Assignment #2

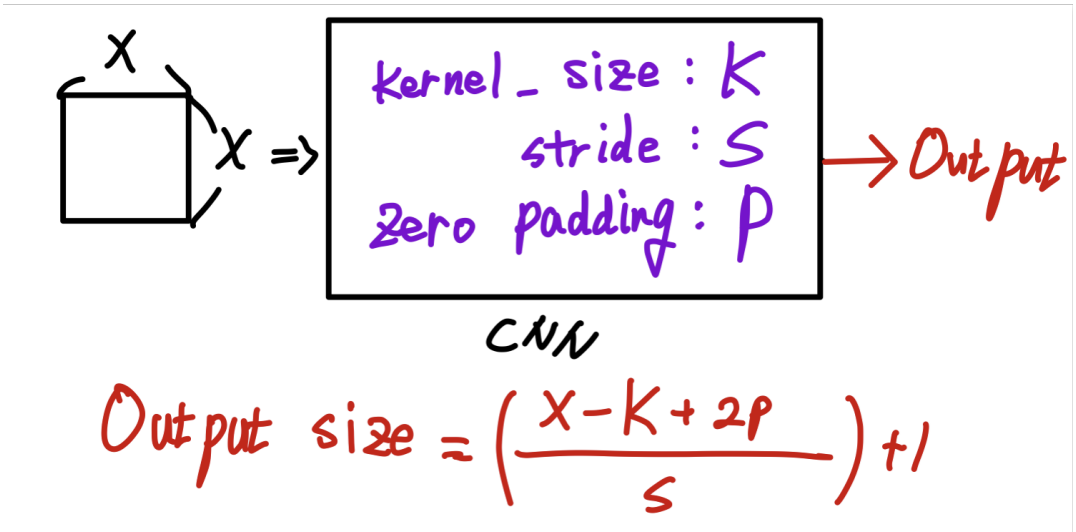
KW-VIP 2018707017 정용훈

●개요

CIFAR10 데이터셋을 활용해 CNN모델을 사용한다.

또한, 설계한 여러 모델들을 비교하면서 어느 모델이 정확도면에서 좋은 결과를 나타내는지 분석한다.

●구현 방법



CNN에 (X by X) 사진이 입력으로 들어가면 커널에 의해 계산된 Output 사이즈는

Output size $= \frac{X-Kernel\ size+2*Padding}{stride} + 1$이다. 이때, Zero Padding은 Output size가 너무 작아지는 것을 방지하는 역할을 한다.

CIFAR10 Data는 32 x 32 이미지이다.

우선, 32 by 32를 Resize()하는 방법은 고려하지 않았다. 왜냐하면 resize()한다는 것은 Interpolation을 거친다는 의미인데 Interpolation을 거친다고 해도 원본의 픽셀 값이 훨씬 정확하다고 생각했다. 따라서 별도의 resize를 하지 않고 이미지 그대로 학습에 사용하는 것을 선택했다.

크기가 작은 것을 고려해서 Conv2D를 거쳐서는 크기가 일정하도록 설계하고, 사이즈를 조절하는 부분은 Maxpooling2D에서 한다.

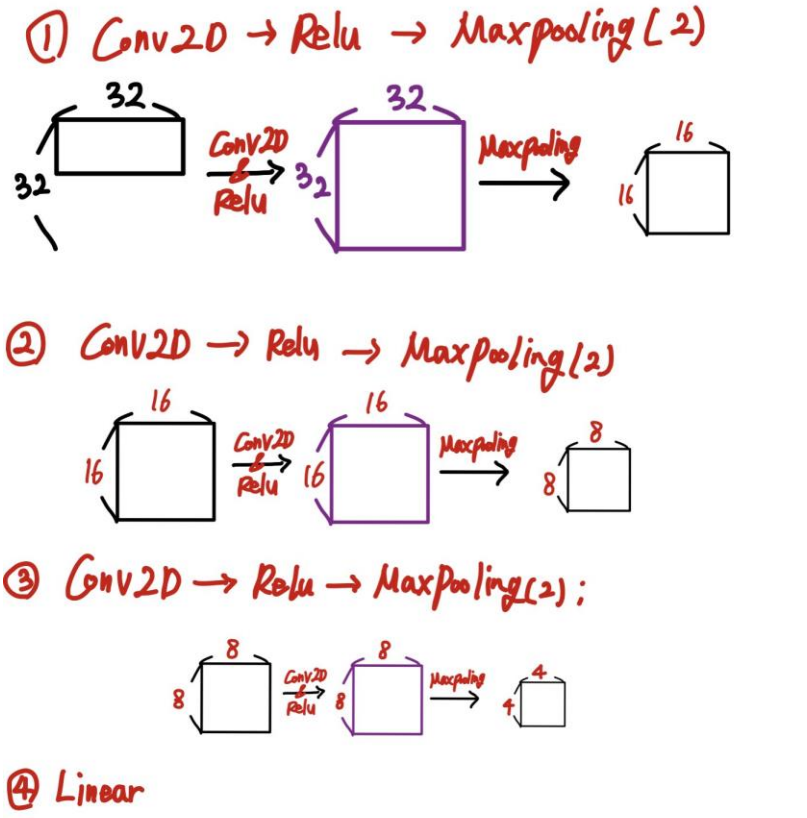
If stride = 1로 설정할 경우;

Output size $= \frac{X-Kernel\ size\ +\ 2*Padding}{1} + 1 = \mathrm{X} - \mathrm{Kernel\ size} + 2 * \mathrm{Padding} + 1$ 이 된다.

따라서 Kernel size = 2 * Padding + 1일 경우 Output size는 Input Size와 일정하다는 것을 확인할 수 있다.

결과적으로 Kernel size는 너무 작지 않으면서 Padding은 너무 크지 않는 것을 고려하여 Kernel size = 5, Padding = 2로 일관된 Conv2D를 설계했다. 이후는 Accuracy를 기준으로 실험적으로 찾은 값이다.

<모델 설계>



3개의 Convolution Layer,를 거친 후 4 x 4 feature map이 나오도록 설계했다.

Linear의 Input은 4 x 4 x T (T : feature map의 수)로 하면 될 것이다.

<모델 구현>

```
class ConvNet(nn.Module):
    def __init__(self, num_classes=10):
        super(ConvNet, self).__init__()
        self.layer1=nn.Sequential(
            nn.Conv2d(3,32,kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.layer2= nn.Sequential(
            nn.Conv2d(32,64,kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.layer3 = nn.Sequential(
            nn.Conv2d(64, 128, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))

        self.fc_layer = nn.Sequential(
            nn.Linear(128*4*4, 120),
            nn.Linear(120, num_classes))

    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out=self.layer3(out)
        out = F.dropout(out, training=self.training)
        out=out.reshape(out.size(0),-1)
        out=self.fc_layer(out)
        return out
```

위에서 설계한 방법대로 코드를 짰다.

특히, Image가 32 x 32로 작은 사이즈이므로 Layer를 깊게 쌓는 방법보다 얕게 쌓되, 하나의 Conv2d에 필터의 개수를 크게 잡았다.

이는 과적합(Overfitting)을 피하기 위해서였다.

●결과 화면

<초기 결과>

```
Accuracy of plane : 82 %
Accuracy of   car : 86 %
Accuracy of  bird : 56 %
Accuracy of   cat : 50 %
Accuracy of  deer : 74 %
Accuracy of   dog : 64 %
Accuracy of  frog : 66 %
Accuracy of horse : 84 %
Accuracy of  ship : 82 %
Accuracy of truck : 85 %

Process finished with exit code 0
```

<최종 결과>

```
Test Accuracy of the model on the 10000 test images : 76.58 %
Accuracy of plane : 91 %
Accuracy of   car : 92 %
Accuracy of  bird : 70 %
Accuracy of   cat : 57 %
Accuracy of  deer : 58 %
Accuracy of   dog : 59 %
Accuracy of  frog : 76 %
Accuracy of horse : 82 %
Accuracy of  ship : 86 %
Accuracy of truck : 78 %

Process finished with exit code 0
```