DSAC Module4

Deep Learning (3)-2

2023년 11월 18일~12월 16일

권오준
(ojkwon@deu.ac.kr)

신경망 성능 개선

과대 적합 (Overfitting)

- 신경망은 파라미터 갯수가 많아서 과대적합할 가능성이 항상 높다. 훈련 데이터 양이 많지 않을 때에는 특히 주의해야
  - 과대적합이 발생하면 신경망의 구조를 단순하게 만들어야 한다.
  - 과소적합 여부는 훈련 데이터에 대한 성능이 얼마나 낮은지 (평균치 수준에 머무는지)를 보고 판단

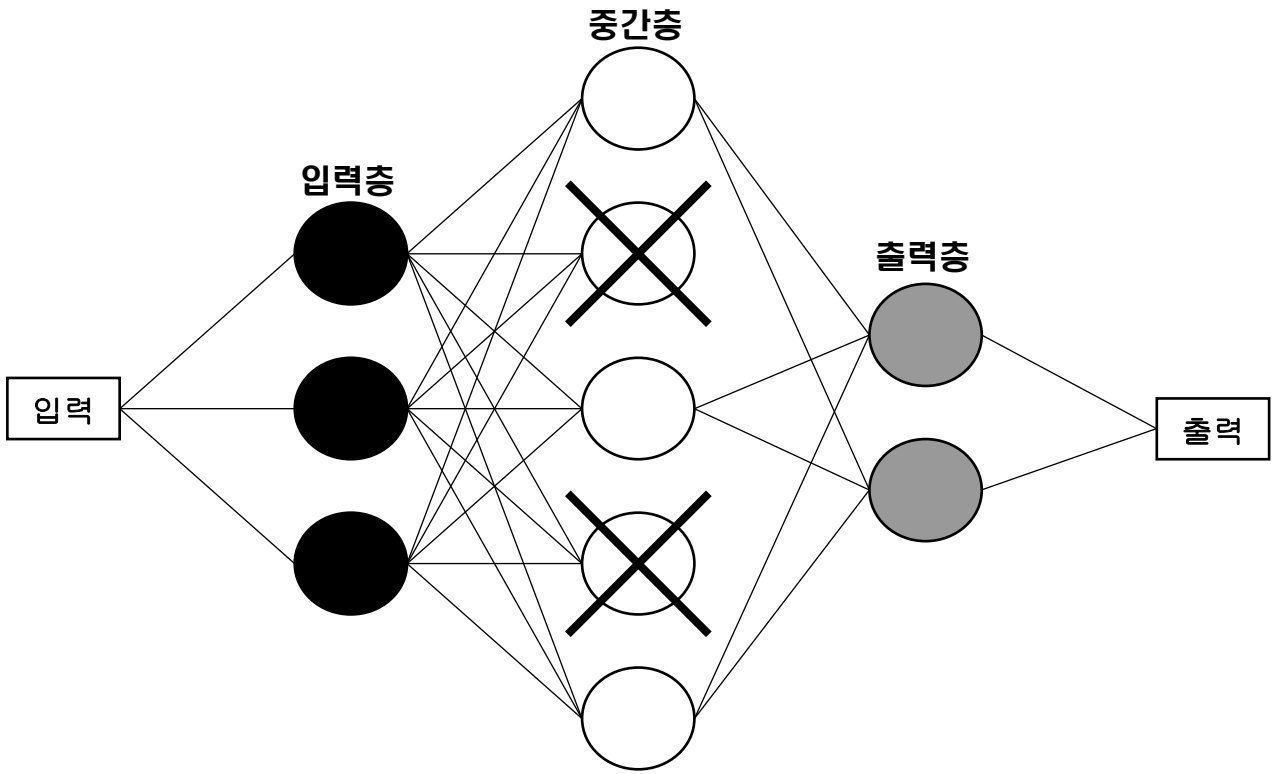
- 과대적합을 줄이는 작업을 일반화(generalize)를 위한 규제화 (regularize) 라고 부르는데,
  - 대표적인 규제화로 L2규제를 적용
    - 네트워크를 구성하는 파라미터 값이 가능한 균등하게 분포하도록 제한
  - 케라스에서는 regularizers.l2() 함수를 사용

드롭 아웃 (Drop out)

- 신호가 계층을 통과할 때 랜덤하게 유닛을 선택하여 신호를 전달하지 않는 방법
- 즉, 모든 신호를 다 사용하지 않고 고의적으로 신호의 일부를 누락시킨다
- 앙상블 효과를 얻어서 과대적합을 효과적으로 줄일 수 있다.
- 드롭아웃을 하면 입력과 출력간의 특정한 관계를 기억하지 못하게 하는 효과가 있어서 신경망이 보다 일반적인 학습을 할 수 있게 한다
- 네트워크의 기억을 랜덤하게 지우는 것이라고 볼 수 있다. 입출력의 특별한 관계를 평준화시키고 다양한 신경망 구조를 이용한 효과(즉, 앙상블 효과를 얻어 성능을 개선하는 방법)

드롭 아웃

- 드롭아웃은 학습을 하는 동안에만 적용
- 학습이 종료된 후 예측을 하는 단계에서는 모든 유닛을 사용하여 예측



출력단 활성화 함수 선택

- 분류를 하는 경우 대부분 소프트맥스 함수를 사용
  - 특정한 카테고리에 속할 확률을 구해준다
  - 즉, 신경망이 하나의 카테고리만 선택할 때는 소프트맥스를 사용

- **복수의 답이 가능한 분류문제**에서는 **시그모이드 함수**를 출력단에 사용하는 것이 나을 수가 있다
  - 시그모이드 함수를 사용하면 각 출력이 0~1의 값을 가지며
  - **출력들의 합은 1을 넘을 수가 있다**

- 회귀분석에 사용될 때에는 출력단에 선형함수(즉, 신호를 그대로 통과시키는 것)를 사용

최적의 신경망 구조

- 최적의 신경망을 구성하는 계층의 수, 유닛의 수, 배치 크기, 학습률의 설정 등이 어려운 과제

- 기본 전략 : 처음에는 구조를 간단히 출발

- 일단 동작을 확인하고 성능을 개선한다.
  - 계층이 2~3만으로도 동작하는지를 확인
  - 만일 2~3개의 계층으로 모델이 동작하지 않으면 계층 수를 늘려도 동작하지 않는다고 알려져 있다.

- 입력 데이터를 간단히 만들어 보는 것도 필요
  - 예를 들어 10가지 동물 이미지를 구분하는 모델이 필요하다고 하여도 우선 몇 가지 대표적인 동물들을 구분하는 모델을 먼저 만들어보는 것

배치 (Batch) 크기

- 배치 크기 : 신경망이 한번에 학습하는 입력 데이터 수

- 배치 크기가 클수록 학습이 정교하고, 기울기를 정확히 구할 수 있으나 계산량이 많아진다.
  - 필요한 메모리 사용량이 많아 메모리 오류가 날 가능성이 높다(특히 GPU를 사용할 때). 처음에는 배치 크기를 작게 16~32정도로 적게 잡고 시작

- 배치 크기가 작을 때에는 기울기가 상대적으로 정확하게 계산되지 못하므로 학습률도 작게 잡아야 한다.
  - 일단 작은 값의 학습률로 동작하는 것을 확인하고 학습률을 조금씩 크게 한다

배치 정규화 (Batch Normalization)

- 딥러닝 학습에서 해결해야할 가장 어려운 문제
  - Vanishing Gradient Problem

- 이는 기울기 값에 비례하여 학습시킬 때 기울기 즉, 미분값이 0에 가까워지면 변화량이 매우 적어지고 이것이 앞단의 계층으로 전파되는 양이 급속히 줄어들어 학습이 잘 되지 않는 현상

- 이러한 문제를 해결하기 위해 배치 정규화가 제시됨
  - 이는 "계층별로", 주어진 배치 데이터를 대상으로 정규화를 다시 수행하여 데이터의 분포가 너무 작거나 너무 커지지 않게 하는 방식
  - **학습 시 미니배치를 단위로 정규화 : (평균 0, 분산 1)**

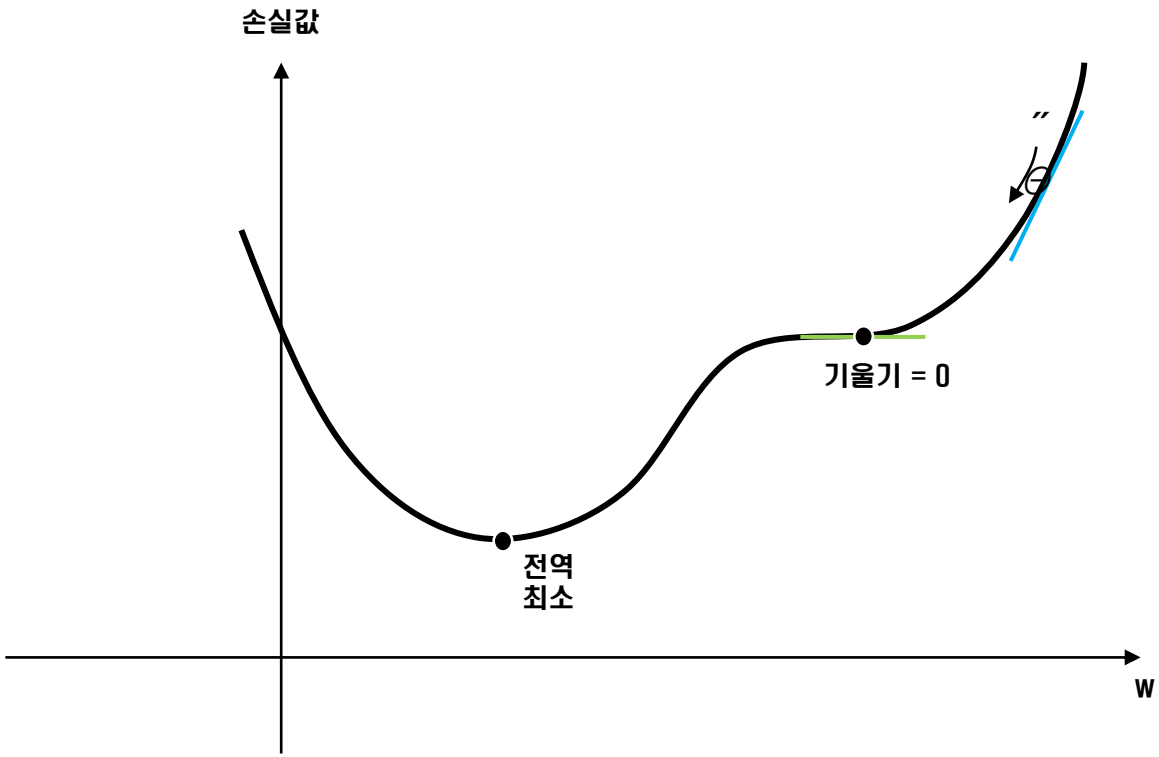
모멘텀 (Momentum) 알고리즘

- 머신러닝에서는 최적화 알고리즘으로 SGD이 널리 사용된다.

- 신경망에서는 좀 더 정교한 방법으로 모멘텀 기법을 사용한다.
  - 모멘텀 기법이란 기울기 뿐 아니라 가속도 항을 고려하여 움직이던 방향으로 계속 움직이려는 관성을 반영한 것이다.
  - 현재의 기울기(gradient) 에 비래하는 학습을 할 뿐 아니라 여기에 더해서 기울기의 “변화량”도 반영하게 된다.

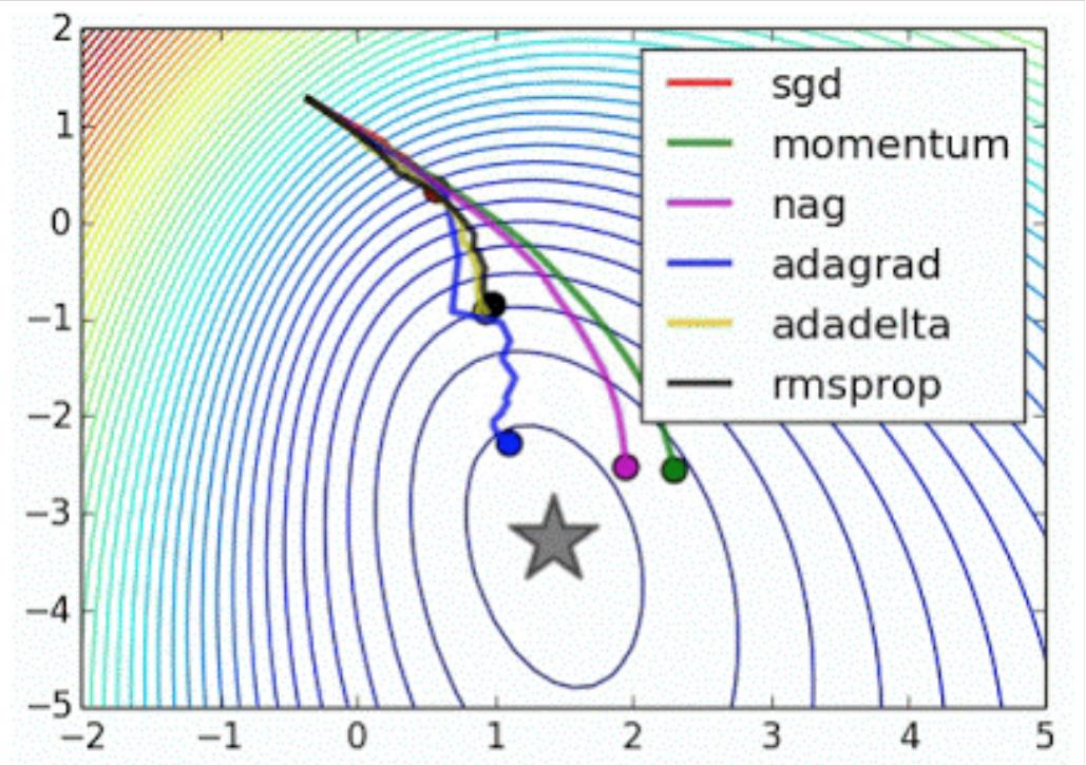
- 학습률을 적응형으로 감소하는(adaptive gradient) 방식인 AdaGrad도 널리 사용된다.
  - 이 방법에서는 학습률을 서서히 낮춘다.
  - 모멘텀 기법과 적응형 방법을 조합한 방법으로 Adam이 2015년 소개되었다. 현재 가장 널리 사용되고 있다.

모멘텀 (Momentum) 알고리즘



최적화 (Optimization) 알고리즘 비교

- 최적화 동작
  - https://goo.gl/Pdu4uW



최적화 (Optimization) 알고리즘

- 가속도 방식을 도입
  - 지역 최소를 지나가게 하여 전역 최소를 찾을 수 있게 한다
  - Momentum
  - Nesterov Momentum
  - Adam
  - RMSProp

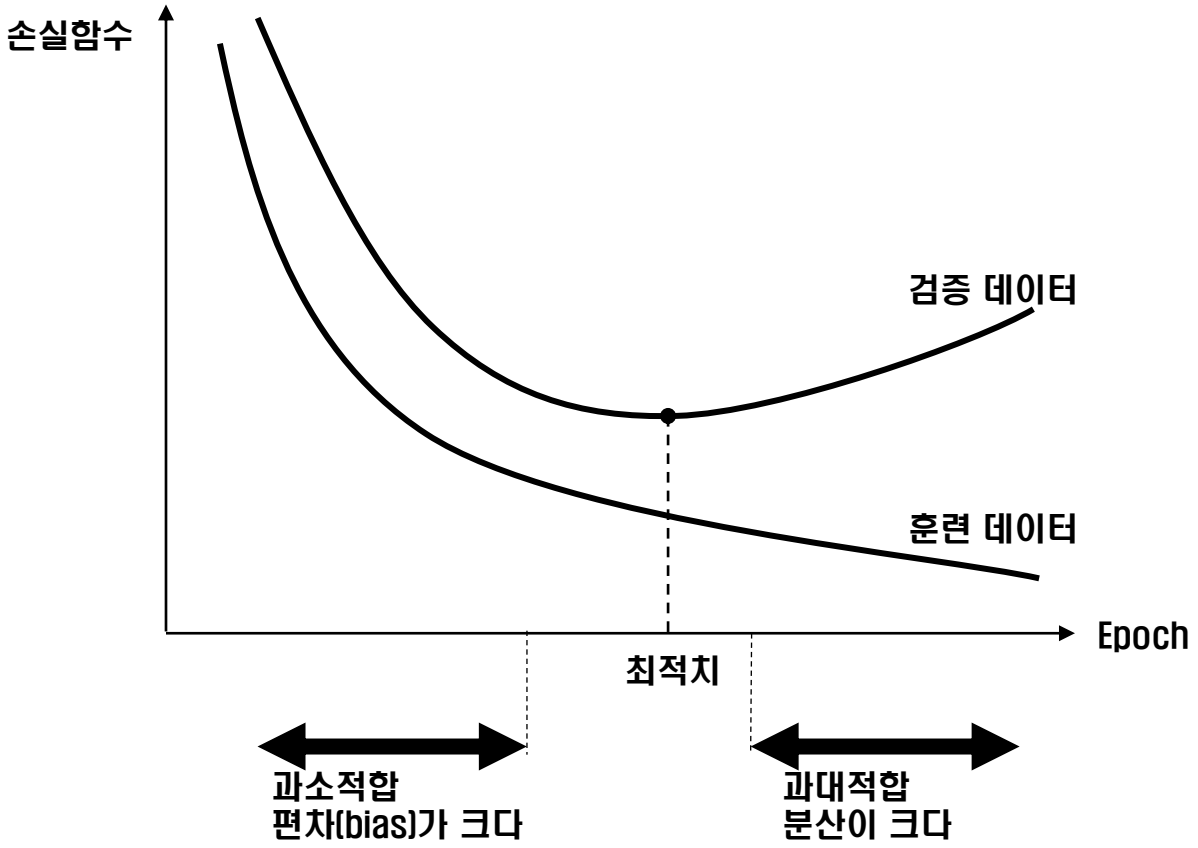
과대적합 검증

- 신경망은 파라미터가 많아서 과대적합되기 쉽다. 즉, 상세한 모델링이 가능하여, 훈련 데이터 수가 적으면 이 훈련 데이터의 속성을 모두 기억할 수 있어 과대적합되기 쉬운 것이다.

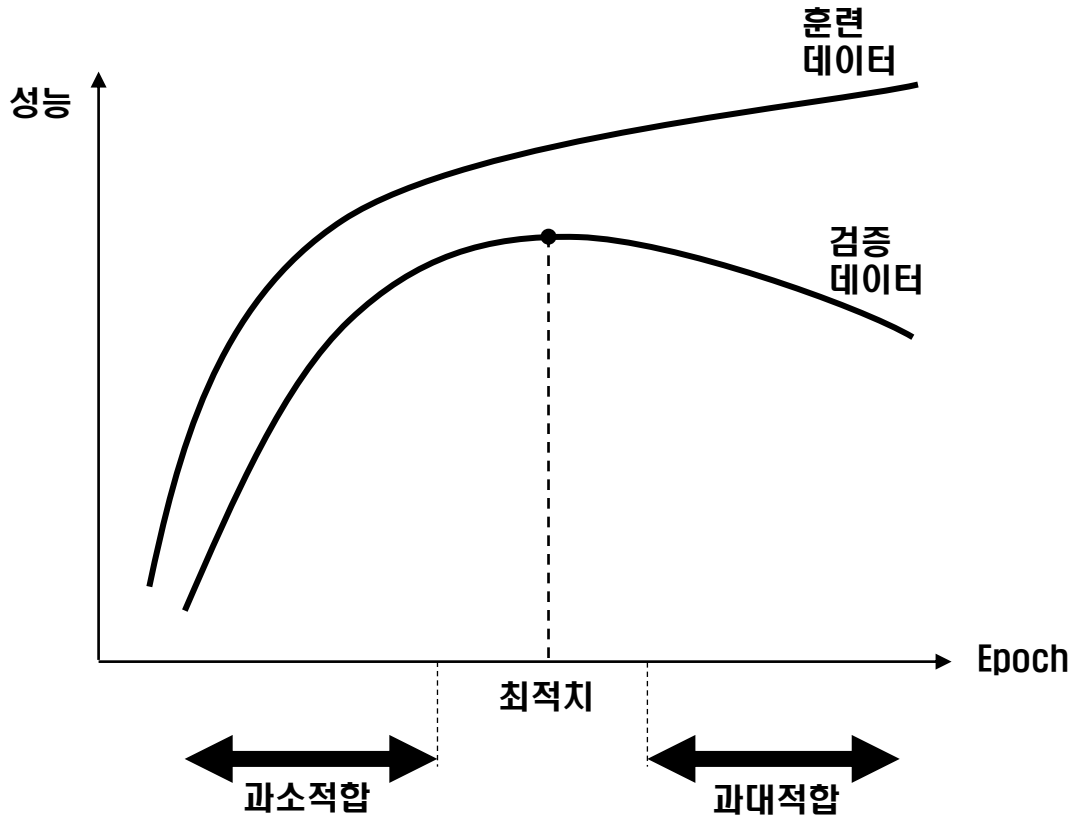
- 훈련데이터에 대해서는 계속 성능이 좋아지지만 검증 데이터에 대해서는 오히려 성능이 나빠진다면 과대적합한 것이다.

- 성능의 개선되는지를 측정하는 방법
  - 정확도 등 최종 성능 지표를 관찰하는 방법
  - 손실함수가 줄어드는지를 관찰하는 방법이 있다.

이포크 (Epoch) 증가에 따른 손실함수의 변화



이포크 (Epoch) 증가에 따른 성능의 변화



과대 적합을 피하는 방법

- 학습조기중단(early stopping)
- L1 이나 L2 규제
- 드롭아웃(dropout)
- 데이터 확장(data augmentation)
  - 훈련 데이터가 많은 것처럼 보이는 효과

데이터 확장 (Data Augmentation)

- 과대적합이 일어나는 이유 중 하나
  - 훈련데이터가 부족하기 때문

- 훈련 데이터가 충분히 많다면 과대적합을 줄일 수 있다.

- 데이터 확장이라 훈련 데이터를 다양하게 변형하여 변형된 새로은 훈련 데이터처럼 사용함으로써 마치 훈련 데이터 수가 늘어난 효과를 얻는 것이다.

- 데이터 확장을 사용하면 여러 이포크를 수행해도 똑같은 데이터를 가지고 학습하지 않게 된다.

데이터 확장 (Data Augmentation)

- rotation: 0° 에서 360° 사이에서 회전
- shifting: 랜덤하게 상하좌우로 이동
- rescaling: 랜덤하게 1.0 ~ 1.6배로 사진 확대
- flipping: 좌우, 또는 상하로 반전
- shearing: -20° 에서 20° 도 사이에서 왜곡
- stretching: 1.0 ~ 1.3배로 확장

데이터 확장 예

- “레이블링 된” 학습 데이터가 부족한 문제를 해결

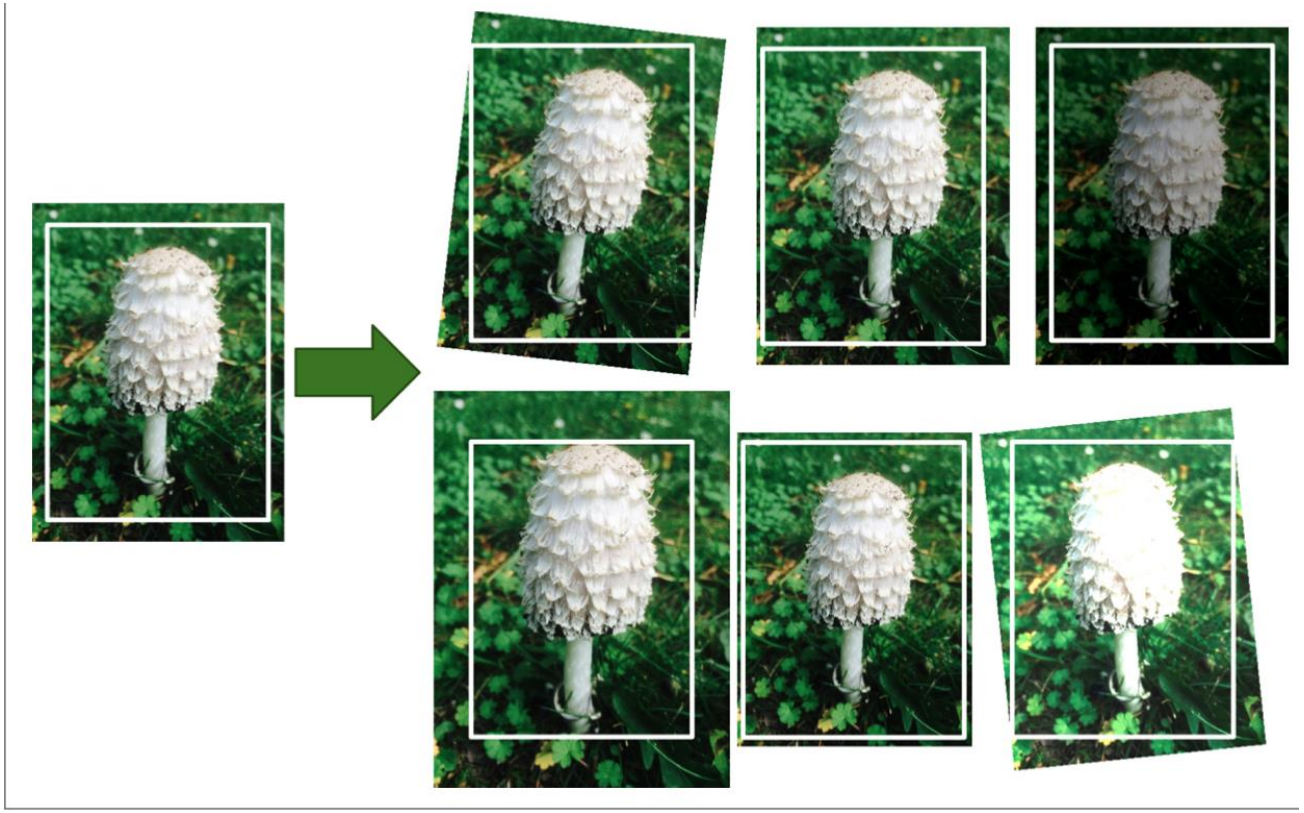
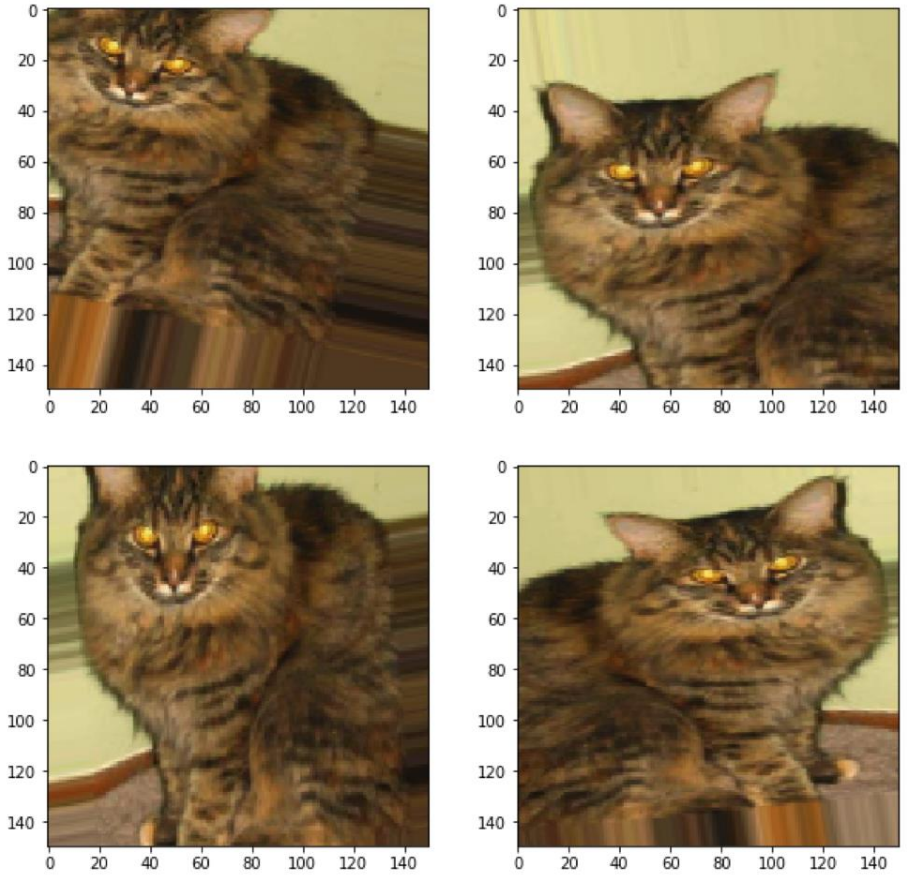


Figure 11-10. Generating new training instances from existing ones

데이터 확장 예

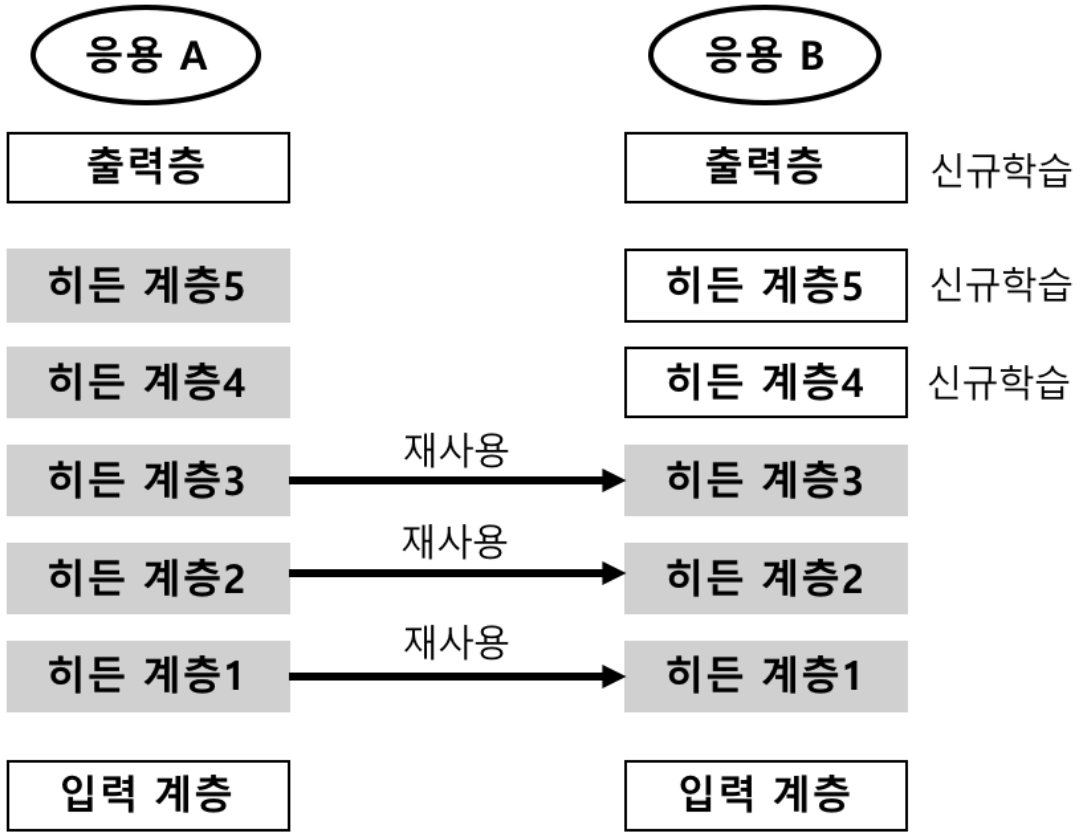


전이 학습

전이 학습 (Transfer Learning) 정의

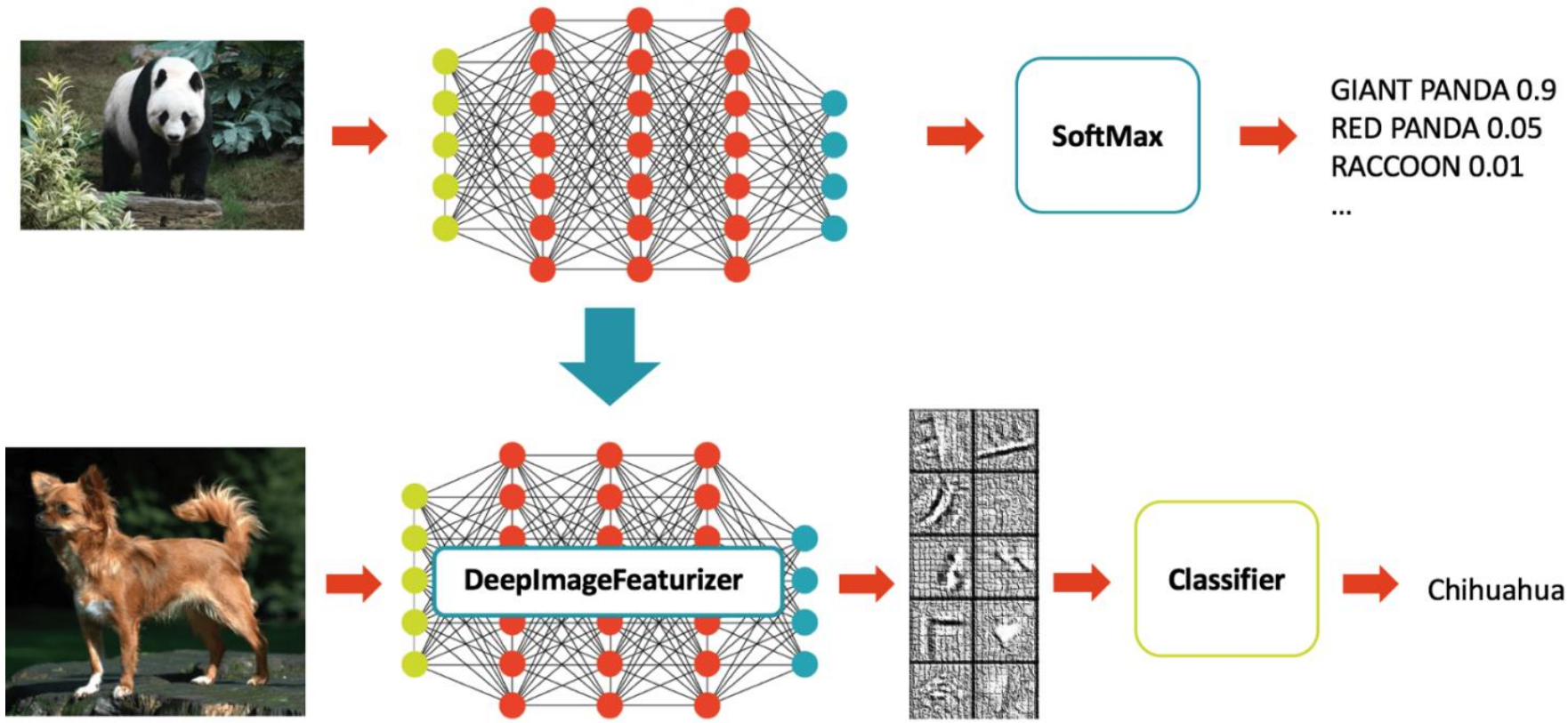
- 전이학습 이란 다른 데이터 셋을 사용하여 이미 학습한 모델을 유사한 다른 데이터를 인식하는데 사용하는 기법이다.
- 이 방법은 특히 새로 훈련시킬 데이터가 충분히 확보되지 못한 경우에 학습 효율을 높여준다.
- 이미지넷에는 동물이나 일상생활의 물건들을 주로 포함하여 1000종의 이미지를 갖고 있으며 140만장의 사진이 있다.
- 이미지넷에는 고양이 강아지를 포함한 많은 동물 이미지도 들어 있으며, 이를 강아지 고양이 분류 문제에 사용할 수 있다.
  - 여기서는 2014년에 소개된 VGG16 모델을 사용하겠다.
- 사전학습모델을 이용하는 방법은 특성 추출(feature extraction) 방식과 미세조정(fine-tuning) 방식이 있다.

전이 학습


응용 A
응용 B
출력층
출력층
신규학습
히든 계층5
히든 계층5
신규학습
히든 계층4
히든 계층4
신규학습
재사용
히든 계층3
히든 계층3
재사용
히든 계층2
히든 계층2
재사용
히든 계층1
히든 계층1
입력 계층
입력 계층

전이 학습

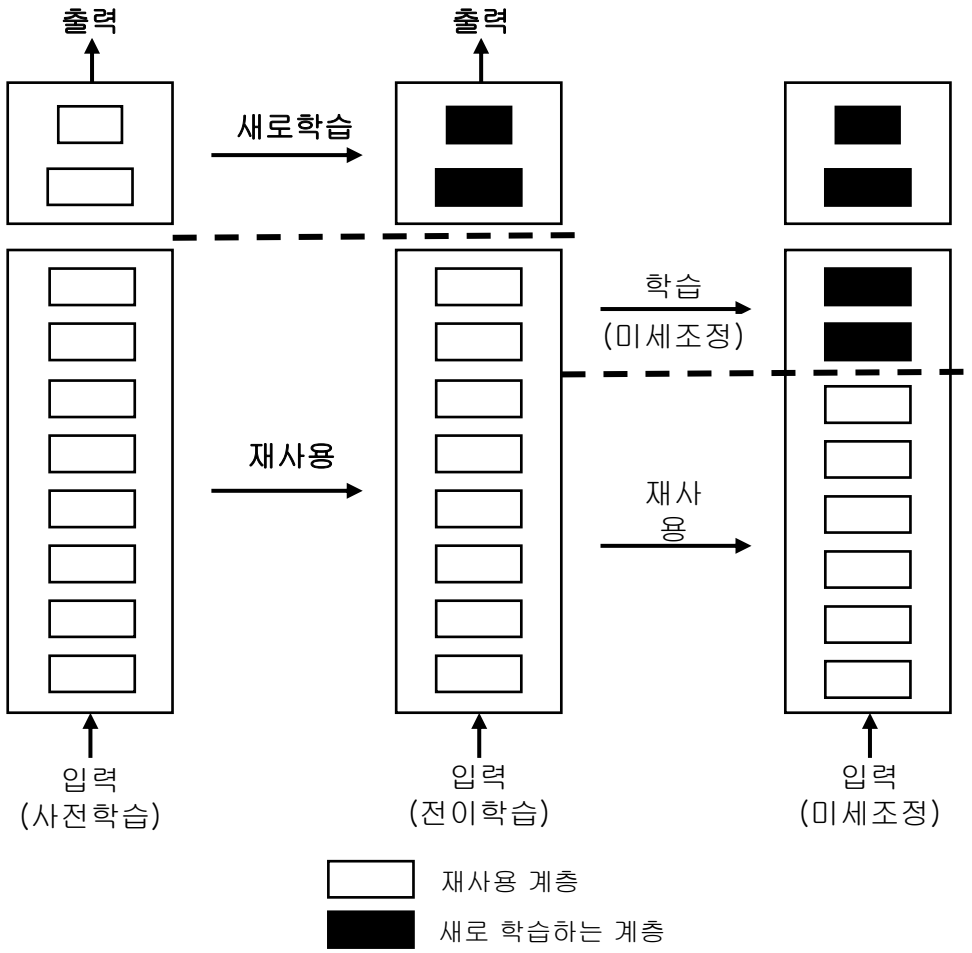
전이 학습


SoftMax
GIANT PANDA 0.9
RED PANDA 0.05
RACCOON 0.01
...
DeepImageFeaturizer
Classifier
Chihuahua

특성 추출 방식

- 특성 추출이란 이전의 네트워크로부터 배운 표현 방식을 사용하여 새로운 데이터 샘플에서 추가로 흥미로운 특성들을 추출하는 것
- 이렇게 얻은 특성들을 사용하여 새로운 분류기를 작동시키고 학습을 수행한다.
- 이미지 분류기는 컨볼류션 네트워트와 풀링 계층의 조합 그리고 전결합층 분류기 등 크게 두 부분으로 구성된다.
- 컨볼류션과 풀링으로 구성된 부분을 컨볼류션 베이스라고한다
- 특성 추출 방식은 컨볼류션 베이스를 그대로 사용하고, 새로운 데이터를 여기에 적용하여 훈련시키는 방식이다.

특성 추출 방식


출력
출력
새로학습
학습
(미세조정)
재사용
재사
용
입력
(사전학습)
입력
(전이학습)
입력
(미세조정)
재사용 계층
새로 학습하는 계층

특성 추출 방식

- 컨볼류션 베이스 부분만 재사용하는 이유는 이 부분은 상당히 일반적인 학습정보를 포함하고 있기 때문이다.

- 컨볼류션 계층에서도 재사용할 수 있는 정보의 수준은 몇 번째 계층인지에 따라 다르다. 모델의 앞단의 계층일수록 에지, 색상, 텍스처 등 일반적인 정보를 담는다.

- 반면에 뒷 부분의 깊은 계층일수록 추상적인 정보를 담는다 (예를 들어 고양이 귀, 강아지 귀 등)

- 새롭게 분류할 클래스의 종류가 사전 학습에 사용된 데이터와 특성이 매우 다르면, 컨볼류션 베이스 전체를 재사용해서는 안되고 앞단의 일부 계층만을 재사용해야 한다.

미세 조정 (Fine tuning) 방식

- 모델 베이스 중 상위 몇개의 계층은 전결합층 분류기와 함께 새로 학습시키는 방식이다.
- 최종 분류기의 계수가 랜덤하게 초기화 되어 있으므로 이를 먼저 학습시키며 이때 VGG16 모델의 컨볼류션 베이스를 초기에는 고정해야 한다.
- 먼저 분류기를 계수를 학습시킨 다음에 (즉, 이 동안은 미세조정을 하지 않도록 상위계층의 계수를 고정시켜 두고), 그 이후에 미세조정을 해야 한다.
- 처음부터 베이스 상위계층의 계수를 같이 훈련시키면 분류기에서 발생하는 큰 에러 값으로 인해, 사전 학습된 정보가 많이 손실된다.

미세 조정 절차

1) 사전 학습된 기본 네트워크 상단에 새로운 네트워크를 추가한다.
2) 기본 네트워크를 고정시킨다.
3) 새로 추가한 부분을 학습시킨다.
4) 기본 계층 중에 학습시킬 상위 부분의 고정을 푼다
5) 고정을 푼 계층과 새로 추가한 계층을 함께 훈련시킨다.

- 미세 조정을 천천히 수행하기 위해서 느린 학습 속도를 선택한다. 갑자기 큰 변화를 주면 사전 학습된 내용이 훼손되기 때문이다.

콜백(Callback)

- 케라스, 콜백 함수
  - 모델 학습을 하는 동안 중간 결과를 저장하거나
  - 조기 종료를 시키거나 할 수 있는 기능

- 콜백은 여러 가지를 동시에 지정할 수 있으며 이를 fit 함수의 callbacks 인자로 넘겨줄 수 있다.

콜백(Callback) – 주요 함수

- History()
  - 훈련중에 발생하는 여러 이벤트를 History 객체에 저장
  - keras.callbacks.History()

- 모델 저장: 매 이포크마다 모델을 저장
  - 저장하는 주기(period)와 베스트 모델만 저장 등을 지정

```
keras.callbacks.ModelCheckpoint(filepath,
                monitor='val_loss', verbose=0,
                save_best_only=False,
                save_weights_only=False, mode='auto', period=1)
```

콜백(Callback) – 주요 함수

- 조기 종료
  - 손실값, 성능 등 관찰 중인 지표가 어떤 조건을 만족하면 이포크 실행을 종료
  - 조기 종료 조건들을 인자로 지정

```
keras.callbacks.EarlyStopping(
        monitor='val_loss',
        min_delta=0, patience=0, verbose=0, mode='auto',
        baseline=None, restore_best_weights=False)
```

- 참고) https://keras.io/callbacks/#modelcheckpoint

실습

신경망 설계 – keras 주요 특징

- 모듈화 (Modularity)
  - 제공하는 모듈은 독립적으로 설정 가능하며, 가능한 최소한의 제약사항으로 서로 연결되어 있고 모델은 시퀀스 또는 그래프로 이러한 모듈들을 구성
  - 특히 신경망 층, 비용함수, 최적화기, 초기화기법, 활성화함수, 정규화기법은 모두 독립적인 모듈이며, 새로운 모델을 만들기 위해 이러한 모듈을 조합할 수 있다
- 최소주의 (Minimalism)
  - 각 모듈은 짧고 간결
  - 모든 코드는 한 번 훑어보는 것으로도 이해가능한 수준
  - 단 반복 속도와 혁신성에는 다소 떨어질 수가 있음

- 참고) https://tykimos.github.io/2017/01/27/Keras_Talk/

신경망 설계 – keras

- 쉬운 확장성
  - 새로운 클래스나 함수로 모듈을 아주 쉽게 추가
  - 따라서 고급 연구에 필요한 다양한 표현이 가능
- 파이썬 기반
  - Caffe 처럼 별도의 모델 설정 파일이 필요없음
  - 파이썬 코드로 모델들이 정의됨

신경망 설계 – keras

- 데이터셋 생성하기
  - 원본 데이터를 불러오거나 시뮬레이션을 통해 데이터를 생성
  - 데이터로부터 train, validation, test 데이터셋을 생성
  - 이 때 딥러닝 모델의 학습 및 평가를 할 수 있도록 포맷 변환

- 모델 구성하기
  - 시퀀스 모델을 생성한 뒤 필요한 레이어를 추가하여 구성
  - 좀 더 복잡한 모델이 필요할 때는 케라스 함수 API를 사용

- 모델 학습과정 설정하기
  - 학습하기 전에 학습에 대한 설정을 수행
  - 손실 함수 및 최적화 방법을 정의
  - 케라스에서는 compile() 함수를 사용

신경망 설계 – keras

- 모델 학습시키기
  - 훈련셋 train 데이터셋을 이용하여 구성한 모델로 학습
  - 케라스에서는 fit() 함수를 사용
- 학습과정 살펴보기
  - 모델 학습 시 train 데이터셋, validation 데이터셋의 손실 및 정확도를 측정
  - 반복횟수에 따른 손실 및 정확도 추이를 보면서 학습 상황을 판단
- 모델 평가하기
  - 준비된 test 데이터셋으로 학습한 모델을 평가
  - 케라스에서는 evaluate() 함수를 사용
- 모델 사용하기
  - 임의의 입력으로 모델의 출력을 예측
  - 케라스에서는 predict() 함수를 사용

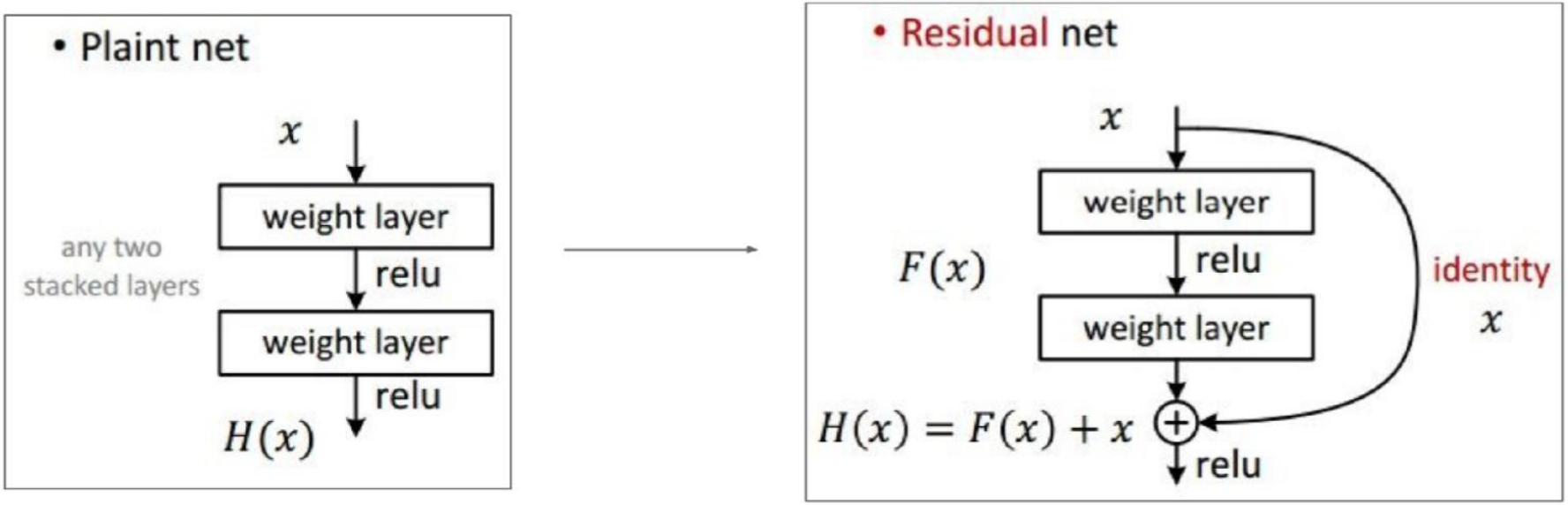
신경망 설계 – keras 요약

- Sequential 모형 클래스 객체 생성
- add () 메서드로 layer 추가
  - 입력단부터 순차적으로 추가
  - 각 layer – 출력 뉴런 갯수를 첫번째 인수
  - 최초의 layer – input_dim 인수로 입력 크기를 설정
  - activation : 활성화함수 설정
- compile () 메서드로 모형 완성
  - Loss : 비용함수 설정
  - optimizer : 최적화 알고리즘 설정
  - metrics : 훈련 단계에서 기록할 성능 기준 설정
- fit () 메서드로 학습
  - nb_epoch : epoch 횟수 설정
  - batch_size : 배치크기 설정
  - Verbose : 학습 중 출력되는 문구를 설정,

  * Jupyter Notebook 사용시 verbose=2로 설정, progress bar 나오지 않도록

Q & A

ResNet


• Plaint net
x
weight layer
relu
weight layer
relu
$H(x)$
any two
stacked layers
• Residual net
x
weight layer
relu
weight layer
$F(x)$
identity
x
$H(x) = F(x) + x$
relu

고양이 강아지 구분

- 고양이와 강아지 이미지를 구분하는 예를 소개하겠다.
- 이러한 분류에 MLP 모델을 사용하면 성능이 나쁘게 나온다.
- 데이터는 아래 Kaggle 사이트에서 다운로드 받을 수 있다.
  https://www.kaggle.com/c/dogs-vs-cats/data
- 캐글에서 원래 제공하는 데이터는 25,000 장이나 여기서는 2천 장의 사진만 사용하겠다.
- 적은 훈련 데이터를 사용하는 경우에 신경망 과대적합이 발생하기 쉽다. 이를 방지하기 위해 데이터 확장(augmentation)을 사용한다
- 전이학습을 소개한다.
  - 전이학습을 사용하면 훈련 시간도 줄이고, 훈련 데이터도 적게 필요하고, 분류 성능도 상당히 개선시킬 수 있다.

CNN을 이용한 MNIST

- MLP를 이용한 MNIST 숫자 인식 프로그램을 소개했었다.
  - 간단한 구조의 MLP를 이용한 MNIST 인식에서도 인식률이 97.8%의 높은 성능을 보였는데 이는 주어진 샘플 이미지가 잘 정리되어서 크기와 위치가 모두 균일했기 때문이었다.

- 여기서는 일반적인 이미지 (즉, 숫자의 위치와 크기가 랜덤한)에 대해서도 인식률을 높일 수 있는 CNN을 소개하겠다.

- CNN의 가장 큰 특징은 입력 신호에 여러 가지 필터(커널)를 적용한다는 것이다.
  - 이미지 처리의 경우 보통 3x3 크기의 작은 필터를 적용한다.
  - 예를 들어 붉은 색을 찾는 필터를 통과시키면 입력 이미지에서 붉은 색이 많은 부분을 찾아내고, 대각선 성분이 있는 곳을 찾는 필터를 통과시킨면 대각선 성분의 크기에 비례하는 출력은 얻는다.