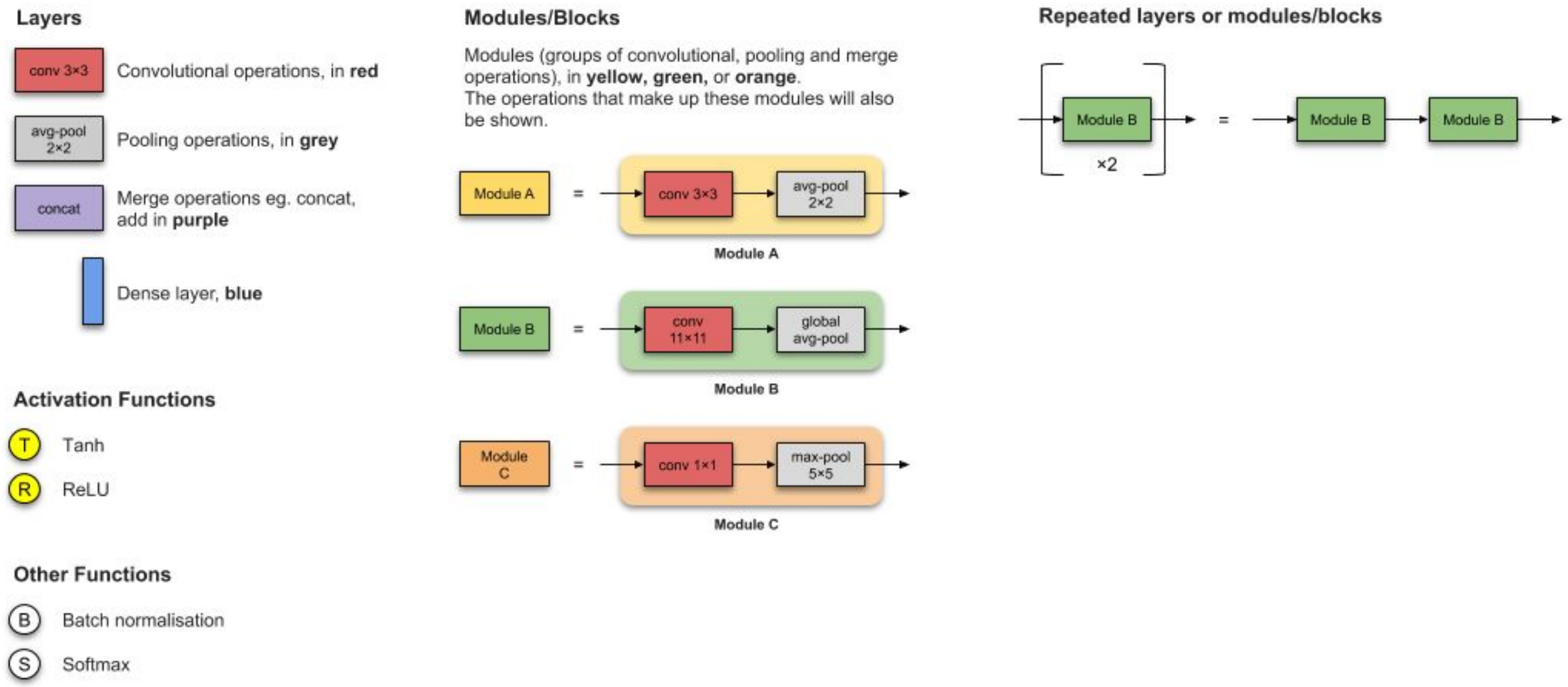


```
train_loss_list = []
fcn.train()
for epoch in range(epochs):
    for i, (X, t) in enumerate(train_loader):
        X = X.cuda()
        t = one_hot_embedding(t, 10).cuda()
        z = pretrained_encoder(X)
        Y = fcn(z)

        loss = loss_function(Y, t)
        train_loss_list.append(loss)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        print("[%d/%d][%d/%d] loss : %f"%(i,len(train_loader),epoch,epochs, loss))
```

test 부분도 이런 구조로 짜줘야합니다!

Deep Convolutional Neural Networks

legends



Layers

conv 3×3 — Convolutional operations, in **red**

avg-pool 2×2 — Pooling operations, in **grey**

concat — Merge operations eg. concat, add in **purple**

Dense layer, **blue**

Activation Functions

T — Tanh

R — ReLU

Other Functions

B — Batch normalisation

S — Softmax

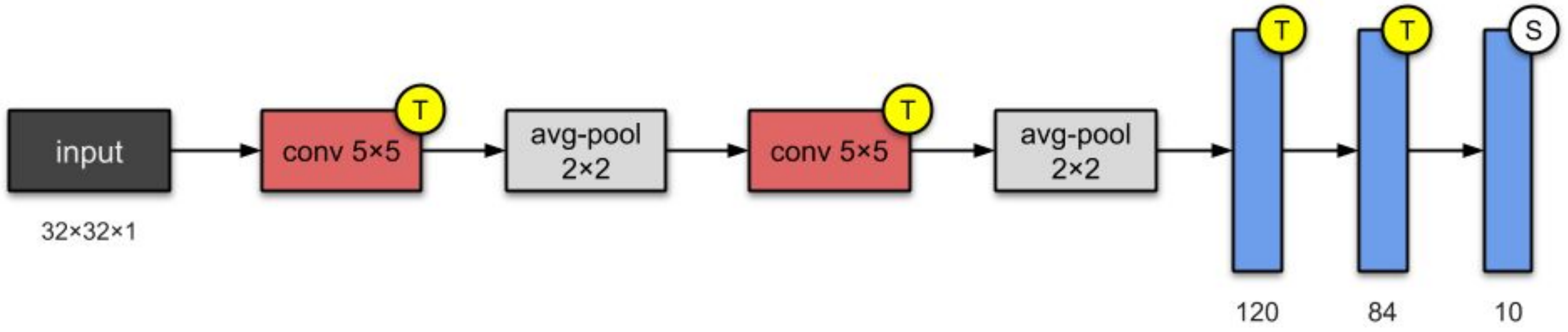
Modules/Blocks

Modules (groups of convolutional, pooling and merge operations), in **yellow, green,** or **orange**.
The operations that make up these modules will also be shown.

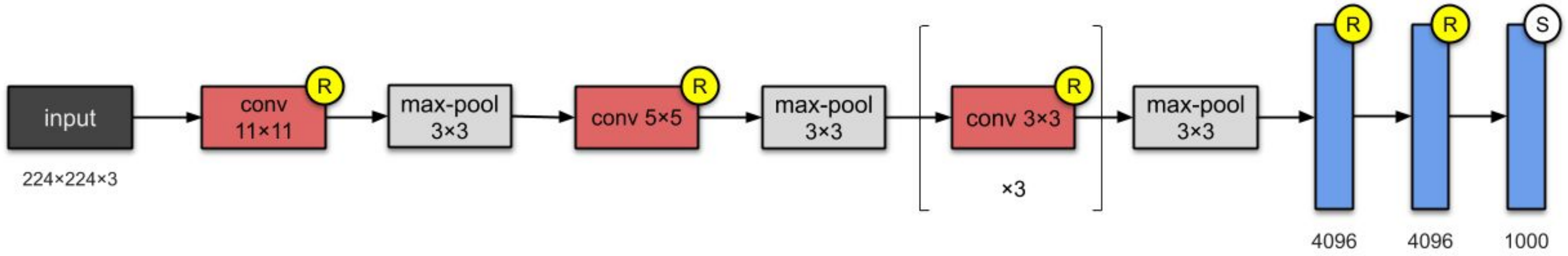
Repeated layers or modules/blocks

출처 : https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

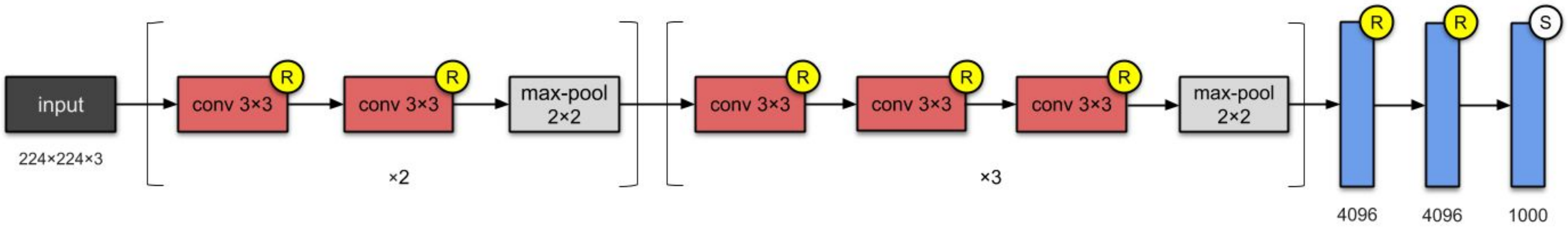
Lenet-5


input
32×32×1
conv 5×5
T
avg-pool 2×2
conv 5×5
T
avg-pool 2×2
T
T
S
120
84
10

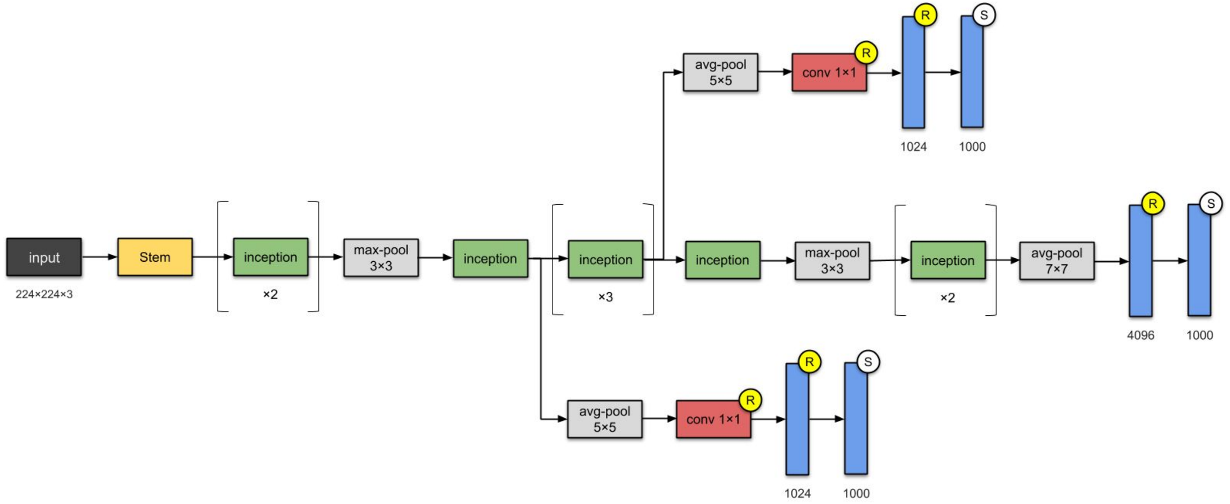
Alexnet

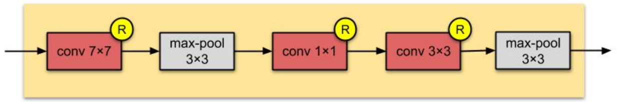

input
224×224×3
conv 11×11
R
max-pool 3×3
conv 5×5
R
max-pool 3×3
conv 3×3
R
×3
max-pool 3×3
R
R
S
4096
4096
1000

VGG-16

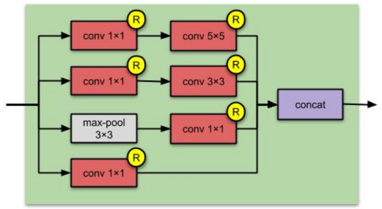

input
224×224×3
conv 3×3
R
conv 3×3
R
max-pool 2×2
×2
conv 3×3
R
conv 3×3
R
conv 3×3
R
max-pool 2×2
×3
R
R
S
4096
4096
1000

Inception-V1



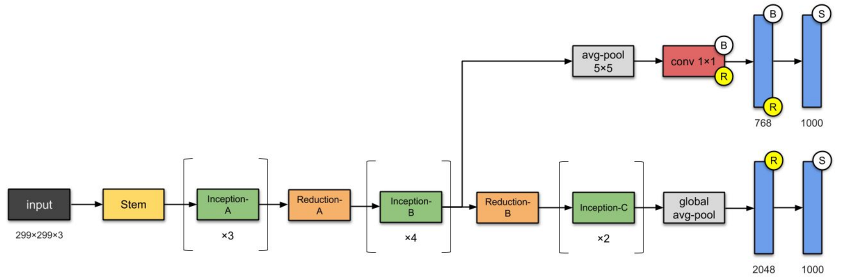


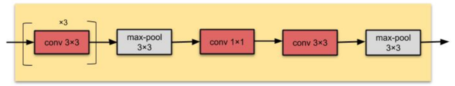
Stem



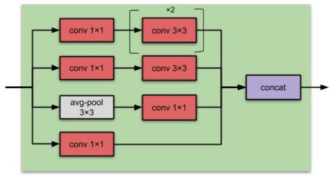
Inception module

Inception-V3

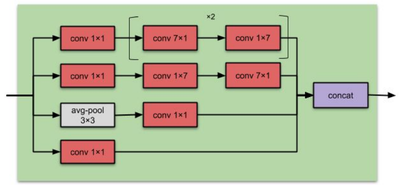




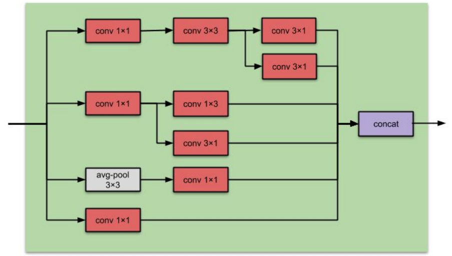
Stem



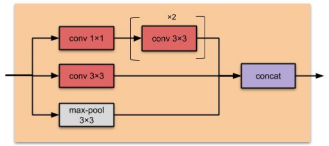
Inception-A



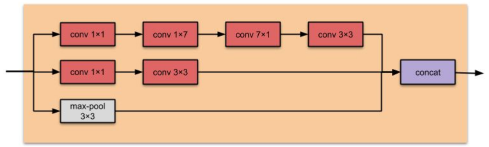
Inception-B



Inception-C

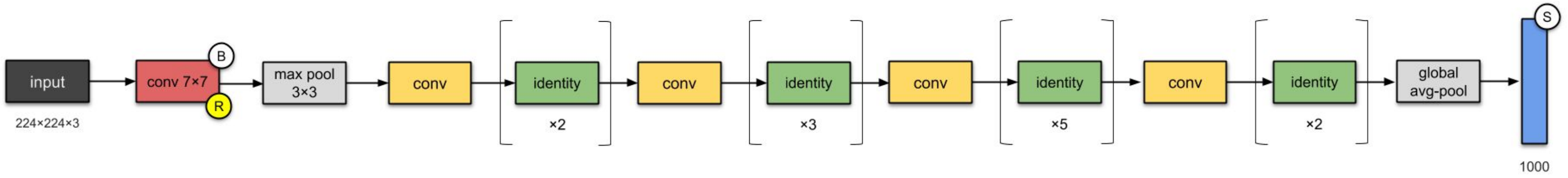


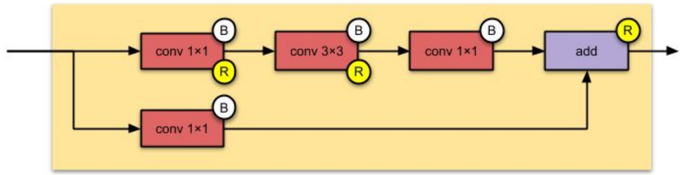
Reduction-A



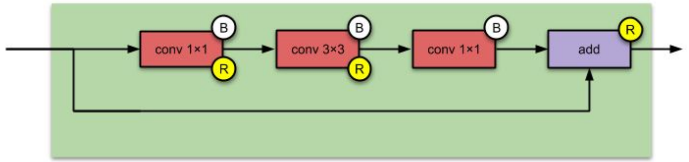
Reduction-B

Resnet-50

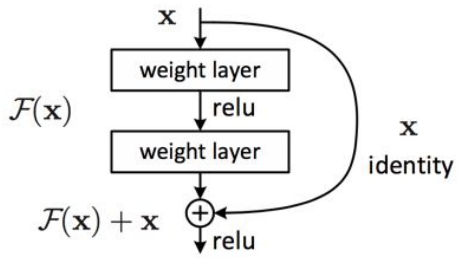




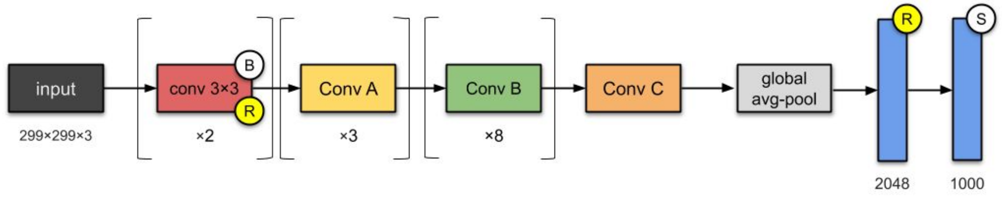
Conv block

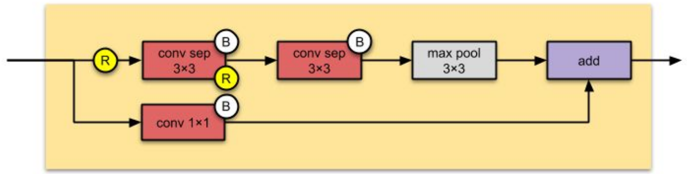


Identity block

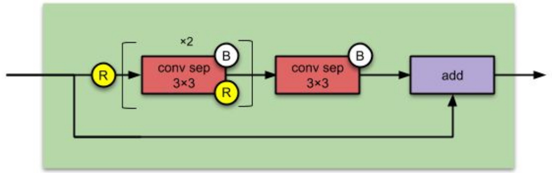

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Xception

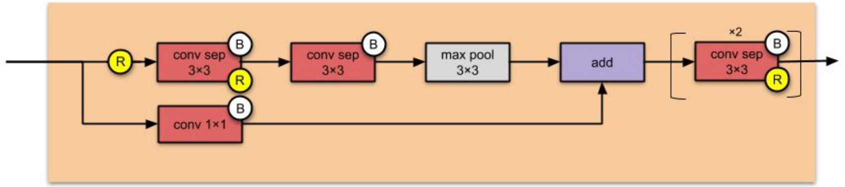




Conv A



Conv B

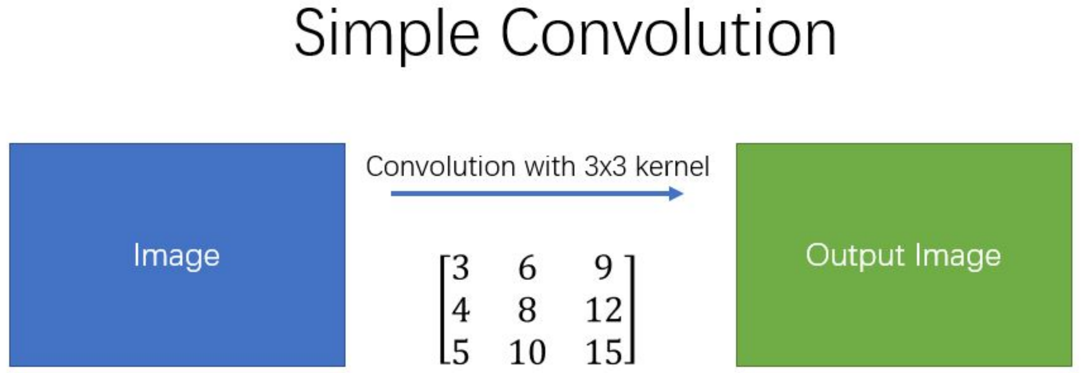


Conv C

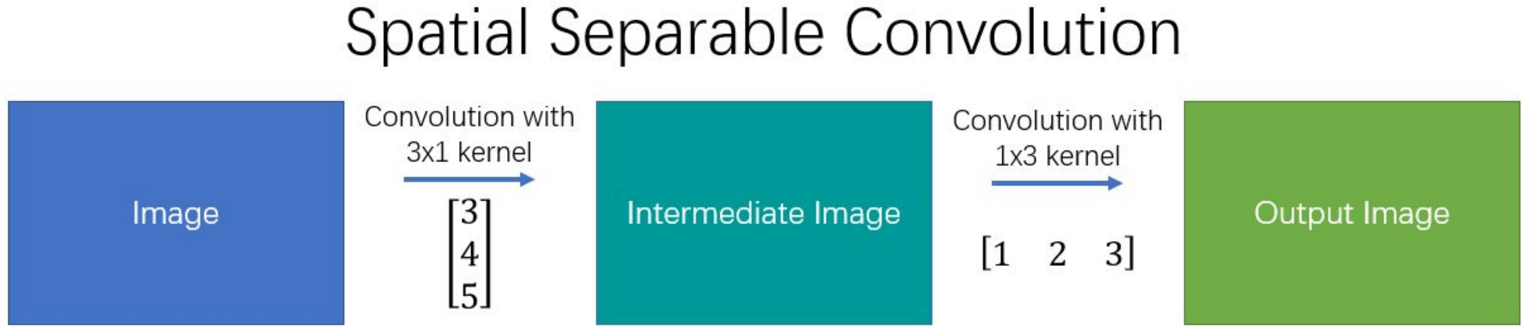
Depthwise separable convolution

$$\begin{bmatrix} 3 & 6 & 9 \\ 4 & 8 & 12 \\ 5 & 10 & 15 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

Depthwise separable convolution

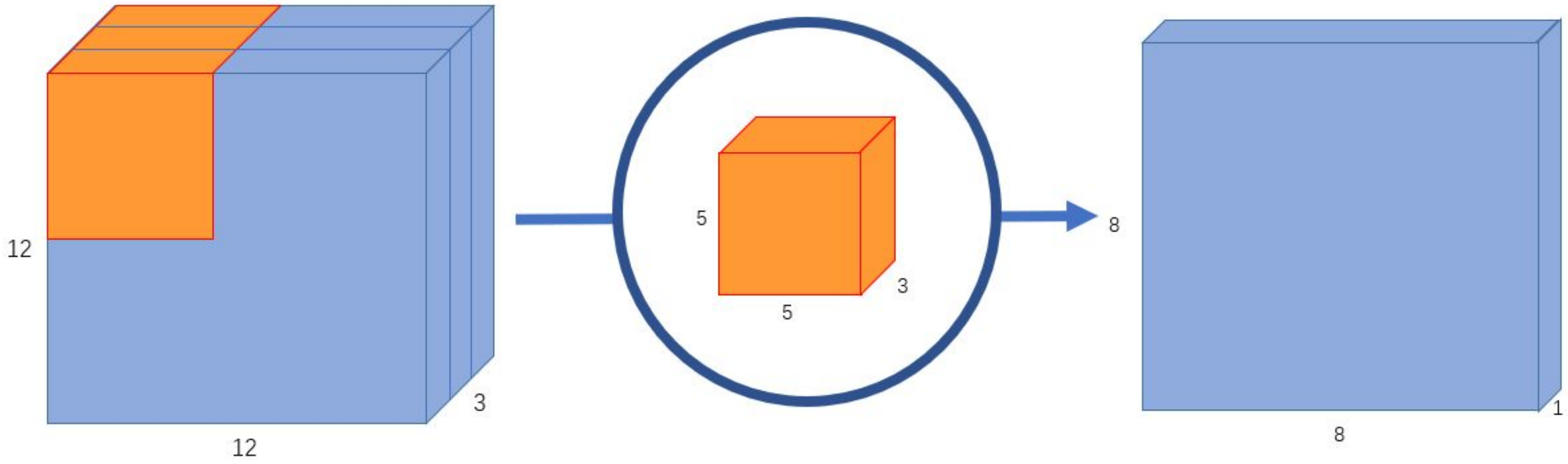


Simple Convolution
Image
Convolution with 3x3 kernel
$\begin{bmatrix} 3 & 6 & 9 \\ 4 & 8 & 12 \\ 5 & 10 & 15 \end{bmatrix}$
Output Image

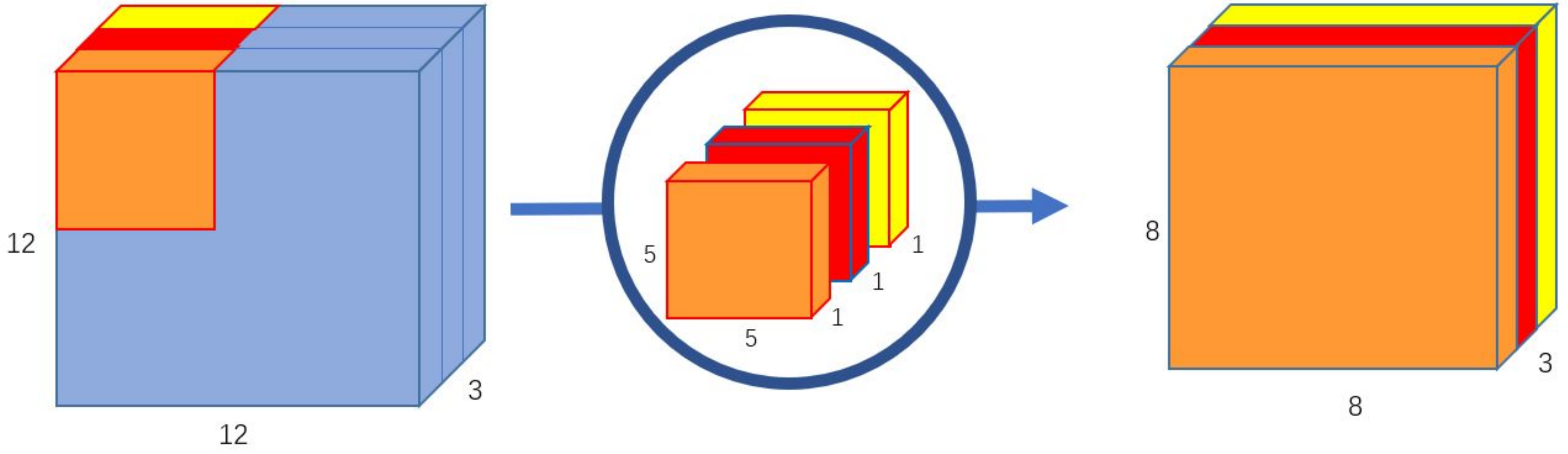


Spatial Separable Convolution
Image
Convolution with 3x1 kernel
$\begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$
Intermediate Image
Convolution with 1x3 kernel
$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$
Output Image

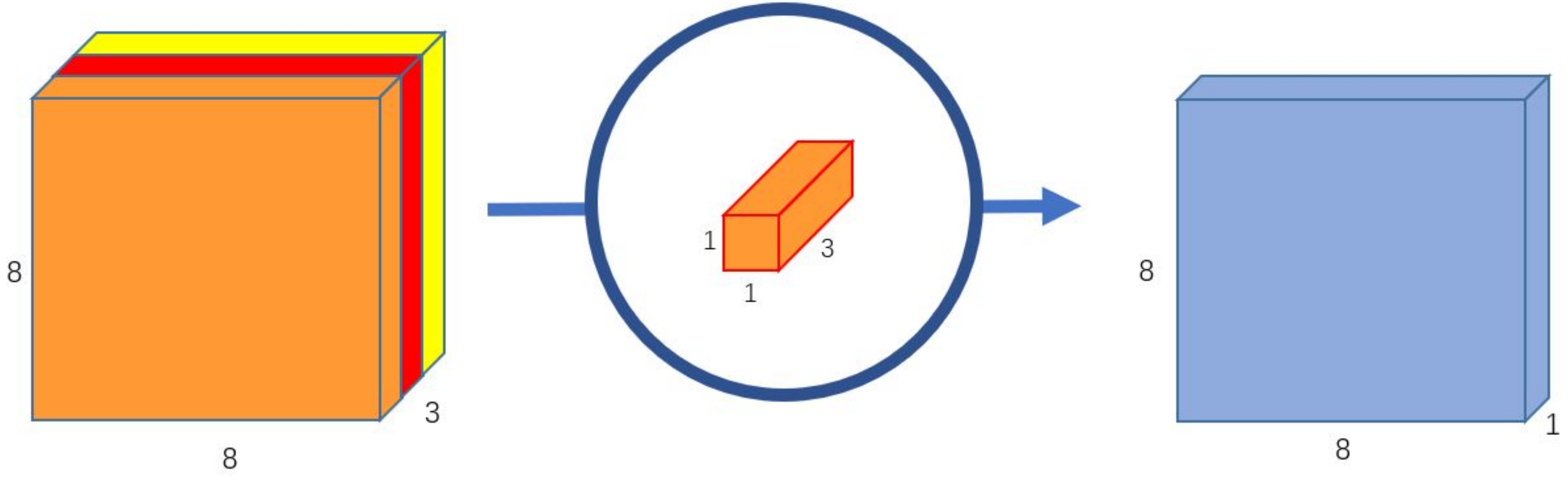
Depthwise separable convolution



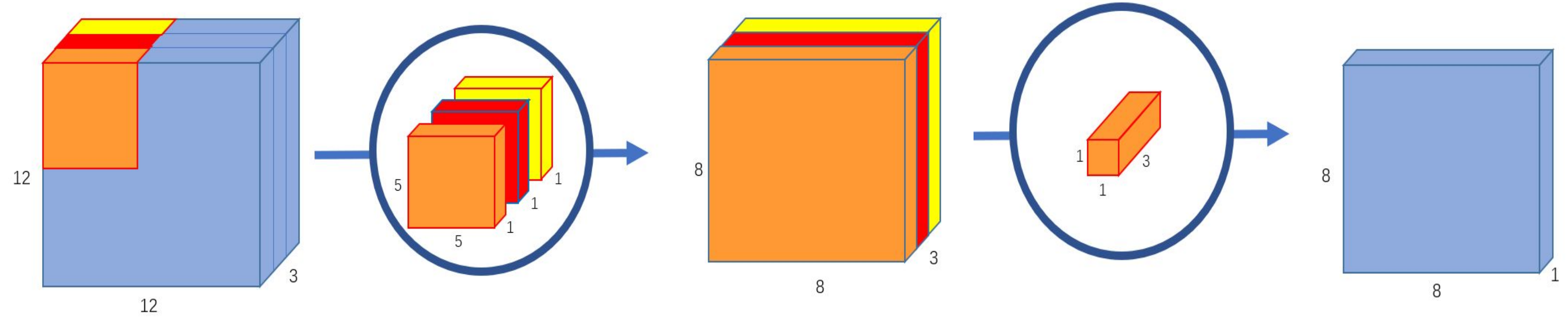
Depthwise separable convolution

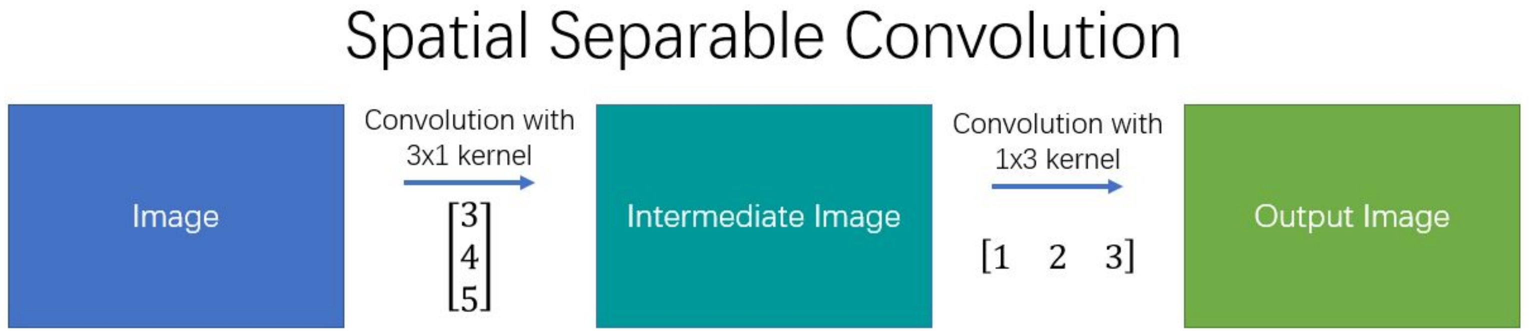


Depthwise separable convolution



Depthwise separable convolution

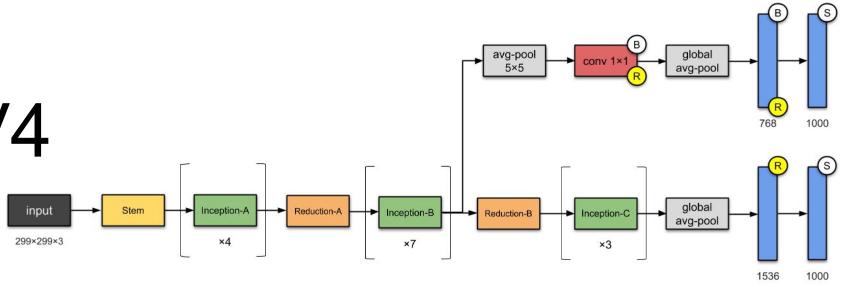


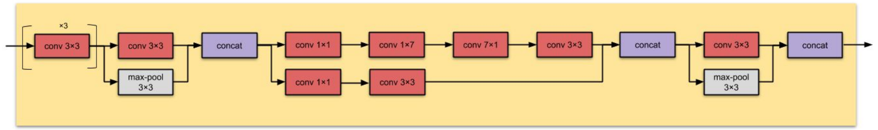


$$\begin{bmatrix}3\\4\\5\end{bmatrix}$$

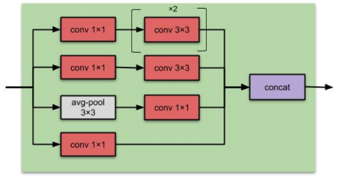
$$\begin{bmatrix}1 & 2 & 3\end{bmatrix}$$

Inception-V4

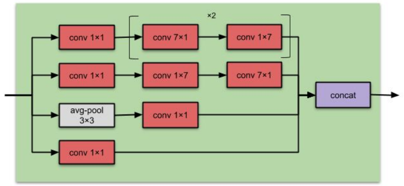




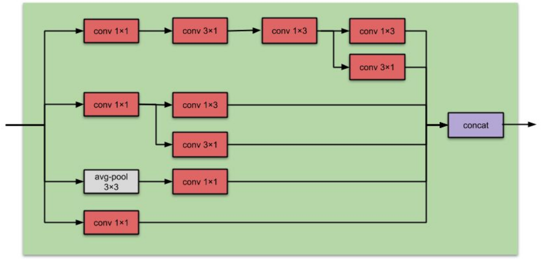
Stem



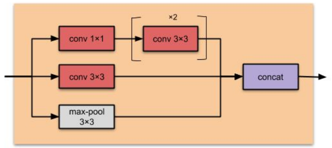
Inception-A



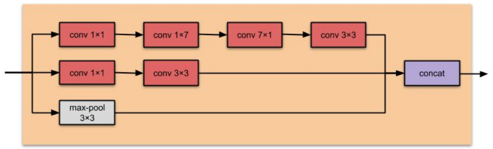
Inception-B



Inception-C

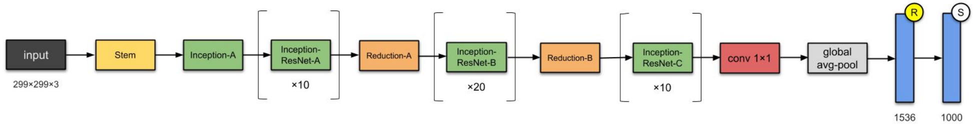


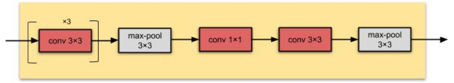
Reduction-A



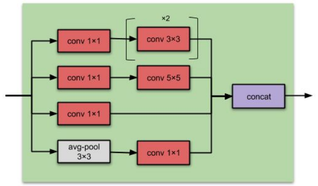
Reduction-B

Inception-Resnet-V2

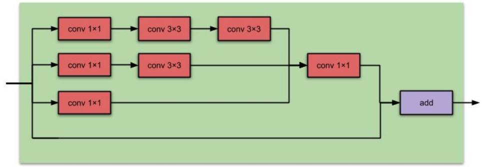




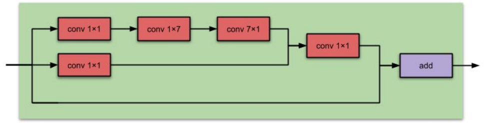
Stem



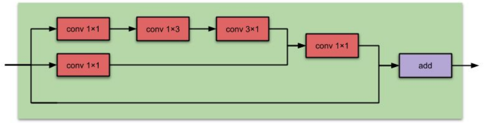
Inception-A



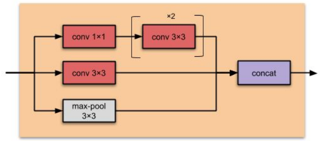
Inception-ResNet-A



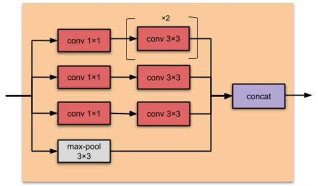
Inception-ResNet-B



Inception-ResNet-C

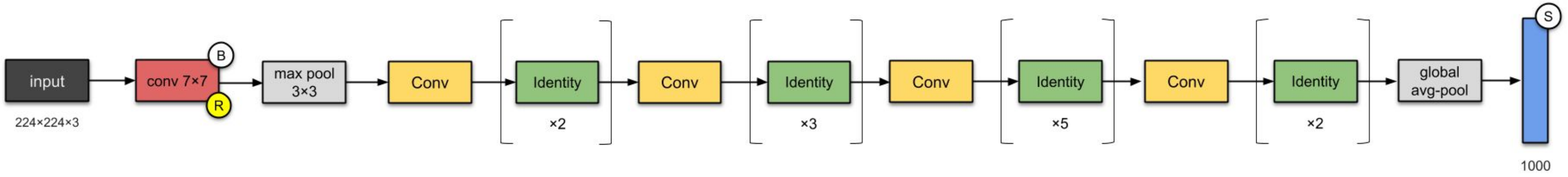


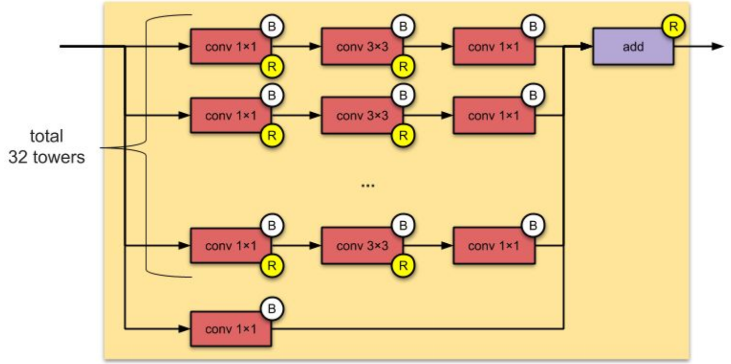
Reduction-A



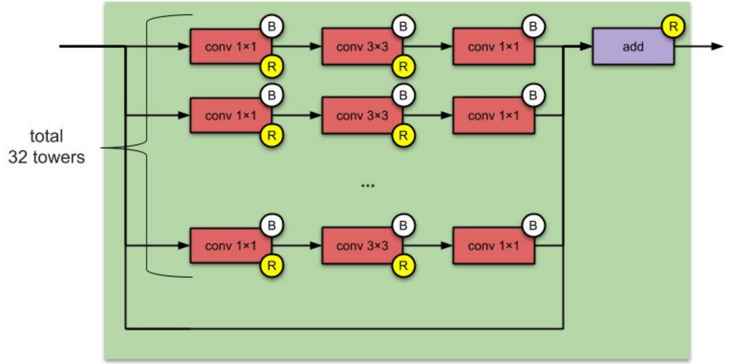
Reduction-B

ResNeXt-50



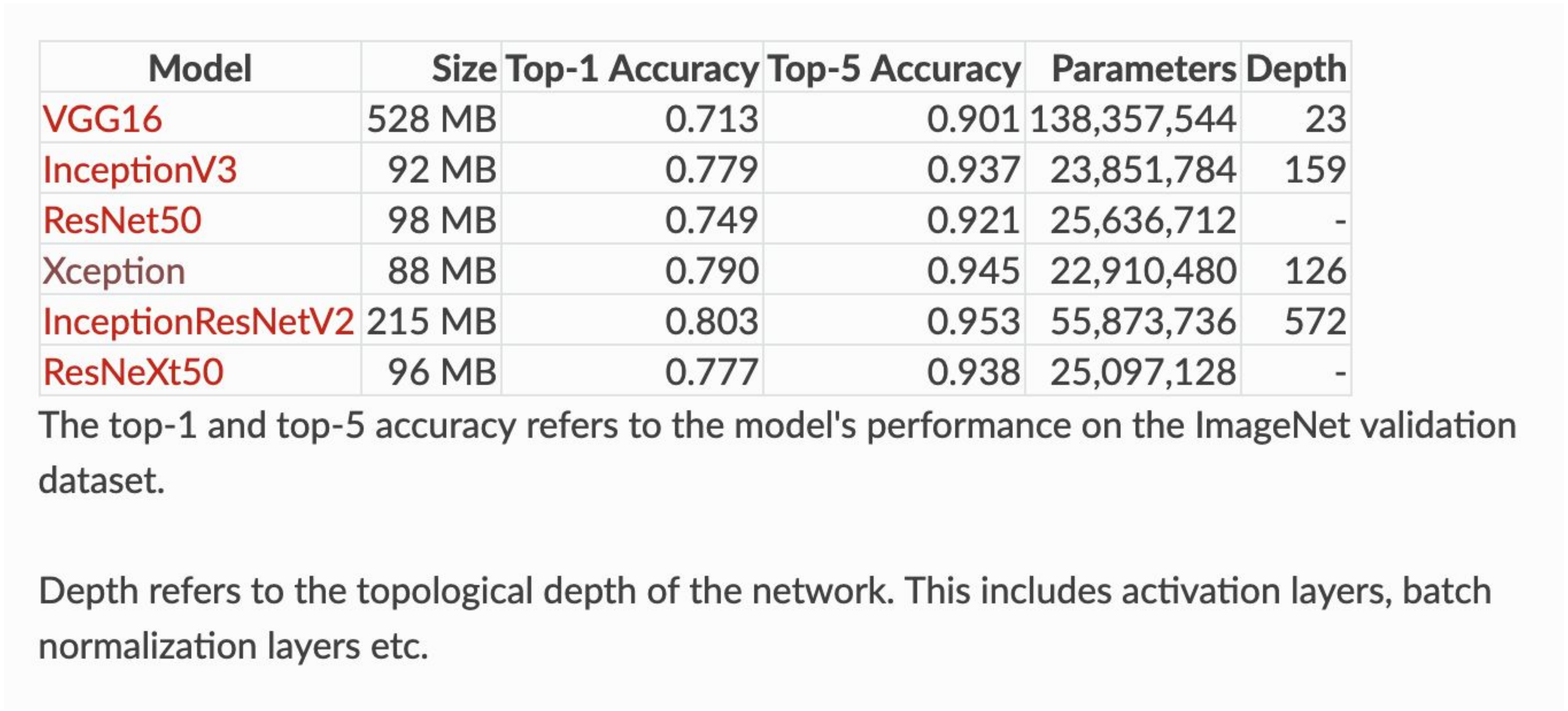


Conv block



Identity block

비교



Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
VGG16	528 MB	0.713	0.901	138,357,544	23
InceptionV3	92 MB	0.779	0.937	23,851,784	159
ResNet50	98 MB	0.749	0.921	25,636,712	-
Xception	88 MB	0.790	0.945	22,910,480	126
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
ResNeXt50	96 MB	0.777	0.938	25,097,128	-

The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset.

Depth refers to the topological depth of the network. This includes activation layers, batch normalization layers etc.

AutoML and NAS(Neural Architecture Search)

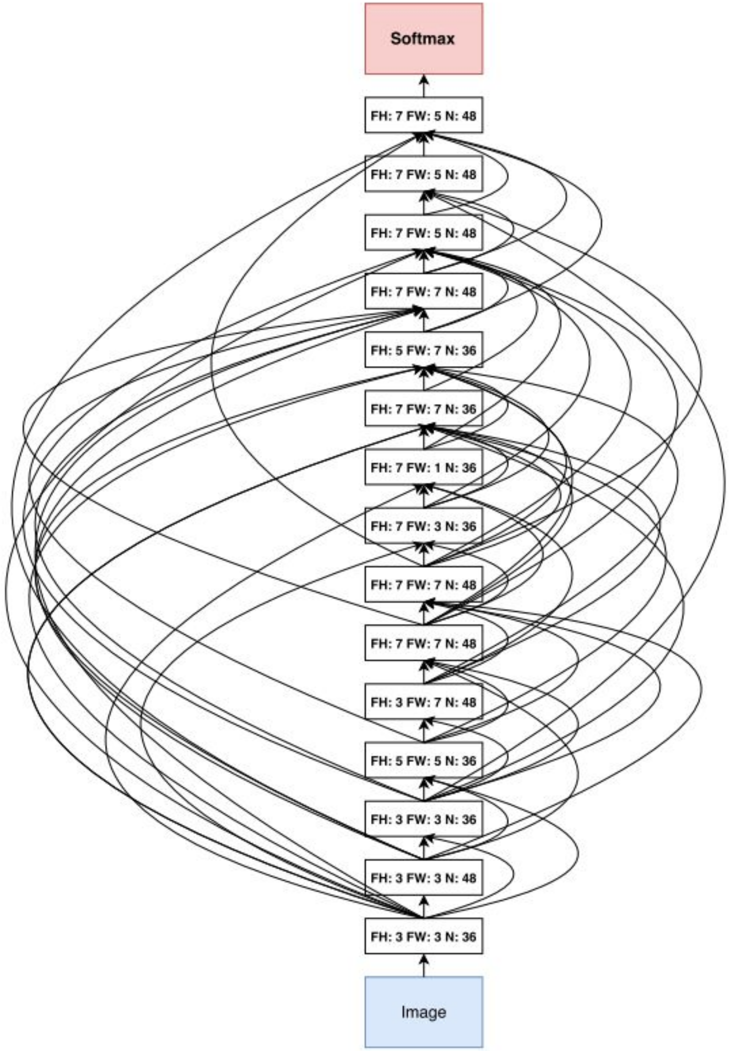
머신러닝 구조 찾는 머신러닝 방법

Automated Feature Learning

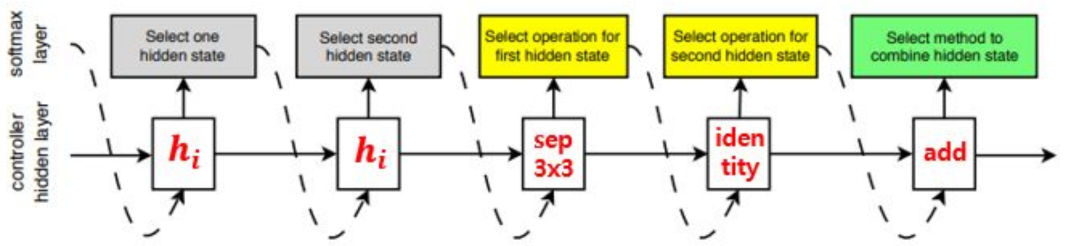
Architecture Search => 강화학습(2017), 유전알고리즘, DARTS…NASnet

Hyperparameter Optimization

NAS


Softmax
FH: 7 FW: 5 N: 48
FH: 7 FW: 5 N: 48
FH: 7 FW: 5 N: 48
FH: 7 FW: 7 N: 48
FH: 5 FW: 7 N: 36
FH: 7 FW: 7 N: 36
FH: 7 FW: 1 N: 36
FH: 7 FW: 3 N: 36
FH: 7 FW: 7 N: 48
FH: 7 FW: 7 N: 48
FH: 3 FW: 7 N: 48
FH: 5 FW: 5 N: 36
FH: 3 FW: 3 N: 36
FH: 3 FW: 3 N: 48
FH: 3 FW: 3 N: 36
Image

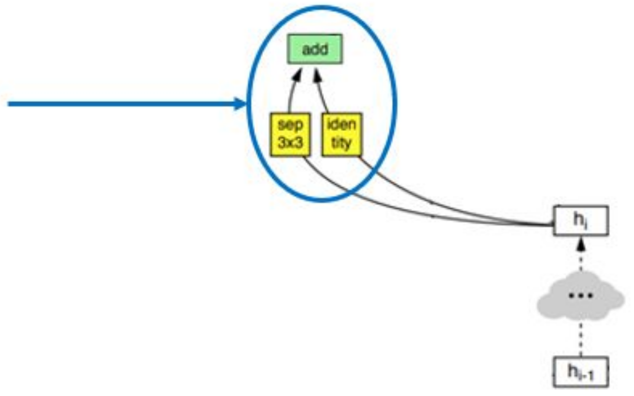
NASnet


softmax layer
controller hidden layer
Select one hidden state
Select second hidden state
Select operation for first hidden state
Select operation for second hidden state
Select method to combine hidden state
h_i
h_i
sep 3x3
iden tity
add

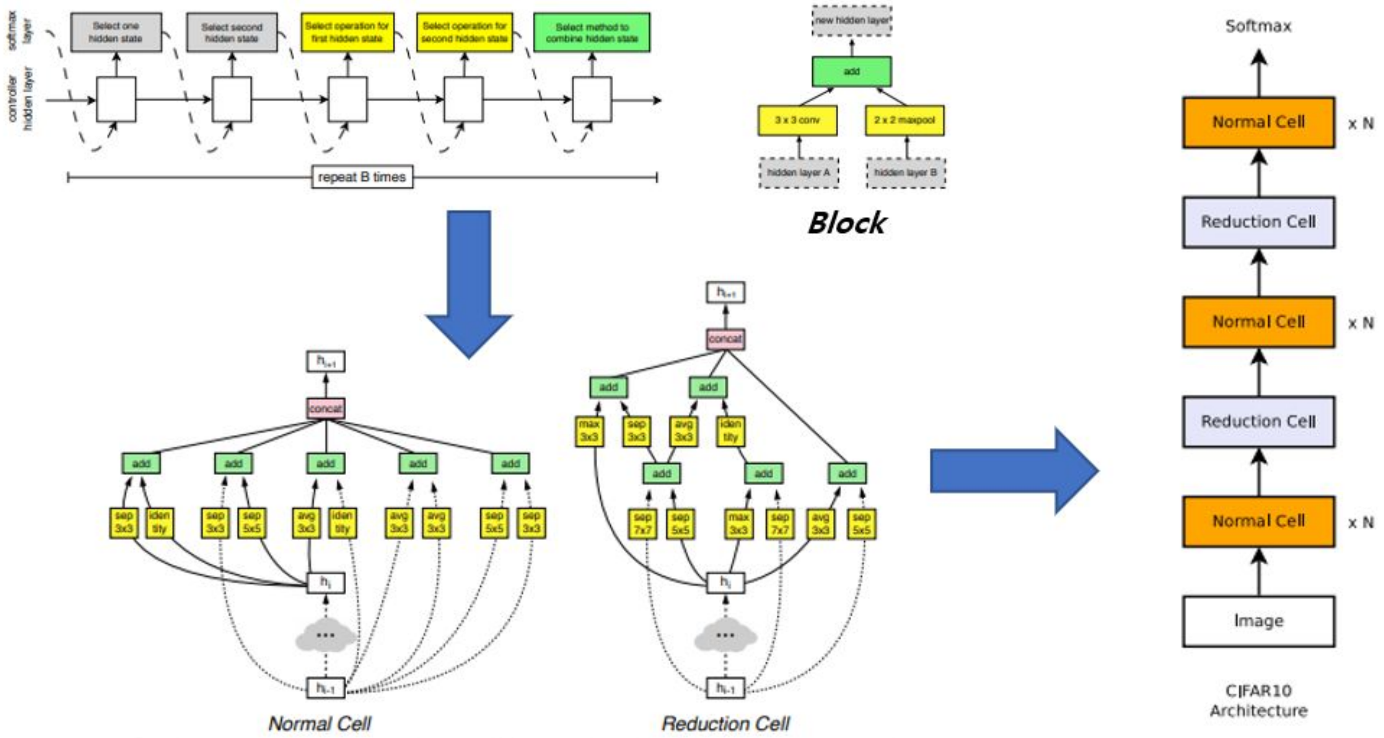
- h_i
- h_{i-1}
- $h_{i,0}$
- $h_{i,1}$
- $h_{i,2}$
- $h_{i,3}$

- Identity
- conv 1x7 + 7x1
- conv 1x3 + 3x1
- avg 3x3
- max 3x3
- max 5x5
- max 7x7
- conv 1x1
- conv 3x3
- sep 3x3
- sep 5x5
- sep 7x7
- dilated 3x3

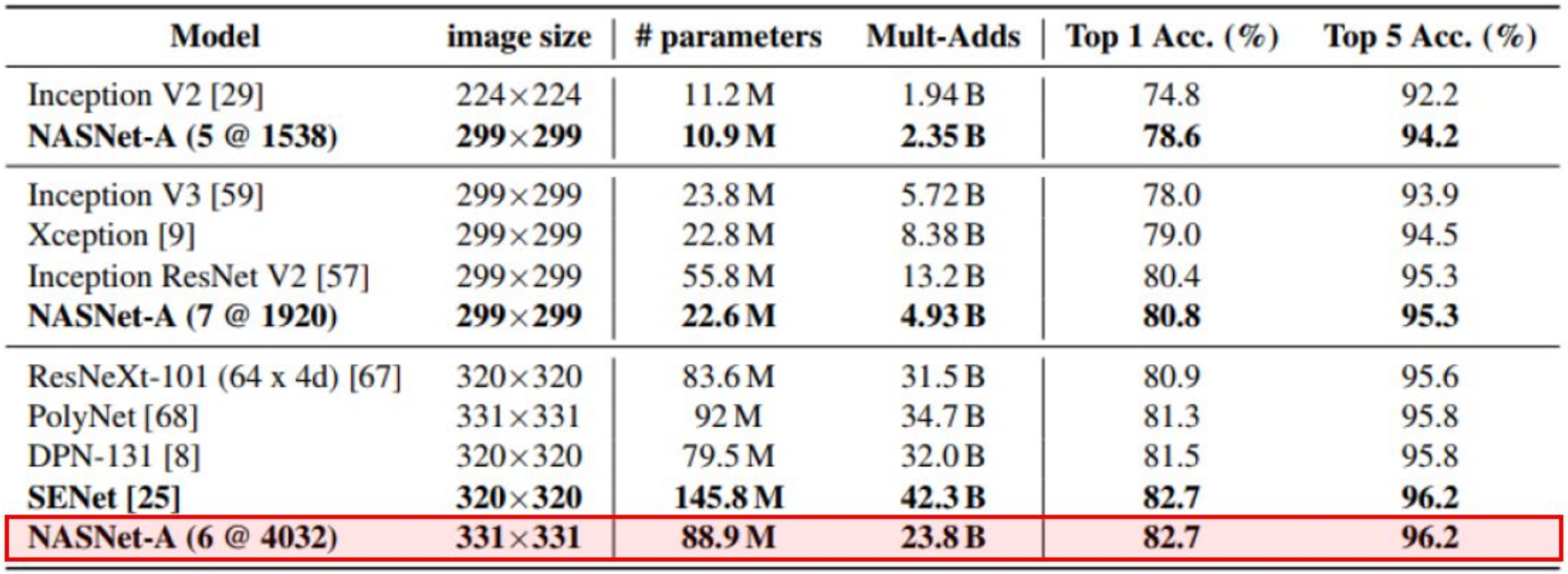
- add
- concat



NASnet


softmax layer
controller hidden layer
Select one hidden state
Select second hidden state
Select operation for first hidden state
Select operation for second hidden state
Select method to combine hidden state
repeat B times
new hidden layer
add
3 x 3 conv
2 x 2 maxpool
hidden layer A
hidden layer B
Block
h_{i+1}
concat
add
sep 3x3
iden tity
sep 5x5
avg 3x3
max 3x3
sep 7x7
h_i
...
h_{i-1}
Normal Cell
Reduction Cell
Softmax
Normal Cell
x N
Reduction Cell
Image
CIFAR10 Architecture

NASnet



Model	image size	# parameters	Mult-Adds	Top 1 Acc. (%)	Top 5 Acc. (%)
Inception V2 [29]	224×224	11.2 M	1.94 B	74.8	92.2
NASNet-A (5 @ 1538)	**299×299**	**10.9 M**	**2.35 B**	**78.6**	**94.2**
Inception V3 [59]	299×299	23.8 M	5.72 B	78.0	93.9
Xception [9]	299×299	22.8 M	8.38 B	79.0	94.5
Inception ResNet V2 [57]	299×299	55.8 M	13.2 B	80.4	95.3
NASNet-A (7 @ 1920)	**299×299**	**22.6 M**	**4.93 B**	**80.8**	**95.3**
ResNeXt-101 (64 x 4d) [67]	320×320	83.6 M	31.5 B	80.9	95.6
PolyNet [68]	331×331	92 M	34.7 B	81.3	95.8
DPN-131 [8]	320×320	79.5 M	32.0 B	81.5	95.8
SENet [25]	**320×320**	**145.8 M**	**42.3 B**	**82.7**	**96.2**
NASNet-A (6 @ 4032)	**331×331**	**88.9 M**	**23.8 B**	**82.7**	**96.2**

AutoML - activation function

Swish is defined as $x \cdot \sigma(\beta x)$, where $\sigma(z) = (1 + \exp(-z))^{-1}$

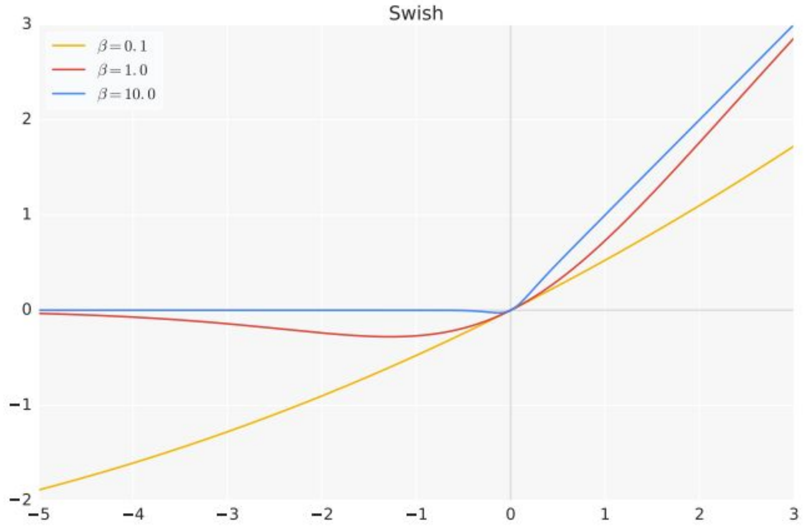


Figure 4: The Swish activation function.

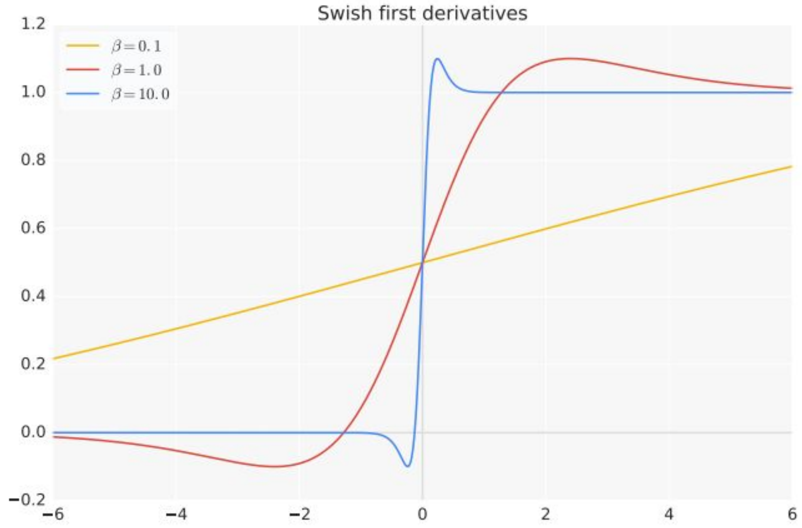
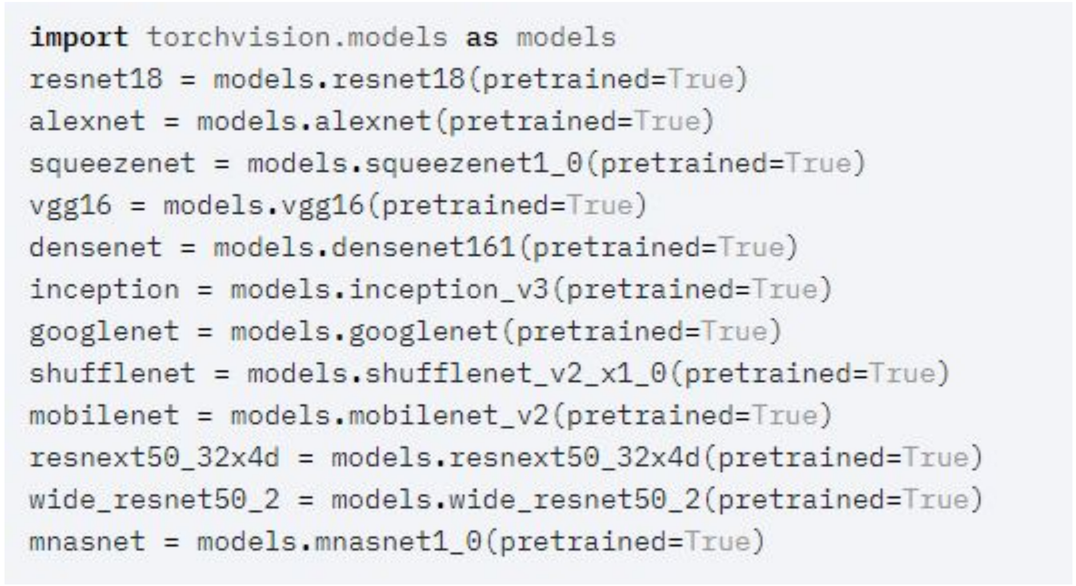


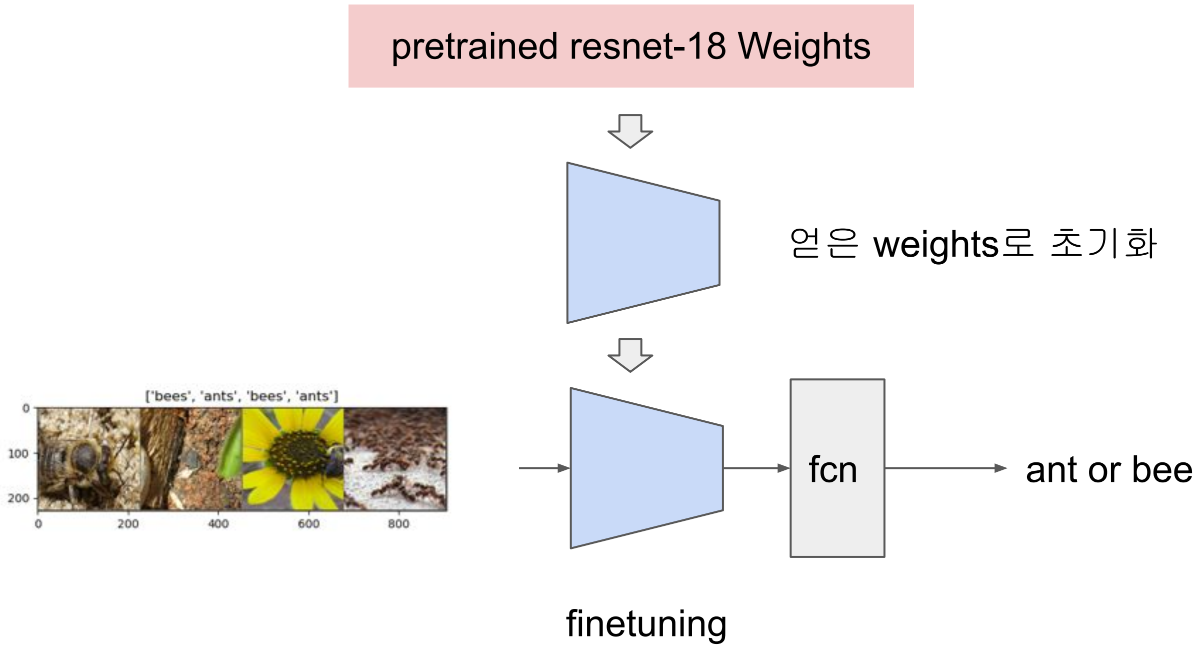
Figure 5: First derivatives of Swish.

실습 13



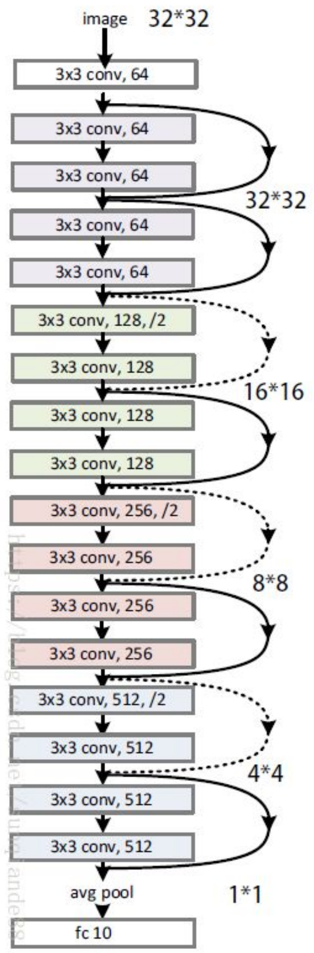
```
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeezenet1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
googlenet = models.googlenet(pretrained=True)
shufflenet = models.shufflenet_v2_x1_0(pretrained=True)
mobilenet = models.mobilenet_v2(pretrained=True)
resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
mnasnet = models.mnasnet1_0(pretrained=True)
```

실습 13


pretrained resnet-18 Weights
얻은 weights로 초기화
['bees', 'ants', 'bees', 'ants']
0
100
200
0
200
400
600
800
fcn
ant or bee
finetuning

실습 13

resnet-18

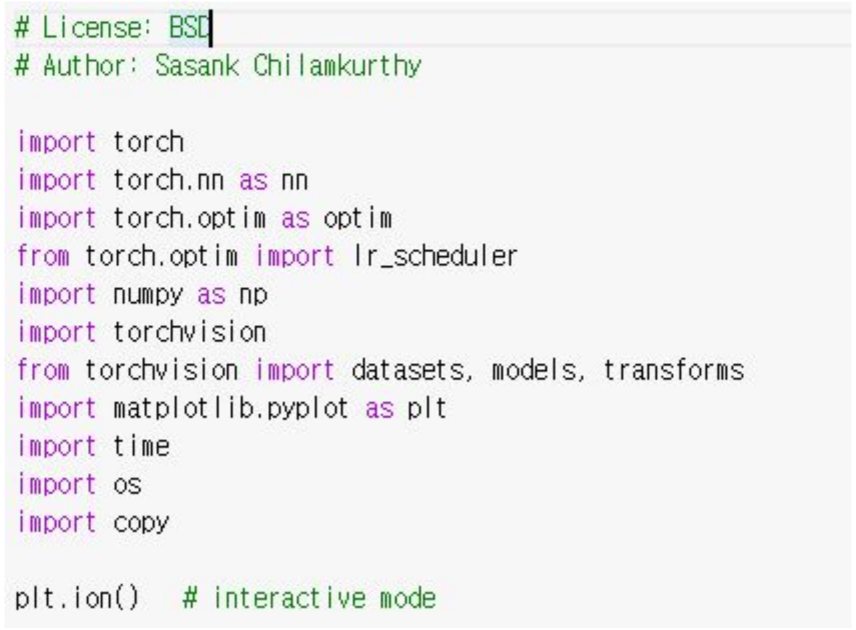


실습 13

dataset link:
https://download.pytorch.org/tutorial/hymenoptera_data.zip

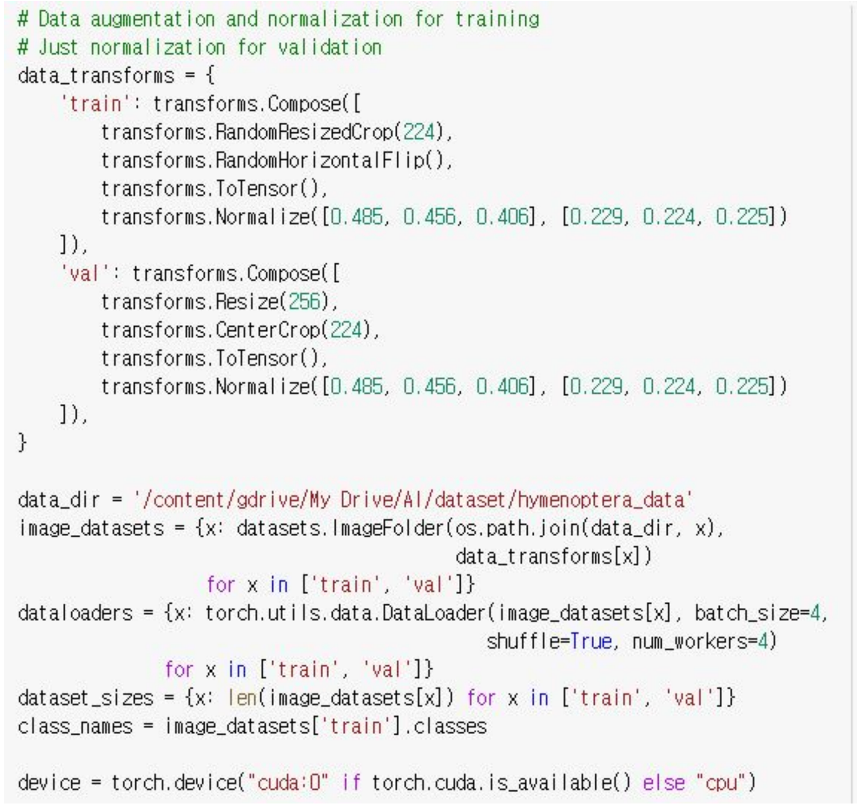
받아서 구글드라이브에 저장해주세요.
저는 dataset 폴더를 만들어 그 안에 저장했습니다.

실습 13



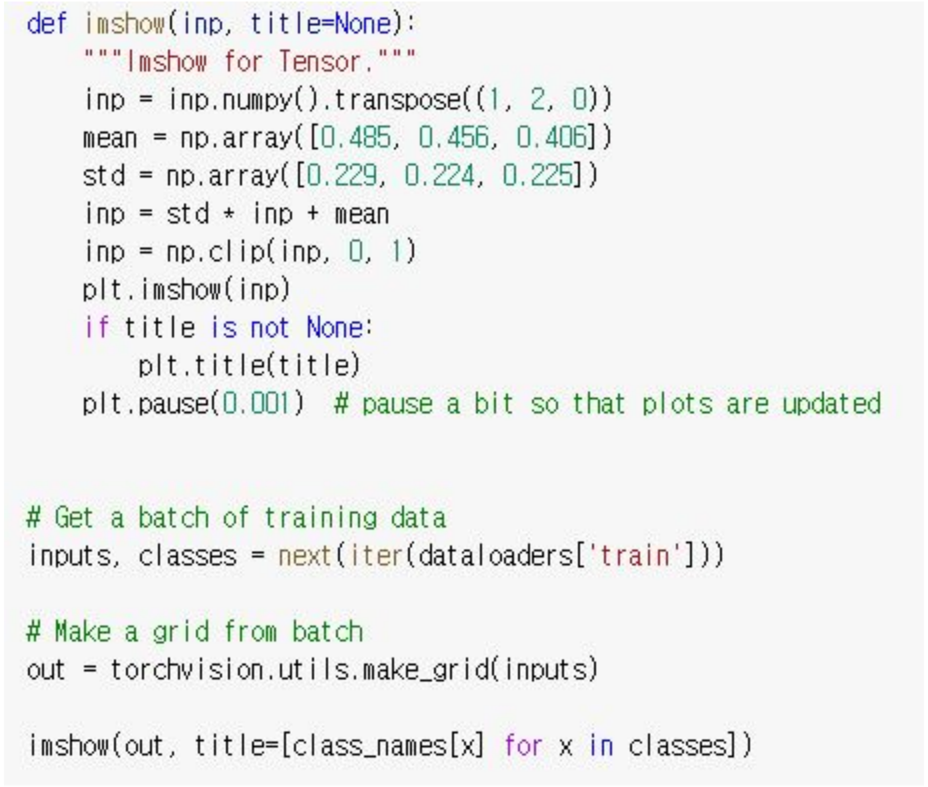
```
# License: BSD
# Author: Sasank Chilamkurthy

import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
import numpy as np
import torchvision
from torchvision import datasets, models, transforms
import matplotlib.pyplot as plt
import time
import os
import copy

plt.ion()   # interactive mode
```

실습 13



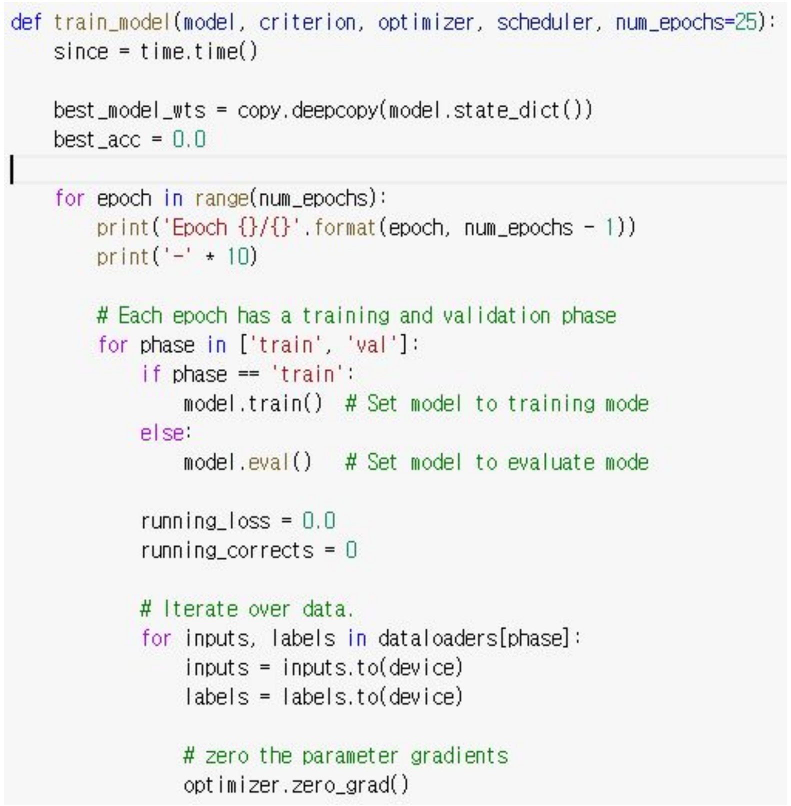
```
# Data augmentation and normalization for training
# Just normalization for validation
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}

data_dir = '/content/gdrive/My Drive/AI/dataset/hymenoptera_data'
image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x),
                                          data_transforms[x])
                  for x in ['train', 'val']}
dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=4,
                                             shuffle=True, num_workers=4)
              for x in ['train', 'val']}
dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'val']}
class_names = image_datasets['train'].classes

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

실습 13



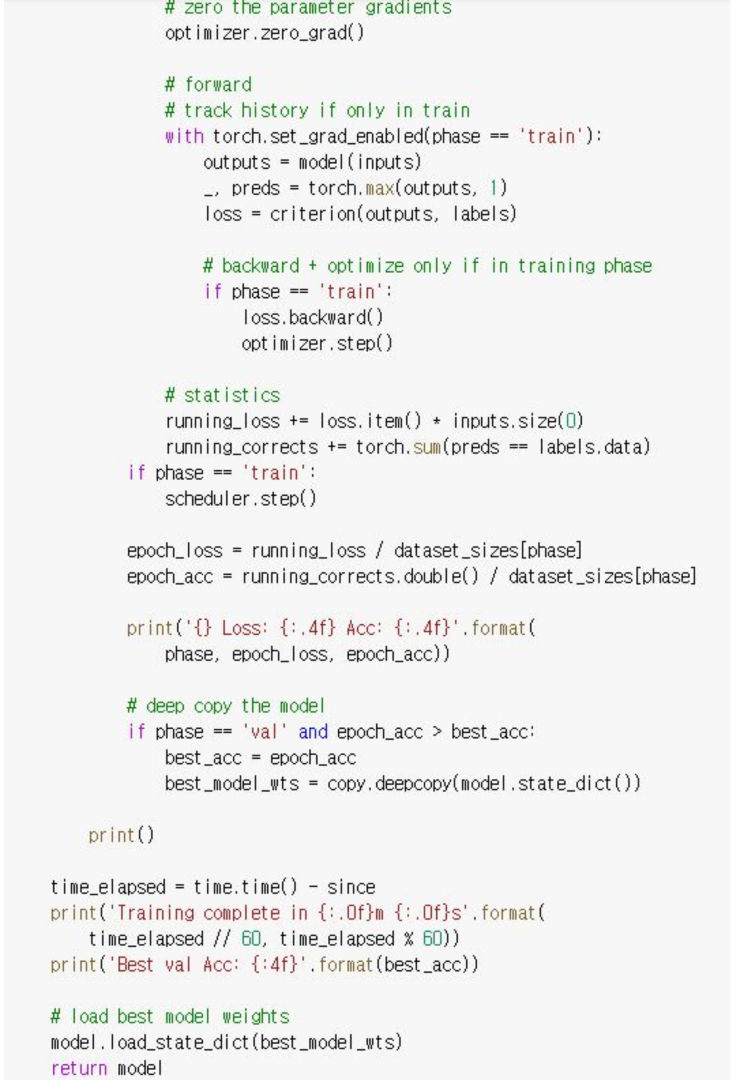
```python
def imshow(inp, title=None):
    """Imshow for Tensor."""
    inp = inp.numpy().transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    if title is not None:
        plt.title(title)
    plt.pause(0.001)  # pause a bit so that plots are updated


# Get a batch of training data
inputs, classes = next(iter(dataloaders['train']))

# Make a grid from batch
out = torchvision.utils.make_grid(inputs)

imshow(out, title=[class_names[x] for x in classes])
```

실습 13



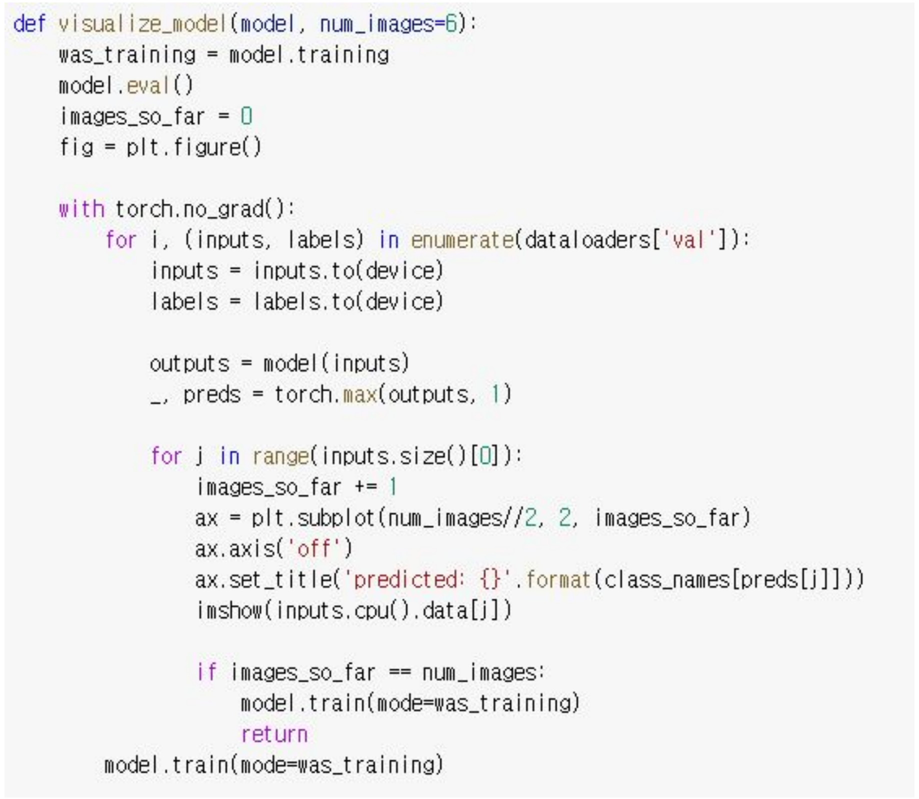
```
def train_model(model, criterion, optimizer, scheduler, num_epochs=25):
    since = time.time()

    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        print('-' * 10)

        # Each epoch has a training and validation phase
        for phase in ['train', 'val']:
            if phase == 'train':
                model.train()  # Set model to training mode
            else:
                model.eval()   # Set model to evaluate mode

            running_loss = 0.0
            running_corrects = 0

            # Iterate over data.
            for inputs, labels in dataloaders[phase]:
                inputs = inputs.to(device)
                labels = labels.to(device)

                # zero the parameter gradients
                optimizer.zero_grad()
```

실습 13



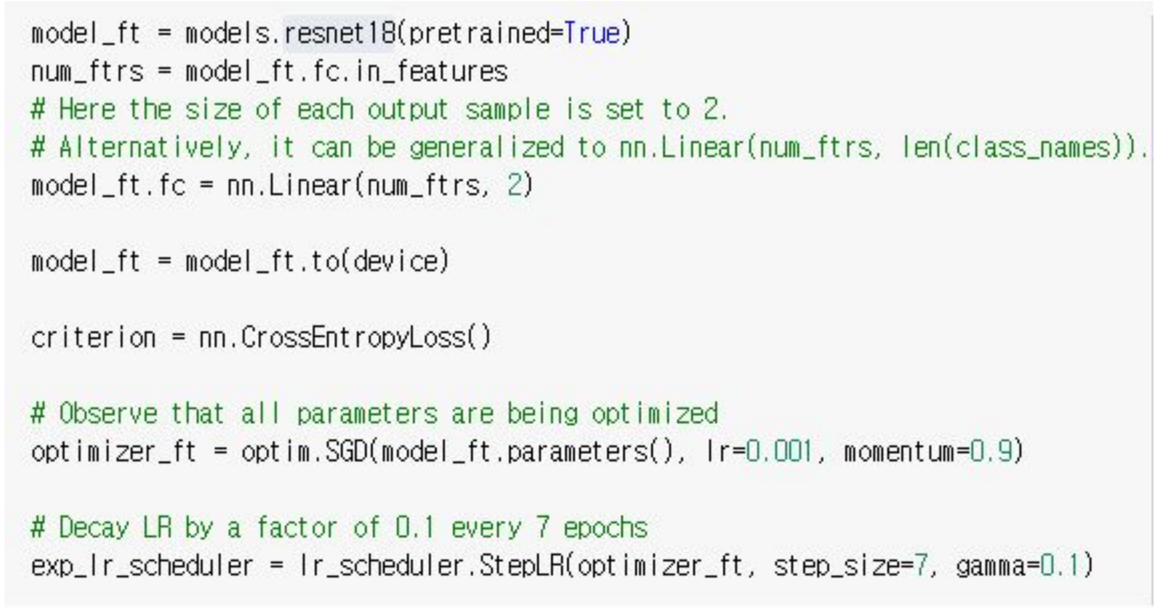
```
                # zero the parameter gradients
                optimizer.zero_grad()

                # forward
                # track history if only in train
                with torch.set_grad_enabled(phase == 'train'):
                    outputs = model(inputs)
                    _, preds = torch.max(outputs, 1)
                    loss = criterion(outputs, labels)

                    # backward + optimize only if in training phase
                    if phase == 'train':
                        loss.backward()
                        optimizer.step()

                # statistics
                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)
            if phase == 'train':
                scheduler.step()

            epoch_loss = running_loss / dataset_sizes[phase]
            epoch_acc = running_corrects.double() / dataset_sizes[phase]

            print('{} Loss: {:.4f} Acc: {:.4f}'.format(
                phase, epoch_loss, epoch_acc))

            # deep copy the model
            if phase == 'val' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(model.state_dict())

        print()

    time_elapsed = time.time() - since
    print('Training complete in {:.0f}m {:.0f}s'.format(
        time_elapsed // 60, time_elapsed % 60))
    print('Best val Acc: {:4f}'.format(best_acc))

    # load best model weights
    model.load_state_dict(best_model_wts)
    return model
```

실습 13

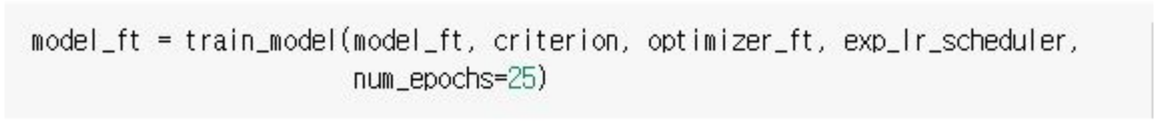


```python
def visualize_model(model, num_images=6):
    was_training = model.training
    model.eval()
    images_so_far = 0
    fig = plt.figure()

    with torch.no_grad():
        for i, (inputs, labels) in enumerate(dataloaders['val']):
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            for j in range(inputs.size()[0]):
                images_so_far += 1
                ax = plt.subplot(num_images//2, 2, images_so_far)
                ax.axis('off')
                ax.set_title('predicted: {}'.format(class_names[preds[j]]))
                imshow(inputs.cpu().data[j])

                if images_so_far == num_images:
                    model.train(mode=was_training)
                    return
        model.train(mode=was_training)
```

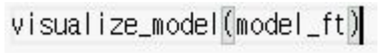
실습 13



```python
model_ft = models.resnet18(pretrained=True)
num_ftrs = model_ft.fc.in_features
# Here the size of each output sample is set to 2.
# Alternatively, it can be generalized to nn.Linear(num_ftrs, len(class_names)).
model_ft.fc = nn.Linear(num_ftrs, 2)

model_ft = model_ft.to(device)

criterion = nn.CrossEntropyLoss()

# Observe that all parameters are being optimized
optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.001, momentum=0.9)

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)
```

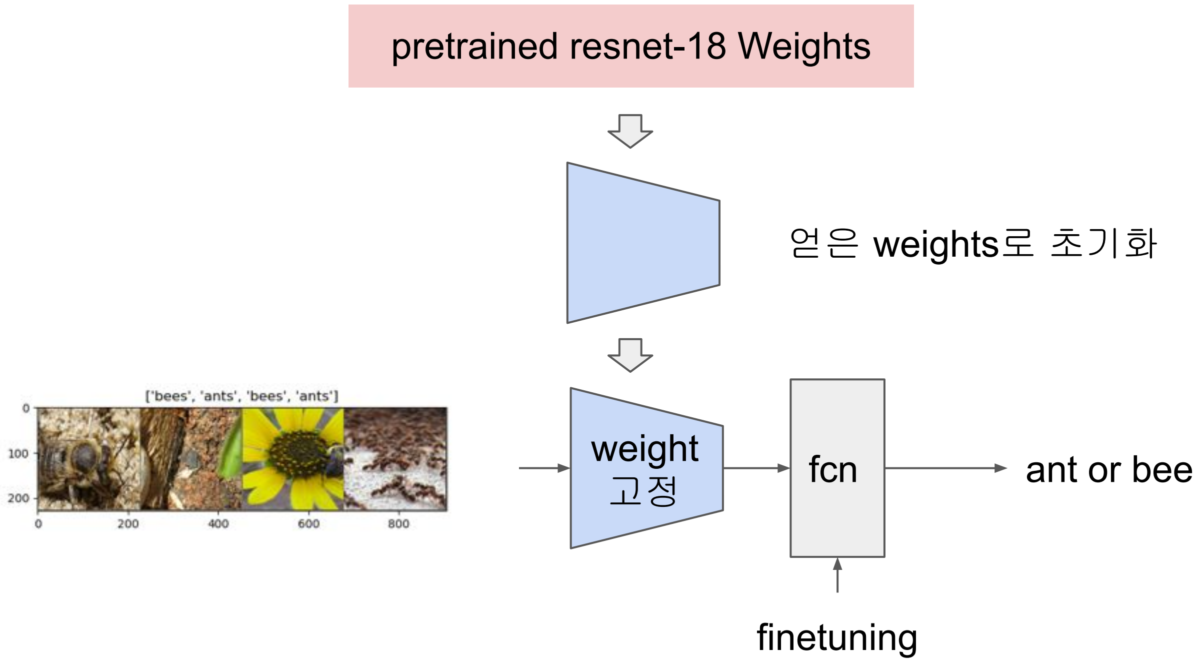


```python
model_ft = train_model(model_ft, criterion, optimizer_ft, exp_lr_scheduler,
                       num_epochs=25)
```

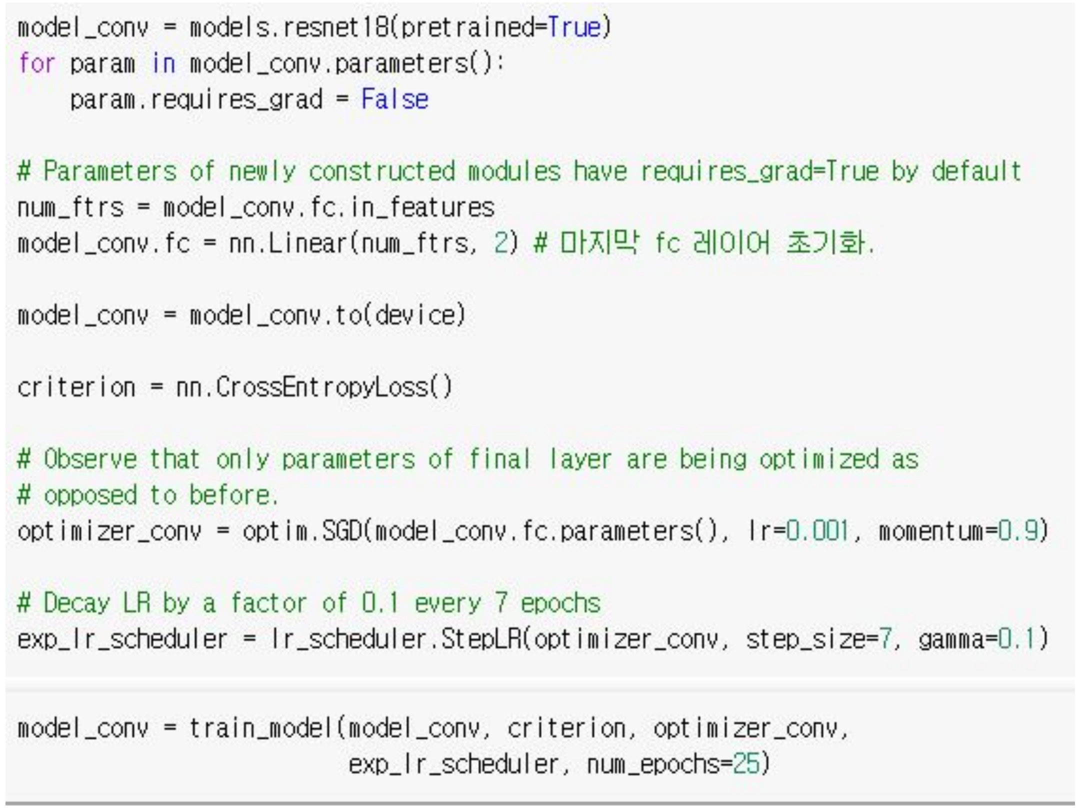


```python
visualize_model(model_ft)
```

실습 13

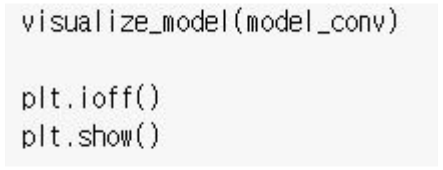

pretrained resnet-18 Weights
얻은 weights로 초기화
['bees', 'ants', 'bees', 'ants']
0
100
200
0
200
400
600
800
weight
고정
fcn
ant or bee
finetuning

실습 13



```python
model_conv = models.resnet18(pretrained=True)
for param in model_conv.parameters():
    param.requires_grad = False

# Parameters of newly constructed modules have requires_grad=True by default
num_ftrs = model_conv.fc.in_features
model_conv.fc = nn.Linear(num_ftrs, 2) # 마지막 fc 레이어 초기화.

model_conv = model_conv.to(device)

criterion = nn.CrossEntropyLoss()

# Observe that only parameters of final layer are being optimized as
# opposed to before.
optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr=0.001, momentum=0.9)

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=7, gamma=0.1)
```

```python
model_conv = train_model(model_conv, criterion, optimizer_conv,
                         exp_lr_scheduler, num_epochs=25)
```



```python
visualize_model(model_conv)

plt.ioff()
plt.show()
```