

생성형 AI

Day 14

딥러닝 II



목차

1. Computer Vision?
2. Computer Vision & DL
3. Convolutional Neural Network
4. 주요 모델
5. 실습과제



Computer Vision

Computer Vision?

- 컴퓨터가 디지털 이미지나 비디오에서 의미 있는 정보를 추출하고 해석할 수 있도록 하는 기술
- 인간이 시각 정보를 인식하고 이해하는 과정을 모방하는 것으로, 다양한 산업 분야에서 중요한 역할
- 이미지, 동영상, ...
- 자율주행차, 의료 영상 분석, 얼굴 인식, 보안 시스템, 산업 자동화 등

이미지와 영상

- 이미지는 정지된 단일 프레임의 그림, 고정된 순간
- 이미지 예시: 사진, 단일 프레임 스크린샷
- 영상은 시간에 따라 연속적으로 변화하는 여러 프레임의 집합, 움직임을 포함
- 영상 예시: 동영상, 라이브 스트림

Computer Vision

Computer Vision 구성요소

픽셀(Pixel)

- 이미지의 가장 작은 단위
- 각 픽셀은 특정한 색상 정보를 가지며, 그 조합으로 전체 이미지 구성
- 이미지 해상도: 이미지의 가로, 세로 픽셀 수로 표현 (예: 1920x1080)
- 그레이스케일 이미지: 픽셀이 흑백 정보만을 가지며, 보통 0(검정)에서 255(흰색) 사이의 값을 가짐
- 컬러 이미지: 각 픽셀이 RGB(Red, Green, Blue) 값을 가지며, 컬러 이미지는 이 세 가지 색상의 조합으로 표현

컬러 공간

- 색상을 숫자로 표현하는 방법
- RGB: 가장 일반적인 컬러 공간, 예를 들어 (255, 0, 0)은 빨간색, (0, 255, 0)은 녹색, (0, 0, 255)는 파란색
- Grayscale: 흑백 이미지로, RGB 채널의 평균 값을 사용하여 단일 채널로 변환, 0은 검정, 255는 흰색
- HSV: 색상(Hue), 채도(Saturation), 명도(Value)로 색상을 표현

Computer Vision & DL

Traditional Computer Vision Method

- 특징 추출 알고리즘을 기반

엣지 검출(Edge Detection):

- 이미지에서 물체의 경계를 찾는 방법
- 예: 소벨(Sobel) 필터, 캐니(Canny) 엣지 검출기

코너 검출(Corner Detection):

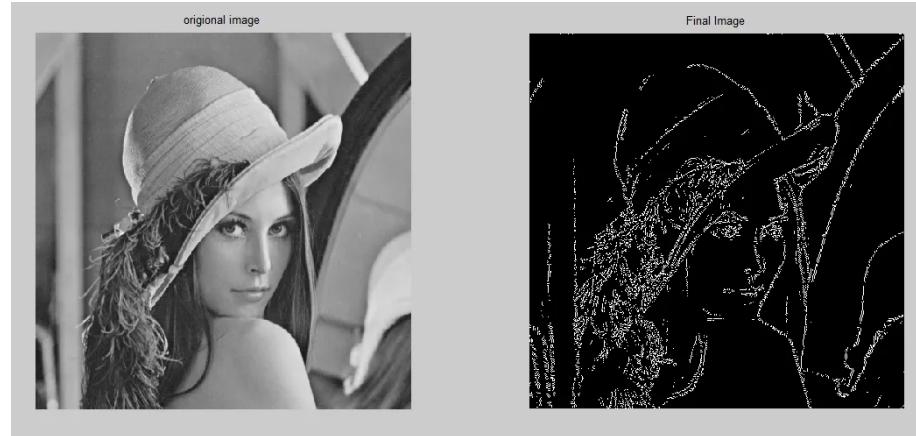
- 이미지에서 코너(모서리)를 찾는 방법
- 예: 해리스(Harris) 코너 검출기

SIFT(Scale-Invariant Feature Transform):

- 이미지의 스케일 변화와 회전에 강인한 특징을 추출
- 주로 객체 인식과 이미지 매칭에 사용

SURF(Speeded Up Robust Features):

- SIFT의 개선된 버전, 더 빠른 계산 속도



Computer Vision & DL

Deep Learning Computer Vision

- 딥러닝은 대량의 데이터를 처리하고 학습할 수 있는 능력 덕분에 전통적인 컴퓨터 비전 기법을 대체해나가는 중

주요 응용 분야

이미지 분류(Image Classification)

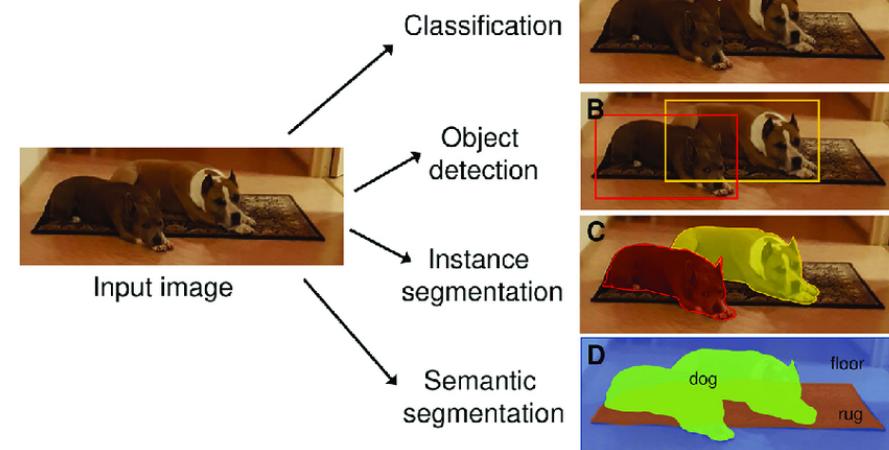
- 이미지를 특정 카테고리로 분류
- 예: 개, 고양이, 자동차 등

객체 검출(Object Detection)

- 이미지 내 여러 객체의 위치와 종류를 식별
- 예: 자율주행차에서 보행자와 차량 인식

이미지 세분화(Image Segmentation)

- 이미지의 각 픽셀을 특정 클래스에 할당
- 예: 의료 영상에서 장기 분할



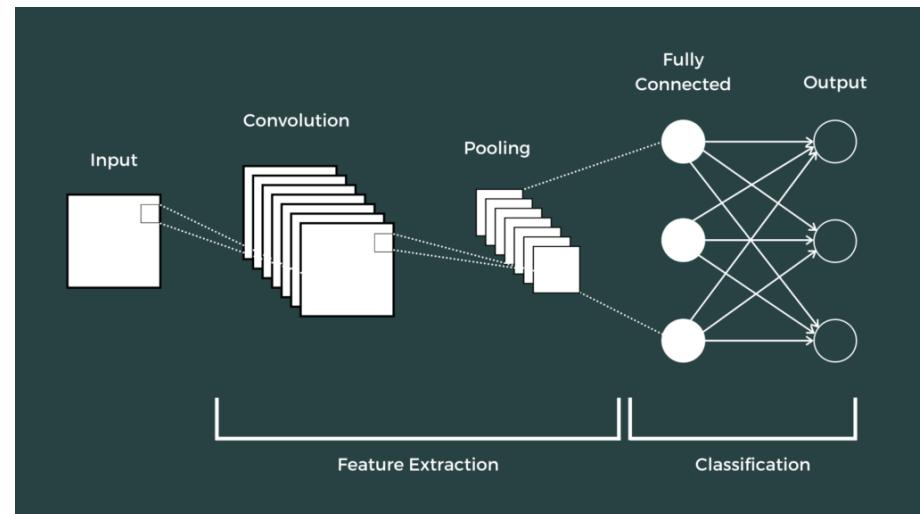
Convolutional Neural Network, 합성곱 신경망

합성곱 신경망 (CNN)?

- 이미지와 같은 격자 구조 데이터를 처리하기 위해 설계된 딥러닝 모델
- 이미지 분류, 객체 검출, 이미지 세분화와 같은 컴퓨터 비전 분야에서 뛰어난 성능을 발휘
- 이미지 내에서 ‘중요한 특징을 자동으로 추출’하고, 이를 바탕으로 예측이나 분류 작업을 수행
- 다수의 합성곱 층과 풀링 층을 포함하며, 지역적 특징을 효과적으로 추출

CNN의 기본 구조

- 합성곱 층(Convolutional Layer)
- 풀링 층(Pooling Layer)
- 완전 연결 층(Fully Connected Layer)
- 활성화 함수(Activation Function)



Convolutional Neural Network, 합성곱 신경망

합성곱 층(Convolutional Layer)

- 합성곱 층은 이미지에서 지역적 특징을 추출하는 역할
- 작은 필터(또는 커널)를 사용하여 입력 이미지에 대해 합성곱 연산을 수행

필터

- 작은 크기의 행렬(예: 3x3, 5x5)로, 입력 이미지에 적용되어 특징 맵을 생성

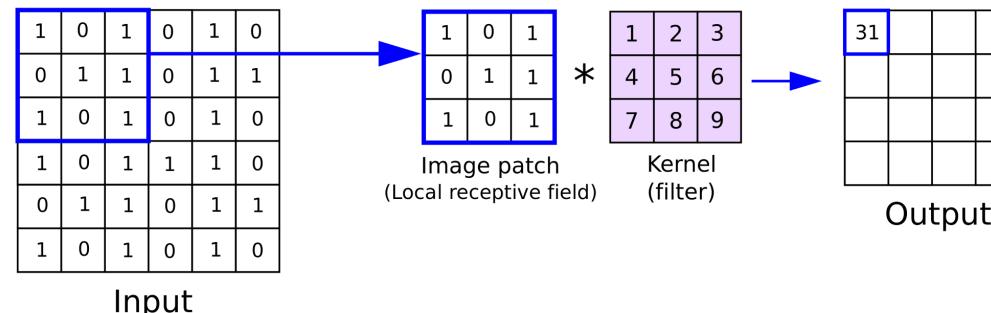
스트라이드

- 필터가 입력 이미지 위를 이동하는 간격

패딩

- 출력 크기를 조절하거나 경계 정보 손실을 방지하기 위해 입력 이미지의 가장자리에 추가된 픽셀 값

iq.opengenus.org



Convolutional Neural Network, 합성곱 신경망

풀링 층(Pooling Layer)

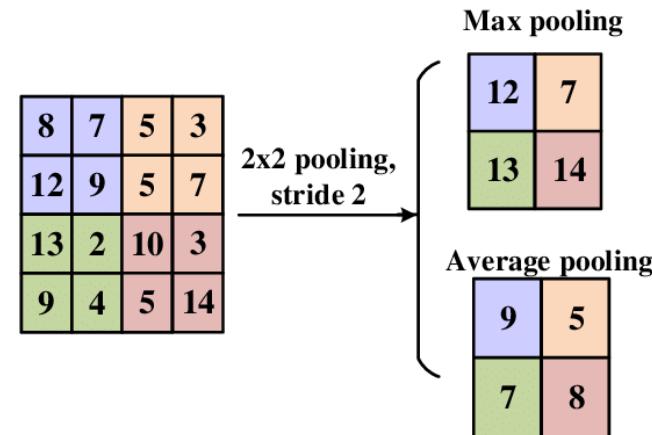
- 풀링 층은 입력 이미지의 공간적 크기를 줄여 계산량을 감소시키고, 모델의 일반화 성능을 향상시키는 역할

맥스 풀링

- 공간 크기를 줄이면서 중요한 특징을 보존하기 위해 풀링 영역 내에서 최대값을 선택

평균 풀링

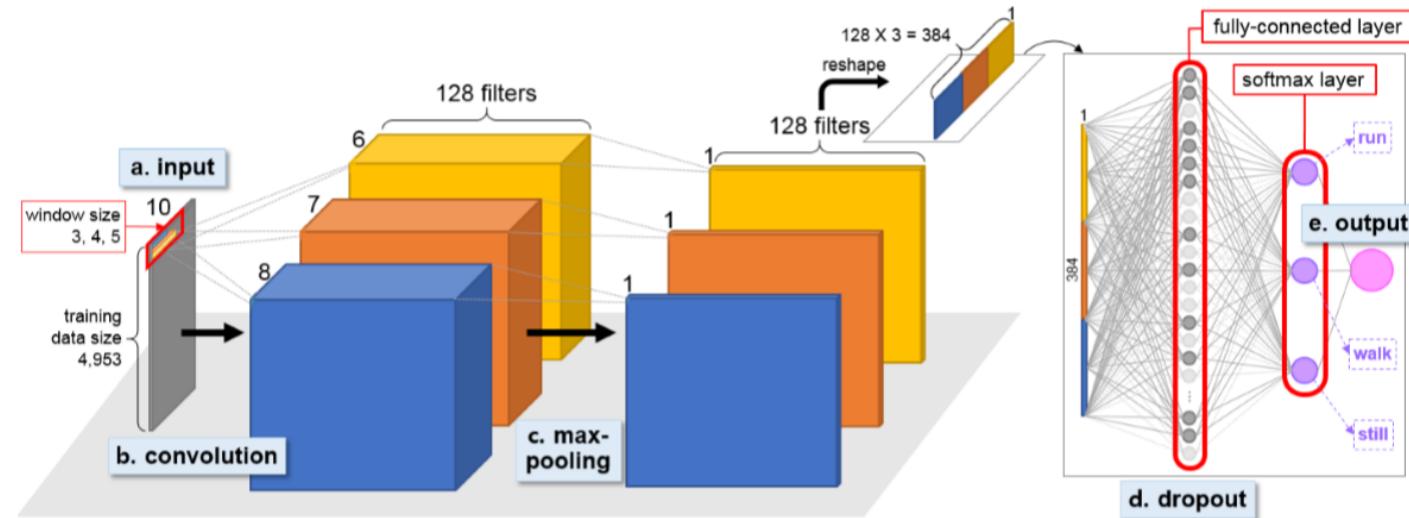
- 풀링 영역 내의 평균값을 선택



Convolutional Neural Network, 합성곱 신경망

완전 연결 층(Fully Connected Layer)

- 완전 연결 층은 신경망의 마지막 단계에서 사용되며, 모든 입력 뉴런이 모든 출력 뉴런과 연결
- 고차원 특징을 결합하여 최종 예측을 수행
- 전통적인 신경망과 유사하게 동작



Convolutional Neural Network, 합성곱 신경망

활성화 함수(Activation Function)

- 활성화 함수는 뉴런의 출력을 비선형 변환하여 신경망의 표현력을 향상

ReLU

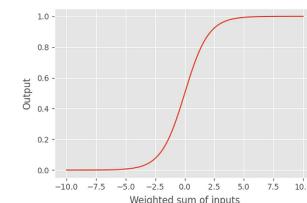
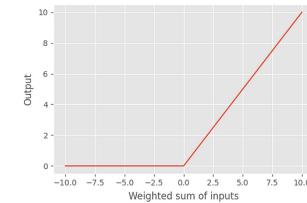
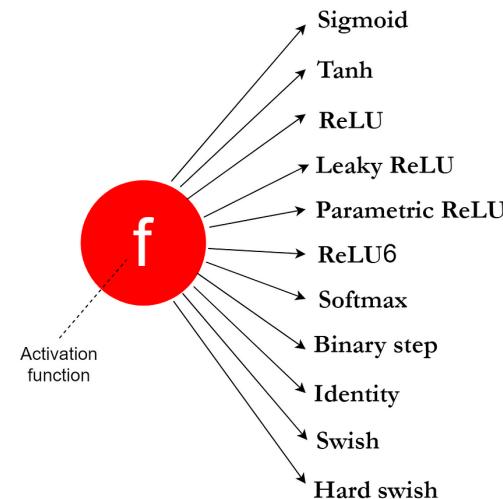
- 가장 널리 사용되는 활성화 함수로, 음수를 0으로 변환

Sigmoid

- 출력값을 0과 1 사이로 변환

Tanh

- 출력값을 -1과 1 사이로 변환



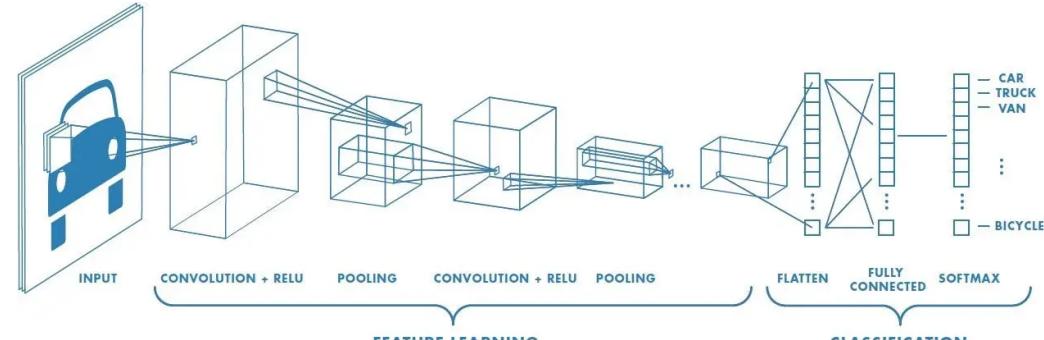
Convolutional Neural Network, 합성곱 신경망

CNN의 작동 원리

- 입력 이미지에 대해 여러 단계의 합성곱과 풀링을 수행하여 점진적으로 추상화된 특징을 추출
- 각 단계에서 추출된 특징 맵은 다음 단계의 입력으로 사용되며, 이는 고차원의 특징을 점진적으로 축적

CNN Flow

1. 이미지 입력: 원본 이미지 또는 전처리된 이미지가 입력
2. 합성곱 층: 필터와 스트라이드를 사용하여 지역적 특징을 추출
3. 풀링 층: 공간 크기를 줄이고 중요한 특징을 보존
4. 2~3 반복: 합성곱 층과 풀링 층을 여러 번 반복하여 고차원 특징을 추출
5. 완전 연결 층: 최종 특징을 결합하여 예측을 수행
6. 출력: 분류 결과나 회귀 값을 출력



Convolutional Neural Network, 합성곱 신경망

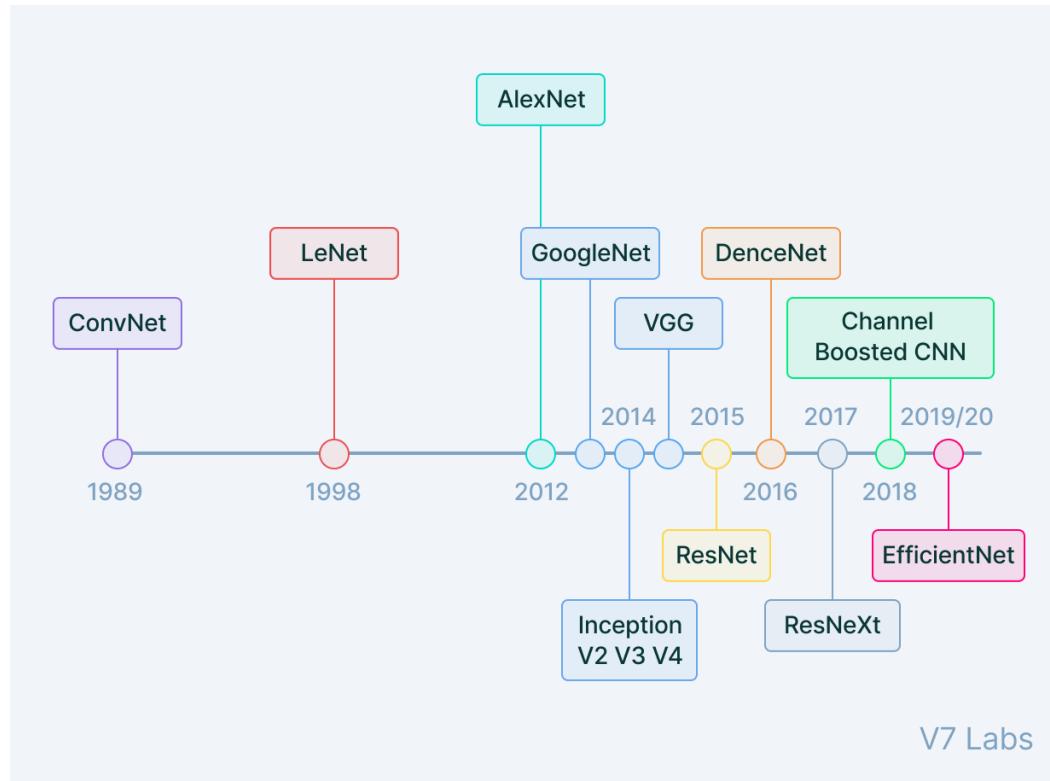
CNN의 장점

- 수작업 없이 중요한 특징을 자동으로 추출하여 학습
- 필터가 지역적 패턴을 학습하므로, 이미지의 공간적 구조를 효과적으로 활용
- 저차원에서 고차원으로 특징을 점진적으로 학습

CNN의 한계

- 대량의 데이터를 필요로 하며, 데이터가 부족할 경우 과적합이 발생할 수 있음
- 다수의 합성곱 연산과 다수의 층을 포함하므로, 계산 비용이 높음
- 딥러닝의 특성상, 높은 성능에도 불구하고, 모델의 내부 동작을 해석하는 것이 어려움

주요모델



주요모델

LeNet-5

- 1998년에 개발된 합성곱 신경망(CNN), 손글씨 숫자 인식에 사용
- LeNet-5는 총 7개의 층으로 구성
- 3개의 합성곱 층과 2개의 풀링 층, 2개의 완전 연결 층으로 구성

구조

- 입력층: 32x32 픽셀의 흑백 이미지를 사용, 입력 이미지를 신경망에 전달
- 합성곱층: 5x5 크기의 6개 필터를 사용, 28x28x6의 출력, Sigmoid 또는 Tanh를 사용, 저수준의 특징을 추출
- 풀링층: 평균 풀링, 2x2 크기의 필터, 14x14x6의 출력, 특징 맵의 크기를 줄여 계산량을 감소, 위치 변화에 대한 강건성을 향상
- 합성곱층: 5x5 크기의 16개 필터를 사용, 10x10x16의 출력, Sigmoid 또는 Tanh를 사용, 더 복잡한 특징을 추출
- 풀링층: 평균 풀링, 2x2 크기의 필터, 5x5x16의 출력, 특징 맵의 크기를 더 줄여, 모델의 복잡도 감소
- 합성곱층: 5x5 크기의 120개 필터를 사용, 1x1x120의 출력, Sigmoid 또는 Tanh를 사용, 전체 이미지를 대표하는 고차원 특징을 추출
- 완전 연결 층: 84개 뉴런, Sigmoid 또는 Tanh, 추출된 특징을 기반으로 클래스별 점수를 계산
- 출력층: 10개 뉴런 (클래스 수, 0-9), Softmax, 각 클래스에 대한 확률을 출력하여, 최종적으로 숫자를 분류

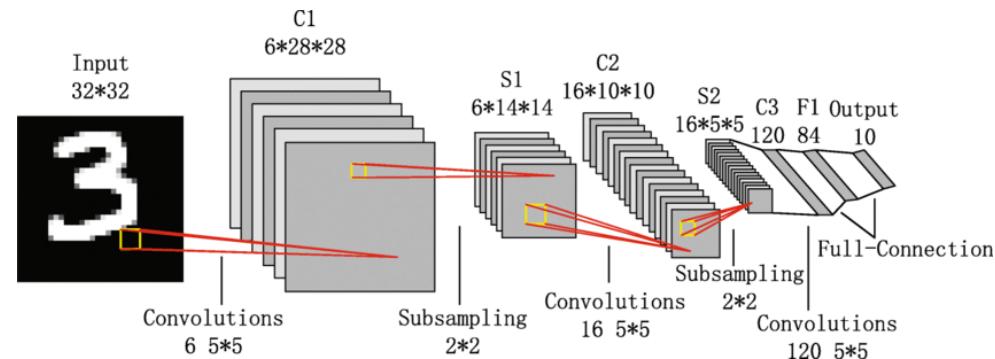
주요모델

LeNet-5의 장점

- 수작업 없이 이미지의 중요한 특징을 자동으로 학습
- 필터가 지역적 패턴을 학습하여 이미지의 공간적 구조를 효과적으로 활용
- 저차원에서 고차원으로 특징을 점진적으로 학습하여, 복잡한 패턴도 효과적으로 인식
- 풀링 층을 통해 파라미터 수를 줄이고, 계산량을 효율적으로 관리

LeNet-5의 단점

- 여러 층을 포함하므로, 모델의 설계와 학습이 복잡
- 계산 비용이 높음
- 대량의 데이터를 필요



주요모델

AlexNet

- 2012년 ImageNet 대회에서 우승하며 딥러닝 분야에 큰 혁신을 가져온 합성곱 신경망(CNN) 모델
- 입력 이미지에서 저수준 특징을 추출한 후, 점진적으로 더 복잡하고 추상적인 특징을 학습
- AlexNet은 8개의 학습 가능한 층으로 구성
- 5개의 합성곱 층과 3개의 풀링 층, 3개의 완전 연결 층으로 구성
- ReLU 활성화 함수와 드롭아웃을 도입하여 모델 성능 향상

구조

- 입력층: 224x224x3 크기의 RGB 이미지, 이미지를 모델에 입력
- 합성곱층: 11x11, 96개 필터, 스트라이드 4, 55x55x96의 출력, ReLU, Local Response Normalization(LRN), 큰 필터와 스트라이드를 사용하여 저수준의 특징 추출
- 풀링층: 맥스 풀링, 3x3 필터, 스트라이드 2, 27x27x96의 출력, 공간 크기를 줄이고, 중요한 특징을 보존
- 합성곱층: 5X5 크기의 256개 필터를 사용, 스트라이드 1, 패딩 1, 27x27x256의 출력, ReLU, LRN, 중간 수준의 특징 추출
- 풀링층: 맥스 풀링, 3x3 필터, 스트라이드 2, 13x13x256의 출력, 공간 크기를 줄이고, 중요한 특징을 보존
- 합성곱층: 3X3 크기의 384개 필터를 사용, 스트라이드 1, 패딩 1, 13x13x384의 출력, ReLU, 고차원 특징 추출
- 합성곱층: 3X3 크기의 384개 필터를 사용, 스트라이드 1, 패딩 1, 13x13x384의 출력, ReLU, 더 깊은 고차원 특징 추출
- 합성곱층: 3X3 크기의 256개 필터를 사용, 스트라이드 1, 패딩 1, 13x13x256의 출력, ReLU, 마지막 고차원 특징 추출
- 풀링층: 맥스 풀링, 3x3 필터, 스트라이드 2, 6x6x256의 출력, 공간 크기를 줄이고, 중요한 특징을 보존
- 완전 연결 층: 4096개 뉴런, ReLU, 드롭아웃 0.5, 고차원의 특징을 결합하여 중요한 정보 추출
- 완전 연결 층: 4096개 뉴런, ReLU, 드롭아웃 0.5, 더 높은 수준의 고차원 특징을 결합
- 완전 연결 층: 1000개 뉴런 (ImageNet의 클래스 수), Softmax, 각 클래스에 대한 확률을 출력하여 최종 분류

주요모델

AlexNet 주요 특징

ReLU 활성화 함수

- Rectified Linear Unit (ReLU) 활성화 함수는 음수를 0으로 변환하고, 양수는 그대로 통과시키는 비선형 함수
- ReLU는 기울기 소실 문제를 완화하고, 학습 속도를 크게 향상
- $\text{ReLU}(x) = \max(0, x)$

드롭아웃 (Dropout)

- 학습 시 특정 뉴런을 무작위로 비활성화하여 과적합을 방지하는 정규화 기법
- 모델의 일반화 성능을 향상시키고, 다양한 뉴런 조합을 학습하도록 유도

Local Response Normalization (LRN)

- 인접한 뉴런들의 응답을 정규화하여 일반화 성능을 향상시키는 방법
- 인접한 뉴런의 활성화를 억제하여 과대 적합을 방지

데이터 증강 (Data Augmentation)

- 학습 데이터의 다양성을 높이기 위해 원본 데이터를 변형하는 방법
- 모델이 더 다양한 패턴을 학습하도록 돋고, 과적합을 방지

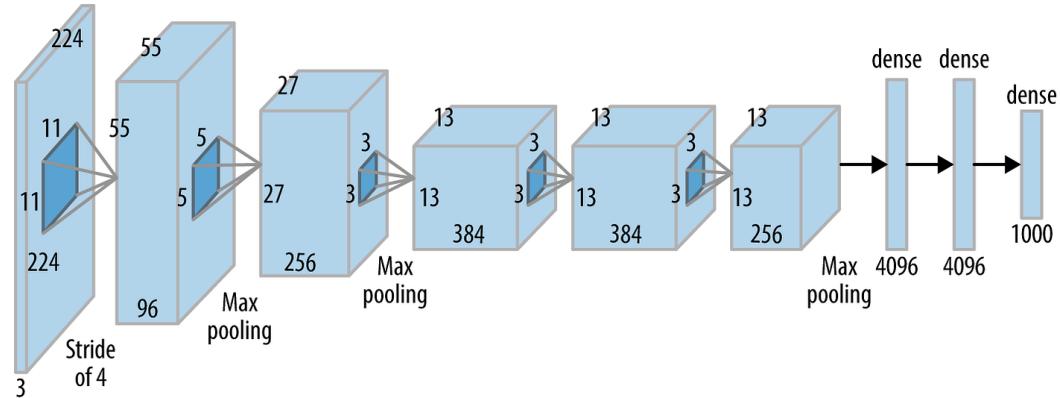
주요모델

AlexNet의 장점

- ImageNet과 같은 대규모 데이터셋을 효과적으로 학습하여 높은 성능을 발휘
- ReLU 활성화 함수를 사용해 기울기 소실 문제를 완화하고, 학습 속도를 크게 향상
- 드롭아웃을 사용해 과적합을 방지하고, 모델의 일반화 성능을 향상
- 데이터 증강을 통해 훈련 데이터의 다양성을 높여 과적합을 방지

AlexNet의 단점

- 많은 층과 필터를 사용하므로 계산 비용이 높음
- 대규모 데이터를 처리하기 위해 많은 메모리가 필요
- 여러 층을 포함하므로 모델의 설계와 학습이 복잡함



주요모델

VGGNet

- 2014년 ILSVRC(ImageNet Large Scale Visual Recognition Challenge) 대회에서 우수한 성능을 보여준 합성곱 신경망(CNN) 모델
- 신경망의 깊이를 증가시킴으로써 성능을 향상시키는 방법을 제시
- 매우 깊은 신경망 구조를 가지며, VGG-16과 VGG-19의 두 가지 주요 변형 존재
- 작은 크기(3x3)의 필터를 사용하여 깊이를 증가시키는 것이 핵심 아이디어

구조

- 입력층: 224x224x3 크기의 RGB 이미지, 이미지를 모델에 입력
- 합성곱층: 3x3, 스트라이드 1, 패딩 1, ReLU
- 풀링층: 맥스 풀링, 2x2 필터, 스트라이드 2, 공간 크기를 줄이고, 중요한 특징을 보존
- 완전 연결 층: 4096개 뉴런, ReLU, 드롭아웃 0.5
- 출력층: 1000개 뉴런 (ImageNet의 클래스 수), Softmax

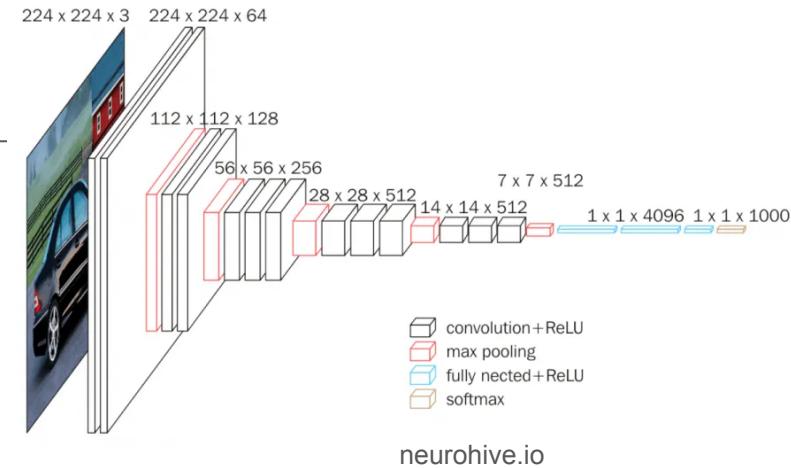
주요모델

VGGNet - 16

- VGG-16은 13개의 합성곱 층과 3개의 완전 연결 층으로 구성
- VGG구조가 반복되는 구조
- 블록으로 구분

VGG-16 구조

- 입력층: 224x224x3 크기의 RGB 이미지, 이미지를 모델에 입력
- Block1: 3x3, 64개 필터를 갖는 합성곱층 2개, 2x2, 스트라이드 2의 최대풀링층 1개
- Block2: 3x3, 128개 필터를 갖는 합성곱층 2개, 2x2, 스트라이드 2의 최대풀링층 1개
- Block3: 3x3, 256개 필터를 갖는 합성곱층 3개, 2x2, 스트라이드 2의 최대풀링층 1개
- Block4: 3x3, 512개 필터를 갖는 합성곱층 3개, 2x2, 스트라이드 2의 최대풀링층 1개
- Block5: 3x3, 512개 필터를 갖는 합성곱층 3개, 2x2, 스트라이드 2의 최대풀링층 1개
- 완전연결층: 4096개의 뉴런, ReLU, 드롭아웃 0.5의 완전연결층 2개, 1000개의 뉴런, Softmax를 갖는 완전연결층 1개



neurohive.io

주요모델

VGGNet 주요 특징

작은 필터(3x3)

- VGGNet은 3x3 크기의 작은 필터를 사용하여 신경망의 깊이를 증가
- 작은 필터를 여러 번 적용하면 더 큰 수용 영역을 효과적으로 커버할 수 있으며, 더 복잡한 패턴을 학습 가능

심층 구조

- VGGNet은 매우 깊은 신경망 구조를 통해 고차원 특징을 학습
- 깊은 신경망은 더 복잡한 특징을 추출하고, 모델의 표현력을 향상시킴

ReLU 활성화 함수

- 각 합성곱 층 뒤에 ReLU 활성화 함수를 사용하여 비선형성을 추가

드롭아웃

- 완전 연결 층에서 드롭아웃을 사용하여 과적합 방지

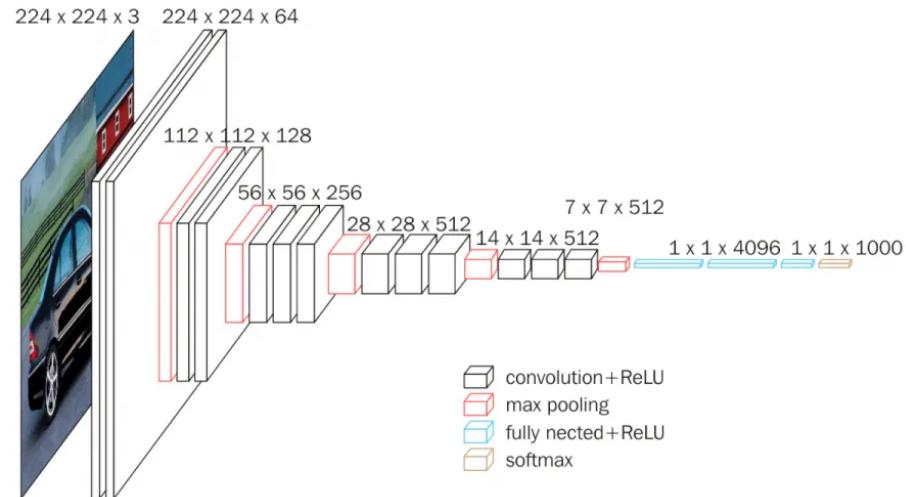
주요모델

VGGNet의 장점

- 작은 필터를 반복적으로 사용하여 모델의 구조를 단순화하면서도 깊이를 증가
- ImageNet과 같은 대규모 데이터셋에서 높은 정확도를 달성
- AlexNet에 비해 더 깊은 구조를 사용하여 성능을 향상

VGGNet의 단점

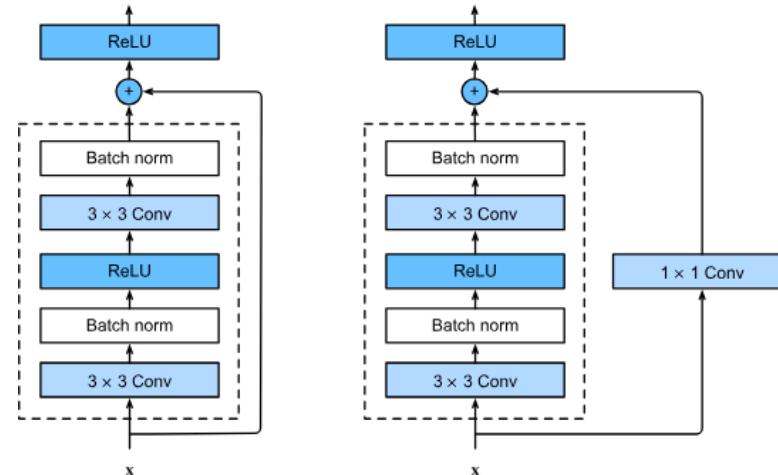
- 많은 층과 필터를 사용하므로 계산 비용이 높음
- 대규모 데이터를 처리하기 위해 많은 메모리가 필요
- 여러 층을 포함하므로 모델의 설계와 학습이 복잡함



주요모델

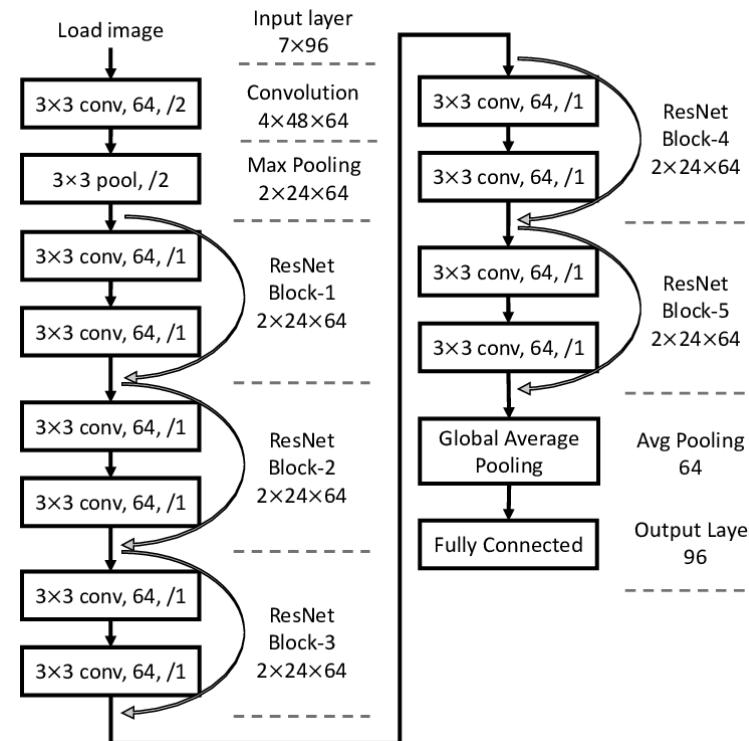
ResNet(Residual Networks)

- ResNet은 매우 깊은 신경망을 학습할 때 발생하는 기울기 소실 문제를 해결하기 위해 잔차 연결을 도입
- 수백 개의 층을 가진 매우 깊은 신경망을 효과적으로 학습 가능하게 함
- 기본적으로 VGG-19의 구조를 이어받음



주요모델

ResNet-12



주요모델

ResNet 주요 특징

잔차 연결(Residual Block)

- 각 층이 학습할 함수를 잔차로 표현하여, 입력을 출력에 더하는 스kip 연결을 도입
- 신경망의 각 층이 학습해야 할 정보를 원본 입력 값과의 차이(잔차)로 표현하는 방식
- 기울기 소실 문제를 완화하고, 더 깊은 신경망을 효과적으로 학습가능

병목 블록

- 1×1 합성곱 층을 사용하여 차원을 축소 및 복원하여 연산량을 줄이는 구조
- 연산 효율성을 높이고, 더 깊은 네트워크 설계가 가능해짐

배치 정규화

- 각 합성곱 층 뒤에 배치 정규화를 적용하여 학습을 안정화시키고 속도도 향상시킴

심층 구조

- 매우 깊은 신경망을 통해 고차원 특징을 학습해서 더 복잡한 특징을 추출하고, 모델의 표현력을 향상시킴

주요모델

ResNet의 장점

- 잔차 연결을 통해 기울기 소실 문제를 완화하고, 깊은 신경망을 효과적으로 학습할 수 있음
- ImageNet과 같은 대규모 데이터셋에서 높은 정확도를 달성
- 병목 블록을 사용하여 연산 효율성을 높이고, 더 깊은 네트워크를 설계할 수 있음

ResNet의 단점

- 많은 층과 필터를 사용하므로 계산 비용이 높음
- 대규모 데이터를 처리하기 위해 많은 메모리가 필요
- 여러 층을 포함하므로 모델의 설계와 학습이 복잡함

주요모델

SSD(Single Shot MultiBox Detector)

- 객체 검출 모델
- 단일 신경망 패스로 여러 객체의 위치와 클래스를 동시에 예측
- VGG16의 구조를 이어받음
- 최근에는 ResNet 모델 구조로 대체되는 경우도 있음

주요개념

싱글 샷 검출(Single Shot Detection)

- 이미지 전체를 한 번에 처리하여 객체의 위치와 클래스를 동시에 예측하고 속도가 매우 빠름

멀티스케일 특징 맵(Multi-scale Feature Maps)

- 다양한 크기의 특징 맵을 사용하여 객체를 검출

디폴트 박스(Default Boxes)

- 다양한 크기와 종횡비를 가진 박스를 미리 정의하여 각 위치에서 객체를 검출

주요모델

SSD 구조

기본 네트워크(Base Network)

- VGG-16 또는 ResNet과 같은 사전 학습된 모델을 사용
- 기본 네트워크의 마지막 몇 층은 제거되고, 합성곱 층을 추가해서 사용

추가 합성곱 층

- 기본 네트워크 위에 여러 합성곱 층을 추가하여 다양한 크기의 특징 맵을 생성

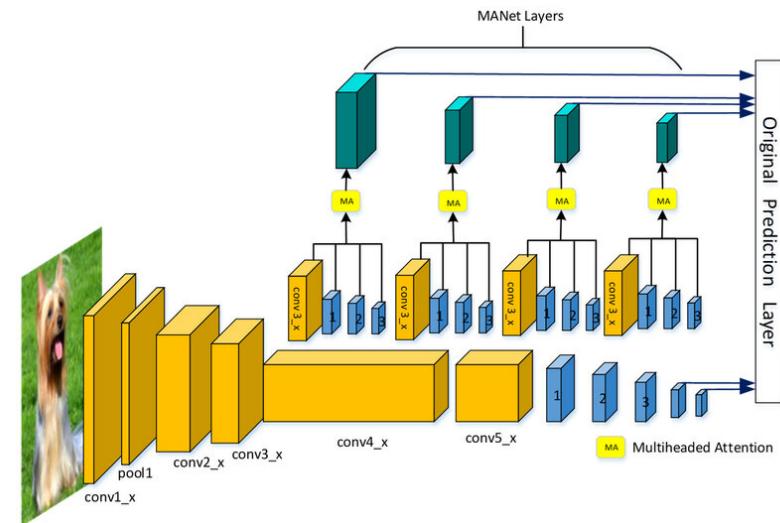
멀티스케일 특징 맵

- 다양한 크기의 특징 맵을 통해 객체를 검출

주요모델

SSD Flow

1. 입력: 원본 이미지가 모델에 입력
2. 기본 네트워크: VGG-16을 통해 초기 특징 추출
3. 추가 합성곱 층: 기본 네트워크에 추가된 합성곱 층을 통해 다양한 크기의 특징 맵을 생성
4. 디폴트 박스 생성: 각 특징 맵의 각 셀 위치에서 다양한 크기와 종횡비를 가진 디폴트 박스를 생성
5. 예측: 각 디폴트 박스에서 위치 오프셋과 클래스 확률을 예측
6. 손실 계산: 예측된 위치와 클래스 확률을 실제 값과 비교
7. 학습: 손실을 최소화하도록 모델 학습
8. 객체 검출: 새로운 이미지에서 객체를 검출



주요모델

SSD의 장점

- 단일 패스로 객체 검출을 수행하므로 매우 빠름
- 다양한 크기의 특징 맵을 사용하여 다양한 크기의 객체를 효과적으로 검출
- 단일 신경망으로 모든 예측을 수행하므로 구현이 상대적으로 간단

SSD의 단점

- 작은 객체를 검출하는 데 어려움을 겪을 수 있음
- 다양한 크기의 특징 맵과 많은 디폴트 박스를 사용하므로 메모리 사용량이 많을 수 있음

이론 실습

colab(PyTorch tutorial 참고):

<https://colab.research.google.com/drive/1Q3TjTuGdcgGZQpRDpHM2t36Sn4zOTNpa?usp=sharing>

실습 과제 (SSD)

1. SSD의 기본 구조 이해: SSD의 기본 원리와 구조를 이해하기
2. 사전 학습된 모델 사용: 사전 학습된 SSD 모델을 사용하여 이미지에서 객체를 검출하기
3. 실제 데이터셋 사용: COCO와 같은 실제 데이터셋을 사용하여 SSD 모델을 학습시키고 평가하기
4. 결과 시각화: 검출된 객체를 이미지에 시각화하여 결과를 확인하기

실습 진행