심층학습
[실습05] 합성곱 신경망(2)
Transfer Learning

SW융합학부 양희경

GitHub 로 실습코드 관리하길 추천합니다

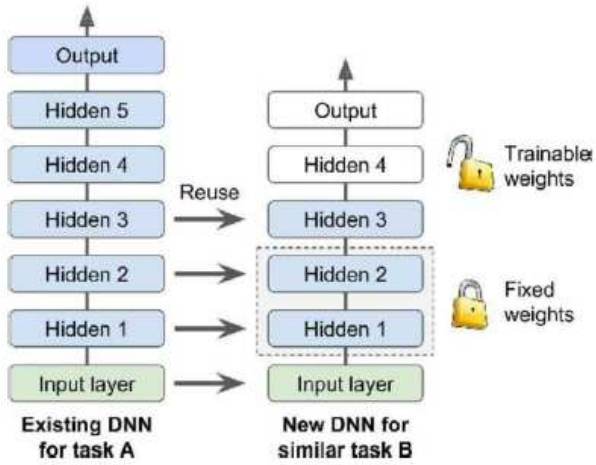
- AWS SageMaker 5GB 제약
- 포트폴리오 작성법 익힘(미래 나의 재산)
- 오픈소스에 기여

[리뷰] 4. 심층 신경망 훈련

4.5 미리훈련된 층 재사용

- 파이어니어의 팁pioneer's tip
  - 보통 큰 규모의 DNN을 처음부터 새로 훈련시키는 것 비추천

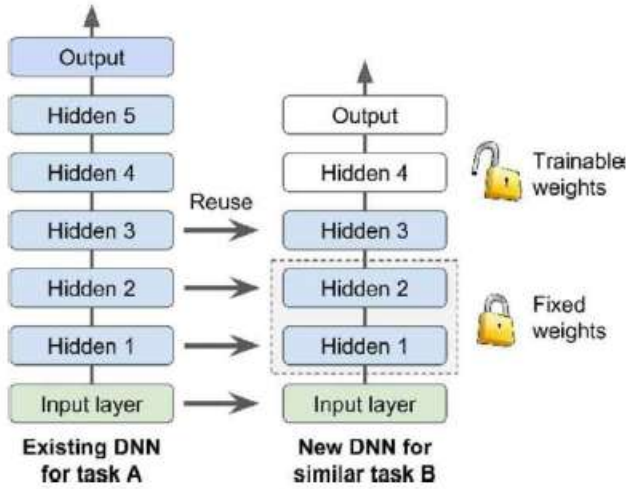
- 전이 학습Transfer learning
  - 풀고자 하는 작업과 비슷한 유형의 문제를 처리한 신경망이 있는 지 찾아보고, 그 신경망의 하위층을 재사용! (5장)
  - 훈련 속도 ↑, 훈련 데이터 적게 듦
  - 예) 동/식물, 자동차, 생필품 등 100 개 카테고리 이미지 분류하도록 훈련된 DNN이 있음 → 자동차 종류 분류하는 DNN 에서 재사용

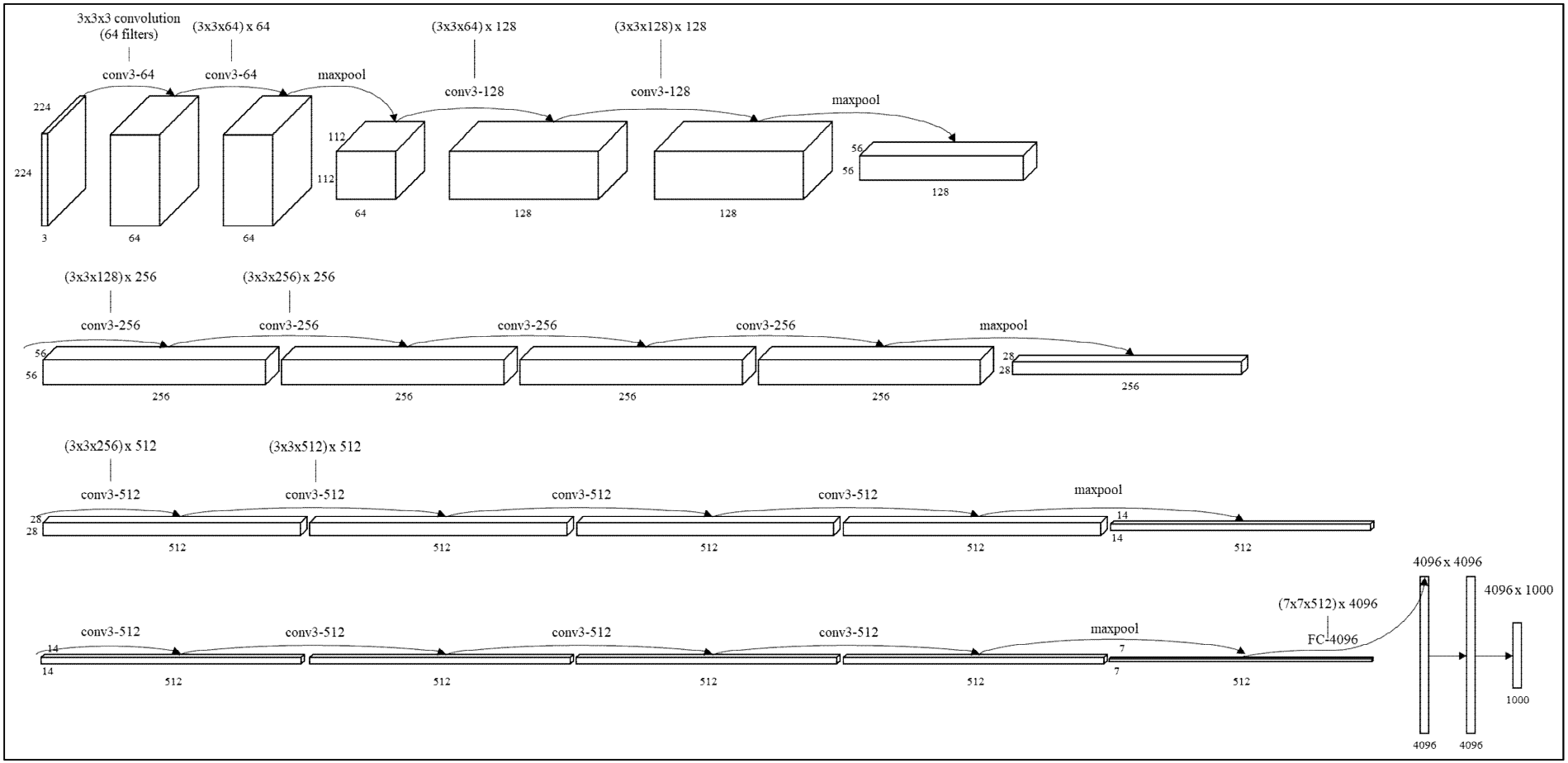


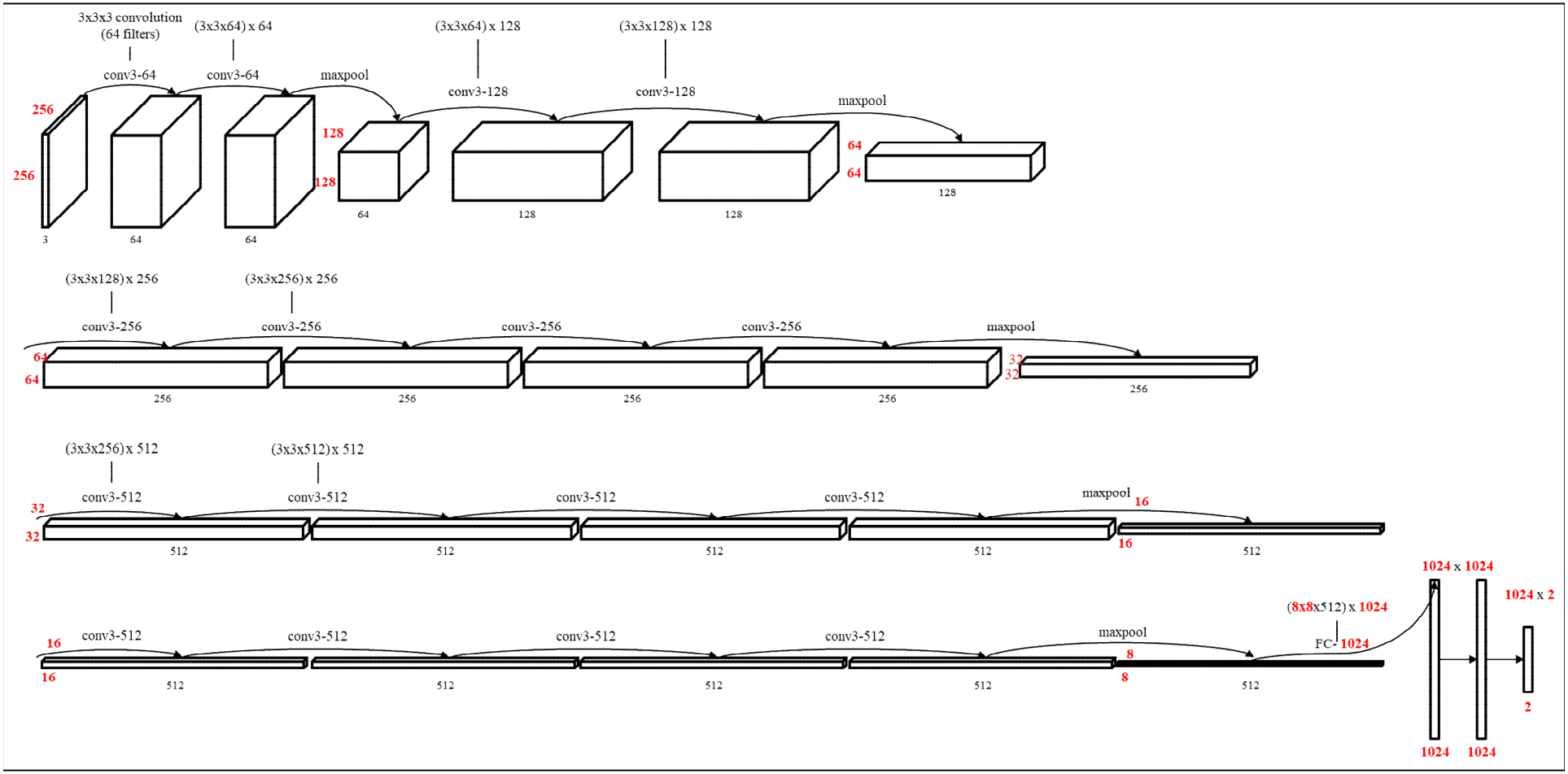
[리뷰] 4. 심층 신경망 훈련

4.5 미리훈련된 층 재사용

- 전이 학습Transfer learning
  - 원래 문제의 이미지와 다른 크기의 이미지를 사용한다면, 원본 모델에 맞도록 크기를 조절하는 전처리 단계 필요
  - 일반적으로 전이학습은 입력 데이터가 유사한 저수준 특성을 가질 때 잘 작동
  - 보통 마지막 1~2 개 레이어를 붙여, 이 레이어만 학습
  - 모델 저장소model zoo




3x3x3 convolution
(64 filters)
(3x3x64) x 64
(3x3x64) x 128
(3x3x128) x 128
conv3-64
conv3-64
maxpool
conv3-128
conv3-128
maxpool
224
224
3
64
64
112
112
64
128
128
56
56
128
(3x3x128) x 256
(3x3x256) x 256
conv3-256
conv3-256
conv3-256
conv3-256
maxpool
56
56
256
256
256
256
28
28
256
(3x3x256) x 512
(3x3x512) x 512
conv3-512
conv3-512
conv3-512
conv3-512
maxpool
28
28
512
512
512
512
14
14
512
conv3-512
conv3-512
conv3-512
conv3-512
maxpool
14
14
512
512
512
512
7
7
512
(7x7x512) x 4096
FC-4096
4096 x 4096
4096 x 1000
1000
4096
4096


3x3x3 convolution
(64 filters)
(3x3x64) x 64
(3x3x64) x 128
(3x3x128) x 128
conv3-64
conv3-64
maxpool
conv3-128
conv3-128
maxpool
256
256
128
128
64
64
3
64
64
64
128
128
128
(3x3x128) x 256
(3x3x256) x 256
conv3-256
conv3-256
conv3-256
conv3-256
maxpool
64
64
32
32
256
256
256
256
256
(3x3x256) x 512
(3x3x512) x 512
conv3-512
conv3-512
conv3-512
conv3-512
maxpool
32
32
16
16
512
512
512
512
512
16
16
conv3-512
conv3-512
conv3-512
conv3-512
maxpool
8
8
512
512
512
512
512
(8x8x512) x 1024
FC- 1024
1024 x 1024
1024 x 2
2
1024
1024

1. Settings

1) Important required libraries

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.init as init
import torch.utils.data as data
import torchvision.datasets as dset
import torchvision.models as models
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
from torch.autograd import Variable
import time
import matplotlib.pyplot as plt
import utils
```

2) Hyperparameter

```
batch_size= 16 #64  #1
learning_rate = 0.0001
epoch = 20

n_node = 1024  # customized last layer 의 노드 수. 64, 128, 256, 512, 1024
dropratio = 0.5   # 얼마나 드랍시킬지 inverse keepratio

imgsize = 256
```

2. Data Loader

트레이닝 데이터

```
#img_dir = "../../../images/painting_dataset/real_artwork_divided_shffl_4K/Train"
img_dir = "animal/train"
train_data = dset.ImageFolder(img_dir, transforms.Compose([
            # ①(512)③②RCrop  <-- Best !!
            transforms.CenterCrop(imgsize*2),      # ① CenterCrop(512)
            transforms.RandomCrop(imgsize),        # ③ RandomCrop
            transforms.RandomHorizontalFlip(),     # ② RandomHorizontalFlip

            transforms.Resize(imgsize),
            transforms.ToTensor()
            ]))
print(train_data.__len__())

train_batch = data.DataLoader(train_data, batch_size=batch_size,
                              shuffle=True, num_workers=2)
```

```
46
```

고정된 데이터 셋

```
# 2. Dev data
#img_dir = "../../../images/painting_dataset/real_artwork_divided_shffl_4K/Valid"
img_dir = "animal/val"
dev_data = dset.ImageFolder(img_dir, transforms.Compose([
            #transforms.Scale(256),
            #transforms.RandomSizedCrop(224),

            transforms.CenterCrop(size=imgsize),
            transforms.Resize(imgsize),
            transforms.ToTensor()
            ]))
dev_batch = data.DataLoader(dev_data, batch_size=batch_size,
                            shuffle=True, num_workers=2)
```

```
# 3. Test data
img_dir = "animal/test"
test_data = dset.ImageFolder(img_dir, transforms.Compose([
            #transforms.Scale(256),
            #transforms.RandomSizedCrop(224),

            transforms.CenterCrop(size=imgsize),
            transforms.Resize(imgsize),
            transforms.ToTensor()
            ]))
test_batch = data.DataLoader(test_data, batch_size=batch_size,
                            shuffle=True, num_workers=2)
```

```
28 nclass = len(train_data.classes)
29 print("# of classes: %d" %nclass)
30 print(train_data.classes)
31 print(train_data.class_to_idx)
32 print(train_data.__len__())
33
34 print("Training: %d, Dev: %d, Test: %d"
35       %(train_data.__len__(), dev_data.__len__(), test_data.__len__())),
36
37 # for imgs, labels in train_batch:
38 #     for j in range(len(imgs)):
39 #         img = transforms.ToPILImage()(imgs[j])
40 #         plt.title("label: %d" % labels[j])
41 #         plt.imshow(img)
42 #         plt.show()
```

```
# of classes: 2
['cats', 'dogs']
{'cats': 0, 'dogs': 1}
46
Training: 46, Dev: 17, Test: 41
```

```
# '.ipynb_checkpoints' 가 클래스로 나오는 경우, 새로운 폴더(train, val, test) 를 만들어 이동 시킬 것
print(train_data.classes)
print(dev_data.classes)
print(test_data.classes)
```

```
['cats', 'dogs']
['cats', 'dogs']
['cats', 'dogs']
```

Zip 파일로 데이터를 업로드하고, 서버 환경에서
unzip 명령어로 풀면 이러한 에러가 발생하지 않아요.

```
['cats', 'dogs']
['.ipynb_checkpoints', 'cats', 'dogs']
['cats', 'dogs']
```

3. Model

1) Pretrained VGG Model

```
1  vgg = models.vgg19(pretrained=True)
2
3  for name,module in vgg.named_children():
4      print(name)
5
6  print(list(vgg.children())[0])
7  print(list(vgg.children())[-1])
8
9  # cnt = 0
10 # for i in model.children():
11 #     print("yhk[%d]" %cnt),
12 #     print(i)
13 #     cnt = cnt+1
```

```
features
avgpool
classifier
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (6): ReLU(inplace=True)
  (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (8): ReLU(inplace=True)
  (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (11): ReLU(inplace=True)
  (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (13): ReLU(inplace=True)
  (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
```

```
Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
  (1): ReLU(inplace=True)
  (2): Dropout(p=0.5, inplace=False)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace=True)
  (5): Dropout(p=0.5, inplace=False)
  (6): Linear(in_features=4096, out_features=1000, bias=True)
)
```

```
print(list(vgg.children())[0][0])
```

```
Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
```

2) Customized Fully Model

```
base_dim = 64
fsize = imgsize/32

class MyVGG(nn.Module):
    def __init__(self):
        super(MyVGG, self).__init__()
        # [0]: features(conv), [1]: classifier(fc)
        self.layer0 = nn.Sequential(*list(vgg.children())[0])

        self.layer1 = nn.Sequential(
            nn.Linear(8*base_dim * fsize * fsize, n_node),
            nn.BatchNorm1d(n_node),
            nn.ReLU(),
            nn.Dropout2d(dropratio),  # 0.3 만큼 drop 하자.

            nn.Linear(n_node, n_node),
            nn.BatchNorm1d(n_node),
            nn.ReLU(),
            nn.Dropout2d(dropratio),

            nn.Linear(n_node, n_node),
            nn.BatchNorm1d(n_node),
            nn.ReLU(),
            nn.Dropout2d(dropratio),

            nn.Linear(n_node, nclass),
        )
        # weight initialization
        for m in self.layer1.modules():
            #print(m)
            if isinstance(m, nn.Conv2d):
                init.kaiming_normal(m.weight.data)  # RELU 일 때
                m.bias.data.fill_(0)
            if isinstance(m, nn.Linear):
                init.kaiming_normal(m.weight.data)
                m.bias.data.fill_(0)
    def forward(self, x):
        # layer0의 사이즈를 무식하게 프린트 하여 알아낼 수 있음(batchsize, x,x,x)
        #print(x.size())
        out = self.layer0(x)
        out = out.view(out.size(0), -1)
        out = self.layer1(out)
        return out
```

2) Customized Fully Model

```
base_dim = 64
fsize = imgsize/32

class MyVGG(nn.Module):
    def __init__(self):
        super(MyVGG, self).__init__()
        # [0]: features(conv), [1]: classifier(fc)
        self.layer0 = nn.Sequential(*list(vgg.children())[0])

        self.layer1 = nn.Sequential(
            nn.Linear(8*base_dim * fsize * fsize, n_node),
            nn.BatchNorm1d(n_node),
            nn.ReLU(),
            nn.Dropout2d(dropratio),  # 0.3 만큼 drop 하자.

            nn.Linear(n_node, n_node),
            nn.BatchNorm1d(n_node),
            nn.ReLU(),
            nn.Dropout2d(dropratio),

            nn.Linear(n_node, n_node),
            nn.BatchNorm1d(n_node),
            nn.ReLU(),
            nn.Dropout2d(dropratio),

            nn.Linear(n_node, nclass),
        )
```

```
        # weight initialization
        for m in self.layer1.modules():
            #print(m)
            if isinstance(m, nn.Conv2d):
                init.kaiming_normal(m.weight.data)  # REUL 일 때
                m.bias.data.fill_(0)
            if isinstance(m, nn.Linear):
                init.kaiming_normal(m.weight.data)
                m.bias.data.fill_(0)
    def forward(self, x):
        # layer0의 사이즈를 무식하게 프린트 하여 알아낼 수 있음(batchsize, x,x,x)
        #print(x.size())
        out = self.layer0(x)
        out = out.view(out.size(0), -1)
        out = self.layer1(out)
        return out
```

2) Customized Fully Model

```
base_dim = 64
fsize = imgsize/32

class MyVGG(nn.Module):
    def __init__(self):
        super(MyVGG, self).
        # [0]: features(con
        self.layer0 = nn.Se

        self.layer1 = nn.Se
            nn.Linear(8*bas
            nn.BatchNorm1d(
            nn.ReLU(),
            nn.Dropout2d(dr

            nn.Linear(n_nod
            nn.BatchNorm1d(
            nn.ReLU(),
            nn.Dropout2d(dr

            nn.Linear(n_node, n_node),
            nn.BatchNorm1d(n_node),
            nn.ReLU(),
            nn.Dropout2d(dropratio),

            nn.Linear(n_node, nclass),
        )
        # weight initialization
        for m in self.layer1.modules():
            #print(m)
            if isinstance(m, nn.Conv2d):
                init.kaiming_normal(m.weight.data)  # REUL 일 때
                m.bias.data.fill_(0)
            if isinstance(m, nn.Linear):
                init.kaiming_normal(m.weight.data)
                m.bias.data.fill_(0)
    def forward(self, x):
        # layer0의 사이즈를 무식하게 프린트 하여 알아낼 수 있음(batchsize, x,x,x)
        #print(x.size())
        out = self.layer0(x)
        out = out.view(out.size(0), -1)
        out = self.layer1(out)
        return out
```

3) Model on GPU

```
model = MyVGG().cuda()

for params in model.layer0.parameters():
    params.required_grad = False

for params in model.layer1.parameters():
    params.required_grad = True
```

```
for name in model.children():
    print(name)
```

```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (6): ReLU(inplace=True)
  (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (8): ReLU(inplace=True)
  (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
```

```
Sequential(
  (0): Linear(in_features=32768, out_features=1024, bias=True)
  (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): ReLU()
  (3): Dropout2d(p=0.5, inplace=False)
  (4): Linear(in_features=1024, out_features=1024, bias=True)
  (5): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (6): ReLU()
  (7): Dropout2d(p=0.5, inplace=False)
  (8): Linear(in_features=1024, out_features=1024, bias=True)
  (9): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (10): ReLU()
  (11): Dropout2d(p=0.5, inplace=False)
  (12): Linear(in_features=1024, out_features=2, bias=True)
)
```

4. Optimizer & Loss

```
1  loss_func = nn.CrossEntropyLoss()
2  optimizer = optim.Adam(model.layer1.parameters(),lr=learning_rate)
```

5. Train

```python
import utils

total_time = 0
disp_step = 10

to_train = True
if (to_train==False):
    #netname = './nets/media_vgg19_fixed.pkl'
    netname = './nets/catdog_vgg19_10.pkl'
    model = torch.load(netname)
else:
    print("3 layer, n_node: %d, dropratio: %.2f" %(n_node, dropratio))
    model.eval()  # evaluation(test) mode 로 바꾸기 -> dropout, batch normalization 에 영향을 줌.
    train_corr = utils.ComputeCorr(train_batch, model)
    dev_corr = utils.ComputeCorr(dev_batch, model)
    test_corr = utils.ComputeCorr(test_batch, model)
    print("Correct of train: %.2f, dev: %.2f, test: %.2f"
          %(train_corr, dev_corr, test_corr))
    model.train()
```

```
import utils

total_time = 0
disp_step = 10

to_train = True
if (to_train==False):
    #netname = './nets/media_vgg19_fixed.pkl'
    netname = './nets/catdog_vgg19_10.pkl'
    model = torch.load(netname)
else:
    print("3 layer, n_node: %d, dropratio: %.2f" %(n_node, dropratio))
    model.eval()  # evaluation(test) mode 로 바꾸기 -> dropout, batch normalization 에 영향을 줌.
    train_corr = utils.ComputeCorr(train_batch, model)
    dev_corr = utils.ComputeCorr(dev_batch, model)
    test_corr = utils.ComputeCorr(test_batch, model)
    print("Correct of train: %.2f, dev: %.2f, test: %.2f"
          %(train_corr, dev_corr, test_corr))
    model.train()

    netname = './nets/catdog_vgg19'

    # graph 그리기
    x_epoch = []
    y_train_err = []
    y_dev_err = []
    y_test_err = []

    x_epoch.append(0)
    y_train_err.append(100.0-train_corr)
    y_dev_err.append(100.0-dev_corr)
    y_test_err.append(100.0-test_corr)

#    # 학습을 재시작한다면
#    netname = './nets/media_pre_vgg19.pkl'
#    model = torch.load(netname)
#    # 파라미터 학습 여부 결정
#    for params in model.layer0.parameters():
#        params.required_grad = False
#    for params in model.layer1.parameters():
#        params.required_grad = True
#    for i in range(84, epoch):

    # 재시작하지 않는 다면
    for i in range(epoch):
        start_time = time.time()
        print("%d.." %i),
        for img,label in train_batch:
            img = Variable(img).cuda()
            label = Variable(label).cuda()

            optimizer.zero_grad()
            output = model(img)
            loss = loss_func(output,label)
            loss.backward()
            optimizer.step()

        end_time = time.time()
        duration = end_time - start_time
        total_time += duration
        if (i % disp_step == 0) or (i==epoch-1):
            torch.save(model, netname+'_%d.pkl'%i, )
            print("₩n[%d/%d] loss: %.3f, " %(i, epoch, (loss.cpu()).data.numpy())),

            # train, dev, train accr
            model.eval()  # evaluation(test) mode 로 바꾸기 -> dropout, batch normalization 에 영향을 줌.
            train_corr = utils.ComputeCorr(train_batch, model)
            dev_corr = utils.ComputeCorr(dev_batch, model)
            test_corr = utils.ComputeCorr(test_batch, model)
            print("Correct of train: %.2f, dev: %.2f, test: %.2f, "
                  %(train_corr, dev_corr, test_corr)),
            model.train()
            print("time: %.2f sec.." %(total_time))

            # graph 그리기
            x_epoch.append(i+1)
            y_train_err.append(100.0-train_corr)
```

```
21      netname = './nets/catdog_vgg19'
22
23      # graph 그리기
24      x_epoch = []
25      y_train_err = []
26      y_dev_err = []
27      y_test_err = []
28
29      x_epoch.append(0)
30      y_train_err.append(100.0-train_corr)
31      y_dev_err.append(100.0-dev_corr)
32      y_test_err.append(100.0-test_corr)
```

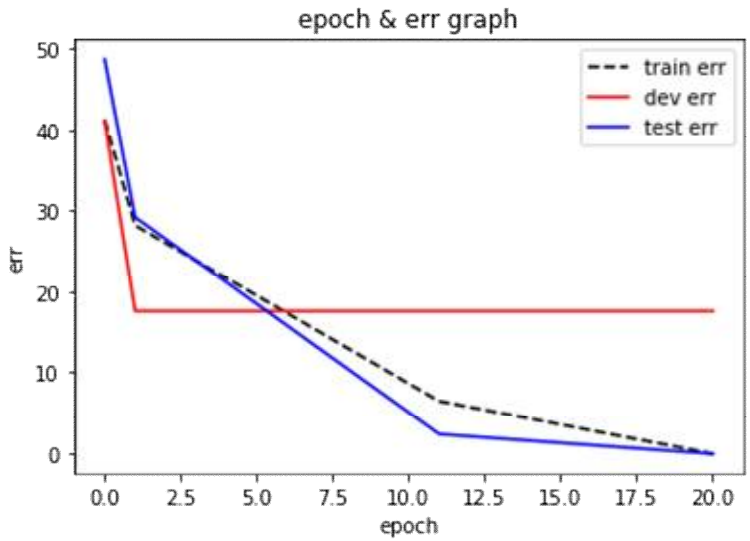
```python
import utils

total_time = 0
disp_step = 10

to_train = True
if (to_train==False):
    #netname = './nets/media_vgg19_fixed.pkl'
    netname = './nets/catdog_vgg19_10.pkl'
    model = torch.load(netname)
else:
    print("3 layer, n_node: %d, dropratio: %.2f" %(n_node, dropratio))
    model.eval()  # evaluation(test) mode 로 바꾸기 -> dropout, batch normalization 에 영향을 줌.
    train_corr = utils.ComputeCorr(train_batch, model)
    dev_corr = utils.ComputeCorr(dev_batch, model)
    test_corr = utils.ComputeCorr(test_batch, model)
    print("Correct of train: %.2f, dev: %.2f, test: %.2f"
          %(train_corr, dev_corr, test_corr))
    model.train()

    netname = './nets/catdog_vgg19'

    # graph 그리기
    x_epoch = []
    y_train_err = []
    y_dev_err = []
    y_test_err = []

    x_epoch.append(0)
    y_train_err.append(100.0-train_corr)
    y_dev_err.append(100.0-dev_corr)
    y_test_err.append(100.0-test_corr)

#     # 학습을 재시작한다면
#     netname = '../nets/media_pre_vgg19.pkl'
#     model = torch.load(netname)
#     # 파라미터 학습 여부 결정
#     for params in model.layer0.parameters():
#         params.required_grad = False
#     for params in model.layer1.parameters():
#         params.required_grad = True
#     for i in range(34, epoch):

    # 재시작하지 않는 다면
    for i in range(epoch):
        start_time = time.time()
        print("%d.." %i),
        for img,label in train_batch:
            img = Variable(img).cuda()
            label = Variable(label).cuda()

            optimizer.zero_grad()
            output = model(img)
            loss = loss_func(output,label)
            loss.backward()
            optimizer.step()

        end_time = time.time()
        duration = end_time - start_time
        total_time += duration
        if (i % disp_step == 0) or (i==epoch-1):
            torch.save(model, netname+'_%d.pkl'%i, )
            print("₩n[%d/%d] loss: %.3f, " %(i, epoch, (loss.cpu()).data.numpy())),

            # train, dev, train accr
            model.eval()  # evaluation(test) mode 로 바꾸기 -> dropout, batch normalization 에 영향을 줌.
            train_corr = utils.ComputeCorr(train_batch, model)
            dev_corr = utils.ComputeCorr(dev_batch, model)
            test_corr = utils.ComputeCorr(test_batch, model)
            print("Correct of train: %.2f, dev: %.2f, test: %.2f, "
                  %(train_corr, dev_corr, test_corr)),
            model.train()
            print("time: %.2f sec.." %(total_time))

            # graph 그리기
            x_epoch.append(i+1)
            y_train_err.append(100.0-train_corr)
```

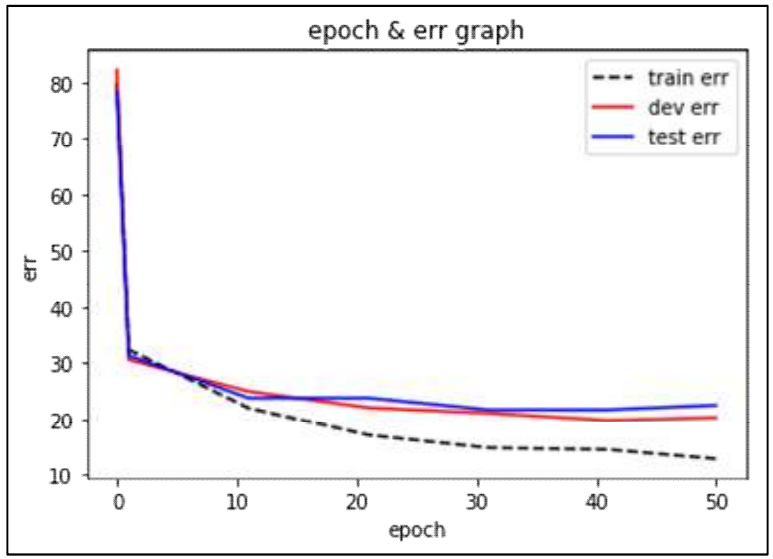
```python
#     # 학습을 재시작한다면
#     netname = '../nets/media_pre_vgg19.pkl'
#     model = torch.load(netname)
#     # 파라미터 학습 여부 결정
#     for params in model.layer0.parameters():
#         params.required_grad = False
#     for params in model.layer1.parameters():
#         params.required_grad = True
#     for i in range(34, epoch):

    # 재시작하지 않는 다면
    for i in range(epoch):
        start_time = time.time()
        print("%d.." %i),
        for img,label in train_batch:
            img = Variable(img).cuda()
            label = Variable(label).cuda()

            optimizer.zero_grad()
            output = model(img)
            loss = loss_func(output,label)
            loss.backward()
            optimizer.step()
```

```
58            end_time = time.time()
59            duration = end_time - start_time
60            total_time += duration
61            if (i % disp_step == 0) or (i==epoch-1):
62                torch.save(model, netname+'_%d.pkl'%i, )
63                print("₩n[%d/%d] loss: %.3f, " %(i, epoch, (loss.cpu()).data.numpy())),
64
65                # evaluation(test) mode 로 바꾸기 -> dropout, batch normalization 에 영향을 줌.
66                model.eval()
67
68                # train, dev, train accr
69                train_corr = utils.ComputeCorr(train_batch, model)
70                dev_corr = utils.ComputeCorr(dev_batch, model)
71                test_corr = utils.ComputeCorr(test_batch, model)
72                print("Correct of train: %.2f, dev: %.2f, test: %.2f, "
73                      %(train_corr, dev_corr, test_corr)),
74                model.train()
75                print("time: %.2f sec.." %(total_time))
76
77                # graph 그리기
78                x_epoch.append(i+1)
79                y_train_err.append(100.0-train_corr)
80                y_dev_err.append(100.0-dev_corr)
81                y_test_err.append(100.0-test_corr)
82        print("Total time: %.2f sec" %total_time)
```

```
3 layer, n_node: 1024, dropratio: 0.50
Correct of train: 58.70, dev: 58.82, test: 51.22
0..

/home/ec2-user/anaconda3/envs/pytorch_p27/lib/python2.7/site-packages/torch/serialization.py
g: Couldn't retrieve source code for container of type MyVGG. It won't be checked for correc
ng.
  "type " + obj.__name__ + ". It won't be checked "


[0/20] loss: 1.206,  Correct of train: 71.74, dev: 82.35, test: 70.73,  time: 0.39 sec..
1.. 2.. 3.. 4.. 5.. 6.. 7.. 8.. 9.. 10..
[10/20] loss: 0.673,  Correct of train: 93.48, dev: 82.35, test: 97.56,  time: 4.37 sec..
11.. 12.. 13.. 14.. 15.. 16.. 17.. 18.. 19..
[19/20] loss: 0.207,  Correct of train: 100.00, dev: 82.35, test: 100.00,  time: 7.79 sec..
Total time: 7.79 sec
```

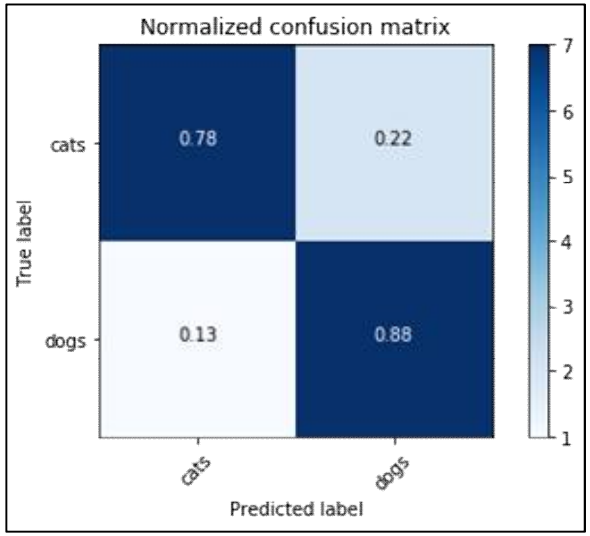
```
# epoch-err curve
if (to_train):
    plt.plot(x_epoch, y_train_err, color='black', label='train err', linestyle='--')
    plt.plot(x_epoch, y_dev_err, color='red', label='dev err')
    plt.plot(x_epoch, y_test_err, color='blue', label='test err')

    plt.xlabel('epoch')
    plt.ylabel('err')
    plt.title('epoch & err graph')
    plt.legend(loc="upper right")
    plt.show()
```

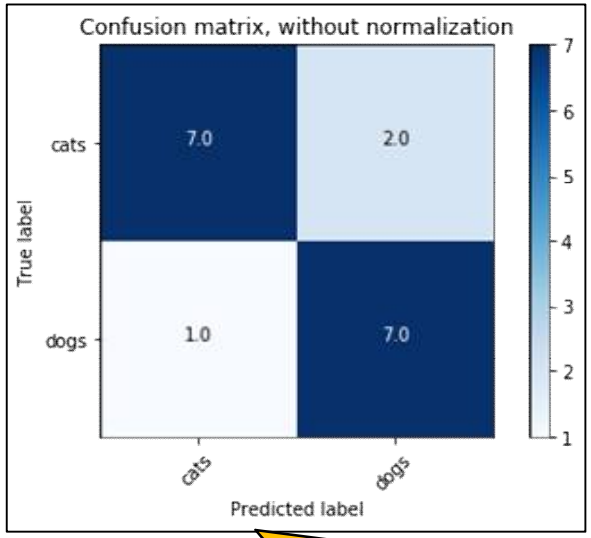




6. Evaluation for dev & test data

```
1  model.eval()   # evaluation(test) mode 로 바꾸기 -> dropout, batch norma
2  utils.EvaluateClassifier(dev_batch, model, dev_data.classes, batch_size)
```



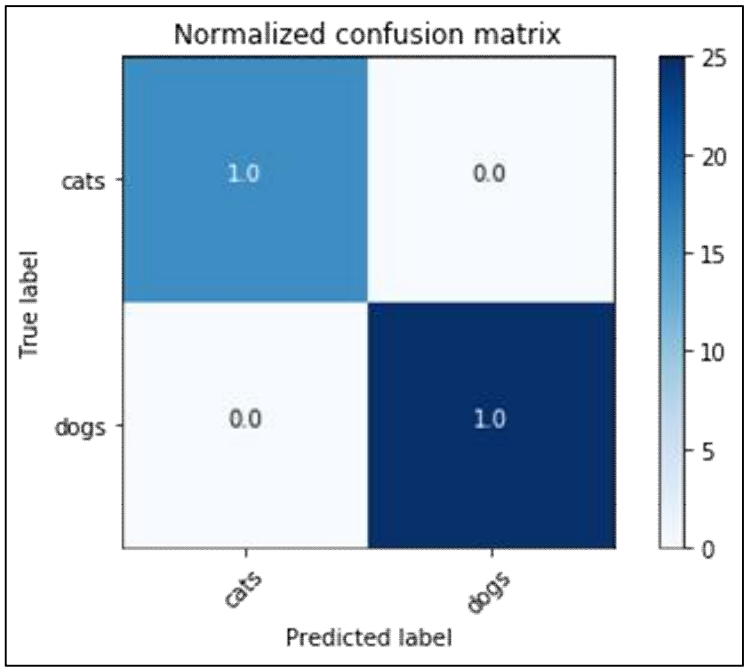


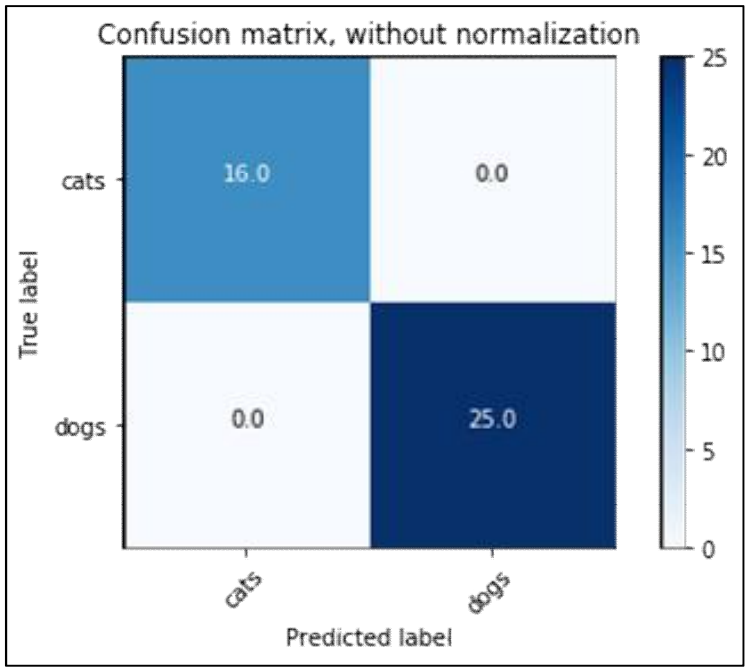

```
         acc   |  pre   |  rec   |  f1
 cats: 0.82 |    0.88 |  0.78 |  0.82
 dogs: 0.82 |    0.78 |  0.88 |  0.82
*accuracy: 0.82, precision: 0.83, recall: 0.83, *f1 score: 0.82
[AP] cats: 0.96 dogs: 0.97
[mAP] 0.964
[miAP] 0.912
```

- Confusion matrix, precision, recall, F1 score 등의 개념은 기계학습시간에 배우는 것으로, 이 강의에서는 ' 분류 문제 성능 측정 metric 이구나 ' 정도만 알고 넘어갑시다.
- 심층학습 과목의 시험에는 출제하지 않겠습니다.

```
(array([0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0]),
 array([0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0]),
 array([0.98118323, 0.99716514, 0.51210082, 0.55733442, 0.80914319,
        0.98838562, 0.70135534, 0.98953331, 0.5148167 , 0.58564794,
        0.57090443, 0.86580342, 0.78719759, 0.99546295, 0.99332422,
        0.73675084, 0.50962478]))
```

```
model.eval()
_, _,_ = utils.EvaluateClassifier(test_batch, model, test_data.classes, batch_size)
```

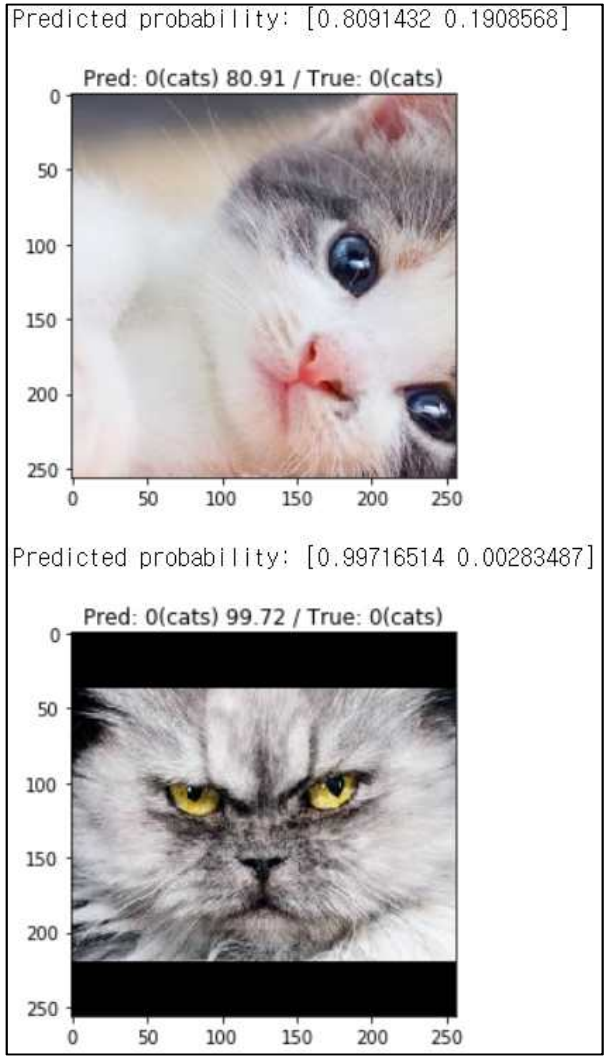





```
         acc  |  pre  |  rec  |  f1
 cats: 1.00 |    1.00 |  1.00 |  1.00
 dogs: 1.00 |    1.00 |  1.00 |  1.00
*accuracy: 1.00, precision: 1.00, recall: 1.00, *f1 score: 1.00
[AP] cats: 1.00 dogs: 1.00
[mAP] 1.000
[miAP] 1.000
```

```
1  utils.VisTFPred(dev_batch, model, test_data.classes, batch_size, i_n=2)
```

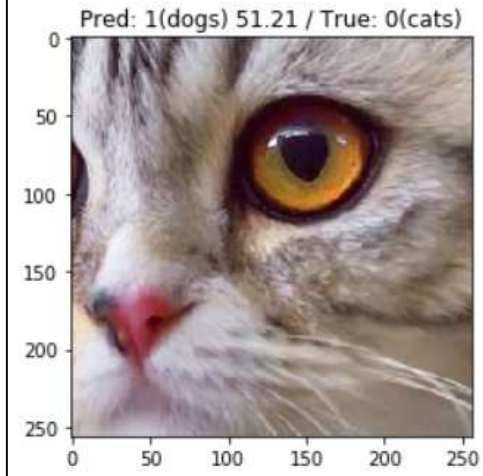
```
Category: cats
 True predicted images/total cats category: 7 / 9
```



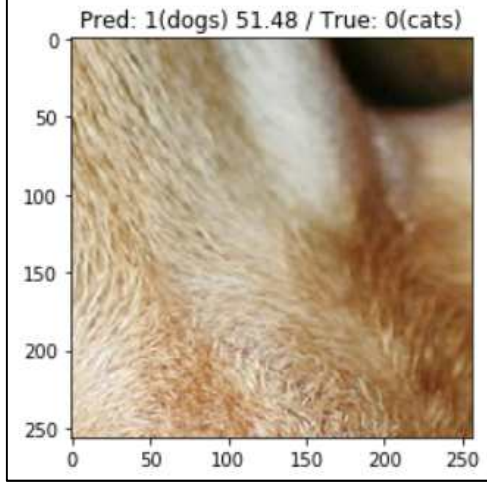
```
1 utils.VisTFPred(dev_batch, model, test_data.classes, batch_size, i_n=2)
```

```
False predicted images/total cats category: 2 / 9
```

Predicted probability: [0.4878992 0.5121008]

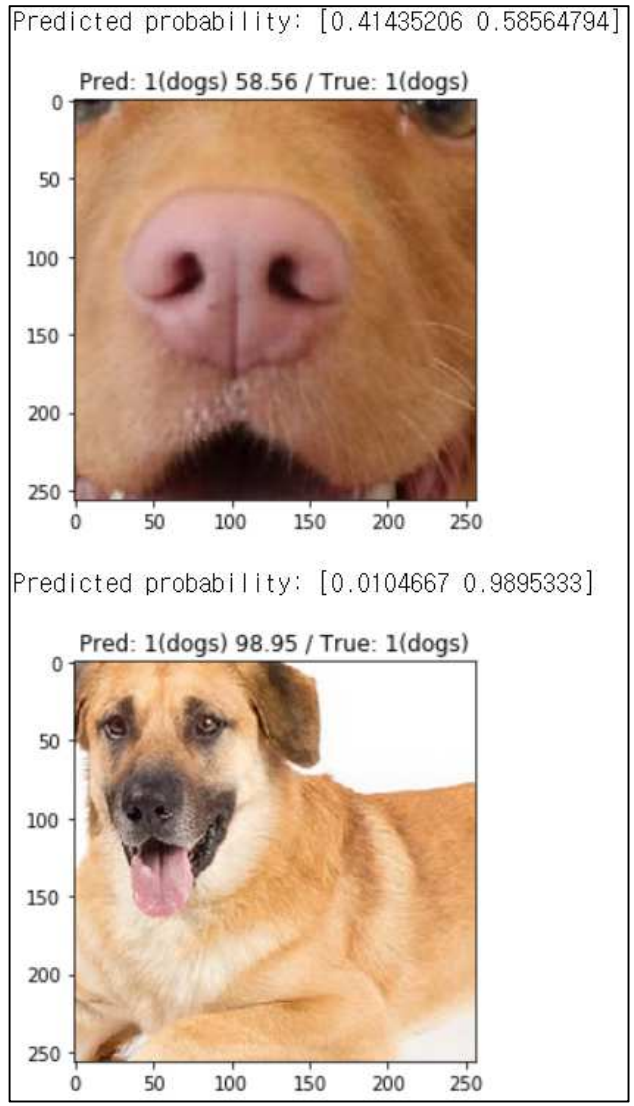


Predicted probability: [0.48518327 0.5148167]



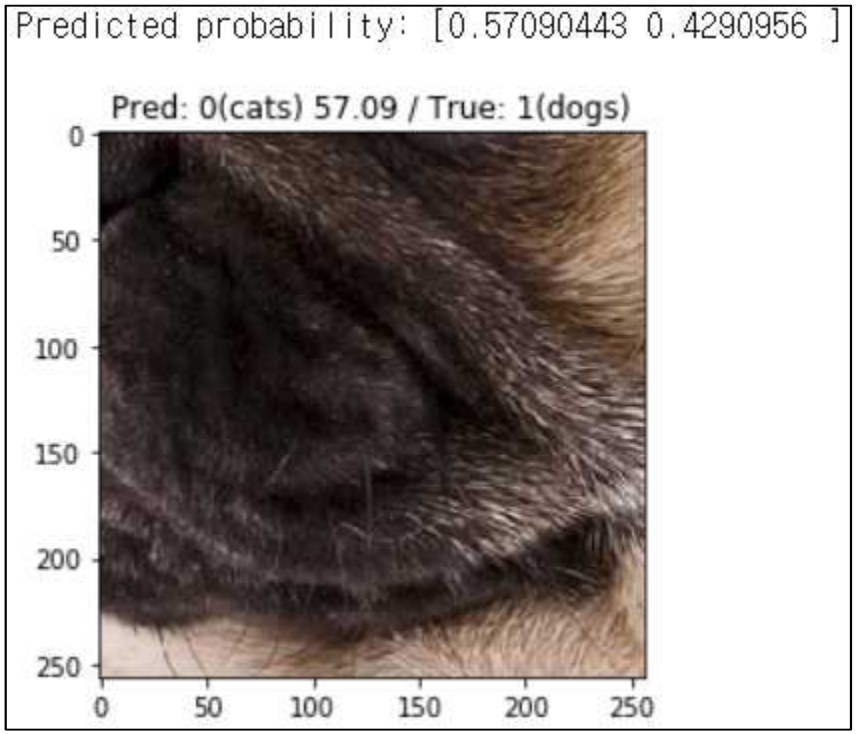
```
1 utils.VisTFPred(dev_batch, model, test_data.classes, batch_size, i_n=2)
```

```
Category: dogs
 True predicted images/total dogs category: 7 / 8
```



```
False predicted images/total dogs category: 1 / 8
```

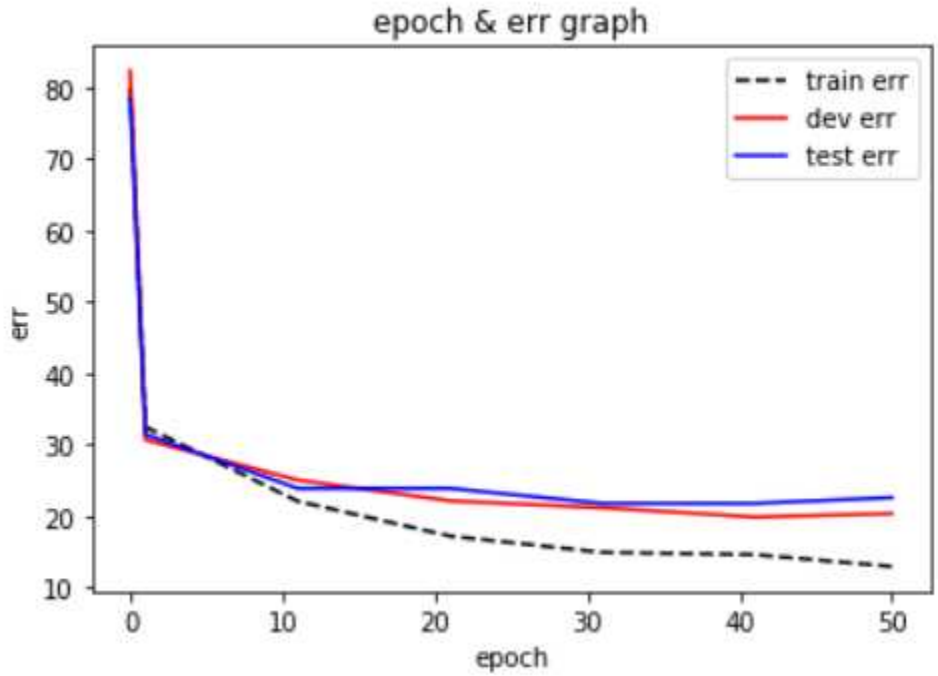
```
Predicted probability: [0.57090443 0.4290956 ]
```



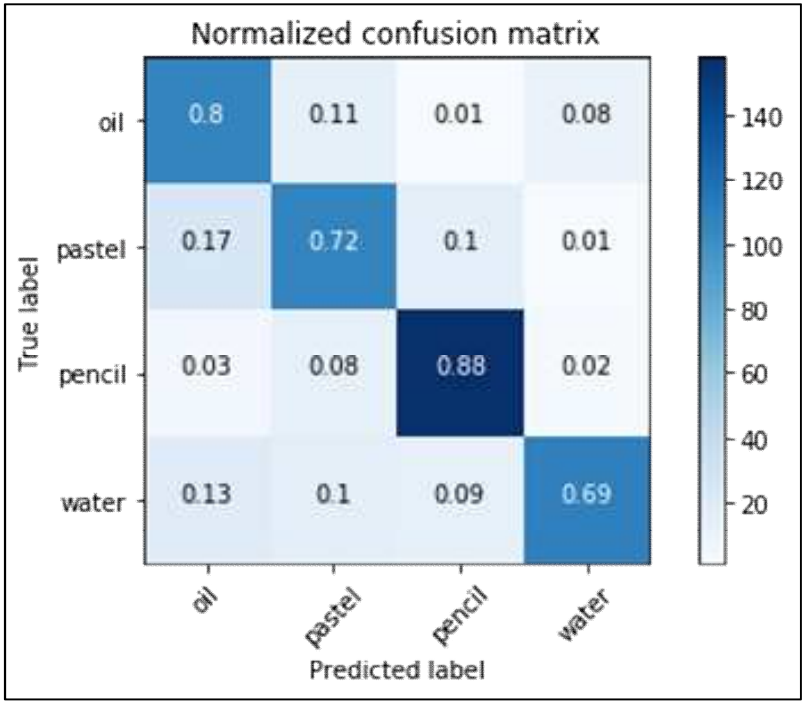
미술도구 효과 분류 (참고)

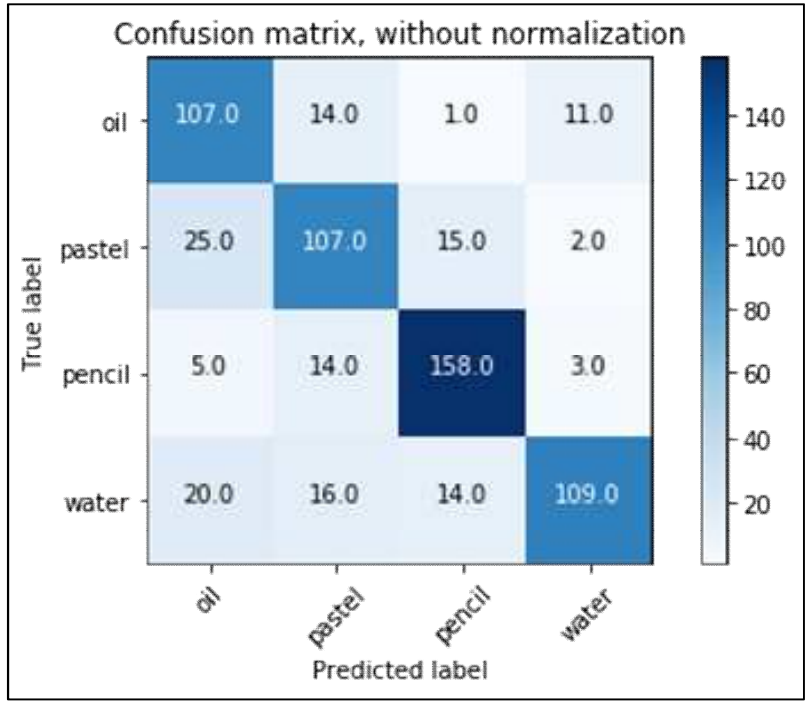
```
Correct of train: 20.03, dev: 17.74, test: 21.74
0..

/usr/local/lib/python2.7/dist-packages/torch/serialization.py:256: UserWarning: Couldn't retr
e for container of type MyVGG. It won't be checked for correctness upon loading.
  "type " + obj.__name__ + ". It won't be checked "


[0/50] loss: 1.187,  Correct of train: 67.56, dev: 69.35, test: 68.60,  time: 36.84 sec..
1.. 2.. 3.. 4.. 5.. 6.. 7.. 8.. 9.. 10..
[10/50] loss: 0.835,  Correct of train: 77.96, dev: 75.00, test: 76.17,  time: 455.65 sec..
11.. 12.. 13.. 14.. 15.. 16.. 17.. 18.. 19.. 20..
[20/50] loss: 0.505,  Correct of train: 82.83, dev: 77.90, test: 76.17,  time: 870.88 sec..
21.. 22.. 23.. 24.. 25.. 26.. 27.. 28.. 29.. 30..
[30/50] loss: 0.527,  Correct of train: 85.11, dev: 78.87, test: 78.26,  time: 1280.66 sec..
31.. 32.. 33.. 34.. 35.. 36.. 37.. 38.. 39.. 40..
[40/50] loss: 0.780,  Correct of train: 85.42, dev: 80.16, test: 78.26,  time: 1688.65 sec..
41.. 42.. 43.. 44.. 45.. 46.. 47.. 48.. 49..
[49/50] loss: 0.261,  Correct of train: 87.08, dev: 79.68, test: 77.46,  time: 2054.75 sec..
Total time: 2054.75 sec
```

미술도구 효과 분류 (참고)


epoch & err graph
train err
dev err
test err
err
epoch
80
70
60
50
40
30
20
10
0
10
20
30
40
50

미술도구 효과 분류 (참고)

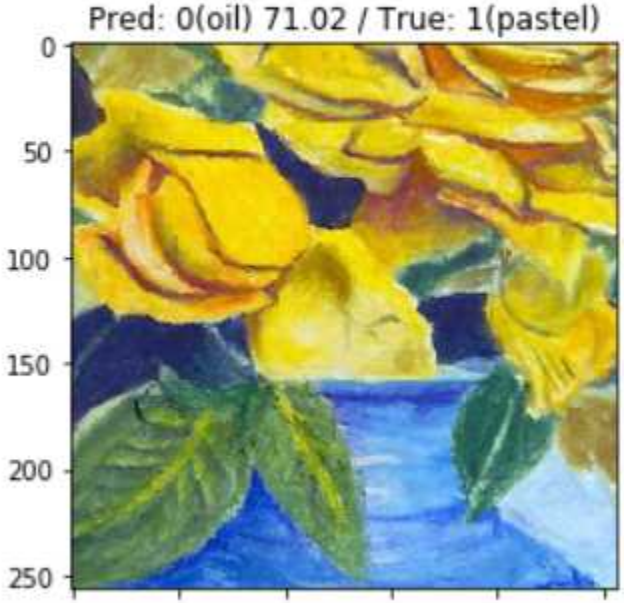




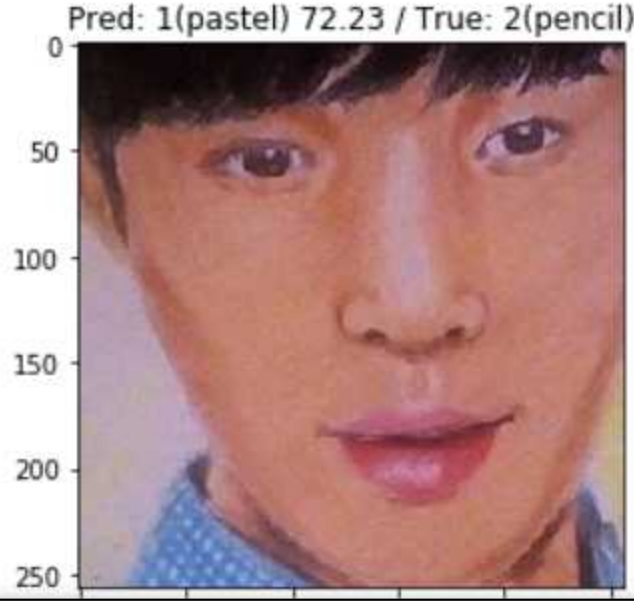

```
        acc   |  pre   |  rec   |  f1
oil: 0.88 |     0.68 |  0.80 |  0.74
pastel: 0.80 |  0.71 |  0.72 |  0.71
pencil: 0.84 |  0.84 |  0.88 |  0.86
water: 0.89 |   0.87 |  0.69 |  0.77
*accuracy: 0.85, precision: 0.78, recall: 0.77, *f1 score: 0.77
[AP] oil: 0.83 pastel: 0.79 pencil: 0.94 water: 0.90
[mAP] 0.864
[miAP] 0.927
```

미술도구 효과 분류 (참고)

```
False predicted images/total pastel category: 42 / 149
Predicted probability: [0.710221   0.12368666 0.08959635 0.07649599]
```




```
False predicted images/total pencil category: 22 / 180
Predicted probability: [7.0927717e-04 7.2234809e-01 2.6
```



```
class MyResnet(nn.Module):
    def __init__(self):
        super(MyResnet, self).__init__()
        self.layer0 = nn.Sequential(*list(resnet.children())[0:8])  # [0:8]: features(conv), [9]: classifier(fc)
        self.layer1 = nn.Sequential(
            #nn.Linear(8*base_dim * fsize * fsize, 1024),
            nn.Linear(2048*8*8, n_node),   # 256*6*6: out = self.layer0(x) 의 out 을 프린트하는 무식한 방법으로 알아냄.
            nn.BatchNorm1d(n_node),
            nn.ReLU(),
            nn.Dropout2d(dropratio),  # 0.3 만큼 drop 하자.

            nn.Linear(n_node, n_node),
            nn.BatchNorm1d(n_node),
            nn.ReLU(),
            nn.Dropout2d(dropratio),

#             nn.Linear(n_node, n_node),
#             nn.BatchNorm1d(n_node),
#             nn.ReLU(),
#             nn.Dropout2d(dropratio),

            nn.Linear(n_node, nclass),
        )
        # weight initialization
        for m in self.layer1.modules():
            if isinstance(m, nn.Conv2d):
                init.kaiming_normal(m.weight.data)  # RELU 일 때
                m.bias.data.fill_(0)
            if isinstance(m, nn.Linear):
                init.kaiming_normal(m.weight.data)
                m.bias.data.fill_(0)
    def forward(self, x):
        #print(x.size())  # layer0의 사이즈를 무식하게 프린트 하여 알아낼 수 있음(batchsize, x,x,x)
        out = self.layer0(x)
        out = out.view(out.size(0), -1)
        out = self.layer1(out)
        return out
```

오늘의 과제

- 나만의 커스텀 데이터 셋을 구축한다.
  - 예) car/{train, val, test}/{sedan, sports_car, suv, truck}
  - 용량 주의
- 데이터셋을 업로드한다
  - '.ipynb_checkpoints' 폴더 주의
  - 압축된 파일(.zip)을 업로드하고, 실습 관경에서 압축 풀면 에러 발생 안됨
    'unzip file.zip' 하여
- '[실습05] 합성곱 신경망(2)' 을 실습한다.
  - P. 7~28
  - 참고 코드를 활용하여 VGG19 이 외 다른 네트워크 구조를 커스터마이즈 해도 좋음 (가/감점 없음)
- HTML, .ipynb 파일을 다운받아 e-campus 에 제출한다.
  - utils.VisTFPred(test_batch, model, test_data.classes, batch_size, i_n=2)
  - 시각화 이미지 총 10장 이하로
- 마감: e-campus 과제 마감일 확인