심층학습 [실습05] 합성곱 신경망(2)

201911019 최현민 - GitHub url

1. Settings

```
!unzip "/content/animal+utils.zip"
```

```
  inflating: animal/train/fox/image28.jpg
  inflating: animal/train/fox/image29.jpg
  inflating: animal/train/fox/image30.jpg
  inflating: animal/train/fox/image31.jpg
  inflating: animal/train/fox/image32.jpg
  inflating: animal/train/fox/image33.jpg
  inflating: animal/train/fox/image34.jpg
  inflating: animal/train/fox/image35.jpeg
  inflating: animal/train/fox/image36.jpeg
  inflating: animal/train/fox/image37.jpeg
  inflating: animal/train/fox/image38.jpeg
  inflating: animal/train/fox/image39.jpg
  inflating: animal/train/fox/image40.jpeg
   creating: animal/train/rabbit/
  inflating: animal/train/rabbit/image21.jpeg
  inflating: animal/train/rabbit/image22.jpeg
  inflating: animal/train/rabbit/image23.jpg
  inflating: animal/train/rabbit/image24.jpg
  inflating: animal/train/rabbit/image25.jpg
  inflating: animal/train/rabbit/image26.jpg
  inflating: animal/train/rabbit/image27.jpg
  inflating: animal/train/rabbit/image28.jpeg
  inflating: animal/train/rabbit/image29.jpg
  inflating: animal/train/rabbit/image30.jpeg
  inflating: animal/train/rabbit/image31.jpg
  inflating: animal/train/rabbit/image32.jpg
  inflating: animal/train/rabbit/image33.jpg
  inflating: animal/train/rabbit/image34.jpg
  inflating: animal/train/rabbit/image35.jpg
  inflating: animal/train/rabbit/image36.jpeg
  inflating: animal/train/rabbit/image37.jpg
  inflating: animal/train/rabbit/image38.jpeg
  inflating: animal/train/rabbit/image39.jpg
  inflating: animal/train/rabbit/image40.jpg
   creating: animal/val/
   creating: animal/val/fox/
  inflating: animal/val/fox/image41.jpeg
  inflating: animal/val/fox/image42.jpeg
  inflating: animal/val/fox/image43.jpeg
  inflating: animal/val/fox/image43.jpg
  inflating: animal/val/fox/image44.jpg
  inflating: animal/val/fox/image45.jpg
  inflating: animal/val/fox/image46.jpg
  inflating: animal/val/fox/image47.jpg
  inflating: animal/val/fox/image48.jpg
  inflating: animal/val/fox/image49.jpeg
  inflating: animal/val/fox/image50.jpg
```

```
 creating: animal/val/rabbit/
inflating: animal/val/rabbit/image41.jpg
inflating: animal/val/rabbit/image42.jpg
inflating: animal/val/rabbit/image43.png
inflating: animal/val/rabbit/image44.jpeg
inflating: animal/val/rabbit/image45.jpg
inflating: animal/val/rabbit/image46.jpg
inflating: animal/val/rabbit/image47.jpg
inflating: animal/val/rabbit/image48.jpg
inflating: animal/val/rabbit/image49.jpg
inflating: animal/val/rabbit/image50.jpeg
inflating: utils.py
```

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 = 50

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 = "/content/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.RandomCrop(imgsize),     # ③ RandomCrop
            transforms.RandomHorizontalFlip(),  # ② RandomHorizontalFlip

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

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

```
40
/usr/local/lib/python3.7/dist-packages/torchvision/transforms/transforms.py:285: UserWarning
  warnings.warn("The use of the transforms.Scale transform is deprecated, " +
```

고정된 데이터 셋

```
# 2. Dev data
img_dir = "/content/animal/val"
dev_data = dset.ImageFolder(img_dir, transforms.Compose([
    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 = "/content/animal/test"
test_data = dset.ImageFolder(img_dir, transforms.Compose([
    transforms.CenterCrop(size=imgsize),
    transforms.Resize(imgsize),
    transforms.ToTensor()
]))
test_batch = data.DataLoader(test_data, batch_size=batch_size, shuffle=True, num_workers=2)

nclass = len(train_data.classes)
print("# of classes: %d" %nclass)
print(train_data.classes)
print(train_data.class_to_idx)
print(train_data.__len__())

print("Training: %d, Dev: %d, Test: %d" %(train_data.__len__(), dev_data.__len__(), test_data.__ler
```

```
# of classes: 2
['fox', 'rabbit']
{'fox': 0, 'rabbit': 1}
40
Training: 40, Dev: 21, Test: 39
```

```
print(train_data.classes)
print(dev_data.classes)
print(test_data.classes)
```

```
['fox', 'rabbit']
['fox', 'rabbit']
['fox', 'rabbit']
```

3. Model

1) Pretrained VGG Model

```
vgg = models.vgg19(pretrained=True)

for name, module in vgg.named_children():
    print(name)

print(list(vgg.children())[0])
print(list(vgg.children())[-1])
```

```
Downloading: "https://download.pytorch.org/models/vgg19-dcbb9e9d.pth" to /root/.cache/torch/l
100%                                      548M/548M [00:03<00:00, 147MB/s]

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)
```

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

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

2) Customized Fully Model

```
  (17): ReLU(inplace=True)
```

```
base_dim = 64
fsize = imgsize//32

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

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
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:33: UserWarning: nn.init.kaimin
```

```
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))
  (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))
  (15): ReLU(inplace=True)
  (16): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (17): ReLU(inplace=True)
  (18): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (19): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (20): ReLU(inplace=True)
  (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (22): ReLU(inplace=True)
  (23): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (24): ReLU(inplace=True)
  (25): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (26): ReLU(inplace=True)
  (27): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
```

```
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): ReLU(inplace=True)
    (32): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (33): ReLU(inplace=True)
    (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (35): ReLU(inplace=True)
    (36): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  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

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

5. Train

```
import utils

total_time = 0
disp_step = 10

to_train = True
if (to_train==False):
    #netname = './nets/media_vgg19_fixed.pkl'
    #netname = './nets/media_vgg19_RCrop_fixed.pkl'
    netname = '/content/media_vgg19_50.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()
```

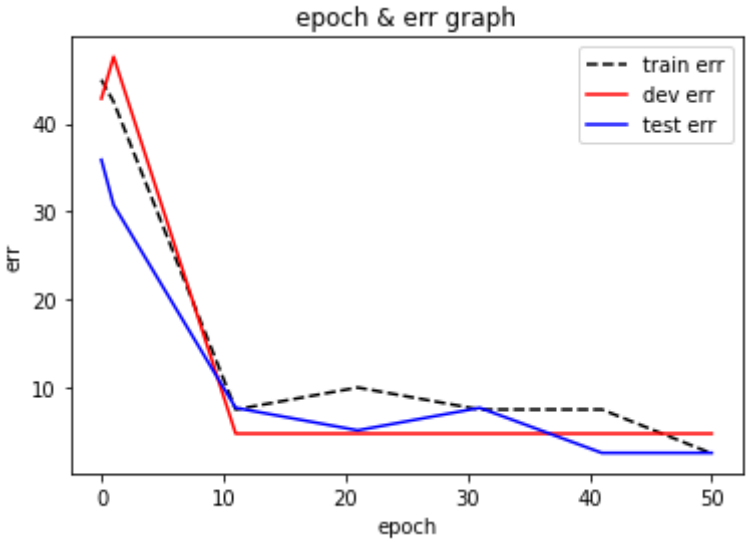
```
    netname = '/content/media_vgg19'

    # 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 영향
            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_cc
            model.train()
            print("time: %.2f sec.." %(total_time))

            # graph 그리기
```

```
            x_epoch.append(i+1)
            y_train_err.append(100.0-train_corr)
            y_dev_err.append(100.0-dev_corr)
            y_test_err.append(100.0-test_corr)
    print("Total time: %.2f sec" %total_time)
```

```
3 layer, n_node: 1024, dropratio: 0.50
Correct of train: 55.00, dev: 57.14, test: 64.10
0..

[0/50] loss: 0.580,
Correct of train: 57.50, dev: 52.38, test: 69.23,
time: 1.01 sec..
1..
2..
3..
4..
5..
6..
7..
8..
9..
10..

[10/50] loss: 0.137,
Correct of train: 92.50, dev: 95.24, test: 92.31,
time: 10.82 sec..
11..
12..
13..
14..
15..
16..
17..
18..
19..
20..

[20/50] loss: 0.609,
Correct of train: 90.00, dev: 95.24, test: 94.87,
time: 20.89 sec..
21..
22..
23..
24..
25..
26..
27..
28..
29..
30..

[30/50] loss: 0.225,
Correct of train: 92.50, dev: 95.24, test: 92.31,
time: 31.12 sec..
31..
32..
33..
34..
35..
36..
```

```
37..
38..
39..
40..
```

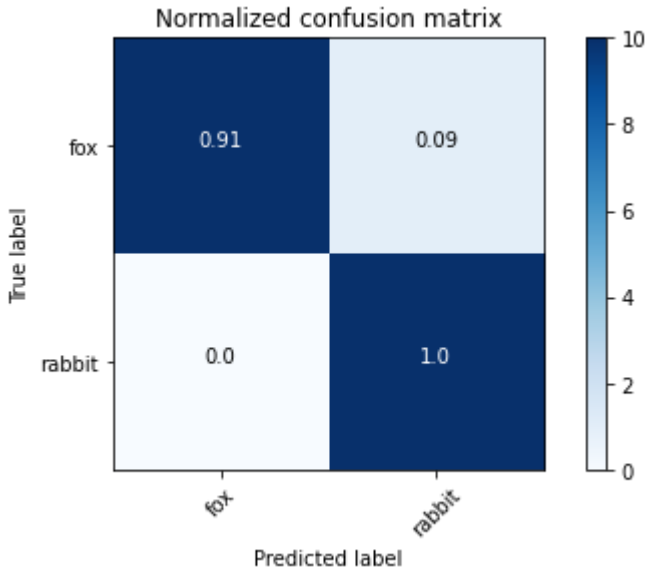
```
# epoch-err curve
if (to_train):
    plt.plot(x_epoch, y_train_err, color='black', label='train err', linestyle='--')
    plt.plot(x_epoch, y_dev_err, color='red', label='dev err')
    plt.plot(x_epoch, y_test_err, color='blue', label='test err')

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



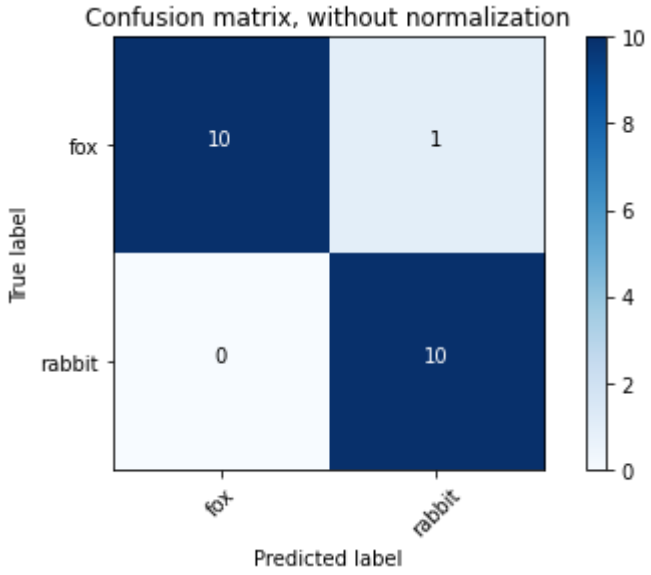
6. Evaluation for dev & test data

```
model.eval()   # evaluation(test) mode 로 바꾸기 -> dropout, batch normalization 에 영향을 줌.
utils.EvaluateClassifier(dev_batch, model, dev_data.classes, batch_size)
```

```
Normalized confusion matrix
```



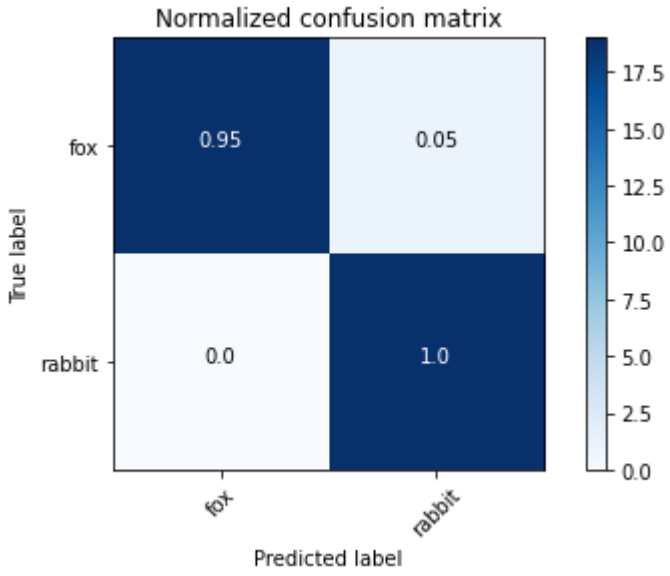

```
Confusion matrix, without normalization
```



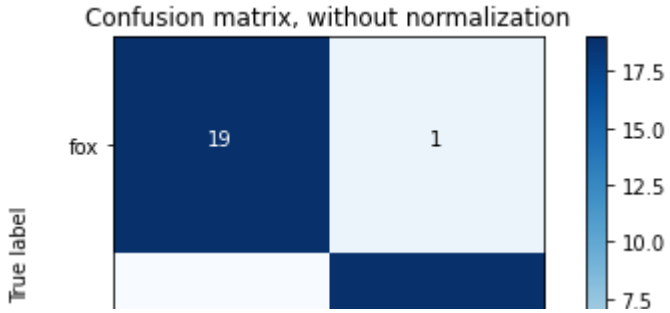

```
          acc  |  pre  |  rec  |  f1
```

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

```
Normalized confusion matrix
```

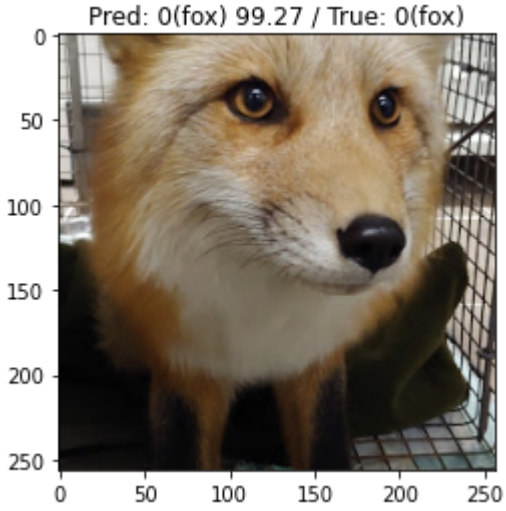



```
Confusion matrix, without normalization
```

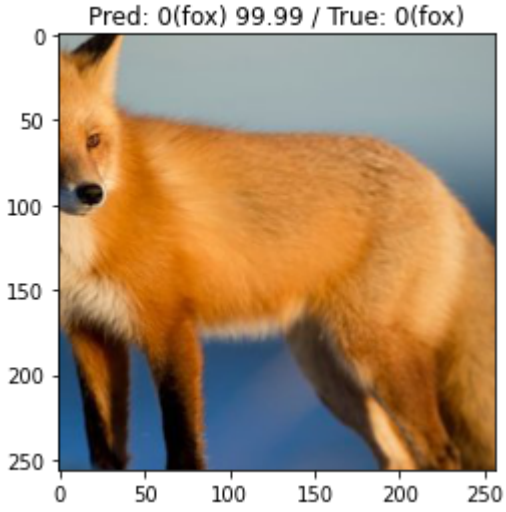



```
utils.VisTFPred(test_batch, model, test_data.classes, batch_size, i_n=2)
```

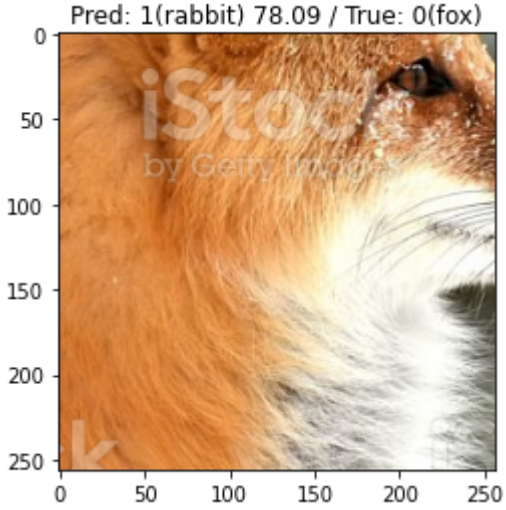
```
Category: fox
 True predicted images/total fox category: 19 / 20
Predicted probability:
[0.9927403  0.00725973]
```




```
Predicted probability:
[9.9986690e-01 1.3311153e-04]
```




```
 False predicted images/total fox category: 1 / 20
Predicted probability:
[0.21905164 0.78094834]
```




```
Category: rabbit
 True predicted images/total rabbit category: 19 / 19
Predicted probability:
[0.06400003 0.936     ]
```