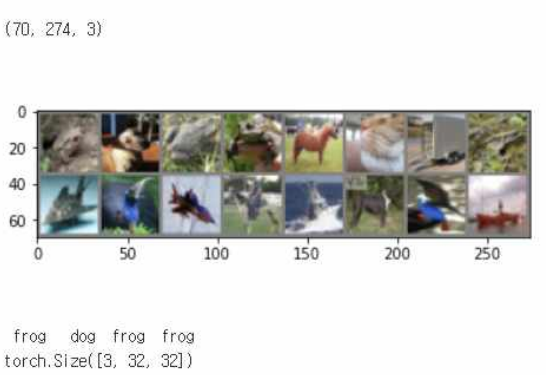
기계학습의 기초 및 전기정보 응용 Assignment03 보고서

전기정보공학부 2017-13758 강정민

Problem 0

Problem 2에서의 batch_size 변경(4 -> 16)으로 이미지 출력 크기에 변동이 생김



Problem 1

```
################################################################################
#                          IMPLEMENT YOUR CODE                                 #
################################################################################
self.conv1 = nn.Sequential(
    nn.Conv2d(in_channels=3, out_channels=8, kernel_size=(7,7), stride=(1,1)),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=(2,2), stride=(2,2))
)
self.conv2 = nn.Sequential(
    nn.Conv2d(in_channels=8, out_channels=16, kernel_size=(4,4), stride=(1,1)),
    nn.ReLU(),
    nn.MaxPool2d(kernel_size=(2,2), stride=(2,2))
)
self.fc1 = nn.Sequential(
    nn.Linear(in_features=400, out_features=100),
    nn.ReLU()
)
self.fc2 = nn.Sequential(
    nn.Linear(in_features=100, out_features=80),
    nn.ReLU()
)
self.fc3 = nn.Sequential(
    nn.Linear(in_features=80, out_features=10)
)
################################################################################
#                          END OF YOUR CODE                                    #
################################################################################
```

```
def forward(self, x):
    ################################################
    #                          IMPLEMENT YOUR CODE
    ################################################
    x = self.conv1(x)
    x = self.conv2(x)
    x = x.view(x.size(0), -1)
    x = self.fc1(x)
    x = self.fc2(x)
    x = self.fc3(x)
    ################################################
    #                          END OF YOUR CODE
    ################################################
    return x
```

문제의 조건에 맞게 CNN 구조를 위와 같이 만들었고 epoch 2에 대해서 수행한 결과는 위와 같다.

아래는 Loss Function과 Optimizer를 정의한 것이다.

```
################################################################################
#                          IMPLEMENT YOUR CODE                                 #
################################################################################
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(net.parameters(), lr=0.001)
################################################################################
#                          END OF YOUR CODE                                    #
################################################################################
```

이를 학습한 후 수행한 결과는 아래와 같다. 좋은 성능을 보이지는 않는다.

```
[1,  2000] loss: 2.302
[2,  2000] loss: 2.294
Finished Training
Saved Trained Model
```

```
Accuracy of the network on the 10000 test images: 18 %
```

Problem 2

아래의 코드는 GoogLeNet의 구조를 간략화해서 Inception 모듈의 수를 줄이고 파라미터 수를 줄여서 간략화한 구조이다.

```
################################################################################
#                          IMPLEMENT YOUR CODE                                 #
################################################################################
self.layer_1 = nn.Sequential(
    nn.Conv2d(3, 192, 3, 1),
    nn.BatchNorm2d(192),
    nn.ReLU(True),
)
self.layer_2 = nn.Sequential(
    Inception(192,  64,  96, 128, 16, 32, 32),
    Inception(256, 192,  96, 208, 16,  48,  64),
    nn.MaxPool2d(3,2,1),
)
self.layer_3 = nn.Sequential(
    Inception(512, 256, 160, 320, 32, 128, 128),
    nn.MaxPool2d(3,2,1),
)
self.layer_4 = nn.Sequential(
    Inception(832, 384, 192, 384, 48, 128, 128),
    nn.AvgPool2d(8,1),
)
self.layer_5 = nn.Dropout2d(0.4)
self.fc_layer = nn.Linear(1024,10)
################################################################################
#                          END OF YOUR CODE                                    #
################################################################################
```

```
def forward(self, x):
    ##################################################
    #                          IMPLEMENT YOUR CODE
    ##################################################
    out = self.layer_1(x)
    out = self.layer_2(out)
    out = self.layer_3(out)
    out = self.layer_4(out)
    out = self.layer_5(out)
    out = out.view(out.size(0),-1)
    out = self.fc_layer(out)
    ##################################################
    #                          END OF YOUR CODE
    ##################################################
    return out
```

그리고 Loss Function은 위의 모델과 동일하게 분류를 하는 과제를 수행하기 때문에 CrossEntropy를 사용했고 Optimizer는 Adam, learning rate는 0.001을 사용했다. 이는 아래의 코드를 통해 확인할 수 있다.

```
################################################################################
#                           IMPLEMENT YOUR CODE                                #
################################################################################
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(betternet.parameters(), lr=0.001)
################################################################################
#                           END OF YOUR CODE                                   #
################################################################################
```

이 모델의 성능은 아래와 같이 약 75%가 나오게 된다.

```
[1,  2000] loss: 1.353
[2,  2000] loss: 0.842
Finished Training
Saved Trained Model
Accuracy of the network on the 10000 test images: 75 %
```

모델의 정보를 확인하기 위해서는 torchsummary 라이브러리를 이용하거나 print()를 이용할 수 있다.

Tell us here

Model's idea is from GoogLeNet Architecture and for fast learning process, number of Inception modules is reduced so that size of parameters is reduced

```
# print shape of all layers' output and size of parameters

import torchsummary
torchsummary.summary(betternet, (3, 32, 32))
```

```
# print information of all layers

print(betternet)
```