CNN 과제

YBIGTA 20기 임현정

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 줄여 주기 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 False 를 선택하시오.

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

(True)

3. 다음과 같이 conv 의 이름으로 convolution layer 필터를 만들어 주고 inputs 를 넣어주었다.

```
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 : Batch size, Channel, Height, Width

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

O = floor[(I-K+2P)/S + 1] = (63-7+2P)/1 + 1 = 63, 56+2P = 62, 2P = 6, P = 3

정답 : 3

5. 다음 용어들에 대한 간단한 정의 혹은 설명을 쓰시오

Convolution 연산: 하나의 필터를 입력 이미지 값에 전체적으로 훑어주는 과정을 말한다. 커널이 이동하며 기존 이미지 matrix와의 내적을 통해 output matrix를 계산하게 되며 그 결과로 Feature map이 생성된다.

Padding: 이미지 주위에 추가로 경계를 덧대어 이미지의 크기를 키우고 윤곽정보를 더욱 활용할 수 있도록 하는 기법을 말한다. 경계처리방법을 "VALID'로 지정하면 유효한 영역만 출력되며, "SAME"으로 지정하면 입력 이미지와 같은 크기로 출력된다.

Channel: gray scale 이미지는 1 Channel, RGB color 이미지는 3 Channel로 구성된다. 입력 데이터가 여러 Channel을 갖는 경우 Filter가 각 Channel을 순회하며 Convolution을 계산해 Channel별로 결과물을 생성한다. 입력 데이터는 Channel 수와는 상관없이 Filter 별로 1개의 Feature map을 생성하게 된다. 3개의 Kernel로 해석하지 않고, 3개의 Channel을 가진 1개의 Kernel로 해석한다.

Stride: 커널의 이동 범위(간격)를 말한다. 사용자에 의해 지정된 간격인 Stride 만큼 전체 입력 데이터와 Filter의 Convolution연산이 진행되며 Feature map을 생성하게 된다.

Filter: 이미지의 특징을 찾아내기 위한 파라미터로 구성되며 이는 CNN에서의 학습 대상이다. Filter는 입력 데이터를 stride만큼씩 이동하며 Convolution 연산을 하게 된다. Filter size는 일반적으로 홀수이다. (중심위치 존재 & padding의 비대칭성 방지를 위해) 또한, 하나의 Convolution layer에 크기가 같은 여러 개의 Filter를 적용할 수 있으며, 이 경우 Feature map에는 Filter 개수만큼의 Channel이 만들어진다. 즉, 입력 데이터에 적용한 Filter의 개수는 출력 데이터인 Feature map의 Channel이 된다고 볼 수 있다.

Poolling: Downsampling (Subsampling)의 일종으로, 출력의 전체 크기를 줄여주는 역할을 한다. 대표적인 방법으로는 일정 크기의 구간 내에서 가장 큰 값만 전달하고 다른 정보는 버리는 Max Pooling과 일정 크기의 구간 내의 값들의 평균을 전달하는 Average Pooling이 있다.

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

(True)

Tensorflow 과제

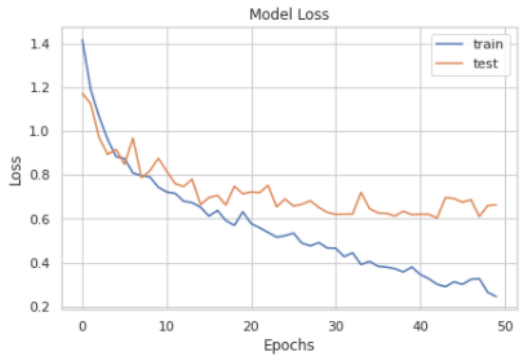
(1) CNN model

```
model = Sequential()
model.add(Conv2D(filters = 32, kernel_size = (3,3), padding = 'same',activation ='relu', input_shape = (150,150,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters = 64, kernel_size = (3,3), padding = 'same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters =96, kernel_size = (3,3), padding = 'same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters = 96, kernel_size = (3,3), padding = 'same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dense(5, activation = "softmax"))
```

(2) Result

```
Epoch 42/50
25/25 [==============================] - 18s 720ms/step - loss: 0.3265 - accuracy: 0.8758 - val_loss: 0.6204 - val_accuracy: 0.8102
Epoch 43/50
25/25 [==============================] - 18s 719ms/step - loss: 0.3005 - accuracy: 0.8910 - val_loss: 0.6033 - val_accuracy: 0.8167
Epoch 44/50
25/25 [==============================] - 18s 722ms/step - loss: 0.2897 - accuracy: 0.8916 - val_loss: 0.6965 - val_accuracy: 0.8037
Epoch 45/50
25/25 [==============================] - 18s 720ms/step - loss: 0.3118 - accuracy: 0.8829 - val_loss: 0.6927 - val_accuracy: 0.7889
Epoch 46/50
25/25 [==============================] - 18s 713ms/step - loss: 0.3001 - accuracy: 0.8865 - val_loss: 0.6751 - val_accuracy: 0.8056
Epoch 47/50
25/25 [==============================] - 18s 734ms/step - loss: 0.3229 - accuracy: 0.8763 - val_loss: 0.6871 - val_accuracy: 0.7898
Epoch 48/50
25/25 [==============================] - 18s 716ms/step - loss: 0.3263 - accuracy: 0.8778 - val_loss: 0.6105 - val_accuracy: 0.8157
Epoch 49/50
25/25 [==============================] - 18s 722ms/step - loss: 0.2639 - accuracy: 0.9003 - val_loss: 0.6599 - val_accuracy: 0.8102
Epoch 50/50
25/25 [==============================] - 19s 742ms/step - loss: 0.2442 - accuracy: 0.9081 - val_loss: 0.6635 - val_accuracy: 0.8102
```

```
plt.plot(History.history['loss'])
plt.plot(History.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'test'])
plt.show()
```




```
plt.plot(History.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')S
plt.legend(['train', 'test'])
plt.show()
```