

8. 딥러닝

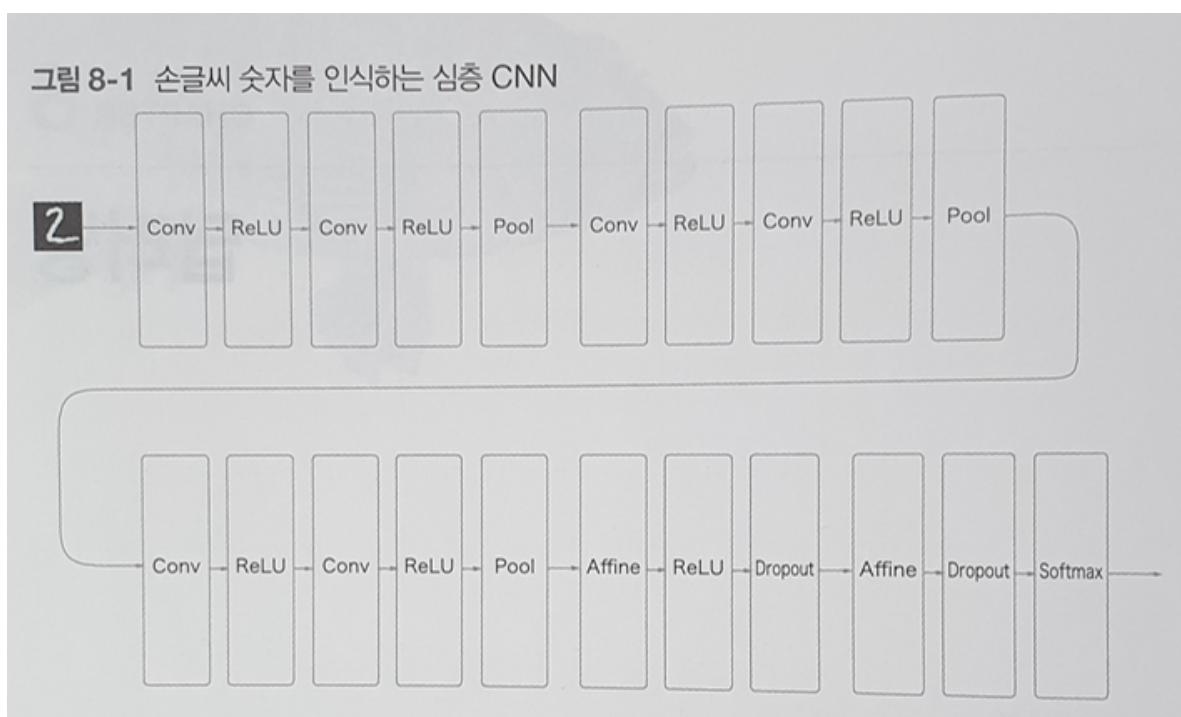
딥러닝은 층을 깊게 한 심층 신경망이다. 심층 신경망은 지금까지 설명한 신경망을 바탕으로 뒷단에 층을 추가하기만 하면 만들 수 있지만, 커다란 문제가 몇 개 있다. 이번 장에서는 딥러닝의 특징과 과제 그리고 가능성을 살펴보겠다.

8.1 더 깊게

신경망에 관해 그동안 많은 것을 배웠다. 신경망을 구성하는 다양한 계층과 학습에 효과적인 기술, 영상 분야 특히 유효한 CNN과 매개변수 최적화 기법 등이 떠오를 것이다. 이 모드가 딥러닝에서 중요한 기술이다. 이번 절에서는 그동안 배운 기술을 집약하고 심층 신경망을 만들어 MNIST 데이터셋의 손글씨 숫자 인식에 도전하려 한다.

8.1.1 더 깊은 신경망으로

이번 절에서는 다음 그림과 같이 구성된 CNN을 만들고자 한다(이 신경망은 다음 절에서 설명하는 VGG 신경망을 참고했다).



딱 봐도 지금까지 구현한 신경망보다 층이 깊다. 여기에서 사용하는 합성곱 계층은 모두 3×3 크기의 작은 필터로, 층이 깊어지면서 채널 수가 더 늘어나는 것이 특징이다(합성곱 계층의 채널 수는 앞 계층에서부터 순서대로 16, 16, 32, 32, 64, 64로 늘어간다). 또 그림과 같이 풀링 계층을 추가하여 중간 데이터의 공간 크기를 점차 줄여나간다. 그리고 마지막 단의 완전연결 계층에서는 드롭아웃 계층을 사용한다.

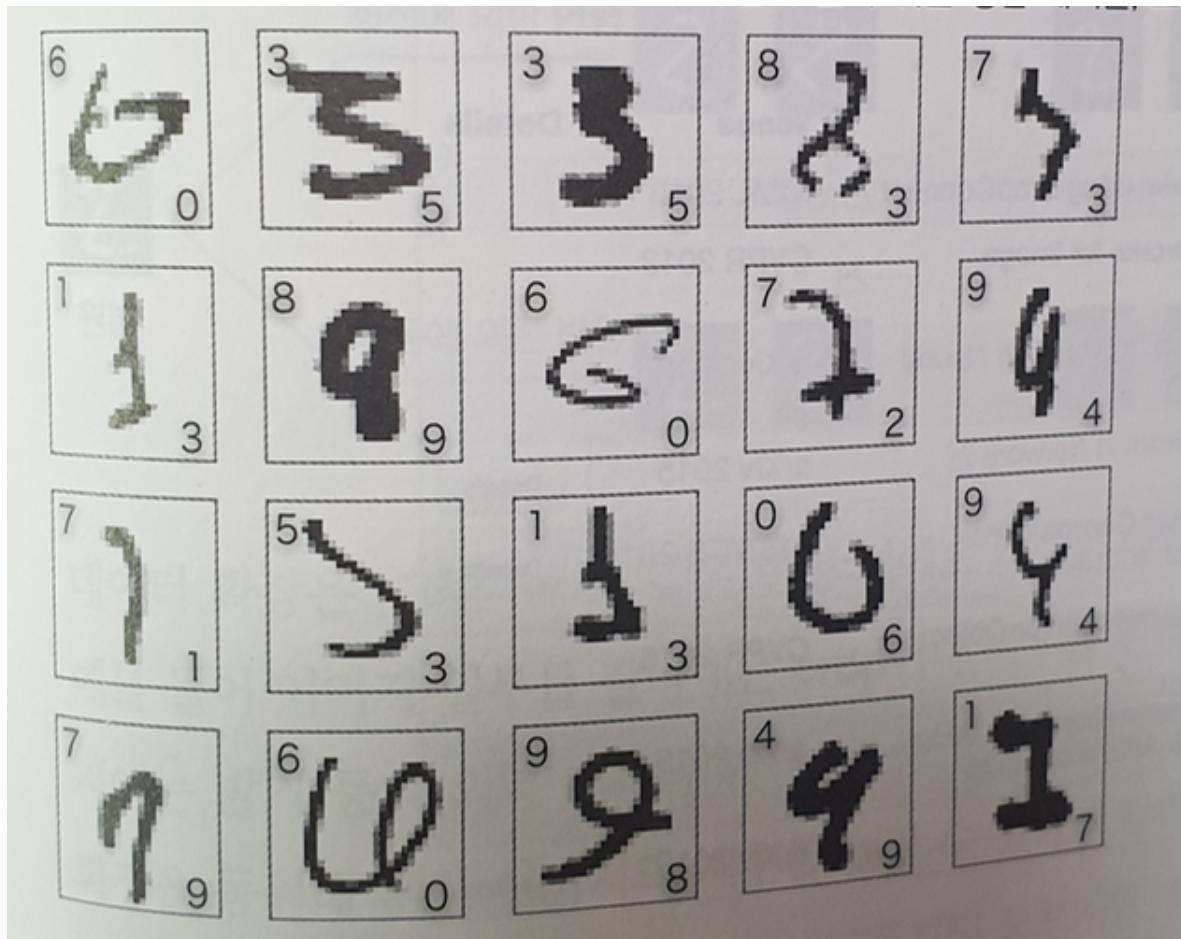
가중치 초기값으로 He 초기값을 사용하고, 가중치 매개변수 갱신에는 Adam을 이용한다.

이상을 정리하면 이 신경망의 특징은 다음과 같다.

- 3×3 의 작은 필터를 사용한 합성곱 계층
- 활성화 함수는 ReLU
- 완전연결 계층 뒤에 드롭아웃 계층 사용
- Adam을 사용해 최적화
- 가중치 초기값은 'He의 초기값'

이상의 특징에서 보듯 이 신경망에는 그동안 배운 신경망 기술이 잔뜩있다. 그럼 이 신경망을 학습시켜보자. 결과부터 말하면 이 신경망의 정확도는 99.38%이다.

이 신경망이 잘못 인식할 확률은 0.62%이다.



위 그림은 신경망이 잘못 인식한 것들이다. 위 사진들은 인간들도 판단하기 어려운 것들이다.

이번의 심층 CNN은 정확도가 높고, 잘못 인식한 이미지들도 인간과 비슷한 인식 오류를 저지르고 있다. 이런 점에서도 심층 CNN의 잠재력이 얼마나 크다는 것을 느낄 수 있을 것이다.

8.1.2 정확도를 더 높이려면

<What is the class of this image?> 웹사이트는 다양한 데이터셋을 대상으로 발표한 정확도를 정리하고 있다.

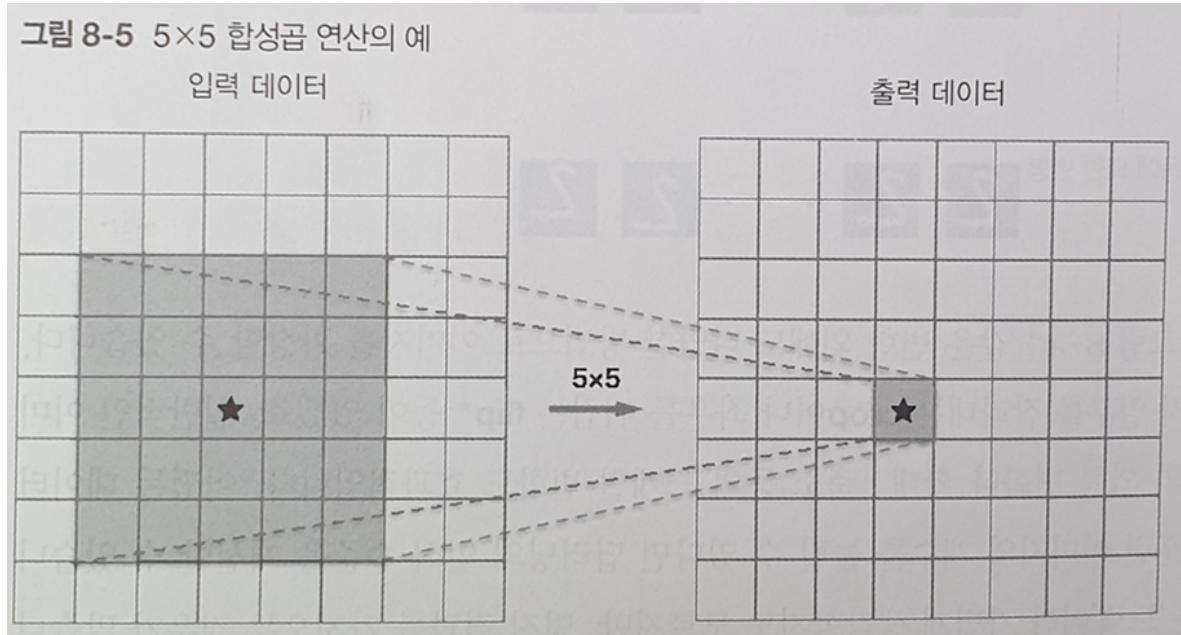
Note MNIST 데이터셋에 대해서는 층을 아주 깊게 하지 않고도 (현시점에서는) 최고 수준의 결과가 나온다. 이는 손글씨 숫자라는 문제가 비교적 단순해서 신경망의 표현력을 극한까지 높일 필요가 없기 때문이다. 이에 층을 깊게 해도 혜택이 적다고 할 수 있다. 반면, 나중에 소개하는 대규모 일반 사물 인식에서는 문제가 훨씬 복잡해지므로 층을 깊게 하면 정확도를 크게 끌어올릴 수 있다.

데이터 개수를 더 많이 하면 정확도가 올라간다.

8.1.3 층을 깊게 하는 이유

층을 깊게 할 때의 이점에 대해 논하겠다. 그것은 신경망의 매개변수 수가 줄어든다는 것이다. 층을 깊게 한 신경망은 깊지 않은 경우보다 적은 매개변수로 같은 (혹은 그 이상) 수준의 표현력을 달성할 수 있다. 합성곱 연산에서의 필터 크기에 주목해 생각해보면 쉽게 이해될 것이다. 예를 하나 보자.

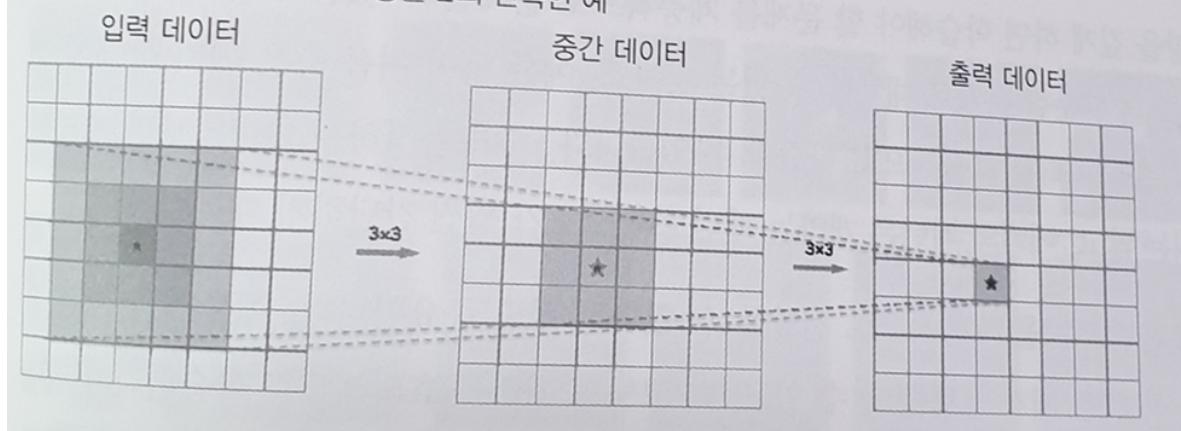
그림 8-5 5×5 합성곱 연산의 예



여기서 주목할 점은 출력 데이터의 각 노드가 입력 데이터의 어느 영역으로부터 계산되었느냐는 것이다. 당연하지만 위 그림의 예에서는 각각의 출력 노드는 입력 데이터의 5×5 크기 영역에서 계산된다.

이어서 다음 그림처럼 3×3 의 합성곱 연산을 2회 반복하는 경우를 생각해보자. 이 경우 출력 노드 하나는 중간 데이터의 3×3 영역에서 계산된다. 그럼 중간 데이터의 3×3 영역은 그 전 입력 데이터의 어느 영역에서 계산될까? 다음 그림을 잘 보면 5×5 크기의 영역에서 계산되어 나오는 것을 알 수 있다. 즉, 다음 그림의 출력 데이터는 입력 데이터의 5×5 영역을 '보고' 계산한다.

그림 8-6 3×3 의 합성곱 계층을 2회 반복한 예



5×5 의 합성곱 연산 1회는 3×3 의 합성곱 연산을 2회 수행하여 대체할 수 있다. 게다가 전자의 매개변수 수가 25개(5×5)인 반면, 후자는 총 18개 ($2 \times 3 \times 3$)이며, 매개변수 수는 층을 반복할수록 적어진다. 그리고 그 개수의 차이는 층이 깊어질수록 커진다. 예를 들어 3×3 의 합성곱 연산을 3회 반복하면 매개변수는 모두 27개가 되지만, 같은 크기의 영역을 1회의 합성곱 연산으로 '보기' 위해서는 7×7 크기의 필터, 즉 매개변수 49개가 필요하다.

Note 작은 필터를 겹쳐 신경망을 깊게 할 때의 장점은 매개변수 수를 줄여 넓은 '수용 영역'을 소화할 수 있다는 데 있다. (수용 영역은 뉴런에 변화를 일으키는 국소적인 공간 영역이다.) 게다가 층을 거듭하면서 ReLU 등의 활성화 함수를 합성곱 계층 사이에 끼움으로써 신경망의 표현력이 개선된다. 이는 활성화 함수가 신경망에 '비선형' 힘을 가지고, 비선형 함수가 겹치면서 더 복잡한 것도 표현할 수 있게 되기 때문이다.

학습의 효율성도 층을 깊게 하는 것의 이점이다. 층을 깊게 함으로써 학습 데이터의 양을 줄여 학습을 고속으로 수행할 수 있다는 뜻이다. 신경망을 깊게 하면 학습해야 할 문제를 계층적으로 분해할 수 있다. 각 층이 학습해야 할 문제를 더 단순한 문제로 대체할 수 있다는 것이다. 예를 들어, 처음 층은 에지 학습에 전념하여 적은 학습 데이터로 효율적으로 학습할 수 있다. 개가 등장하는 이미지보다 에지를 포함한 이미

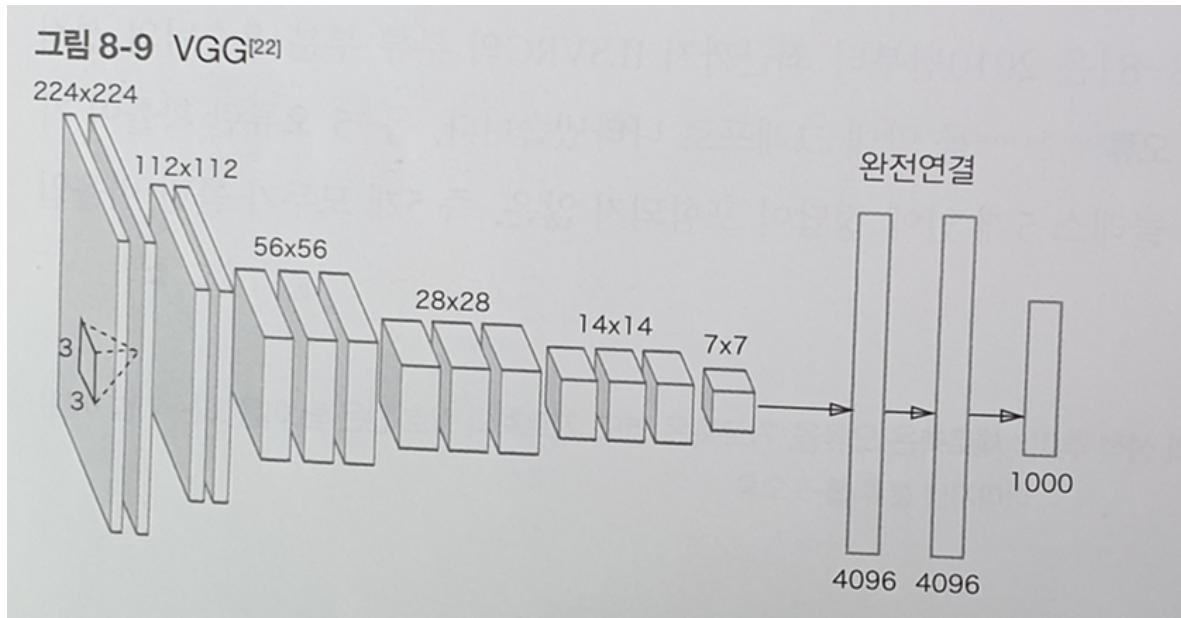
지는 않고, 에지의 패턴은 개라는 패턴 보다 구조가 훨씬 더 간단하기 때문이다.

또, 층을 깊게 하면 정보를 계층적으로 전달할 수 있다는 점도 중요하다. 예를 들어, 에지를 추출한 층의 다음 층은 에지 정보를 쓸 수 있고, 더 고도의 패턴을 효과적으로 학습할 하라 기대할 수 있다. 즉, 층을 깊이 함으로써 각 층이 학습해야 할 문제를 '풀기 쉬운 단순한 문제'로 분해할 수 있어 효율적으로 학습하리라 기대할 수 있다.

8.2 딥러닝의 초기 역사

8.2.2 VGG

VGG는 합성곱 계층과 풀링 계층으로 구성된 '기본적'인 CNN이다.

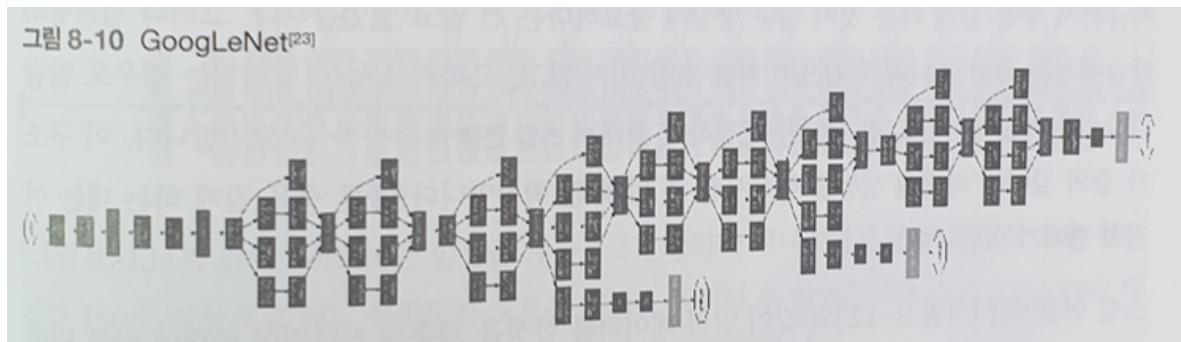


다만 위 그림과 같이 비중 있는 층(합성곱 계층, 완전연결 계층)을 모두 16층(혹은 19층)으로 심화한 게 특징이다(층의 깊이에 따라서 'VGG16', 'VGG19'로 구분하기도 한다).

VGG에서 주목할 점은 3x3의 작은 필터를 사용한 합성곱 계층을 연속으로 거친다는 것이다. 그림에서 보듯 합성곱 계층을 2~4회 연속으로 풀링 계층을 두어 크기를 절반으로 줄이는 처리를 반복한다. 그리고 마지막에는 완전연결 계층을 통과시켜 결과를 출력한다.

8.2.3 GoogLeNet

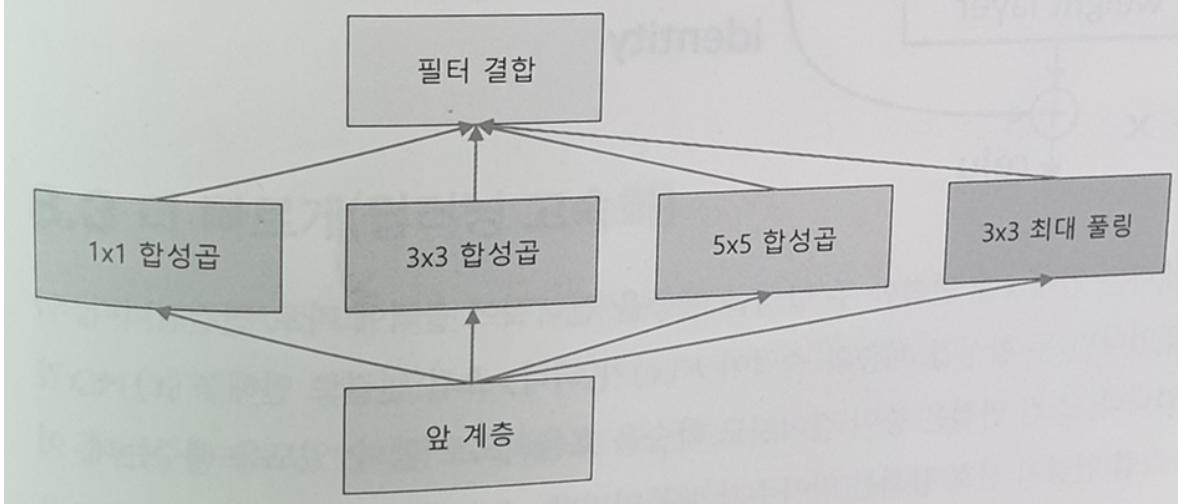
GoogLeNet의 구성은 아래와 같다. 그림의 사각형이 합성곱 계층과 풀링 계층 등의 계층을 나타낸다.



그림을 보면 구성이 매우 복잡해 보이는데, 기본적으로는 지금까지 보아온 CNN과 크게 다르지 않다. 단, GoogLeNet은 세로 방향 깊이 뿐 아니라 가로 방향도 깊다는 점이 특징이다.

GoogLeNet에는 가로 방향에 '폭'이 있다. 이를 인셉션 구조라 하며, 그 기반 구조는 아래 그림과 같다.

그림 8-11 GoogLeNet의 인셉션 구조^[23]



인셉션 구조는 위와 같이 크기가 다른 필터(와 풀링)을 여러 개 적용하여 그 결과를 결합한다. 이 인셉션 구조를 하나의 빌딩 블록 (구성요소)으로 사용하는 것이 GoogLeNet의 특징인 것이다. 또, GoogLeNet에서는 1x1 크기의 필터를 사용한 합성곱 계층을 많은 곳에서 사용한다. 이 1x1의 합성곱 연산은 채널쪽으로 크기를 줄이는 것으로, 매개변수 제거의 고속 처리에 기여한다.

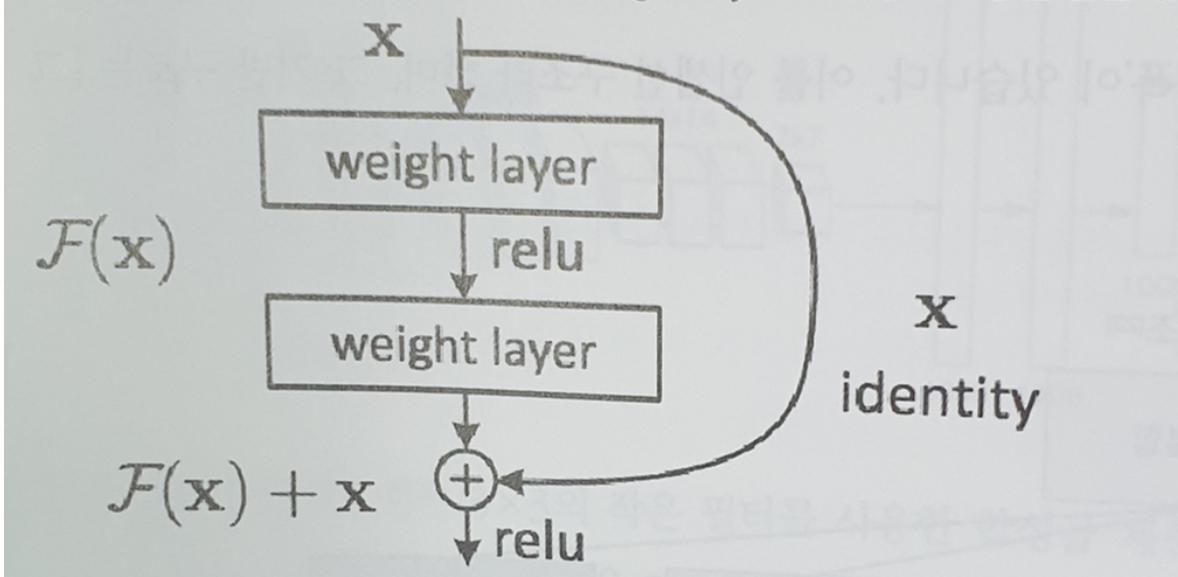
8.2.4 ResNet

ResNet은 마이크로소프트 팀이 개발한 네트워크다. 그 특징은 지금까지 보다 층을 깊게 할 수 있는 특별한 장치에 있다.

지금까지 층을 깊게 하는 것이 성능 향상에 중요하다는 것은 알고 있었다. 그러나 딥러닝의 학습에서는 층이 지나치게 깊으면 학습이 잘 되지 않고, 오히려 성능이 떨어지는 경우도 많다. ResNet은 그런 문제를 해결하기 위해 **스킵 연결**을 도입한다. 이 구조가 층의 깊이에 비례해 성능을 향상시킬 수 있게 한 핵심이다(물론 층을 깊게 하는 데는 여전히 한계가 있다).

스킵 연결이란 아래 그림과 같이 입력 데이터를 합성곱 계층을 건너뛰어 출력에 바로 더하는 구조를 말한다.

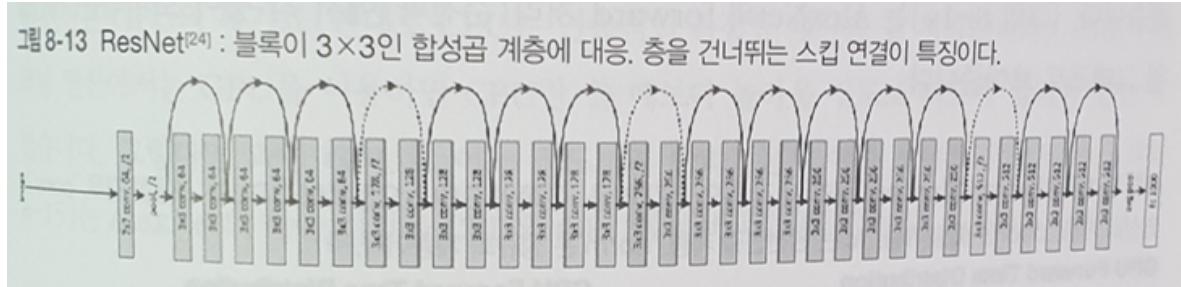
그림 8-12 ResNet의 구성요소^[24] : 'weight layer'는 합성곱 계층을 말한다.



위 그림에서는 입력 x 를 연속한 두 합성곱 계층을 건너뛰어 출력에 바로 연결한다. 이 단축 경로가 없었다면 두 합성곱 계층의 출력이 $F(x) + x$ 가 되는 게 핵심이다. 스kip 연결은 층이 깊어져도 학습을 효율적으로 할 수 있도록 해주는데, 이는 역전파 때 스kip 연결이 신호 감쇠를 막아주기 때문이다.

Note 스kip 연결은 입력 데이터를 '그대로' 흘리는 것으로, 역전파 때도 상류의 기울기를 그대로 하류로 보낸다. 여기에서의 핵심은 상류의 기울기에 아무런 수정도 가하지 않고 '그대로' 흘린다는 것이다. 그래서 스kip 연결로 기울기가 작아지거나 지나치게 커질 걱정 없이 앞 층에' 의미 있는 기울기가 전해지리라 기대할 수 있다. 층을 길게 할수록 기울기가 작아지는 소실 문제를 이 skip 연결이 줄여주는 것이다.

ResNet은 먼저 설명한 VGG 신경망을 기반으로 skip 연결을 도입하여 층을 깊게 했다. 결과는 다음 처럼 된다.



위와 같이 ResNet은 합성곱 계층을 2개 층마다 건너뛰면서 층을 깊게 한다. 실험 결과 150층 이상으로 해도 정확도가 오르는 모습을 확인할 수 있었다. 그리고 ILSVRC 대회에서는 톱-5 오류율이 겨우 3.5%라는 경이적인 결과를 냈다.

Note 이미지넷이 제공하는 거대한 데이터셋으로 학습한 가중치 값들은 실제 제품에 활용해도 효과적이고, 또 많이들 그렇게 이용하고 있다. 이를 전이 학습이라고 해서, 학습된 가중치(혹은 그 일부)를 다른 신경망에 복사한 다음, 그 상태로 재학습을 수행한다. 예를 들어 VGG와 구성이 같은 신경망을 준비하고, 미리 학습된 가중치를 초기값으로 설정한 후, 새로운 데이터셋을 대상으로 재학습(fine tuning)을 수행한다. 전이 학습은 보유한 데이터셋이 적을 때 특히 유용한 방법이다.

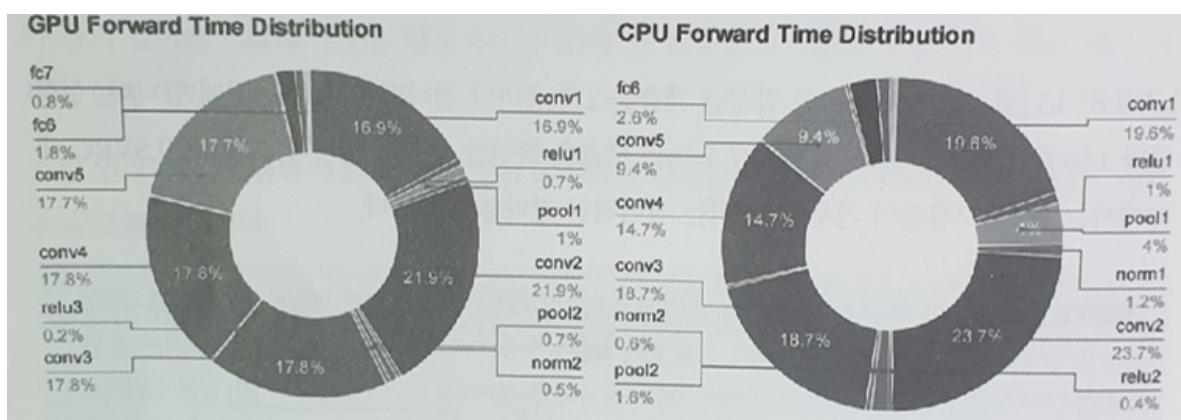
8.3 더 빠르게(딥러닝 고속화)

빅데이터와 네트워크의 발전으로 딥러닝에서는 대량의 연산을 수행해야 한다. 과거에는 주로 CPU가 계산을 담당했으나, CPU만으로 딥러닝을 처리하기엔 부족한 게 현실이다. 실제로 주위를 둘러보면 딥러닝 프레임워크 대부분은 GPU(Graphics Processing Unit)를 활용해 대량의 연산을 고속으로 처리할 수 있다. 최근 프레임워크에서는 학습을 복수의 GPU와 여러 기기로 분산 수행하기 시작했다.

이번 절에서는 딥러닝의 고속화에 관해 논해보고자 한다.

8.3.1 풀어야 할 숙제

딥러닝의 고속화 얘기를 시작하기 앞서, 딥러닝에서는 어떤 처리에 시간이 소요되는지를 보겠다. 다음 그림은 AlexNet의 forward 처리(순전파)에서 각 층이 소비하는 시간을 원 그래프로 보여준다.



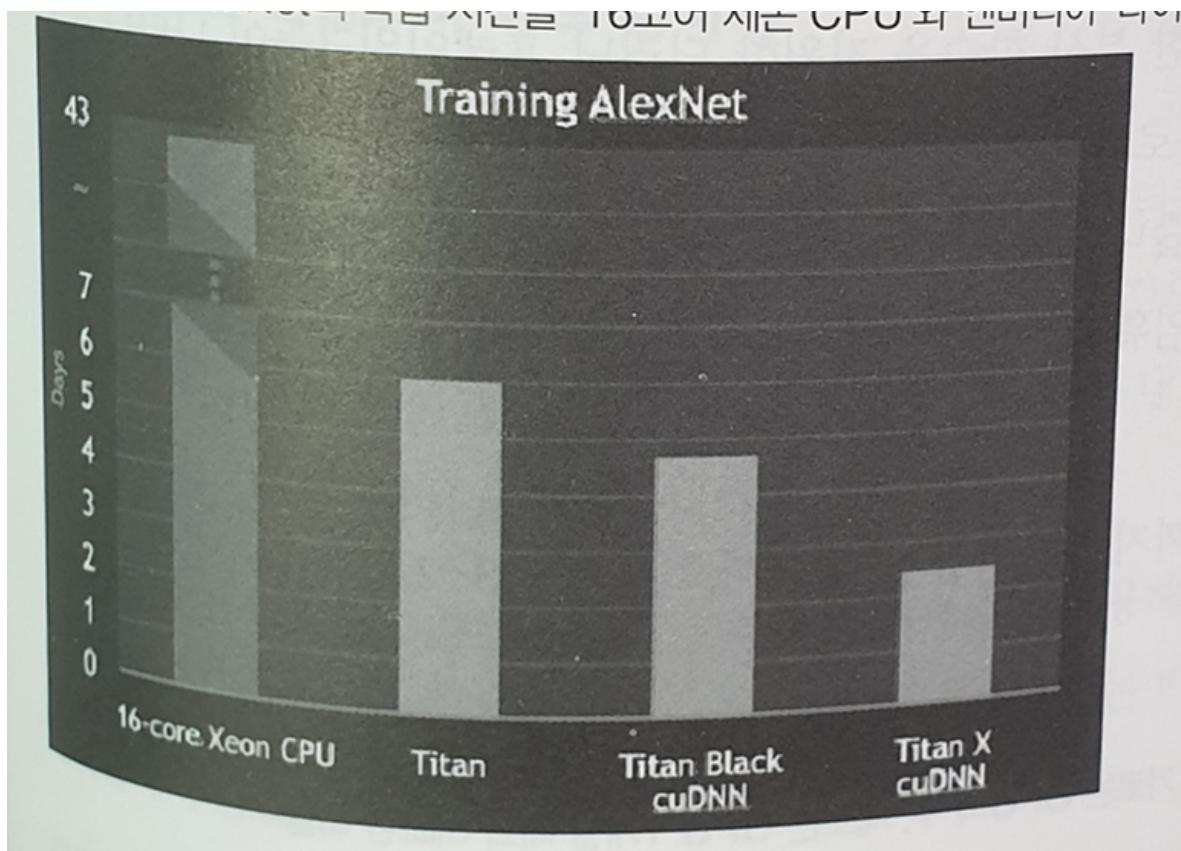
그림에서 보듯 AlexNet에서는 오랜 시간을 합성곱 계층에서 소요한다. 실제로 합성곱 계층의 처리 시간을 다 더하면 GPU에서는 전체의 95%, CPU에서는 전체의 89%까지 달한다. 그래서 합성곱 계층에서 이뤄지는 연산을 어떻게 고속으로 효율적으로 하느냐가 딥러닝의 과제이다. 위 그림은 추론 때의 결과지만, 학습 시에도 마찬가지로 합성곱 계층에서 많은 시간을 소비한다.

Note 합성곱 계층에서 수행하는 연산은 '7.2 합성곱 계층'에서 설명했듯, 결국 '단일 곱셈-누산'이다. 이에 딥러닝 고속화라는 주제는 대량의 '단일 곱셈-누산'을 어찌 고속으로 효율적으로 계산하느냐는 것이다.

8.3.2 GPU를 활용한 고속화

GPU는 원래 그래픽 전용 보드에 이용해왔다. 그러나 최근에는 그래픽 처리뿐 아니라 범용 수치 연산에도 이용한다. GPU는 병렬 수치 연산을 고속으로 처리할 수 있으니, 그 압도적인 힘을 다양한 용도로 활용하자는 것이 GPU 컴퓨팅의 목적이다. 이처럼 GPU로 범용 수치 연산을 수행하는 것을 GPU 컴퓨팅이라고 한다.

딥러닝에서는 대량의 단일 곱셈-누산(또는 큰 행렬의 곱)을 수행해야 한다. 이러한 대량 병렬 연산은 GPU의 특기이다(반대로 CPU는 연속적인 복잡한 계산을 잘 처리한다). 이에 딥러닝 연산에서는 GPU를 이용하면 CPU만 쓸 때보다 놀라울 정도로 빠르게 결과를 얻을 수 있다. 그렇다면 과연 GPU로 어느 정도 까지 빨라지는지 예를 보도록 하자. 아래 그림은 AlexNet의 학습 시간을 CPU와 GPU에서 비교한 결과이다.



위 그림과 같이 CPU에서는 40여일이나 걸리지만 GPU로는 6일까지 단축되었다. 또, cuDNN이라는 딥러닝에 최적화된 라이브러리를 이용하면 더욱 빨라짐을 확인할 수 있다.

딥러닝에 최적화 된것은 아직까진 엔비디아이다. 실제로 대부분의 딥러닝 프레임워크는 엔비디아 GPU에서만 혜택을 받을 수 있다. 엔비디아의 GPU 컴퓨팅용 통합 개발 환경인 CUDA를 사용하기 때문이다. 위 그림에서 등장하는 cuDNN은 CUDA 위에서 동작하는 라이브러리로, 딥러닝에 최적화된 함수 등이 구현되어 있다.

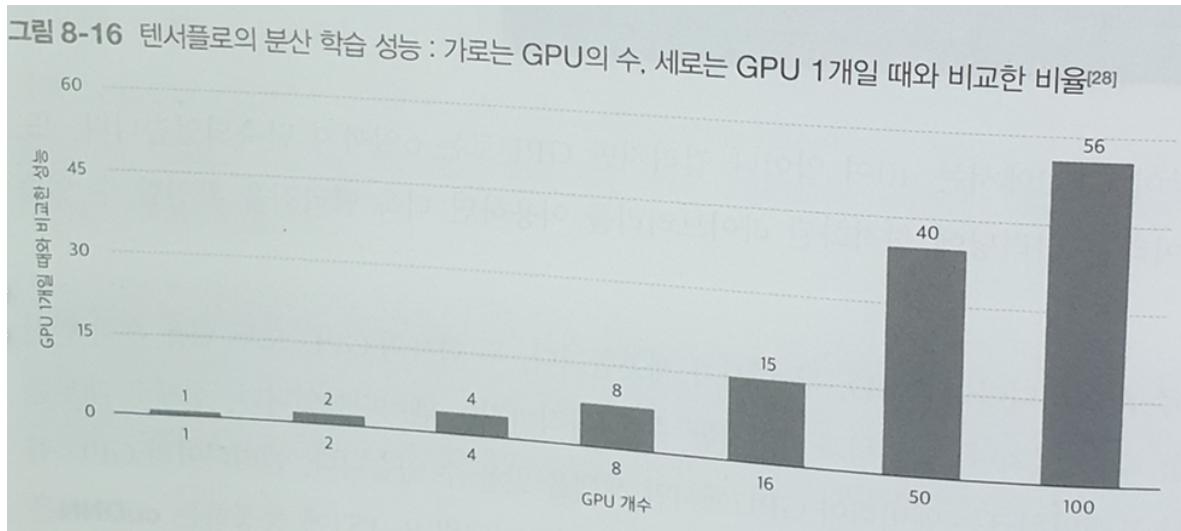
NOTE 합성곱 계층에서 행하는 연산은 *im2col*을 이용해 큰 행렬의 곱으로 변환할 수 있었다. 이러한 *im2col*의 방식은 GPU로 구현하기에도 적합하다. GPU는 '작은' 단위로 계산하기보다는 큰 덩어리를 한번에 계산하는 데 유리하기 때문이다. 즉, *im2col*로 거대한 행렬의 곱으로 한번에 계산하여 CPU의 잡재력을 끌어내는 것이다.

8.3.3 분산 학습

GPU로 딥러닝 연산을 꽤 가속할 수 있지만, 그대로 심층 신경망에서는 학습에 며칠 혹은 몇 주가 걸리기도 한다. 그리고 지금까지 살펴본 것처럼 딥러닝은 많은 시행착오를 동반한다. 뛰어난 신경망을 만들려면 시험을 수없이 반복해야 하고, 그러려면 1회 학습에 걸리는 시간을 최대한 단축하고 싶다는 요구가 필연적으로 생겨난다. 이에 딥러닝 학습을 수평 확장하자는 아이디어(즉, '분산 학습')가 중요해지는 것이다.

딥러닝 계산을 더욱 고속화하고자 다수의 GPU와 기기로 계산을 분산하기도 한다. 최근에는 다수의 GPU와 컴퓨터를 이용한 분산 학습을 지원한 딥러닝 프레임워크들이 나타나고 있다. 그중에서도 구글의 텐서플로와 마이크로소프트의 CNTK(Computational Network Toolkit)은 분산 학습에 역점을 두고 개발하고 있다. 거대한 데이터센터의 저지연, 고처리량 네트워크 위에서 이 프레임워크들이 수행하는 분산 학습은 놀라운 효과를 보이고 있다.

분산 학습까지 더하면 어느 수준까지 고속화할 수 있을까? 아래 그림은 텐서플로로 알아본 분산 학습의 효과이다.



위 그림에서 보듯 GPU 수가 늘어남에 따라 학습도 빨라진다. 실제로 여러 기기를 연결하여 GPU를 100개 까지 사용하니 하나일 때 보다 56배 빨라졌다. 7일 짜리 작업을 불과 3시간 만에 끝낸다는 거승로, 분산 학습의 놀라운 효과를 증명하고 있다.

분산 학습에서도 '계산을 어떻게 분산시키느냐'는 몹시 어려운 문제이다. 컴퓨터 사이의 통신과 데이터 동기화 등, 쉽게 해결할 수 없는 문제를 얼마든지 끌어안고 있다. 이에 이런 어려운 문제는 텐서플로와 같은 뛰어난 프레임워크에 맡기는 것이 좋다. 여기에서는 분산 학습의 상세 내용은 다루지 않겠다. 분산 학습 관련 기술적인 내용은 텐서플로 기술 논문을 참고해라.

8.3.4 연산 정밀도와 비트 줄이기

계산 능력 외에도 메모리 용량과 버스 대역폭 등이 딥러닝 고속화에 병목이 될 수 있다. 메모리 용량 면에서는 대량의 가중치 매개변수와 중간 데이터를 메모리에 저장해야 한다는 것을 생각해야 한다. 버스 대역폭 면에서는 GPU(혹은 CPU)의 버스를 흐르는 데이터가 많아져 한계를 넘어서면 병목이 된다. 이러한 경우를 고려하면 네트워크로 주고받는 데이터의 비트 수는 최소로 만드는 것이 바람직하다. 컴퓨터에서는 주로 64비트나 32비트 부동소수점 수를 사용해 실수를 표현한다. 많은 비트를 사용할수록 계산 오차는 줄어들지만, 그만큼 계산에 드는 비용과 메모리 사용량이 늘고 버스 대역폭에 부담을 준다.

다행히 딥러닝은 높은 수치 정밀도(수치를 몇 비트로 표현하느냐)를 요구하지 않는다. 이는 신경망의 중요한 성질 중 하나로, 신경망의 견고성에 따른 특성이다. 예를 들어 신경망은 입력 이미지에 노이즈가 조금 섞여 있어도 출력 결과가 잘 달라지지 않는 강건함을 보여준다. 이런 견고성 덕분에 신경망을 흐르는 데이터를 '퇴화'시켜도 출력에 주는 영향은 적다.

컴퓨터에서 실수를 표현하는 방식으로 **32비트 단정밀도**와 **64비트 배정밀도** 부동소수점 등의 포맷이 있지만, 지금까지의 실험으로는 딥러닝은 **16비트 반정밀도**만 사용해도 학습에 문제가 없다고 알려져 있다. 실제로 엔비디아의 2016년도 GPU인 **파스칼** 아키텍처는 이 포맷을 지원하여, 이제는 반정밀도 부동소수점이 표준적으로 이용되리라 생각한다.

Note 엔비디아의 맥스웰 세대 GPU는 반정밀도 부동소수점 수를 스토리지(데이터를 저장하는 기능)로 지원하고 있었지만, 연산 자체는 16비트로 수행하지 않았다.

이 책은 지금까지 딥러닝을 구현하며 수치 정밀도에는 특별히 주의하지 않았다. 그럼 몇 가지를 짚어보자. 우선 파이썬에서는 일반적으로 64비트 배정밀도 부동소수점 수를 사용한다. 하지만 넘파이는 16비트 반정밀도 부동소수점도 지원하며, 이를 사용해도 정확도가 떨어지지 않는다는 것을 쉽게 확인할 수 있다.

딥러닝의 비트 수를 줄이는 연구가 몇 가지 진행되고 있다. 최근에는 가중치와 중간 데이터를 1비트로 표현하는 **Binarized Neural Networks**라는 방법도 등장했다. 딥러닝을 고속화하기 위해 비트를 줄이는 기술은 앞으로 주시해야 할 분야이며, 특히 딥러닝을 임베디드용으로 이용할 때 중요한 주제이다.

8.4 딥러닝의 활용

지금까지 딥러닝을 활용한 예를 손글씨 숫자 인식이라는 이미지 분류를 중심으로 살펴봤다. 이는 '사물 인식'의 한 분야이다. 그러나 딥러닝은 사물 인식 뿐 아니라 온갖 문제에 적용할 수 있다. 이미지, 음성, 자연어 등 수많은 분야에서 딥러닝은 뛰어난 성능을 발휘한다. 이번 절에서는 딥러닝이 할 수 있는 것을 컴퓨터 비전 분야를 중심으로 몇 가지 소개하겠다.

8.4.1 사물 검출

사물 검출은 다음 그림과 같이 이미지 속에 담긴 사물의 위치와 종류(클래스)를 알아내는 기술이다.

그림 8-17 사물 검출의 예^[34]

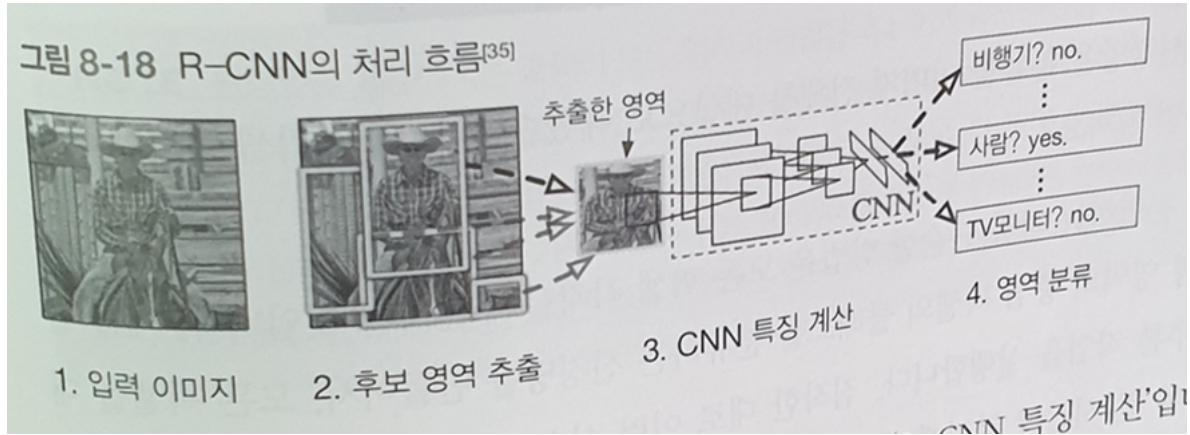


이 그림에서 보듯 사물 검출은 사물 인식보다 어려운 문제이다. 지금까지 본 사물 인식은 이미지 전체를 대상으로 했는데, 사물 검출에서는 이미지 어딘가에 있을 사물의 위치까지 알아내야 한다. 게다가 한 이미지에 여러 사물이 존재할 수도 있다.

이런 사물 검출 문제에 CNN을 기반으로 한 기법이 몇 가지 제안되었다. 이 기법들이 발군의 성능을 보여 사물 검출에도 딥러닝이 효과적임을 시사하고 있다.

자, CNN을 이용하여 사물 검출을 수행하는 방식은 몇 가지가 있는데, 그 중에서도 R-CNN이 유명하다. 아래 그림은 R-CNN의 처리 흐름이다.

그림 8-18 R-CNN의 처리 흐름^[35]



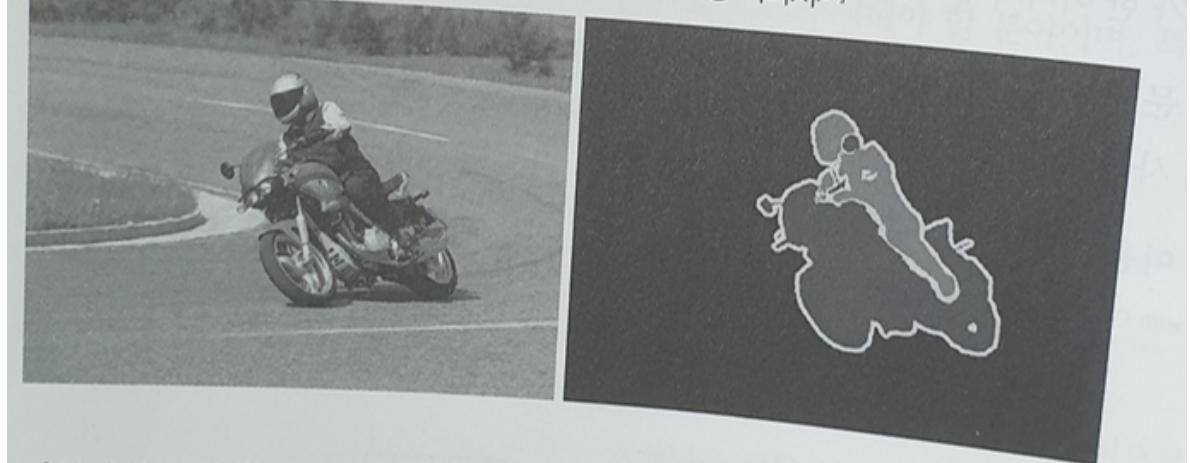
R-CNN 그림에서 주목할 곳은 '2. 후보 영역 추출'과 '3. CNN 특징 계산'이다. 먼저 사물이 위치한 영역을 (어떤 방법으로) 찾아내고, 추출한 각 영역에 CNN을 적용하여 클래스를 분류하는 것이다. 여기서 이미지를 사각형으로 변형하거나 분류할 때 서포트 벡터 머신을 사용하는 등 실제 처리 흐름은 다소 복잡하지만, 큰 틀에서는 이 두 가지 처리(후보 영역 추출과 CNN 특징 계산)로 구성된다.

후보 영역 추출(사물처럼 보이는 물체를 찾아 처리)에는 컴퓨터 비전 분야에서 발전해온 다양한 기법을 사용할 수 있고, R-CNN 논문에서는 Selective Search 기법을 사용했다. 최근에는 이 후보 영역 추출까지 처리하는 **Faster R-CNN** 기법도 등장했다. Faster R-CNN은 모든 일을 하나의 CNN에서 처리하기에 아주 빠르다.

8.4.2 분할

분할이란 이미지를 픽셀 수준에서 분류하는 문제이다. 다음 그림과 같이 픽셀 단위로 객체마다 채색된지도 데이터를 사용해 학습한다.

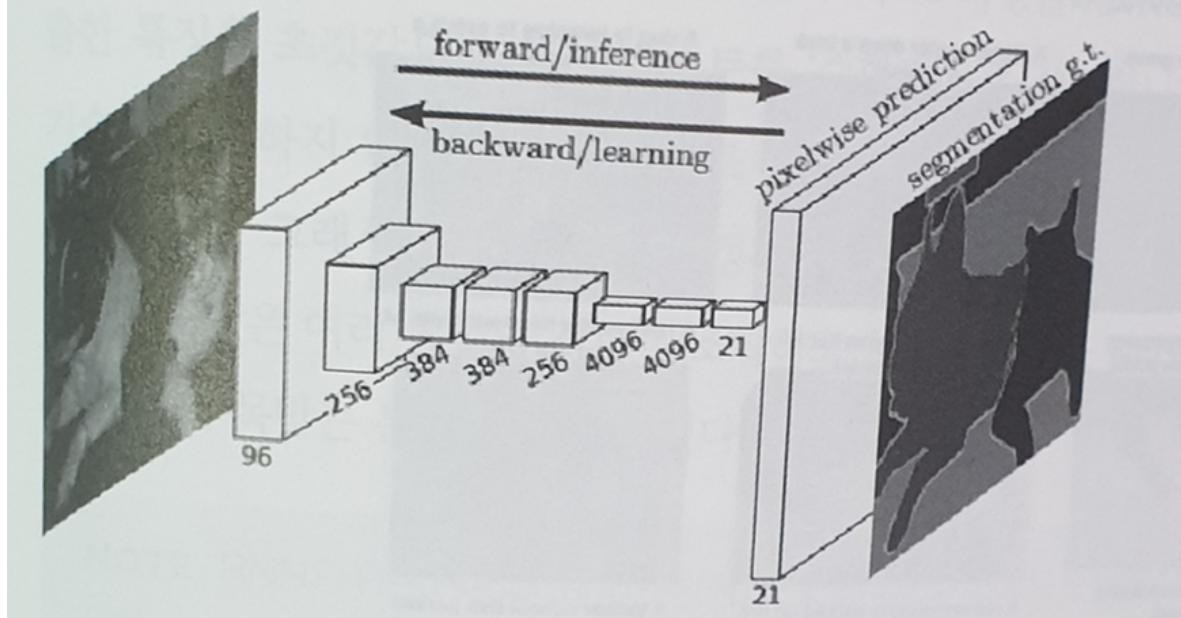
그림 8-19 분할의 예 : 왼쪽이 입력 이미지, 오른쪽이 지도용 이미지^[34]



지금까지 구현한 신경망은 분류를 이미지 전체를 대상으로 해왔다. 이를 픽셀 수준에 적용하려면 어떻게 하면 될까?

신경망을 이용해 분할하는 가장 단순한 방법은 모든 픽셀 각각을 추론하는 것이다. 예를 들어 어떤 직사각형 영역의 중심 픽셀의 클래스를 분류하는 신경망을 만들어서, 모든 픽셀을 대상으로 하나씩 추론 작업을 실행한다. 짐작한 대로 이런 식으로는 픽셀의 수만큼 forward 처리를 해야 하여 긴 시간이 걸린다(정확히는 합성곱 연산에서 많은 영역을 쓸데없이 다시 계산하는 것이 문제가 된다). 이러한 낭비를 줄여주는 기법으로 FCN(Fully Convolutional Network)이 고안되었다. 이는 단 한 번의 forward 처리로 모든 픽셀의 클래스를 분류해주는 놀라운 기법이다.

그림 8-20 FCN의 전체 그림^[37]



Fully Convolutional Network 를 직역하면 '합성곱 계층만으로 구성된 네트워크'가 된다.

일반적인 CNN이 완전연결 계층을 이용하는 반면, FCN은 이 완전연결 계층을 '같은 기능을 하는 합성곱 계층'으로 바꾼다. 사물 인식에서 사용한 신경망의 완전연결 계층에서는 중간 데이터의 공간 볼륨(다차원 형태)을 유지한 채 마지막 출력까지 처리할 수 있다.

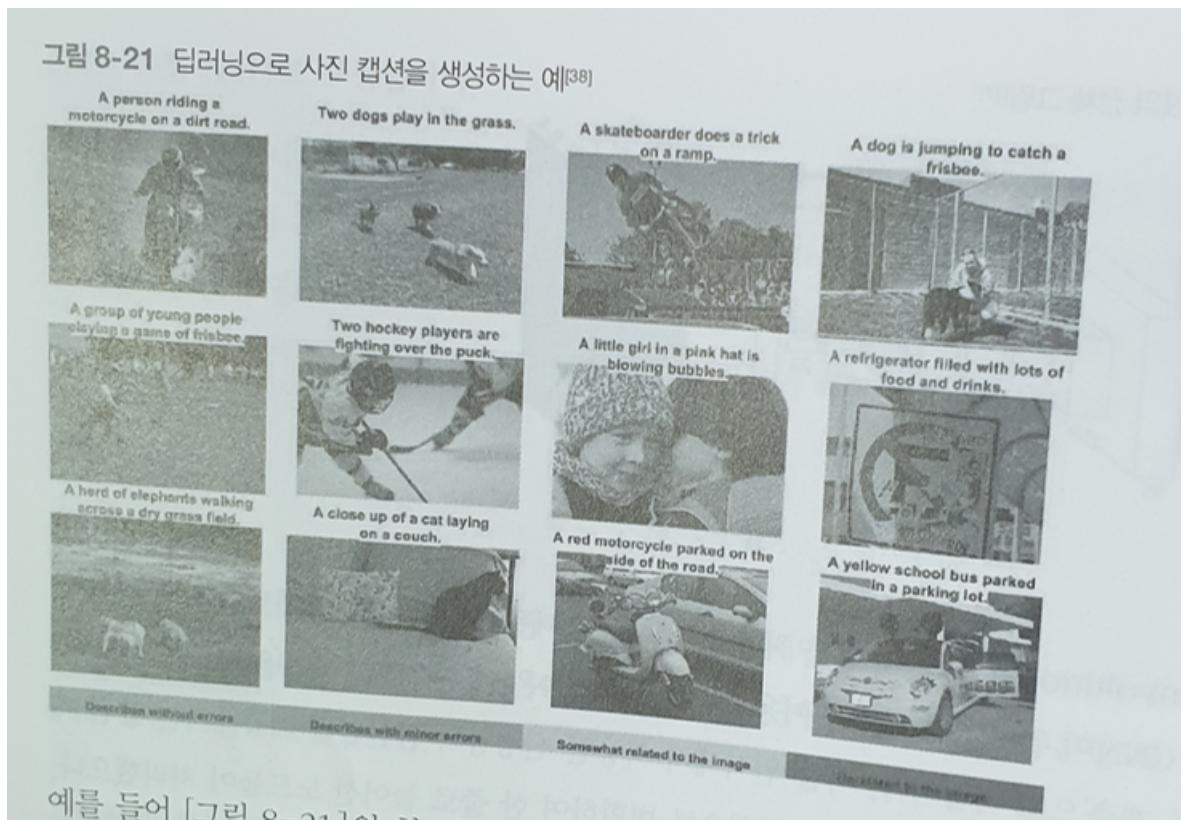
FCN은 위 그림에서 보듯 마지막에 공간 크기를 확대하는 처리를 도입했다는 것도 특징이다. 이 확대 처리로 인해 줄어든 중간 데이터를 입력 이미지와 같은 크기까지 단번에 확대 할 수 있다. FCN의 마지막에 수행하는 확대는 이중 선형 보간에 의한 선형 확대이다. FCN에서는 이 선형 확대를 역합성곱 연산으로 구현해내고 있다.

Note 완전연결 계층에서는 출력이 모든 입력과 연결된다. 이와 같은 구성을 합성곱 계층으로도 구현할 수 있다. 가령 입력 크기가 $32 \times 10 \times 10$ (채널 32개, 높이 10, 너비 10)인 데이터에 대한 완전연결 계층은 필터 크기가 $32 \times 10 \times 10$ 인 합성곱 계층으로 대체할 수 있다. 만약 완전연결 계층의 출력 노드가 100개라면, 합성곱 계층에서는 기존의 $32 \times 10 \times 10$ 필터를 100개 준비하면 완전히 같은 처리를 할 수 있다. 이처럼 완전연결 계층은 같은 일을 수행하는 합성곱 계층을 대체할 수 있다.

8.4.3 사진 캡션 생성

컴퓨터 비전과 자연어를 융합한 재미있는 연구가 있다. 아래 그림과 같은 사진을 주면, 그 사진을 설명하는 글(사진 캡션)을 자동으로 생성하는 연구이다.

그림 8-21 딥러닝으로 사진 캡션을 생성하는 예[38]



예를 들어 [그림 8-21]의 예를 살펴보자.

예를 들어, 위 그림의 첫 번째는 사진만 보고 "비포장도로에서 오토바이를 타는 사람"이라는 문장을 자동으로 생성했다. 설명과 사진이 정확히 일치한다. 오토바이를 타고 있다는 것뿐만 아니고 거친 비포장도로 라는 것도 '이해'하고 있다니 놀라울 따름이다.

딥러닝으로 사진 캡션을 생성하는 방법으로는 NIC(Neural Image Caption)모델이 대표적이다. NIC는 아래 그림과 같이 심층 CNN과 자연어를 다루는 **순환 신경망(Recurrent Neural Network, RNN)**으로 구성된다. RNN은 순환적 관계를 갖는 신경망으로 자연어나 시계열 데이터 등의 연속된 데이터를 다룰 때 많이 활용한다.

NIC는 CNN으로 사진에서 특징을 추출하고, 그 특징을 RNN에 넘긴다. RNN은 CNN이 추출한 특징을 초기값으로 해서 텍스트를 '순환적'으로 생성한다. 여기에서는 더 이상의 상세 기술은 설명하지 않지만, 기본적으로 NIC는 2개의 신경망(CNN과 RNN)을 조합한 간단한 구성이다. 그래서 놀라울 정도로 정확한 사진캡션을 만들어내는 것이다. 또한, 사진이나 자연어와 같은 여러 종류의 정보를 조합하고 처리하는 것을 **멀티모달 처리(multimodal processing)**이라고 하여, 최근 주목받는 분야 중 하나이다.

Note RNN의 R은 Recurrent(순환적)를 뜻한다. 여기에서 순환은 신경망의 순환적 네트워크 구조를 말한다. 이 순환적인 구조로 인해 이전에 생성한 정보에 영향을 받는(바꿔말하면, 과거의 정보를 기억하는) 점이 RNN의 특징이다. 예를 들어 '나는 잤다'라는 문장이 완성되는 식이다. 이처럼 자연어와 시계열 데이터 등 연속성 있는 데이터를 다룰 때 RNN은 과거의 정보를 기억하면서 동작한다.

8.5 딥러닝의 미래

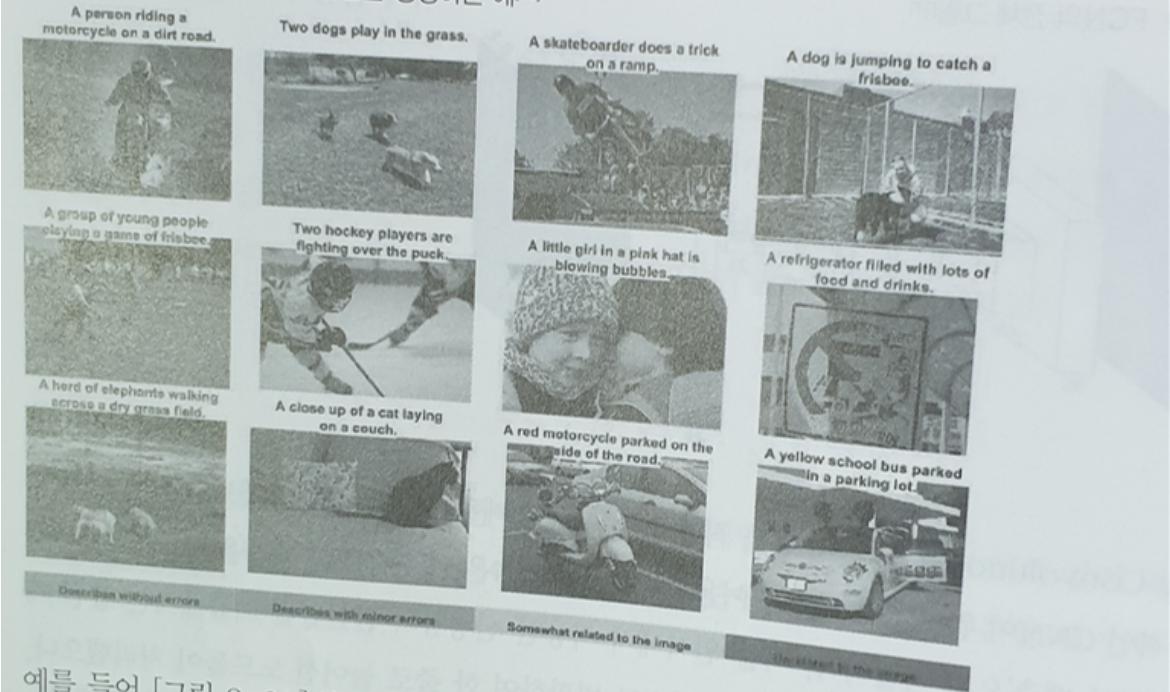
딥러닝은 앞에서 언급한 분야 외에도 여러 분야에서 이용되어 왔다. 이번 절에서는 딥러닝의 가능성과 미래를 느낄 만한 연구를 몇 가지 소개하겠다.

8.5.1 이미지 스타일(화풍) 변환

딥러닝을 활용해 화가처럼 '그림을 그리는' 연구가 있다. 다음 그림은 두 이미지를 입력해서 새로운 그림을 생성하는 연구이다. 하나는 '콘텐츠 이미지', 다른 하나는 '스타일 이미지'라 부르는데, 이 둘을 조합해 새로운 그림을 그려준다.

다음 그림과 같이 고흐의 화풍을 콘텐츠 이미지에 적용하도록 지정하면, 이를 기초로 딥러닝이 새로운 그림을 그린다.

그림 8-21 딥러닝으로 사진 캡션을 생성하는 예^[38]



예를 들어 「그림 8-21」은

여기서는 이 연구를 자세히 설명하지는 않겠다. 큰 틀만 이야기하면, 이 기술은 네트워크의 중간 데이터가 콘텐츠 이미지의 중간 데이터와 비슷해지도록 학습한다. 이렇게 하면 입력 이미지를 콘텐츠 이미지의 형태와 함께 흉내 낼 수 있다. 또, 스타일 이미지의 화풍을 흡수하기 위해 '스타일 행렬'이라는 개념을 도입한다. 그 스타일 행렬의 오차를 줄이도록 학습하여 입력 이미지를 고흐의 화풍과 비슷해지게 만들 수 있는 것이다.

8.5.2 이미지 생성

앞의 이미지 스타일 변환에는 새로운 그림을 생성하려면 이미지 두장을 입력해야 한다. 한편 아무런 입력 이미지 없이도 새로운 이미지를 그려내는 연구도 진행 중이다. 물론 먼저 대량의 이미지를 사용하여 학습하긴 하지만, 학습이 끝난 후에는 아무런 입력 이미지 없이도 새로운 그림을 그려낸다. 가령 딥러닝으로 '침실' 이미지를 무로부터 생성하는 게 가능하다. 아래 이미지는 **DCGAN(Deep Convolutional Generative Adversarial Network)** 기법으로 생성한 침실 이미지들이다.

그림 8-24 DCGAN으로 새롭게 생성한 침실 이미지들^[41]



위 이미지들은 진짜 사진처럼 보일지 모르지만 모두 **DCGAN**을 사용해 새롭게 생성한 이미지이다. 즉, DCGAN이 그린, 아직 아무도 본 적 없는 이미지(학습 데이터에는 존재하지 않는 이미지)이며, 처음부터 새로 생성한 이미지이다.

자, 진짜와 구분할 수 없는 수준의 이미지를 그리는 DCGAN은 이미지를 생성하는 과정을 모델화 한다. 그 모델을 대량의 이미지(가령 침실이 찍힌 대량의 이미지)를 사용해 학습하고, 학습이 끝나면 그 모델을 이용하여 새로운 그림을 생성할 수 있다.

DCGAN도 딥러닝을 사용한다. DCGAN 기술의 핵심은 생성자와 식별자로 불리는 2개의 신경망을 이용한다는 점이다. 생성자가 진짜와 똑같은 이미지를 생성하고 식별자는 그것이 진짜인지(생성자가 생성한 이미지인지, 아니면 실제로 촬영된 이미지인지)를 판정한다. 그렇게 해서 둘을 겨루도록 학습시켜, 생성자는 더 정교한 가짜 이미지 생성 기술을 학습하고 식별자는 더 정확하게 간파할 수 있는 감정사로 성장하는 것이다. 이렇게 둘의 능력을 부지런히 갈고닦게 한다는 개념이 **GAN(Generative Adversarial Network)** 기술의 재미난 점이다.

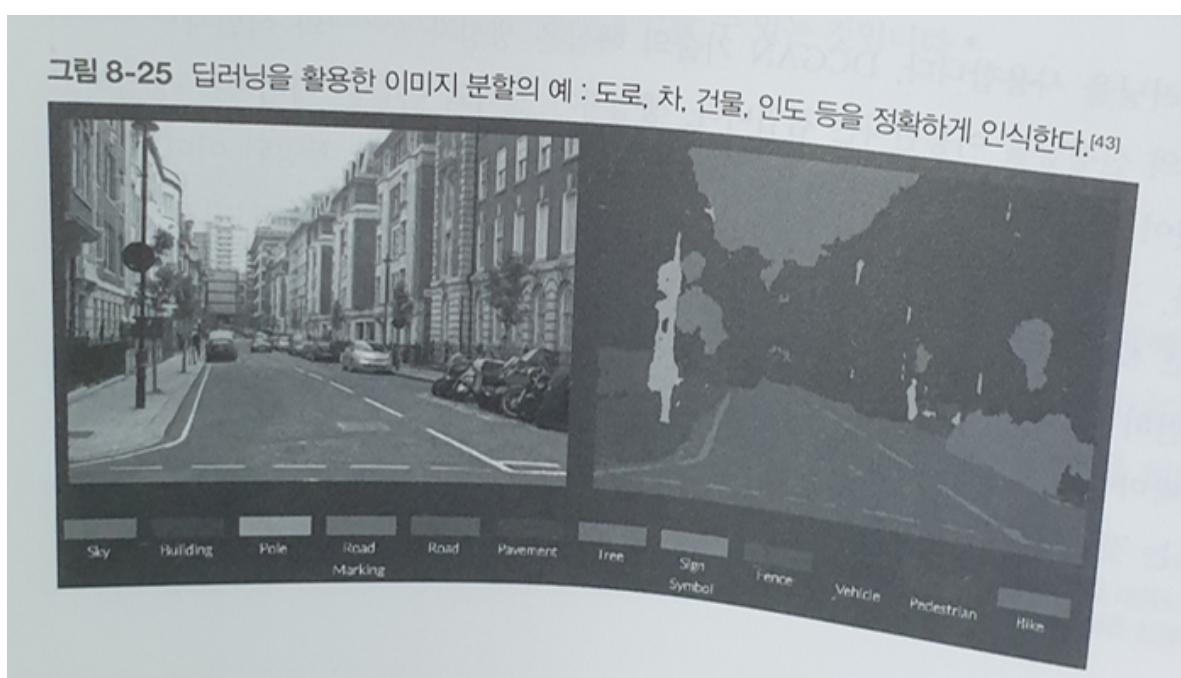
그렇게 절차탁마해서 성장한 생성자는 최종적으로는 진짜와 착각할 정도의 이미지를 그려나는 능력을 기르는 것이다.

Note 이전까지 살펴본 기계학습 문제는 지도 학습이라는 유형의 문제였다. 지도 학습은 손글씨 숫자 인식처럼 이미지 데이터와 정답 레이블을 짹지은 데이터셋을 이용한다. 그러나 이번 절에서 거론한 문제는 지도용 데이터는 주어지지 않고, 단지 대량의 이미지(이미지의 집합)만 주어진다. 즉, 지도 없이 스스로 학습하는 자율 학습문제다. 자율 학습은 비교적 오래전부터 연구된 분야지만 최근에는 그다지 활발하게 연구되지는 않는다는 느낌이다. 최근 딥러닝을 사용한 **DCGAN** 등과 같은 기법이 시선을 끌면서, 앞으로 자율 학습도 새로운 도약을 기대한다.

8.5.3 자율 주행

사람 대신 컴퓨터가 자동차를 운전하는 **자율 주행** 기술이 눈앞으로 다가왔다. 자동차 제조업체 뿐 아니라 IT기업과 대학, 연구소도 자율 주행 상용화 경쟁에 뛰어들었다. 자율 주행은 다양한 기술(주행 경로를 정하는 경로 계획 기술과 카메라나 레이저 등의 탐사 기술 등)을 모아 구현하고 있지만, 그중에서도 주위 환경을 올바르게 인식하는 기술이 가장 중요한 문제이다. 시시각각 변하는 환경과 다른 차와 사람들을 올바르게 인식하기가 매우 어렵다. 다양한 환경에서도 안전한 주행 영역을 올바로 인식하게 되면 자율 주행의 실현도 멀지 않는다. 이에는 딥러닝이 아주 큰 역할을 해줄 것이다.

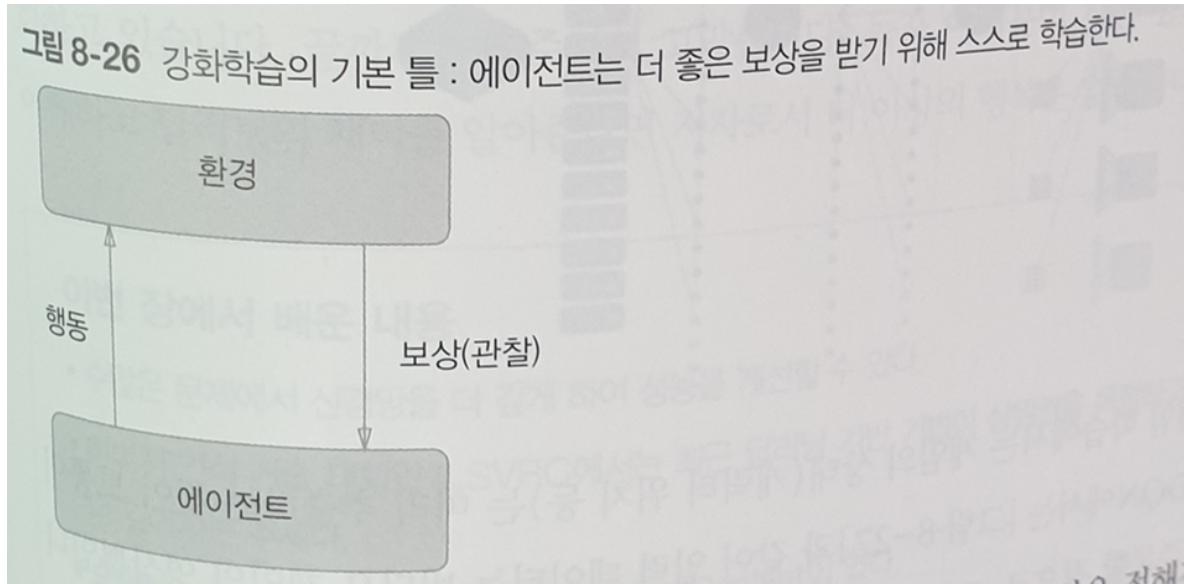
예를 들어 **SegNet**이라는 CNN 기반 신경망은 다음 그림과 같이 주변 환경을 정확하게 인식해낸다.



위 그림과 같이 입력 이미지를 분할(픽셀 수준에서 판정)하고 있다. 결과를 보면 도로와 건물, 보도와 나무, 차량과 오토바이 등을 어느 정도 정확히 판별하고 있다. 이런 인식 기술이 딥러닝에 힘입어 향후 한층 정확해지고 빨라지면 자율 주행이 일상에 파고든 영화 속 모습이 현실이 되겠다.

8.5.4 Deep Q - Network (강화학습)

"가르침"에 의존하는 '지도 학습'과는 다른 분야로, 스스로 학습하게 하려는 분야를 **강화학습**이라고 한다. 강화학습에서는 에이전트라는 것이 환경에 맞게 행동을 선택하고, 그 행동에 의해서 환경이 변한다는 게 기본적인 틀이다. 환경이 변화하면 에이전트는 어떠한 보상을 얻는다. 강화학습의 목적은 더 나은 보상을 받는 쪽으로 에이전트의 행동 지침을 바로 잡는 것이다.

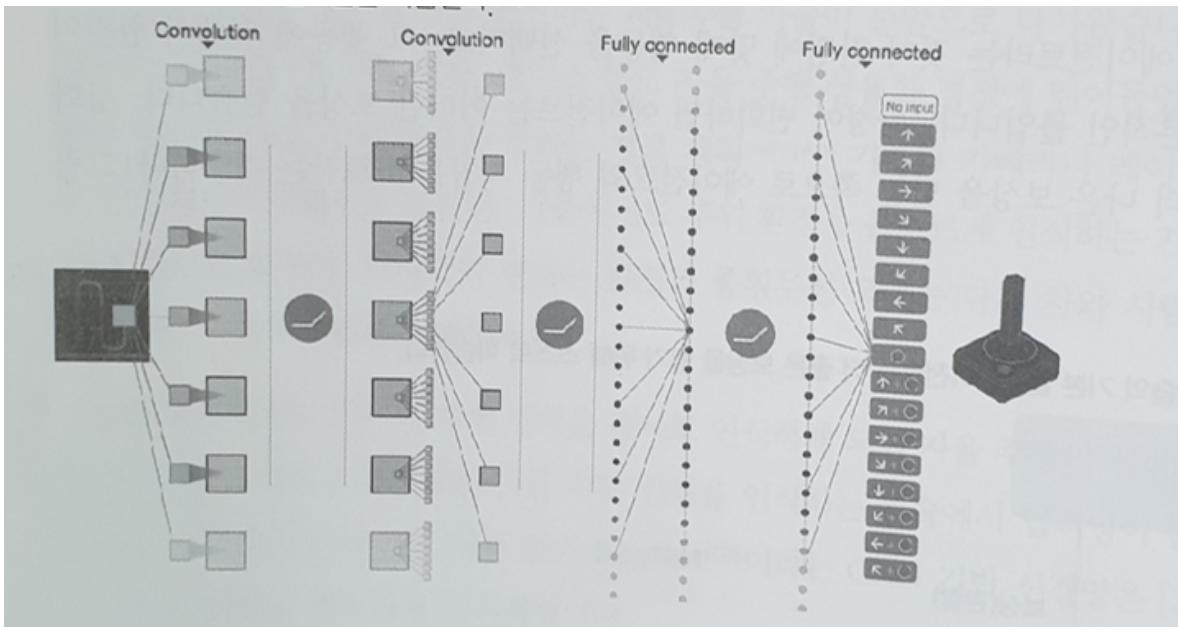


위 그림은 강화학습의 기본 틀이다. 여기서 주의점은 보상은 정해진 것이 아닌 '예상 보상'이라는 점이다. 예를 들면 <슈퍼 마리오 브라더스>에서 마리오를 오른쪽으로 이동했을 때 얻는 보상이 항상 명확한 것이 아니다. 어떤 상황에서 이동한 것이냐에 따라 보상은 천차만별이 될 수 있겠다. 이런 불명확한 상황에서 는 게임 점수(동전을 먹거나 적을 쓰러뜨리는 등)나 게임 종료 등의 명확한 지표로 부터 역산해서 '예상 보상'을 정해야 한다. 만약 지도 학습이었다면 행동에 대한 '지도'를 통해 올바른 평가를 받을 수 있었을 것이다.

딥러닝을 사용한 강화학습 중 **Deep Q - Network (일명 DQN)**이라는 방법이 있다. 이는 Q 학습이라는 강화학습 알고리즘을 기초로 한다. Q학습에서는 최적 행동 가치 함수로 최적인 행동을 정한다.

이 함수를 딥러닝 (CNN)으로 비슷하게 흉내내어 사용하는 것이 DQN이다.

DQN 연구 중에는 비디오 게임을 자율 학습시켜 사람을 뛰어 넘는 수준의 조작을 실현한 사례가 보고되고 있다. 아래 그림과 같이 DQN에서 사용하는 CNN은 게임영상 프레임(4개의 연속한 프레임)을 입력하여 최종적으로 게임을 제어하는 움직임(조이스틱 이동량이나 버튼 조작 여부)에 대해 각 동작의 '가치'를 출력한다.



그동안의 비디오 게임 학습에서는 게임의 상태(캐릭터 위치 등)는 미리 추출하는 것이 보통이었다. 그러나 DQN에서는 위 그림과 같이 입력 데이터는 비디오 게임의 영상뿐이다. 이는 DQN의 주목할 점으로, DQN의 응용 가능성을 현격히 높였다고 할 수 있다. 게임마다 설정을 바꿀 필요없이 단순히 DQN에 게임 영상을 보여주기만 하면 되기 때문이다. 실제 DQN은 구성을 변경하지 않고도 팩맨과 아타리 같은 많은 게임을 학습할 수 있으며, 수많은 게임에서 사람보다 뛰어난 성적을 거두고 있다.

8.6 정리

이번 장에서는 (약간) 깊은 CNN을 구현하고, 순글씨 숫자 인식에서 99%가 넘는 높은 정확도를 얻었다. 또, 네트워크를 깊게 하는 동기를 이야기하고 최근의 딥러닝이 더 깊어지는 방향으로 가고 있다는 것도 설명했다. 그리고 딥러닝의 트랜드와 실제 활용 예, 고속화를 위한 연구와 미래를 느끼게 해주는 연구 사례를 소개했다.

이번 장에서 배운 것

- 수많은 문제에서 신경망을 더 깊게 하여 성능을 개선할 수 있다.
- 유명한 신경망으로는 VGG, GoogLeNet, ResNet이 있다.
- GPU와 분산 학습, 비트 정밀도 감소 등으로 딥러닝을 고속화 할 수 있다.
- 딥러닝은 사물 인식뿐 아니라 사물 검출과 분할에도 이용할 수 있다.
- 딥러닝의 응용 분야로는 사진의 캡션 생성, 이미지 생성, 강화학습 등이 있다. 최근에는 자율 주행에도 딥러닝을 접목하고 있어 기대된다.