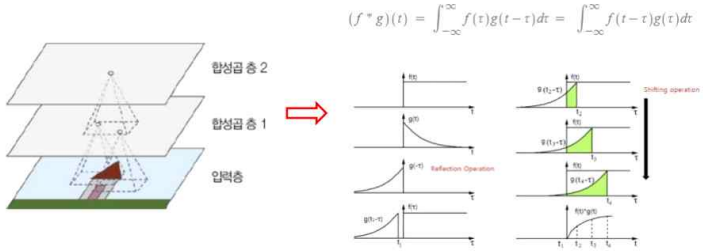
합성곱 신경망을 사용한 컴퓨터 비전 정리노트

4조-(이희구, 유제우)

※ 4조 의견

Q. 합성곱 층에 대해서 설명해줘



CNN의 가장 중요한 구성 요소는 합성곱 층이다. 첫 번째 합성곱 층의 뉴런은 입력 이미지의 모든 픽셀에 연결되는 것이 아니라 합성곱 층 뉴런의 수용장 안에 있는 픽셀에만 연결한다. 두 번째 합성곱 층에 있는 각 뉴런은 첫 번째 층의 작은 사각 영역 안에 위치한 뉴런에 연결한다. 여기서 스트라이드는 한 수용장과 다음 수용장 사이의 간격을 의미한다.

Q. 필터 (합성곱 커널)이 뭐야?

```
# 2개의 필터를 만듭니다.
filters = np.zeros(shape=(7, 7, channels, 2), dtype=np.float32)
filters[:, 3, :, 0] = 1  # 수직선
filters[3, :, :, 1] = 1  # 수평선
```

입력 뉴런에 사용될 가중치 역할을 수행한다. 필터의 모양과 크기가 국부수용장의 모양과 크기를 지정한다. 필터는 코드 상에서 넘파이 어레이로 지정된다. 다양한 필터를 사용하고 필터 수는 하이퍼파라미터로 지정한다.

Q. 특성맵? 컬러채널? 이게 정확히 뭐야?

특성맵은 필터 각각을 사용하여 생성된 출력값이다. 수십, 수백 개의 필터를 사용한다. 각 특성맵의 픽셀이 하나의 뉴런에 해당한다. 필터에 포함된 모든 뉴런은 동일한 가중치와 편향을 사용한다. 필터마다 사용되는 가중치와 편향은 다르다.

컬러채널은 이미지를 대상으로 하는 합성곱 층은 3차원으로 표현이 가능하다. 입력 이미지가 컬러인 경우 R, F, B 세 개의 채널, 흑백인 경우 하나의 채널을 사용한다.

Q. 뉴런의 출력값?

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1}\sum_{v=0}^{f_w-1}\sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \times w_{u,v,k',k} \quad \text{여기서} \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases}$$

각 뉴런의 출력값은 입력에 대한 가중치의 합에 편향을 더한 값이다.

Q. 텐서플로로 구현하는 코드 알려줘

```
import numpy as np
from sklearn.datasets import load_sample_image

# 샘플 이미지를 로드합니다.
china = load_sample_image("china.jpg") / 255
flower = load_sample_image("flower.jpg") / 255
images = np.array([china, flower])
batch_size, height, width, channels = images.shape

# 2개의 필터를 만듭니다.
filters = np.zeros(shape=(7, 7, channels, 2), dtype=np.float32)
filters[:, 3, :, 0] = 1  # 수직선
filters[3, :, :, 1] = 1  # 수평선

outputs = tf.nn.conv2d(images, filters, strides=1, padding="SAME")

plt.imshow(outputs[0, :, :, 1], cmap="gray") # 첫 번째 이미지의 두 번째 특성맵을 그립니다.
plt.axis("off") # 책에는 없습니다.
plt.show()
```

Q. 풀링 층이란게 뭐야?

풀링 층은 계산량과 메모리 사용량을 줄이면서 과대적합의 위험도를 줄여주는 용도로 사용된다. 풀링 층 뉴런은 가중치가 없다. 보폭을 사용하여 차원을 축소시키는 기능을 수행한다. 최대 풀링층은 2x2 크기의 풀링 커널이다. 네 개의 셀에 속한 값들 중에서 가장 큰 값만 상위 층으로 전달된다. 보폭은 2이다. 하위 입력층에서 두 칸씩 건너뛰며 풀링 커널을 적용하여, 상위 층의 뉴런 수가 1/4 로 줄어든다. 패딩은 없다. 보폭에 따라 일부 행과 열이 무시될 수 있다.

Q. 최대 풀링 층의 특징에는 어떤 것들이 있을까?

파라미터 수를 획기적으로 줄여 계산량, 메모리 사용량을 줄여준다. 많은 정보를 잃게 되지만 그래도 잘 작동한다. 작은 변화에 대한 어느 정도의 불변성이 보장된다. 시멘틱 분할의 경우에는 불변성 대신에 등변성이 중요하기도 한다.

Q. 텐서플로로 구현할 때 풀링 층은 어떤 식으로 구성돼?

평균 풀링 층은 풀링커널 구역 내의 평균값을 활용한다. MaxPool2D 대신에 AvgPool2D를 사용한다. 평균 풀링 층은 최대값을 유지하여 보다 강한 특성이 학습에 사용되기 때문에 최대 풀링 층에 비해 성능이 떨어진다.

전역 평균 풀링 층은 각 특성맵의 평균을 계산한다. 샘플의 각 특성맵에 대해 하나의 숫자를 출력하고 출력층에서 유용하게 활용될 수 있다. 현대 신경망 구조에서 종종 활용된다.

깊이별 최대/평균 풀링층은 각각 특성맵에 대해 공간(너비와 높이)별로 최대/평균을 계산하는 것 대신에 지정된 수만큼의 특성맵을 대상으로 최대/평균을 계산하는 풀링층이다. 특성맵 수가 줄어들고 풀링커널 크기를 1로 지정하여 특성맵의 크기를 유지하는 것이 기본이다.

Q. CNN의 구조에 대해 설명해줘

전형적인 CNN 구조는 네트워크를 통과하여 진행할수록 이미지는 점점 작아지지만, 합성곱 층 때문에 일반적으로 점점 더 깊어진다. 즉, 더 많은 특성 맵을 가지게 된다. CNN에서 이미지 인식 문제에서 완전 연결 층의 심층 신경망을 사용하지 않는 이유는 파라미터가 너무 많아지고, CNN은 층을 부분적으로 연결하고 가중치를 공유하기 때문이다. 합성곱 층에 사용하는 커널 크기는 작은 커널이 파라미터와 계산량이 적고 일반적으로 더 나은 성능을 낸다.

Q. LeNet-5?

가장 널리 알려진 CNN 구조이다. 1998년 얀르쿤이 만들었고, 손글씨 숫자 인식(MNIST) 에 사용된다.

Q. LeNet-5의 구조를 알려줘

층	종류	특성 맵	크기	커널 크기	스트라이드	활성화 함수
출력	완전 연결	-	10	-	-	RBF
F6	완전 연결	-	84	-	-	tanh
C5	합성곱	120	1×1	5×5	1	tanh
S4	평균 풀링	16	5×5	2×2	2	tanh
C3	합성곱	16	10×10	5×5	1	tanh
S2	평균 풀링	6	14×14	2×2	2	tanh
C1	합성곱	6	28×28	5×5	1	tanh
입력	입력	1	32×32	-	-	-

출력층이 각 뉴런에서 입력 벡터와 가중치 벡터 사이의 유클리드 거리를 출력한다. 출력은 이미지가 얼마나 특성 숫자 클래스에 속하는지 측정한다.

Q. AlexNet은 뭐야?

2012년 이미지넷 대회에서 우승 (알렉스 크리체프스키, 제프리 힌턴), 톱-5 에러율: 17%, 2위는 26%, 이 구조는 LeNet-5와 비슷하지만 크고 깊음. 처음으로 합성곱 층 위에 풀링 층을 쌓지 않고 바로 합성곱 층끼리 쌓았다.

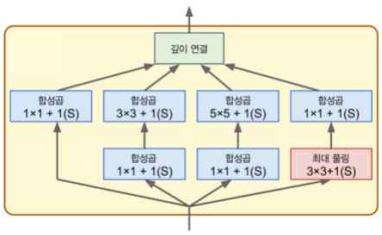
Q. 특징에는 뭐가 있어?

- 과대 적합을 줄이기 위해 두 가지 규제 기법을 사용했다.
- 규제 1 : 드롭아웃을 50% 비율로 적용. (F9와 F10의 출력)
- 규제 2 : 데이터 증식 수행
- 데이터증식은 훈련샘플을 인공적으로 생성하는 기법.
- 수평 뒤집기, 간격 이동, 조명 변경 등 활용
- 정규화: LRN(local response normalization)
- 뉴런의 출력값을 보다 경쟁적으로 만드는 정규화 기법적인 정규화 단계 사용
- 어떤 특성맵에 속한 하나의 뉴런의 활성화 함수의 반환값이 클 경우 주변 특성맵의 동일한 위치에 뉴런의 활성화 함수값을 크게 만들어줌.
- 각각의 특성맵을 보다 특별하게 만들어서 보다 다양한 특성을 탐색할 수 있도록 도와줌.

Q. GoogLeNet?

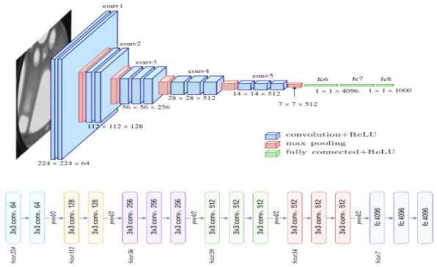
2014년 ILSVRC 이미지넷 대회에서 우승, 톱-5 에러율: 7% 이하로 낮춤 -> 이전 CNN보다 훨씬 더 깊기 때문이다. 인셉션 모듈이라는 서브 네트워크를 사용했다. 이전의 구조보다 훨씬 효과적으로 파마미터를 사용했다. GoogLeNet은 AlexNet 보다 10배나 적은 파라미터를 가진다. (6천만개 -> 6백만개)

Q. 인셉션 모듈이 뭐야?



- 3 x 3 + 1(S): 3x3 커널, 보폭 1,"same" 패딩
- 합성곱 층: ReLU 활성화 함수 사용
- 둘째 층: 다양한 패턴을 다양한 스케일로 파악하기 위한 용도
- 1x1 커널 사용 층
- 깊이별 패턴을 확인하며, 다른 합성곱 층과 연계하는 역할 수행.
- 깊이 연결
- 4개의 합성곱 층의 결과를 쌓은 후 출력
- tf.concat() 함수 활용

Q. VGGNet?



ILSVRC 2014년 대회 2등, 톱-5 에러율: 8~10%, 합성곱 계층과 풀링 계층으로 구성되는 기본적인 CNN, 합성곱 계층, 완전연결 계층을 모두 16층(혹은 19층)으로 심화하였다. 특징은 많은 개수의 필터를 사용하지만, 3x3 필터만을 사용했다.

Q. ResNet?

2015년 ILSVRC 이미지넷 대회에서 우승, 톱-5 에러율: 3.6% 이하, 잔차 네트워크(Residual Network)를 사용했다. 152개 층으로 구성된 극도로 깊은 CNN 사용. 파라미터 수는 그렇게 많지 않다. 더 적은 파라미터를 사용해 점점 더 깊은 네트워크로 모델을 구성했다.

Q. 잔차유닛이 뭐야?

많은 층으로 인한 많은 계산을 줄이기 위해 잔차 유닛(RU) 활용했다. RU를 활용한 잔차학습(residual learning) 효과이다. 스킵 연결로 인한 보다 수월한 학습이 가능하다.

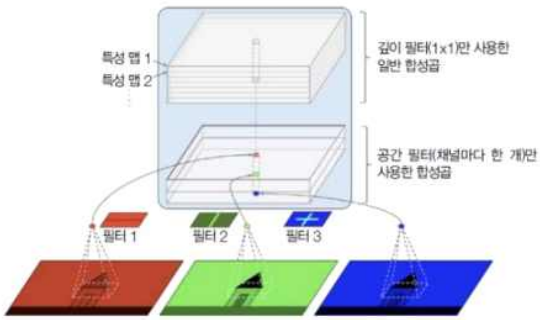
Q. ResNet 구조

각 잔차 유닛은 배치 정규화(BN)와 ReLU, 3x3 커널을 사용한다. 특성맵의 수는 몇 개의 잔차 유닛마다 두 배로 늘어난다. 반면에 뉴런 수를 반씩 줄인다. (3x3 + 2 사용, 즉, 폭이 2임.)
입력과 출력의 모양을 맞추기 위해 빨강색 점선 스킵 부분에 폭이 2인 합성곱을 활용한다.
(ResNet-34, ResNet-152)

Q. Xception

2016년에 소개된 GoogLeNet 과 ResNet 모델의 합성 버전이다. 톱-5 에러율: 3% 정도이다. GoogLeNet의 인셉션 모듈 대신 깊이별 분리 합성곱 층을 사용한다.

Q. 분리 합성곱 층이 뭐야?



공간별 패턴인식 합성곱 층과 깊이별 패턴인식 합성곱 층을 분리하여 연속적으로 적용한다. 공간별 패턴인식: 형태 인식 (입력 특성지도마다 한개만 탐색) 깊이별 패턴인식: 입, 코, 눈으로부터 얼굴을 인식하듯 채널 사이의 패턴을 인식한다.

Q. Xception의 구조는?

입력층에 많은 채널(특성지도)가 존재할 경우에만 활용한다. 보통 두 개 정도의 정상적인 합성곱 층으로 시작한 후에 깊이별 분리합성곱 층을 적용한다. 보다 적은 수의 파라미터, 보다 적은 양의 메모리, 보다 적은 양의 계산 요구되지만, 성능은 더 좋다.

Q. SENet?

2017년 ILSVRC 이미지넷 대회에서 우승, 톱-5 에러율: 2.25%, GoogLeNet의 인셉션 모듈과 ResNet의 잔차유닛(RU)에 SE block을 추가하여 보다 좋은 성능 발휘

Q. SE block 기능 알려줘

입력된 특성맵을 대상으로 깊이별 패턴 특성을 분석, 패턴 특성들을 파악한 후 출력값 보정
예제로 코와 입의 패턴이 보일 때 눈의 특성지도를 강화시킨다.

Q. SE block 구조는?

전역평균 풀링층 -> 밀집층 -> 밀집층
첫째 밀집 층: 뉴런 수를 16분의 1로 줄임(squeeze)
특성맵들 사이의 연관성 학습 강요
둘째 밀집 층: 뉴런 수를 정상화시킴
학습된 연관성을 이용하여 입력 특성지도를 보정할 가중치 출력

※ 각 모델들을 어느 상황에 어떠한 모델을 사용하는 것이 좋을지 조사하여 분석했습니다.

LeNet-5:
사용 상황: 간단한 이미지 분류나 손글씨 숫자(MNIST)와 같은 간단한 작업에 적합합니다.
주의 사항: 현대의 복잡하고 대규모 데이터셋에는 부적합할 수 있습니다.

AlexNet:
사용 상황: 대규모 이미지 데이터셋에서 객체 인식 및 분류에 적합합니다.
주의 사항: 모델이 깊고 많은 파라미터를 가지므로, 작은 데이터셋에서는 과적합의 위험이 있습니다.

GoogLeNet:
사용 상황: 객체 인식 및 분류, 복잡한 시각적 패턴을 다루는데 적합합니다.
주의 사항: 인셉션 모듈을 사용하여 파라미터를 효과적으로 활용하지만, 메모리 사용량이 많을 수 있습니다.

VGGNet:
사용 상황: 간단하면서도 깊은 구조가 필요한 경우, 대규모 데이터셋에서의 이미지 분류에 적합합니다.
주의 사항: 파라미터가 많아 메모리 사용량이 높을 수 있습니다.

ResNet:
사용 상황: 매우 깊은 네트워크가 필요한 경우, 그라디언트 소실 문제를 해결하고자 할 때 사용됩니다.
주의 사항: 더 간단한 작업에는 지나치게 깊은 구조가 필요하지 않을 수 있습니다.

Xception:
사용 상황: 파라미터를 효율적으로 사용하고자 할 때, 깊이별 분리 합성곱 층의 특성을 활용하고자 할 때 적합합니다.
주의 사항: 복잡한 작업에는 적합하지만, 작은 데이터셋에서는 과적합의 위험이 있을 수 있습니다.

SENet:
사용 상황: 각 특성맵의 중요도를 고려하고자 할 때, 모델의 성능을 높이고자 할 때 적합합니다.
주의 사항: 추가적인 계산이 필요하므로 모델의 크기와 속도에 영향을 줄 수 있습니다.