Cnn 과제

YBIGTA 20기 김소정

1. Convolutional Neural Networks(이하 CNN)에 대한 설명으로 옳지 않은 것은?

1번 - Convolution 연산이란, 이미지 위에서 stride 값만큼 filter(kernel)를 이동시키면서 겹쳐지는 부분의 각 원소의 값을 곱해서 모두 더한 값을 출력으로 하는 연산이다.

2번 - CNN은 Filter와 이미지의 Convolution으로 이미지의 Feature를 추출해내는 모델이다.

3. CNN은 parameter를 공유하여 전체 parameter 수를 줄여주기 때문에 overfitting이 일어날 가능성이 DNN보다 더 높다.

정답: 3번

2. CNN 모델을 구축하는 과정에서 다음과 같은 코드를 이용하여 필터(커널)를 만들어주었다.

```
conv = torch.nn.Conv2d(1,1,3)
```

다음에 대해 맞으면 True 틀리면 False를 선택하시오.

"이 필터는 입력채널의 크기가 1, 출력채널의 크기가 1, 필터의 크기가 3*3인 필터이다."

정답: TRUE

3. 다음과 같이 conv 의 이름으로 convolution layer 필터를 만들어 주고 inputs를 넣어 주었다.

```
conv = torch.nn.Conv2d(1,1,3)
inputs = ( A , B , C , D )
output = conv(inputs)
```

A, B, C, D 순서대로 쓰세요. 채널, Width, Height, 배치사이즈

정답: 배치사이즈, 채널, Height, Width

4. 채널이 8인 63x63 input 이미지와 7x7의 16채널 필터를 "stride=1"로 convolution 연산을 하되, input과 같은 크기의 output 결과를 가져오도록 하려고 한다. 이 때, 얼마의 padding을 주어야 하는가?

정답: 3

5. 다음 용어들에 대한 간단한 정의 혹은 설명을 쓰시오

Convolution 연산: 신호를 커널을 이용해 국소적으로 증폭 또는 감소시켜, 정보를 추출 또는 필터링하는 것을 의미

Padding: convolution 후 아웃풋 이미지의 크기 유지와 모서리의 픽셀 정보를 효과적으로 이용하기 위해 사용하는 방법으로, 지정된 픽셀만큼 외곽에 특정한 값을 채워 넣음

Channel: 흑백은 2차원 데이터로서 1개의 채널로 구성되지만, 컬러 이미지는 각각의 픽셀이 RGB로 표현되기 때문에 3차원 데이터이며 3개의 채널로 구성됨.

Stride: 필터가 이동하는 단위로, stride가 2면 필터가 2칸씩 이동함

Filter: 필터는 커널이라고도 부르기도 하며, 특정 이미지의 특징을 찾아내기 위한 변수로 정사각형 행렬이다.

Pooling: 출력 데이터를 입력으로 받아, 데이터의 크기를 줄이거나 특정 데이터를 강조하는 용도로 사용됨. 대표적으로 max pooling, average pooling이 있음.

6. Conv 연산을 한 후 학습을 위해서는 nn.Linear()을 거쳐 1차원 벡터로 변경해야 한다.

정답: TRUE

Tensorflow 과제

```
[32] Epoch 34/50
25/25 [==============================] - 18s 719ms/step - loss: 0.4681 - accuracy: 0.8257 - val_loss: 0.6895 - val_accuracy: 0.7454
Epoch 35/50
25/25 [==============================] - 18s 716ms/step - loss: 0.4609 - accuracy: 0.8176 - val_loss: 0.5917 - val_accuracy: 0.7852
Epoch 36/50
25/25 [==============================] - 18s 712ms/step - loss: 0.4590 - accuracy: 0.8276 - val_loss: 0.6605 - val_accuracy: 0.7713
Epoch 37/50
25/25 [==============================] - 18s 714ms/step - loss: 0.4635 - accuracy: 0.8308 - val_loss: 0.6141 - val_accuracy: 0.7861
Epoch 38/50
25/25 [==============================] - 18s 710ms/step - loss: 0.4228 - accuracy: 0.8389 - val_loss: 0.6911 - val_accuracy: 0.7611
Epoch 39/50
25/25 [==============================] - 18s 710ms/step - loss: 0.4232 - accuracy: 0.8430 - val_loss: 0.6537 - val_accuracy: 0.7769
Epoch 40/50
25/25 [==============================] - 18s 699ms/step - loss: 0.4285 - accuracy: 0.8382 - val_loss: 0.6872 - val_accuracy: 0.7657
Epoch 41/50
25/25 [==============================] - 17s 696ms/step - loss: 0.4058 - accuracy: 0.8485 - val_loss: 0.7193 - val_accuracy: 0.7593
Epoch 42/50
25/25 [==============================] - 18s 698ms/step - loss: 0.4048 - accuracy: 0.8498 - val_loss: 0.6511 - val_accuracy: 0.7648
Epoch 43/50
25/25 [==============================] - 18s 716ms/step - loss: 0.4147 - accuracy: 0.8479 - val_loss: 0.6202 - val_accuracy: 0.7704
Epoch 44/50
25/25 [==============================] - 18s 697ms/step - loss: 0.3990 - accuracy: 0.8459 - val_loss: 0.6692 - val_accuracy: 0.7806
Epoch 45/50
25/25 [==============================] - 18s 725ms/step - loss: 0.4036 - accuracy: 0.8456 - val_loss: 0.6372 - val_accuracy: 0.7648
Epoch 46/50
25/25 [==============================] - 18s 713ms/step - loss: 0.3755 - accuracy: 0.8607 - val_loss: 0.6567 - val_accuracy: 0.7787
Epoch 47/50
25/25 [==============================] - 18s 700ms/step - loss: 0.3616 - accuracy: 0.8659 - val_loss: 0.5810 - val_accuracy: 0.7870
Epoch 48/50
25/25 [==============================] - 18s 725ms/step - loss: 0.3646 - accuracy: 0.8625 - val_loss: 0.6678 - val_accuracy: 0.7602
Epoch 49/50
25/25 [==============================] - 18s 705ms/step - loss: 0.3634 - accuracy: 0.8607 - val_loss: 0.6853 - val_accuracy: 0.7676
Epoch 50/50
25/25 [==============================] - 18s 707ms/step - loss: 0.3570 - accuracy: 0.8697 - val_loss: 0.6271 - val_accuracy: 0.7824
```

✓ 1초 오후 4:08에 완료됨