3-1. 딥러닝 & 영상모델

목차

1. 이미지 딥러닝 모델

학습 목표

- 이미지 기반 학습 모델의 구조와 작동 방식을 이해한다.
- 대표적 CNN 기반 이미지 모델의 변천사를 배운다
- 이미지 분류 문제에서 실습을 통해 주요 이미지 모델을 적용해본다.

학습 시작(OverView)

- CNN 모델의 기본 구조는 무엇이며, 왜 이미지 과업에서 중요한가?
    - CNN 모델의 기원 및 모델 주요 모듈 소개
- CNN의 단위 연산과 디자인 철학은 어떻게 발전했는가?
    - 계층 깊이, 스케일, 학습 방법, 경량화
- CNN 기반 주요 모델은 어떻게 변화해왔고, 왜 중요한가?
    - 깊이와 효율성을 동시에 높이는 방향으로 개선

1. CNN 살펴보기

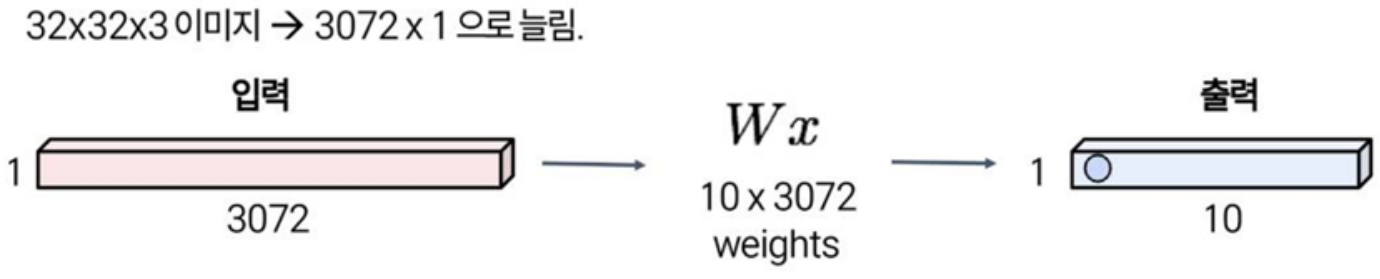
1-1. CNN vs FCN

FCN 모델

- 완전 연결층(Fully-Connected Layer)
- 입력을 출력으로 변환하는 모듈

FCN 동작 방법

- 입력 : (32x32×3) 이미지(HWC) → 1차원 벡터로 변환 (3072 x 1) → FCN 입력
- 출력 : (10x1) 의 1차원 벡터
- 모델 상수 : 입력 → 출력으로 모델 상수 w = (10 × 3072) 가 된다. (FCN 이므로)
    - 입력의 모든 값이 출력에 미치는 영향

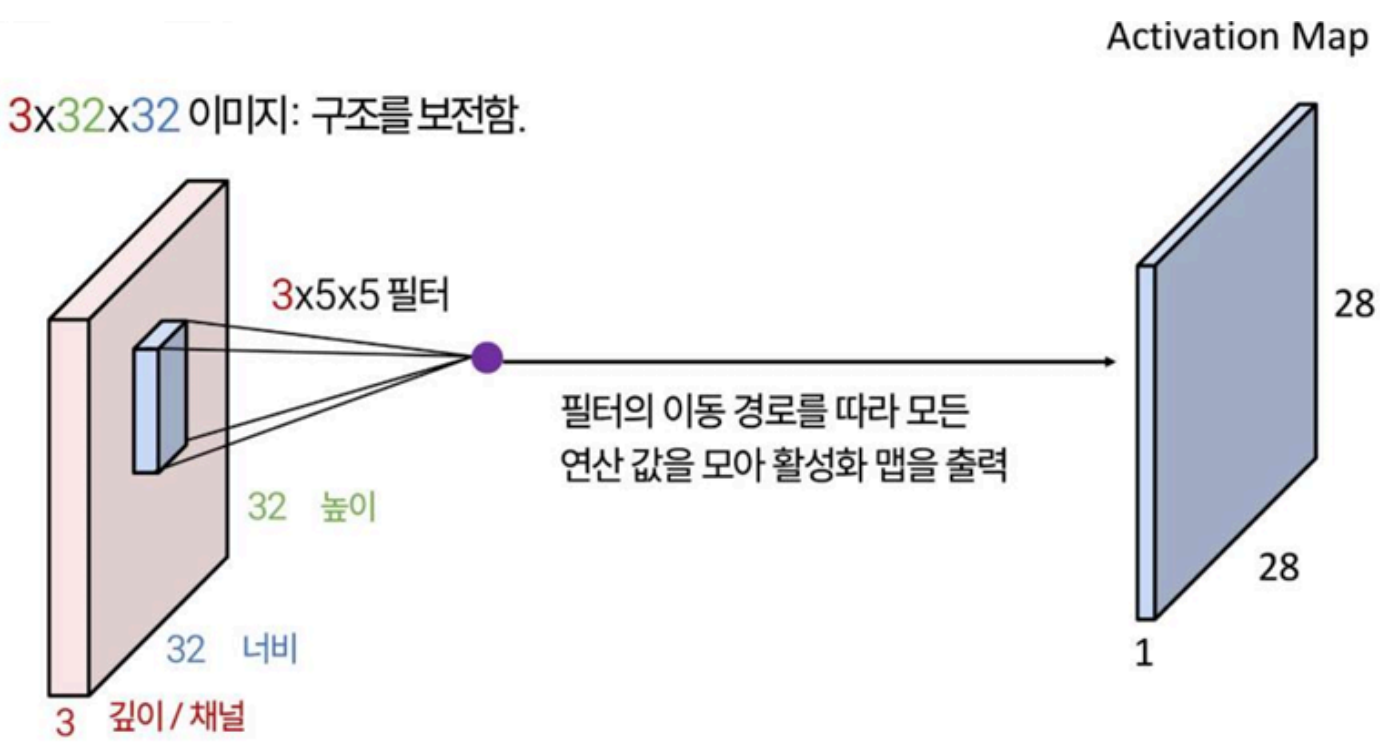


CNN 모델

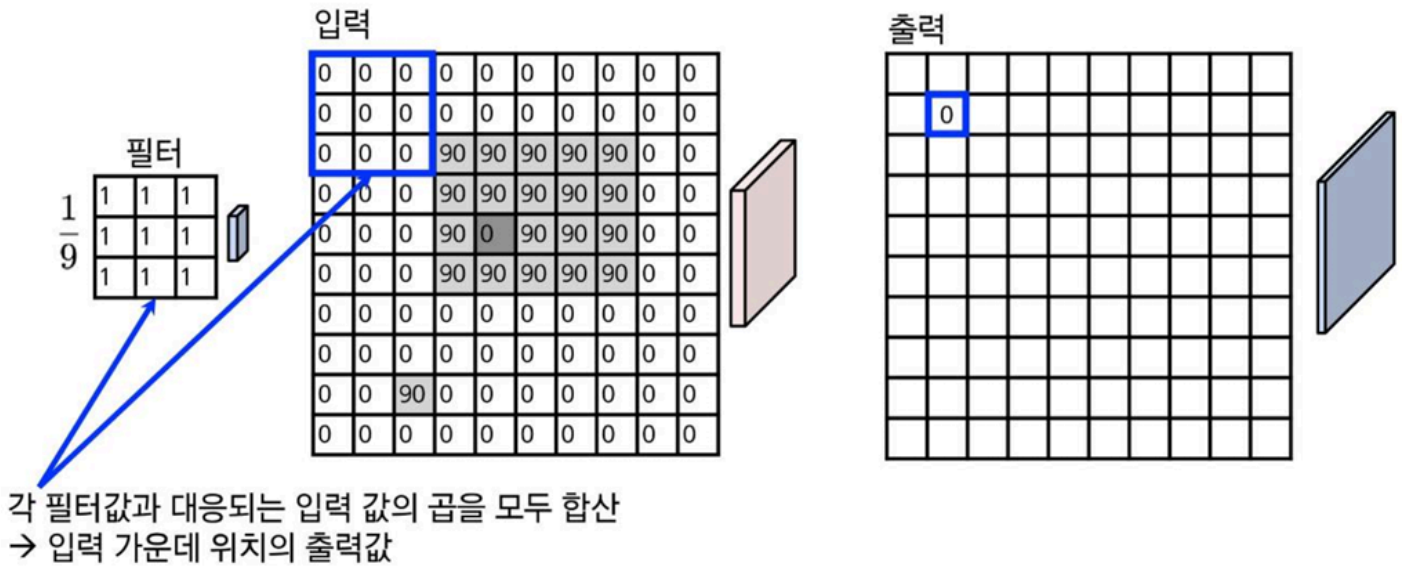
- 합성곱 레이어 (Convolution Layer)
- 입력 이미지를 필터와 연산하여 특징 맵 (Feature map)을 뽑아내는 모듈
- 1차원 구조로 변환하는 FCN과 달리, 3차원 구조를 그대로 보존하면서 연산 (선형 연산)

CNN 동작 방법

- 입력 : (3x32×32) 이미지 (CHW) → 차원 변환 없이 CNN 입력
- 필터의 존재 : Convolution Filter, Box filter (3×5×5), 필터를 이미지 상에서 이동하면서, 내적을 반복 수행
    - 연산 : W^T x X + B
        - T : 필터가 덮는 이미지 영역
        - W : 필터의 가중치
        - B : 편향 (보정값)
- 출력 : 필터를 거쳐 내적으로 수행한 모든 값이 출력으로 제공 (1 x 28 × 28)

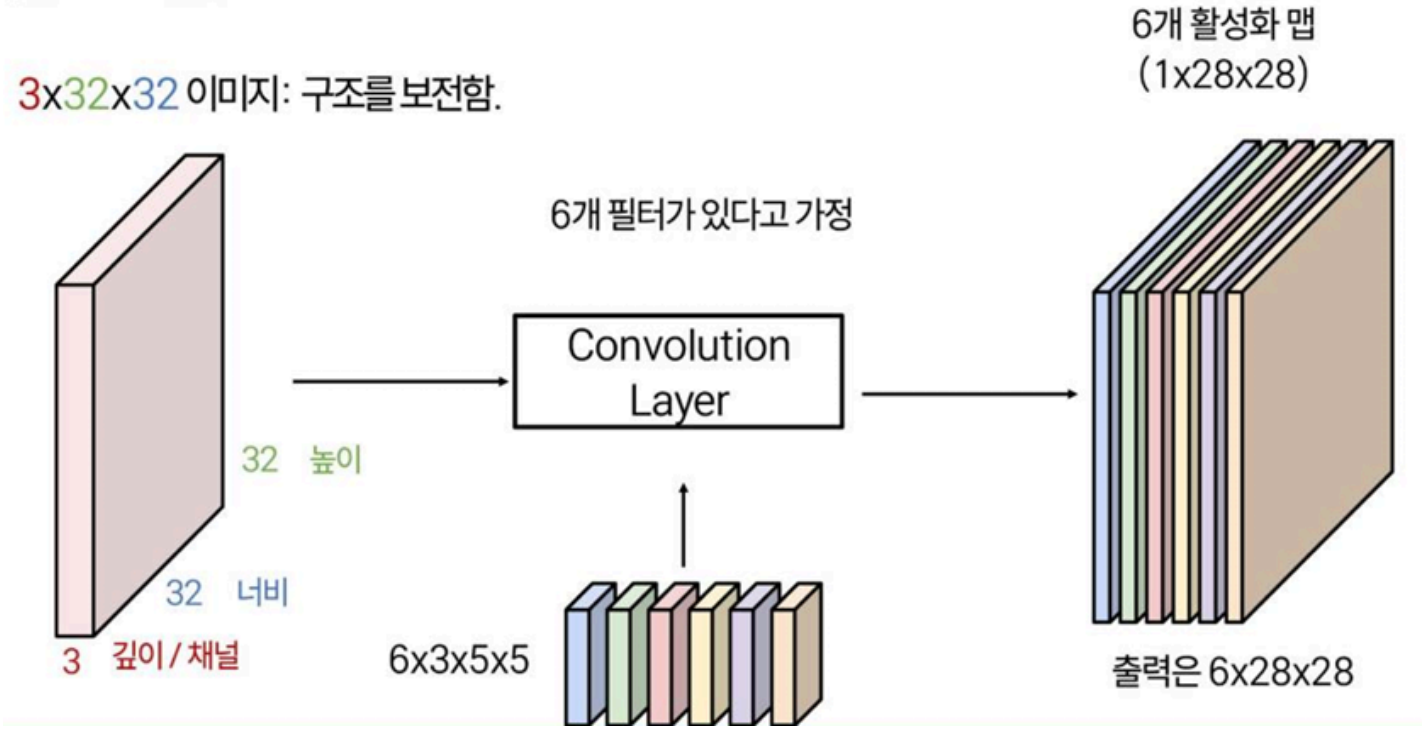


이미지를 필터가 이동하며 내적을 수행한다.



필터 1개를 쓰면 특징 맵이 1개 나온다.

여러 특징을 구하고 싶다면? → 여러 필터를 쓴다!

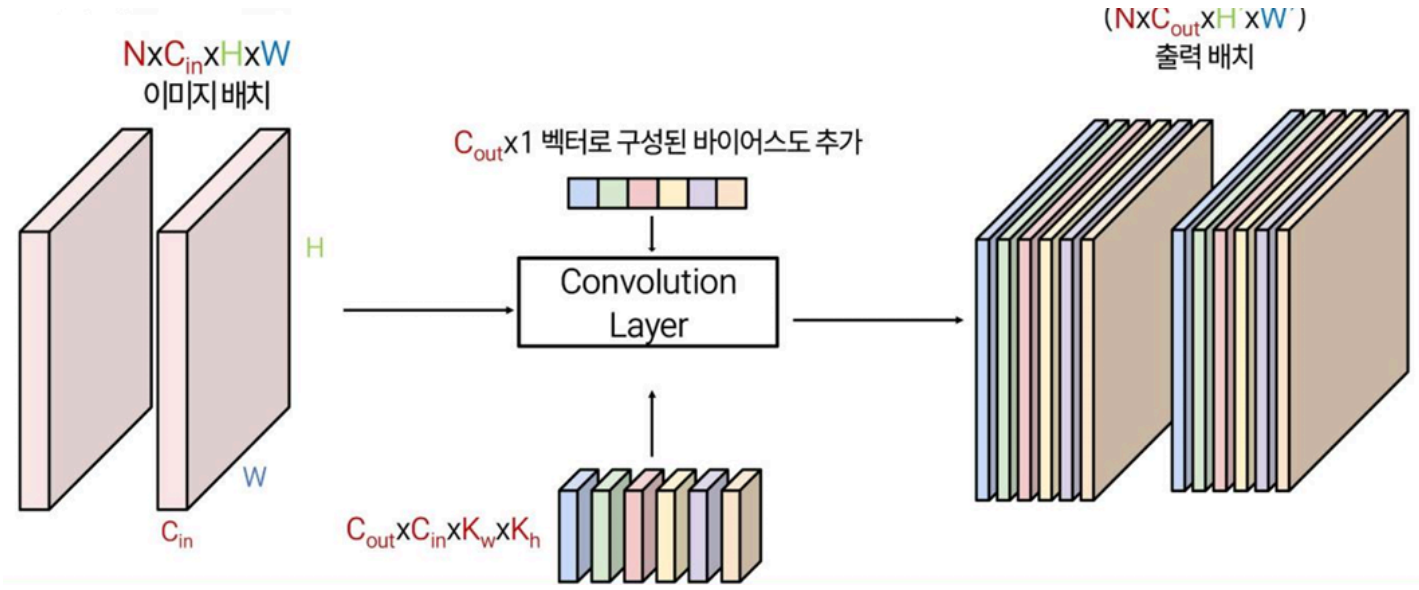


- 출력 해상도 = 입력 해상도 - 필터 해상도 + 1
  - ex) 32 - 5 + 1 = 28

만약, 입출력 해상도를 유지하고 싶다면?

→ 출력값 중 정의되지 않은 경우, 0 또는 가장 가까운 출력값으로 대체 (패딩)

- 보정 값을 위해 bias layer 도 추가
- 입력 이미지 개수 N으로 증가
- 입력 : N x Cin x H x W
- bias layer : Cout x 1
- convolution filter : Cout x Cin x Kw x Kh
- 출력 : N x Cout x H x W

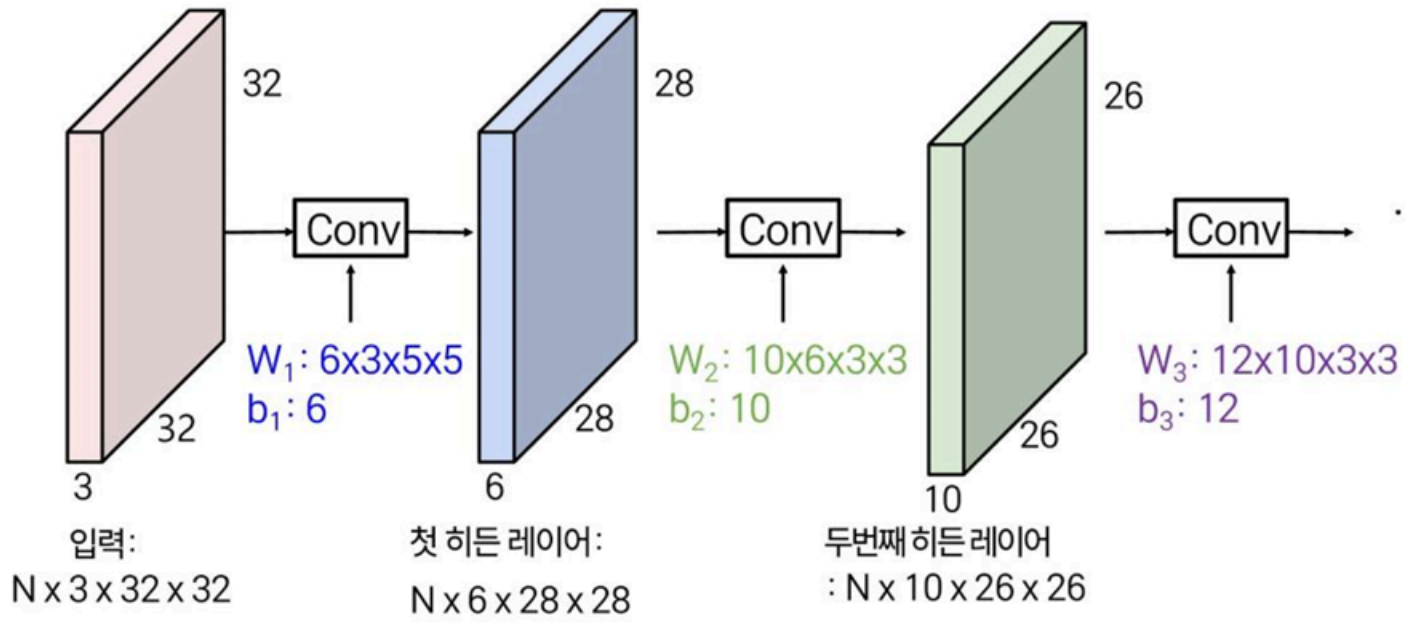


1-2. 모델 구조

CNN 모델을 중첩 시킨다면?

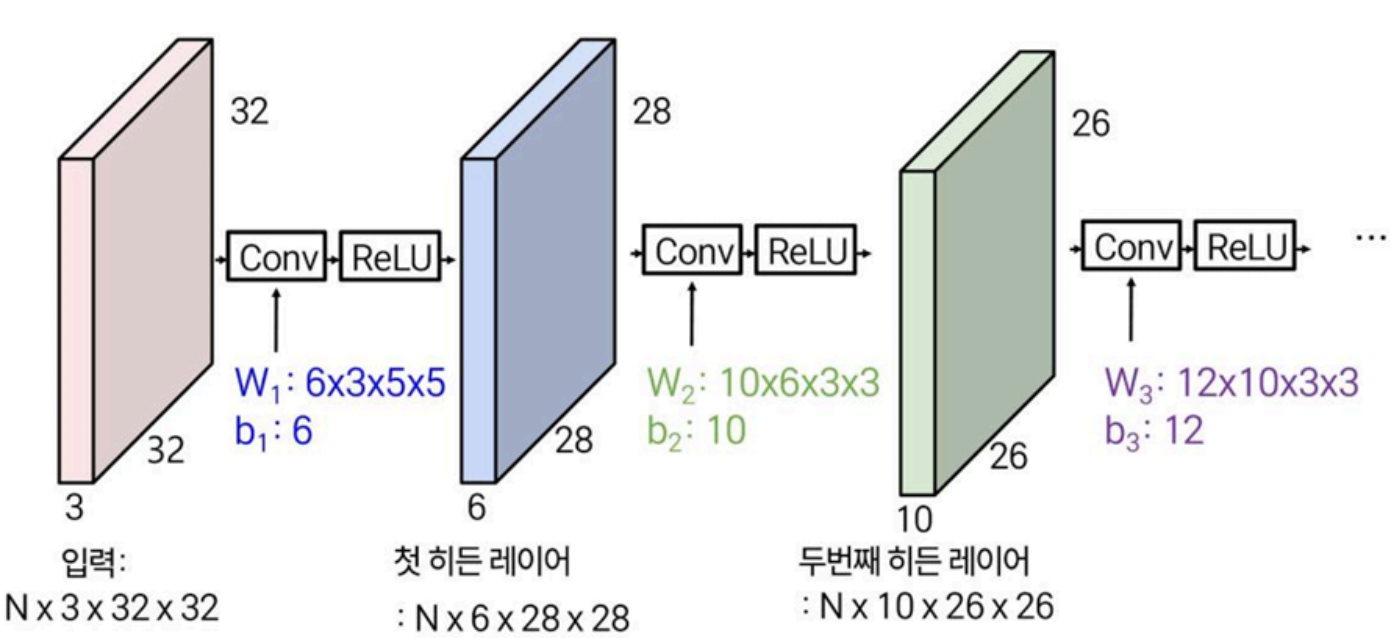
N개의 이미지가 Conv1, Conv2 를 거쳐 차례대로 특징 맵을 추출하는 과정

→ 여러 레이어를 쌓을 수록 고차원적 특징을 이해할 수 있음



활성화 함수의 도입

- 활성화 함수를 넣어, 비선형성 도입



CNN에서 필터의 의미

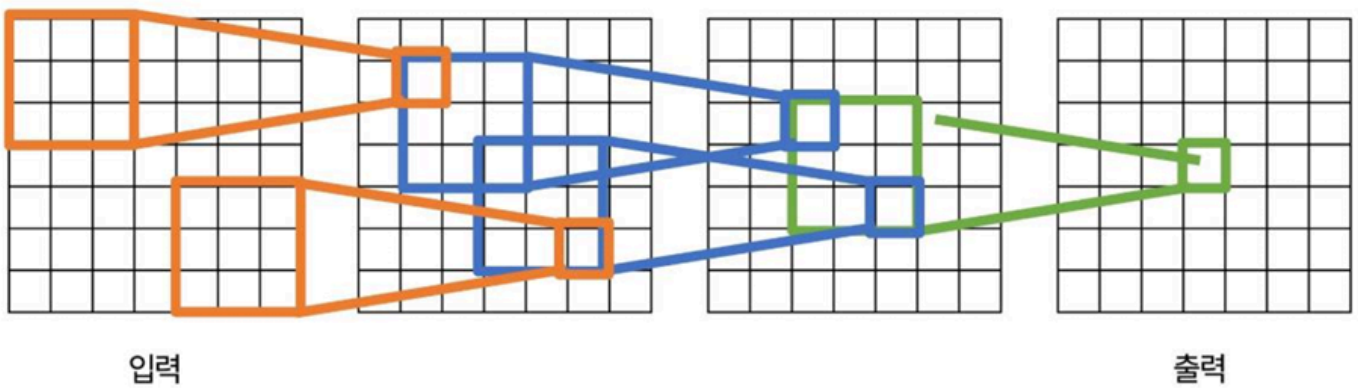
필터 시각화

- 학습된 필터 시각화를 통해 모델이 학습한 정보를 이해 가능
  - 선형모델 : 각 카테고리마다 전형적인 모습 하나를 필터로 학습
    - 그 클래스 전체를 대표하는 평균 템플릿 (ex, bird, plane, car)
  - FCN : 다양한 템플릿 제공하지만, 지역적 패턴 잡는 능력은 제한적
  - CNN : 계층별로 이미지의 세밀한 지역적 템플릿 학습 (엣지, 색상 등)

CNN의 수용영역 (Receptive Field)

- 중첩과 수용영역

  수용영역 : CNN이 이미지를 처리하면서 한번에 볼 수 있는 영역의 크기 (네트워크의 시야)
  - 네트워크가 깊어질 수록 수용영역도 넓어짐 → 폭 넓은 맥락 이해 가능
- CNN의 레이어는 이미지 작은 부분인 지역 정보를 추출하는 데 유리
- 이미지 전체의 의미 파악이 필요
- 지역 정보 + 이미지 전체의 맥락을 이해/활용해야 한다.



문제 : 고해상도 이미지 처리할 경우, 많은 레이어를 통과하며 특징을 잡아야 한다.

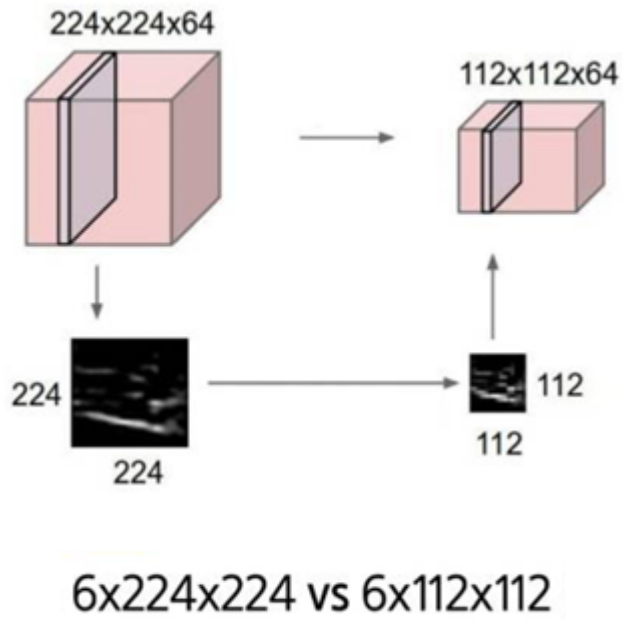
→ 많은 비용, 연산 필요

→ 해결 : 입력 사이즈를 줄여서 모델에 입력하자.

CNN의 풀링 (Pooling)

- 해상도 줄이는 연산 → 값을 버림
- 효율적 연산 및 위치 변화의 강건성 확보
- CNN 레이어의 출력을 줄여 연산 효율성 화고 : 입력 크기가 줄면 CNN의 연산량이 크게 줄어듦

ex) 2×2 풀링 으로 입력의 해상도를 1/2 로 줄이면, 4배의 성능을 높임

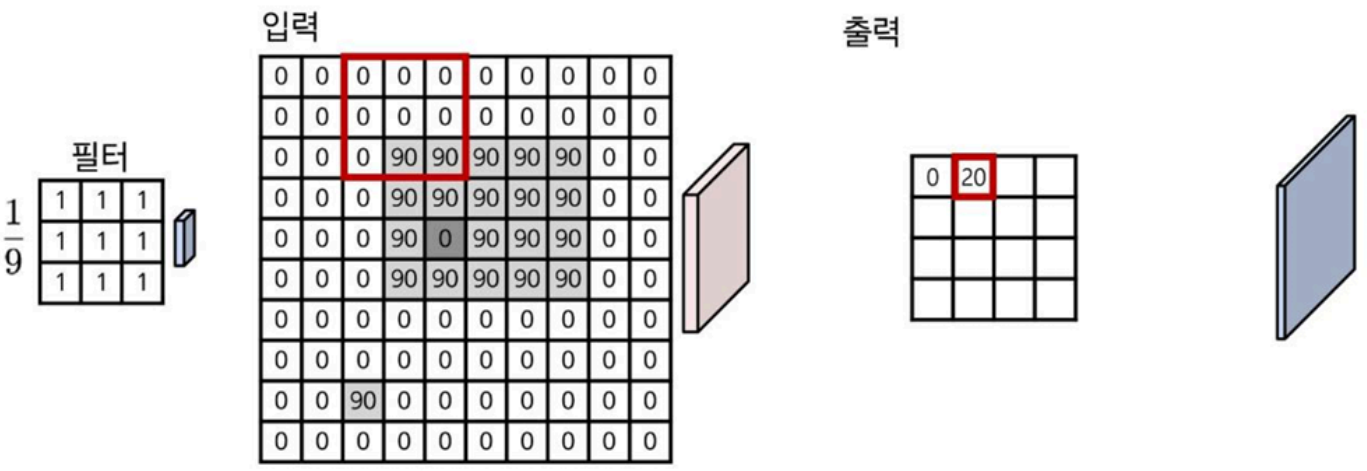


- 위치 변화 강건성 : 입력 내 객체 위차가 다소 변해도 동일한 출력 제공
  - 저해상도 정보에 근거하여 작업을 수행하므로, 몇 화소 이동은 무시됨

- 맥스 풀링 : 정해진 필터(커널) 사이즈로 이미지를 나누어 각 영역 내 가장 큰 값을 선택하는 연산
  - average 풀링 등 다양하다

CNN의 스트라이드 합성곱

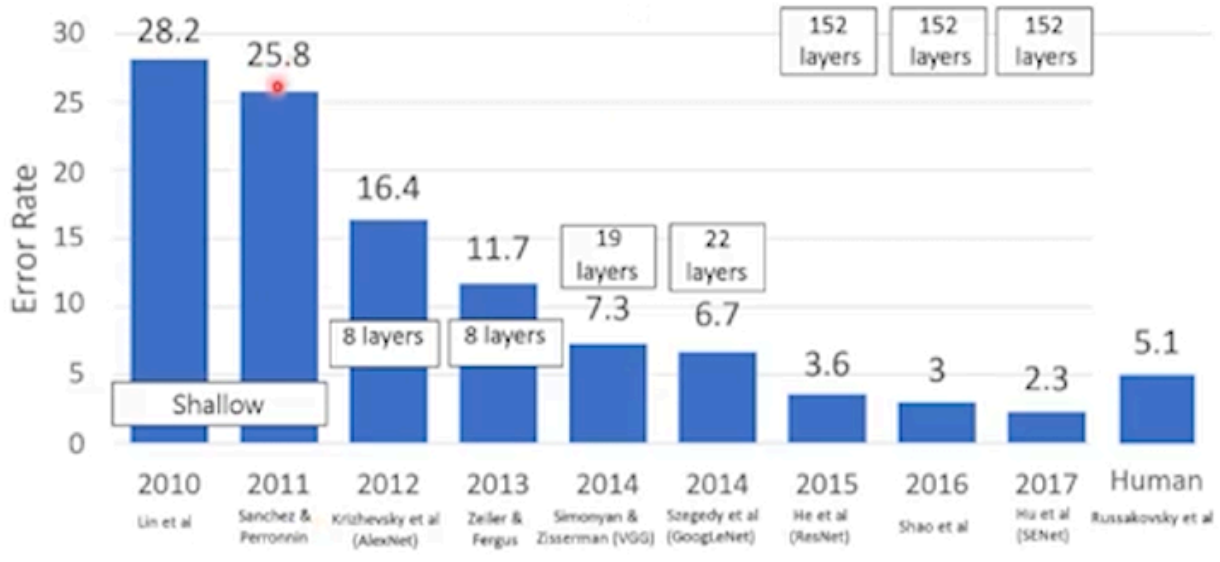
- 풀링의 한계 개선
- 일반 합성곱 : 필터를 1칸 씩 이동
- 스트라이드 합성곱 : 필터를 스트라이드 값(S칸)만큼 이동한 후 출력 연산
  - 풀링처럼 입력 해상도를 줄임 (연산 효율과 위치 강건성 확보)
- 풀링 vs 스트라이드 합성곱
  - 풀링은 학습 상수가 없지만, 스트라이드는 필터(커널)을 동시에 학습
  - 스트라이드 합성곱은 해상도 저하로 인한 정보 손실이 적고, 풀링 + 합성곱을 하나의 레이어로 대체



2. CNN 기반 모델 변천사

2010~2017년까지의 CNN 모델

- 세로축 : 에러율 (Error Rate, %), 낮을수록 성능이 좋은 모델
- 가로축 : (연도 & 모델명)



2-1. AlexNet

- CNN의 가장 고전 방법
- 5개의 합성곱 계층과 3개의 FCN 계층으로 구성된 8계층 CNN 모델

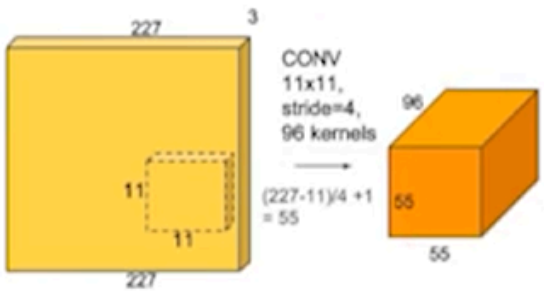
모델 구조 특징

- 5개 합성곱 레이어
- 맥스 풀링
- 3개 연결층 레이어
- ReLU 비선형 활성화

입출력

- 입력 : 이미지 / 3 x 227 × 227 해상도
- 출력 : 레이블 / 1 x 1024 벡터
- 학습 시 정답은 One-hot vector
- 모델 예측 값은 0~1 사이 확률 값
- ImageNet 기준, 각 클래스에 해당할 확률

출력 사이즈 계산법



	입력 사이즈		계층				연산비용		
Layer (층)	C (입력 채널)	H/W (세로/가로)	F (필터 갯수)	K (커널 사이즈)	S (Stride)	P (Padding)	메모리 (KB)	상수 (k)	연산 (FLOPs)
Conv1	3	227	96	11	4	0	1,134	?	?

- 출력 채널 수 = 현재 레이어의 필터(커널) 개수
  - ex) F = 96
- 출력 해상도 W′ = (W-K+2P) / S + 1
  - W : 입력 사이즈

    - K : 커널(필터) 사이즈
    - P : Padding
    - S : Stride
    - W′ = (227 - 11 + 2*0) / 4 + 1 = 55
- 출력 활성 메모리 (KB) = C * H′ * W′ = 96 * 55 * 55 = 290,400KB
- 출력 값 메모리 사용 4Byte → 4 * 출력활성메모리 = 1,134KB
- 상수 k (파라미터수)
    - k = 출력 채널수 x 입력채널수 x K^2 (Filter Weight) + 출력 채널 수 (Bias Weight) = 96 X 3 X 11 X 11 + 96 = 34,944 (34.9 K)
- 연산량 (FLOPs)
    - 총 연산 = 출력 원소수 x 출력 원소별 연산 X 2 (덧셈의 결과는 적당히 2배)
    - (H′ x W′) x (Cin x K x K x Cout) x 2 = (55×55) x (3 × 11× 11 × 96) x 2 = 105.4M FLOPs
- AlexNet의 Memory
    - 메모리 사용은 초기 레이어에 집중
- AlexNet의 모델 상수
    - FC 레이어에 집중
    - 합성곱 레이어가 모델 상수에 효율
- AlexNet의 연산
    - 합성곱에서 주요 발생

2-2. VGGNet

- 5개의 합성곱 블록 + 맥스풀링 구조
- 새로운 설계 철학 제시 : 작고 단순한 필터를 깊게 쌓으면 성능이 올라간다

모델 구조 특징

- 5개 합성곱 블록
- 각 블록당 맥스 풀링
- 3개 연결층 레이어
- ReLU 비선형 활성화
- 총 16개의 합성곱/연결층 레이어
- 최종 소프트맥스 수행

입출력

- 입력 : 이미지 / 3 x 224 × 224 해상도
- 출력 : 레이블 / 1 x 1024 벡

VGG의 디자인 룰

- 3×3 합성곱 / 스트라이드1 / 패딩 1 반복 적용
- 맥스풀링 (2x2) 로 크기 줄이기
- 풀링 후 채널 크기 2배 적용
    - 3×3 합성곱 2개를 연달아 적용하면 5x5 1개와 동일한 결과, 더 적은 연산량 확보

AlexNet과의 차이점

- AlexNet은 레이어별 연산량 차이가 크다
- VGGNet은 해상도를 줄일 때마다 채널을 늘려 레이어별 연산량 차이를 줄

AlexNet vs VGGNet

- VGG는 거대한 모델
- 메모리 25배
- 모델 상수 2.3
- 연산량 19.4배

모델특징

- 단순함 + 깊이의 강력한 성능
- 단순 설계로 모델 해석에 이상적
- 특징 추출기, 전이학습에 강력한 베이스라인
    - 파라미터가 많고 연산량 요구가 매우 큼

2-3. ResNet

합성곱 블록(VGG 유사) + 잔차(Residual) 블록

모델 구조 특징 (ResNet-50 기준)

- 합성곱 블록 (VGG 유사) 과 잔차(Residual) 블록
- 잔차 블록은 블록 입력을 그대로 출력에 더해주는 지름길 연결이 특징
- Inception 모델에서 차원 축소로 활용된 1x1 합성곱을 적용, 연산 효율 개선

입출력

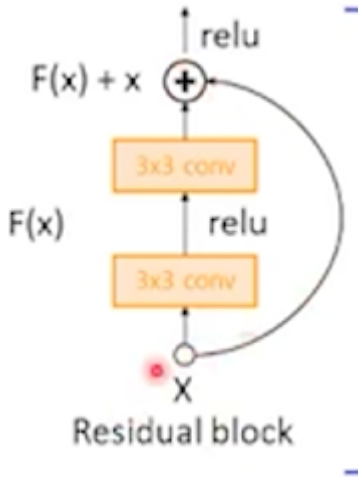
- 입력 : 이미지 / 3x224×224 해상도
- 출력 : 텍스트 / 1x1024

ResNet의 등장 배경

- 배치 정규화 방식으로 레이어 10개 이상 학습 가능
    - 깊다고 무조건 더 잘 학습되는 것이 아님 (Degradation Promblem)
    - 단순히 레이어를 깊게하면 오히려 성능이 떨어짐

ResNet의 아이디어

- 배치 정규화 방식
- 작은 모델의 성능은 최소한 누릴 수 있도록 작은 레이어의 추정값을 그대로 후속 레이어에 제공하여 최소한 작은 레이어 수준의 성능 보장
  → Residual Block의 핵심 아이디어
    - 바로 이전 단계에서 나온 결과를 보존해서 이번 레이어를 통과한 결과로 넘김



입력 X의 경로가 두개

1. 변환 경로 F(x)

   → Conv → ReLU → Conv → ReLU → 출력 F(x)

2. Shortcut 경로 (Identity mapping, 잔차연결, skip connection 이름이 많음)

   → 입력 X를 그대로 전달

마지막에 두 값을 더함 : Output = F(x) + x

만약 학습이 잘 안되어도 (F(x)==0) 최소 성능인 x가 보장

ResNet 잔차블록 버전

- 일반 잔차 블록 Vs 보틀넥 잔차블록
- 연산 효율을 위해 보틀넥 잔차 블록 도입 (차원 축소, 확대) , 입력 값과 출력 값의 크기가 같아야 함

Stem 구조

- 기존 모델의 효율성 레시피를 활용
  - 스템 구조를 모델 초기에 도입, 입구 데이터 해상도를 1/4로 줄이고, 이 후 잔차 블록 적용
  - 여러 개의 FC 레이어를 제거하고 (GoogleLeNet에서 제안) 전역 평균 풀링을 사용

    → 마지막 1개 FC만 적용

일반 잔차 블록 사용한 모델 : ResNet-18, ResNet-34

보틀넥 잔차 블록 : ResNet-50 (지금도 가장 많이 활용 됨, CNN 대표 모델)

ResNet 학습 레시피

- 모든 Conv 레이어 적용 시 배치 정규화 수행
- Xavier 초기화(He 초기화) 적용 / 그래디언트 소실 문제 다룸
- SGD + 모멘텀 (0.9) 사용
- 학습 비율 0.1 적용, 단 Val 오류가 정체되면 1/10으로 비율 축소
- 미니 배치 사이즈는 256 사용
- Weight Decay : 1e^(-5)
- 드롭아웃 미사용

깊은 모델 학습을 위한 중요한 전진 (100+ 레이어 학습 가능)

깊은 모델의 강력함을 다시 확인

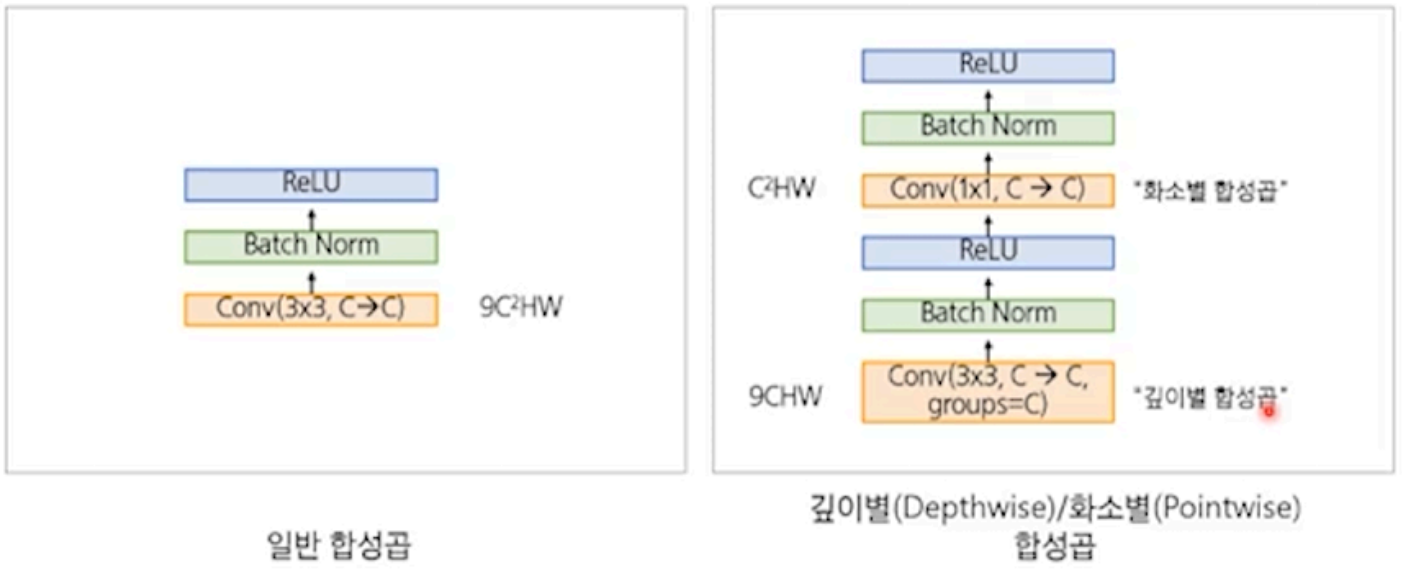
제안 당시 모든 벤치마크에서 최고 성능 달성

지금도 CNN 기반 구조중 가장 활발하게 사용

2-4. MobileNet

목표 : 모바일/임베디드 환경에서 구동가능

- 기존 합성곱은 공간(HxW) 와 채널(C)를 동시 처리하여 연산량이 과도하게 요구
- MobileNet의 핵심 아이디어 : 공간과 채널을 두 단계로 분리하여 처리
  - 깊이별 합성곱 : 각 채널별 독립적 3x3 합성곱 수행
  - 화소별 합성곱 : 1x1 합성곱을 채널 방향으로 적용
- 효과
  - 연산량 기존 대비 9배 가량 절감
  - 모델 상수 대폭 감소



이 후 모델에 영향력

- 가볍고 빠른 모델로 엣지 디바이스에서 딥러닝 모델 실시간 실행 가능함을 보임
- 효율형 모델 구조의 표준 제시 : 깊이별 합성곱과 보틀넥 구조가 이후 경량 모델에서 지속적으로 채택

2. 다양한 신경망 모델

학습 목표

- CNN의 한계를 이해하고, 이미지 인식에서 왜 새로운 접근이 필요한지 확인한다.
- RNN, Attention 기반 Transformer의 핵심 아이디어와 구조를 파악하고, 각 모델이 해결하는 문제를 이해한다.
- 모델 별 설계 철학과 특징을 비교하며, 데이터 유형 (이미지, 시퀀스, 긴 맥락의 중요성) 에 따른 적절한 모델을 선택할 수 있다.

학습시작(Overview)

- 왜 CNN 만으로는 부족할까?
  - 순서가 중요한 데이터 처리 불가
  - 긴 거리 의존성 부족
  - 픽셀 단위 예측 혹은 생성에 부적합
- 시퀀스 데이터 처리에 적절한 방법이 무엇일까? RNN!
  - RNN의 기본 아이디어
  - RNN의 한계
- 긴 거리 의존성을 확보할 방법이 없을까? Attention! ViT!

    - 어텐션 메커니즘 개념
    - 비전 트랜스포머 (ViT)

1. CNN 한계

CNN구조의 장점

- 합성곱 구조의 재조명
    - 지역적 특징 추출에 특화 : 합성곱의 특징
    - 모델 상수 효율성 우수 : 지역적 특징 추출 시 이미지 블록별 가중치를 공유함
    - 위치 변화와 잡음에 강건 : 풀링/스트라이드 합성곱이 이미지 전체 의미에 집중하게 함
    - 이미지 분류/탐지 등 이미지 기반의 과업에서 표준 모델로 활

CNN의 핵심 한계

- CNN은 데이터의 순서를 무시
    - 한계 : 시퀀스 데이터(텍스트, 음성) 특성을 고려 불가
- 긴 거리 의존성 고려 부족
    - 한계 : CNN은 국소적 (로컬) 특징 추출에 강하지만, 이미지나 문장에서 멀리 떨어진 요소 간의 관계를 잘 학습하지 못함

2. 시퀀스 데이터 처리 : RNN

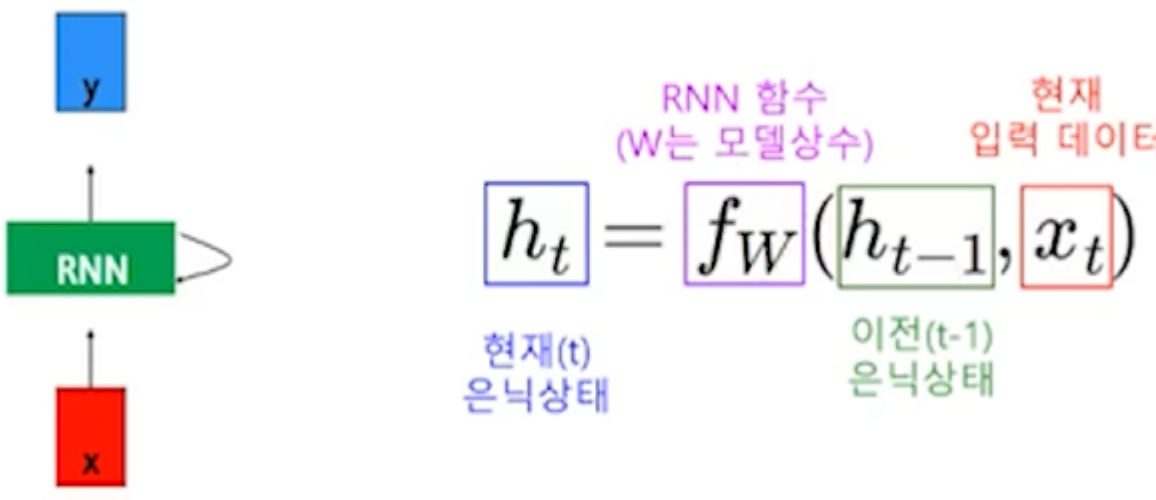
2-1. RNN 처리 방식

데이터 순서를 반영

- RNN 은 순차적 데이터에 적합한 구조
- 이전 단계 데이터를 은닉 상태(hidden state)로 다음 단계에 전달 → 시간/순서의 흐름을 반영
- 문장, 음성신호, 센서 데이터 등 시계열 데이터 처리에 관한 강력한 귀납 편향 제공

시계열 의존성

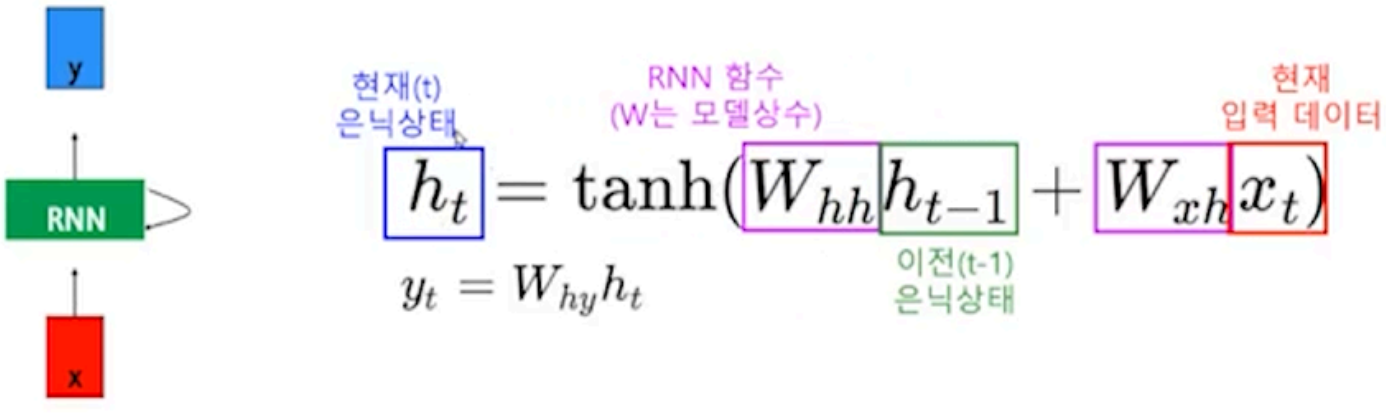
- 데이터의 시계열 특징을 반영하는 모델이나, RNN 상수는 시계열(혹은 순서)에 무관, 동일하게 학습됨.



- 공식 : ht =fw (ht-1 , xt)
    - ht : 현재(t) 은닉 상태
    - fw : RNN 함수 , w는 모델 상수
    - ht-1 : 이전 (t-1) 은닉 상태
    - xt : 현재 입력 데이터

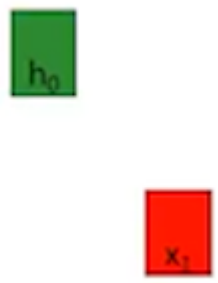
순수 RNN

- 선형 변환, 바이어스를 모델상수/비선형(tanh) 함수 추가로 구성된 단순한 형태



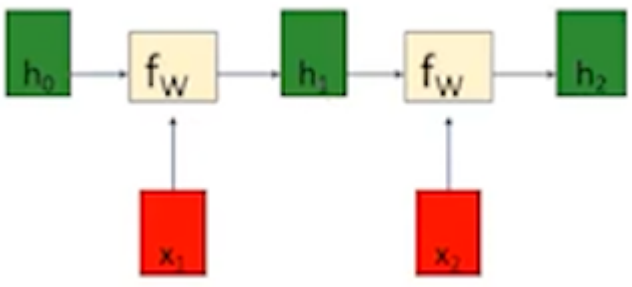
그래프 시각화

- 초기 은닉 상태 : h0
- 입력 : 첫 번째 시계열 데이터/시퀀스 데이

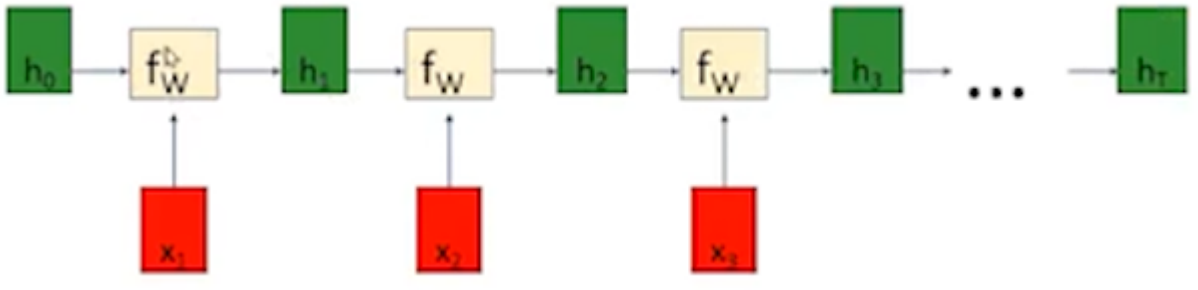


- 동일 함수(fw) 를 모든 시퀀스에 적용

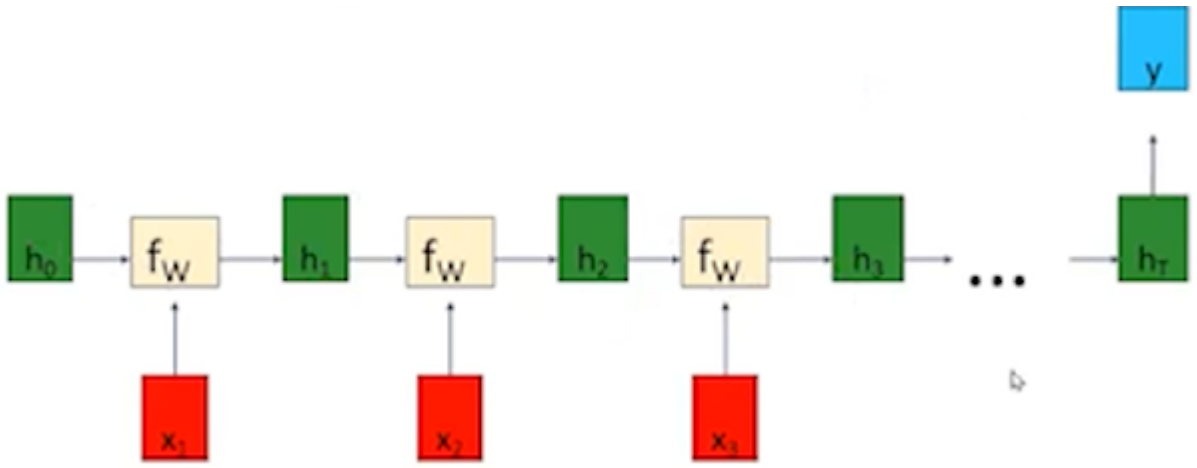
- 반복 적용



- 최종 은닉 상태 → 출력
- W 학습 : 역전파 시 모든 은닉 상태에 발생한 기울기를 더하여 연산

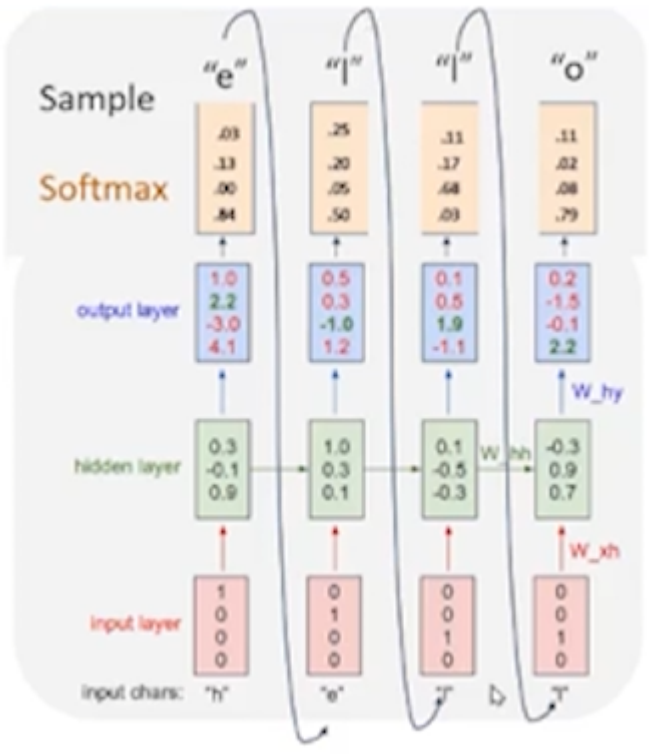


- 시퀀스 입력 → 단일 출력 (y or 각 y1~yn 까지 출력 가능)



- 예시
  - 언어 생성
  - 단위 데이터 : [h, e , l , o]

h 가 순서대로 들어감 → 다음에 나올 y 값으로 e 출력 → e를 입력해서 l → l 입력해서 l → l 입력 해서 o 출력



2-2. 단순 RNN의 한계

기울기 폭발 혹은 소실 발생

- 시퀀스 데이터가 적용되면서 선형 가중치 W가 기울기 계산시, 곱셈으로 적용됨
  - W가 크면 → 기울기가 계속 커짐 (기울기 폭발)
  - W가 작으면 → 기울기가 계속 작아짐 (기울기 소실)

- 해결책 중 하나
  - 기울기 값 강제 절삭
    - 잘못된 기울기를 양산하는 문제
    - 기울기 소실 해결 불가 → 다른 모델 등장

2-3. 대안 모델

LSTM

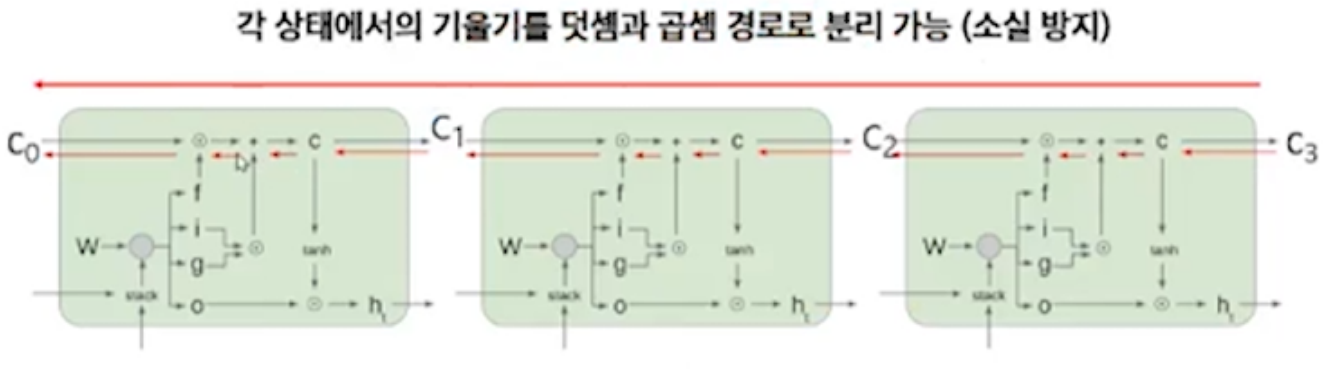
- Long Short Term Memory : 셀 상태, 게이트 유닛으로 구성
  - 셀 상태 (Cell state) : 중요한 정보를 유지/기억하는 역할
  - 게이트 (Gate) : 정보의 흐름을 제어
    - 입력 게이트 (i) : 새로운 정보 저장
    - 망각 게이트 (g) : 과거 정보를 제거
    - 출력 게이트 (o) : 셀 상태를 현재 은닉 상태로 출력
  - 구동 철학 : 긴 시퀀스 데이터에서 필요한 정보는 유지, 불필요한 정보는 지워 안정적 학습 유도

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

셀 상태 $c_t = f \odot c_{t-1} + i \odot g$

은닉 상태 $h_t = o \odot \tanh(c_t)$

- LSTM 은 셀 상태라는 새로운 모듈을 도입, 기울기 전파 시 선형 가중치 곱 이외에 덧셈 경로를 확보



- 한계 : 정보 희석 문제
  - 망각 게이트가 과거 정보를 조금씩 줄임
  - 누적 효과로 정보가 점차 희석되어 오래된 정보가 약화됨

3. 긴거리 의존성 : 어텐션/ViT

3-1. Attention 메커니즘

Attention 메커니즘의 중요 요소

- Query : 질의 대상 (입력 토큰/입력 이미지 패치)
- Key : 특성 (입력 내 기타 토큰)
- Value : 키의 실제 정보
- 어텐션(a) : 쿼리와 키의 유사도를 연산하는 것이 역할 (확률 분포 형태)

이미지에서의 해석

- 픽셀 거리와 무관하게 유사한 패치가 이미지에 존재

- 관련 높은 패치/관련 낮은 패치 정보를 최종 결정에 반영하자

자기 어텐션 (Self-Attention) → Transformer 로 발전

- 쿼리/키/값의 정의
  - 쿼리 /키/값이 모두 입력에서 정의
  - 같은 이미지 내 유사도(쿼리와 키) 반영
  - 입력 내부 패치간 연결망 구축 → 클러스터링 효과

교차 어텐션 (Cross-attention)

- 쿼리/키/값의 정의
  - 둘 이상의 이종 데이터에서 정의
    - 예시 : 쿼리는 이미지 패치, 키는 텍스트, 텍스트 키의 정보
  - 이종 데이터 간 유사도 (쿼리와 키 간 유사도) 반영
    - 쿼리와 키를 비교할 수 있는 공간 필요
  - 서로 다른 입력 간 연결망 구축

3-2. 위치 인코딩

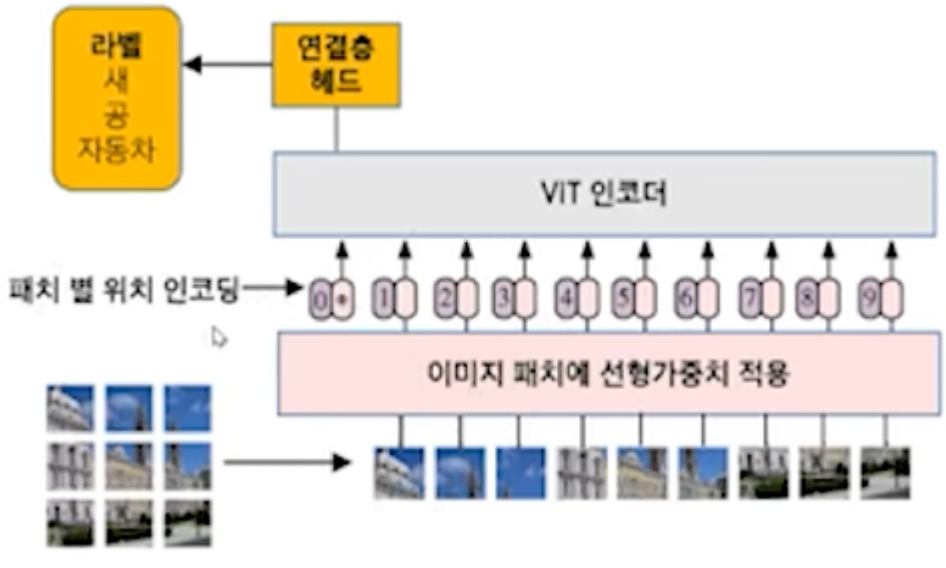
모델 별 특성 저일

- CNN
  - 지역적 특성에 집중
  - 장거리 맥락 이해 떨어짐
  - 데이터 순서 반영 불가
- RNN
  - 데이터 순서를 반영
  - 장거리 맥락 학습이 어려움 (기울기 소실/폭발)
- LSTM
  - 데이터 순서 반영
  - 장거리 맥락 이해 가능
  - 정보 희석 문제
- Attention
  - 장거리 맥락에 탁월/정보 희석 없음
  - 데이터 순서 반영 안됨 → 위치 인코딩이 제안됨

ViT 위치인코딩

순서 정보 제공

- 각 입력 토큰에 위치 정보를 담은 벡터를 추가
- 이 패치가 이미지 (2,2) 에 위치함을 이해 가능

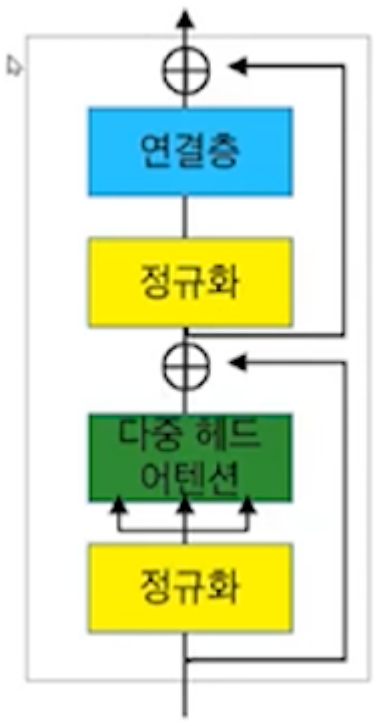


방식

1. 학습 가능한 위치인코딩
    - 데이터에 알맞게 최적화
    - 장점
        - ViT 논문에서 제시된 방식, 특화 학습으로 성능 우수
    - 단점
        - 해상도가 바뀌면 다시 학습 필요
2. 사인파 인코딩
    - 위치 마다 사인파형 이용, 고유한 패턴 부여
    - 장점
        - 데이터 해상도가 바뀌어도 동일한 룰 적용
    - 단점
        - 고정이라 데이터/과업에 특화된 방식이 아님
3. 상대 위치 인코딩
    - 절대 위치가 아닌 상대적 위치를 인코딩
    - 장점
        - 해상도 변화에도 적용 가능
    - 단점
        - 절대적 위치 고려가 어려움 (좌우 반전)

ViT 인코더

- 인코더 내부 구조
    - 정규화 - 다중 헤드 어텐션 - 정규화 - 연결층으로 구성
    - 인코더 L개 통과 (일반적으로 12개 이상)
- 전역 관계 학습
    - 지역적 특징으로부터 수용영역을 늘리지 않음
    - 전역적 맥락 이해 가능



ViT

- 이미지를 이미지 토큰으로 변환
  - 위치 인코딩으로 패치 토큰의 순서를 학습에 반영
- ViT 인코더
  - 정규화 - 다중헤드 어텐션 - 정규화 - 연결층으로 구성
- 최종출력
  - MLP head - 최종 예측 수행

ViT vs ResNet

- ViT가 늘 CNN 모델 보다 좋을까? → X
  - ImageNet의 경우 ResNet이 더 우수
- ViT는 거대 데이터 세트 사전학습 시 우수
  - 대량의 데이터세트 사전 학습 후 미세조정하니 성능이 개선됨

ViT 학습 시 유의점

- ViT 학습 기술이 중요
  - 막대한 상수를 보류
  - 정규화 기법 / 데이터 증강 기술이 매우 중요
- 증류 학습 기술이 효과적
  - 증류 학습 방식
    - 선생님(CNN) 모델을 이미지-레이블 활용해서 학습
    - 학생(ViT) 모델은 선생님 모델의 예측을 따라가도록 학습
      - 정답도 함께 배우도록 학습
    - 학생 ViT 모델은 학습 가능한 토큰을 추가로 입력 받아, 선생님 CNN의 logit 출력과 동일하게 예측하도록 학습됨

ViT의 트렌드

- 다양한 변종이 존재
  - CNN처럼 이미지의 계증적 이해를 활용할 수 있을까?
  - ex) Swin Transformer : 이미지전역에 직접 어텐션 수행하는 것은 너무 고비용, 비효율

1. 윈도우 영역을 지정, 그 내부의 어텐션에 집중
2. 윈도우를 비껴 가게 한번 더 설정

1,2가 중첩되면서 최종적으로 이미지 전역을 고려하는 효과

- 특히, 검출/분할과 같은 화소별, 공간 이해에서 좋은 성능 보임

- Vision 백본 vs 데이터 확장성 vs 멀티모달 모델
  - 현존 최고의 구조? 기준에 따라 다름
  - Vision 백존 : ViT-22B (220억 상수) - 구글 리서치
    - 지도학습 채택 : 대규모 비공개 레이블 데이터 기반 (JFT-3B)
    - 일반화 성능이 우수/모델 상수 비공개 (재현 불가)
  - 자기지도 Vision 백본 : DINOv2 (최대 10억 상수) - 메타AI
    - 공개 웹 데이터 기반 14억장 이미지 활용
    - 비지도 학습 채택 : 자기지도 학습 - 대조학습/증류
    - 공개 모델, 현재 가장 많이 활용되는 백본
  - 비전-언어 : EVA-CLIP - 상하이AI
    - 지도학습 채택 : 이미지- 텍스트 대조 학습 + 마스킹 기반 사전학습 결합 (LAION)
    - 제로샷 분류 및 멀티모달 검색 강력/ 모델 상수 공개
  - 범용 시각 모델 : InternImage-H - 상하이 AI Lab
    - 대규모 지도 학습 채택
    - Conv + Transformer 하이브리드 백본
    - 탐지/분할에서 최고 성능 달성
    - 모델/코드 공개

ViT 장단점

- 장점
  - 전역 문맥을 한 번에 고려 가능
  - 시계열 / 시퀀스 데이터의 경우 순서 고려 가능 (위치 인코딩)
    - 순서 없는 데이터는 선별 제거 가능
- 단점
  - 대규모 학습 자원 필요
    - 대규모 데이터 및 GPU 자원이 요구 (메모리)
  - 학습 데이터 규모가 작을 경우, CNN보다 성능 저하
  - 한번에 고려할 수 있는 토큰 제한적
- 활용
  - 이미지 분류/탐지/분할/생성에서 모두 SoTA 성능 달성
  - 멀티모달 모델에서 기준 아키텍처
  - 기준 모델에 활용