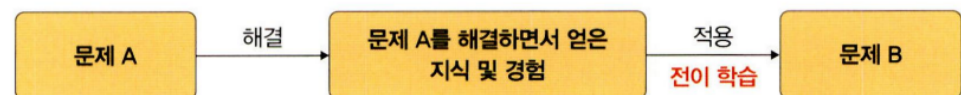


# Week4\_예습과제

## 5.3 전이 학습

이미지넷(ImageNet)처럼 아주 큰 데이터셋을 써 훈련된 모델(사전훈련된 모델, 네트워크)의 가중치를 가져와 우리가 해결하려는 과제에 맞게 보정해 사용하는 것

▼ 그림 5-29 전이 학습

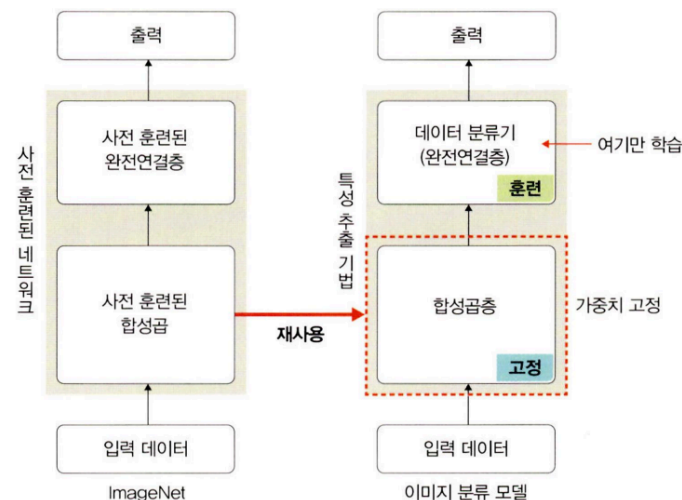


### 5.3.1 특성 추출 기법

- ImageNet 데이터셋으로 사전 훈련된 모델을 가져온 후 마지막에 완전 연결층 부분만 새로 만듦.
- 학습 : 마지막 완전 연결층 (이미지의 카테고리 결정)

#### 📌 특성 추출 구성

▼ 그림 5-30 특성 추출 기법



- 합성곱층 : 합성곱층 + 풀링층  
→ 합성곱층의 가중치는 고정, 데이터 출력
- 데이터 분류기 ( 완전연결층 ) : 추출된 특성을 입력받아 최종적으로 이미지에 대한 클래스 분류  
→ 합성곱층의 출력을 바탕으로 훈련

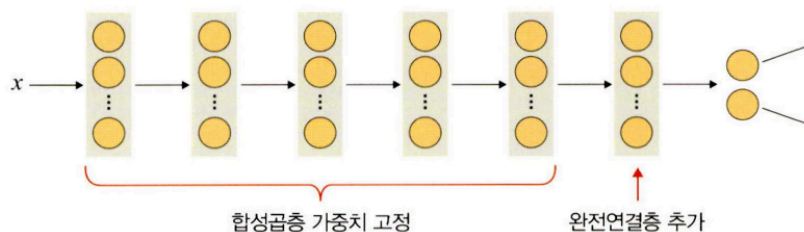
- 이미지 분류 모델 : Xception, Inception V3, ResNet50, VGG16, VGG19, MobileNet

\*\* 예제 부분은 제출한 .ipynb파일에 주석으로 정리, 부족한 부분만 추가로 정리

## ✓ ResNet18

- 50계층으로 구성된 합성곱 신경망
- ImageNet 데이터베이스의 100만 개 넘는 영상을 이용해 훈련된 신경망
- 입력 제약 매우 큼, 충분한 메모리(RAM) 없으면 학습 속도 느림.
- 완전 연결층 추가

▼ 그림 5-36 ResNet18에 완전연결층 추가



## ✓ 사전 훈련된 모델

```
import torchvision.models as models
resnet18 = models.resnet18()
alexnet = models.alexnet()
vgg16 = models.vgg16()
squeezenet = models.squeezenet1_0()
densenet = models.densenet161()
inception = models.inception_v3()
googlenet = models.googlenet()
shufflenet = models.shufflenet_v2_x1_0()
mobilenet_v2 = models.mobilenet_v2()
mobilenet_v3_large = models.mobilenet_v3_large()
mobilenet_v3_small = models.mobilenet_v3_small()
resnext50_32x4d = models.resnext50_32x4d()
wide_resnet50_2 = models.wide_resnet50_2()
mnasnet = models.mnasnet1_0()
```

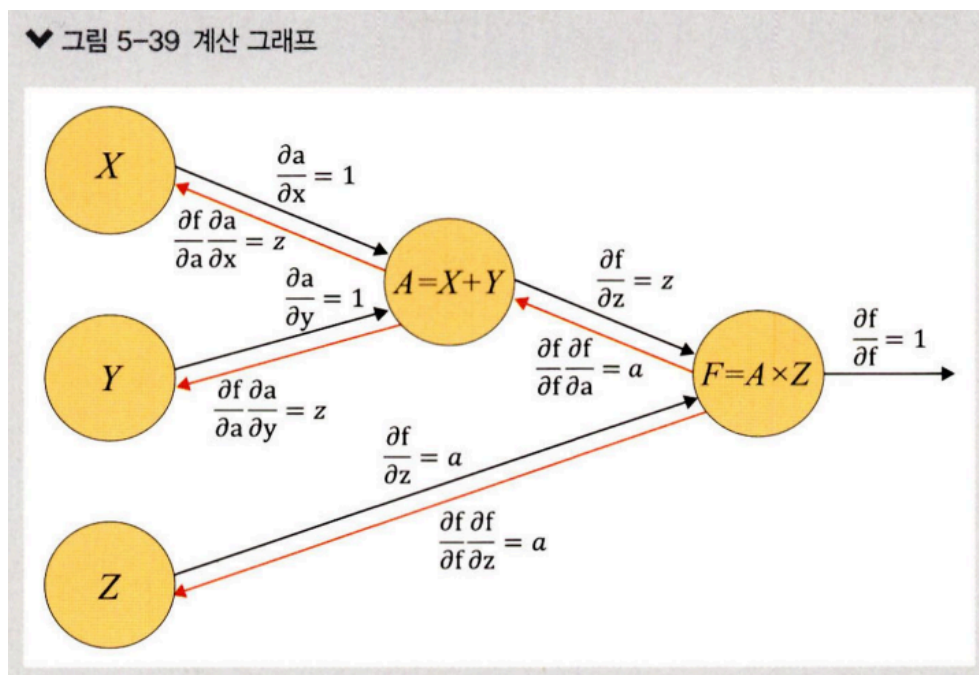
- 안에 `pretrained=True` 를 넣으면 사전 학습된 모델의 가중치값을 사용할 수 있음.
- 없으면 무작위의 가중치

## ✓ `.clone()`, `detach()`, `clone().detach()` 비교

구분	메모리	계산 그래프 상주 유무
<code>tensor.clone()</code>	새롭게 할당	계산 그래프에 계속 상주
<code>tensor.detach()</code>	공유해서 사용	계산 그래프에 상주하지 않음
<code>tensor.clone().detach()</code>	새롭게 할당	계산 그래프에 상주하지 않음

- `.clone()` : 기존 텐서 내용 복사한 새 텐서 생성
- `.detach()` : 기존 텐서에서 기울기 전파 ❌

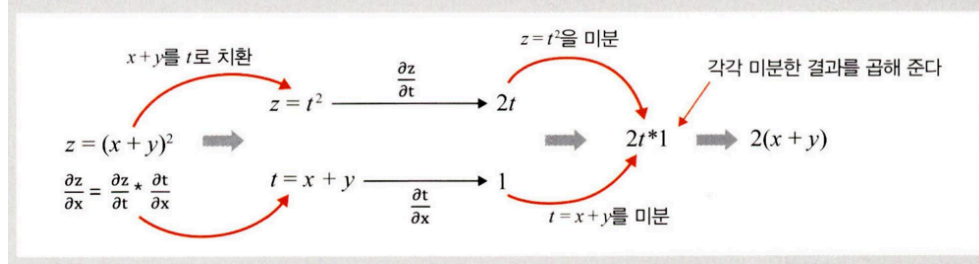
## ✓ 계산 그래프 ( computational graph )



- 노드와 엣지로 구성
- 국소적 계산 가능 (  $Z$  값이 변경돼도  $X, Y$  는 유지한 채로  $F$  계산 가능 )
- 역전파를 통한 미분 계산이 편리함.

## ✓ 연쇄 법칙 ( chain rule )

♥ 그림 5-40 합성 함수의 미분

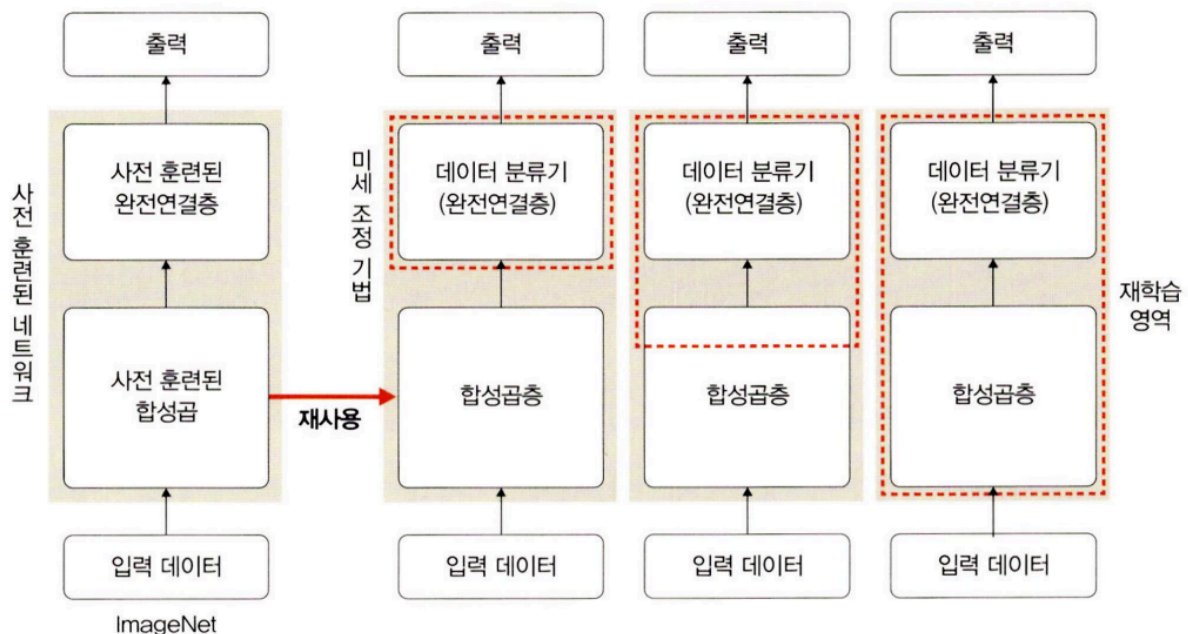


- 합성 함수의 미분법

## 5.3.2 미세 조정 기법




- pretrained model, 합성곱층, 데이터 분류기의 가중치 업데이트 훈련
- 특성 추출 : 목표 특성을 잘 추출하면 좋은 성능  
만약 특성 추출이 잘 안되면 미세조정 기법으로 새로운 이미지 데이터 사용, 네트워크 가중치 업데이트 → 특성 재추출
- 사전 훈련된 네트워크를 미세 조정해 분석 대상인 데이터셋에 잘 맞게 모델의 파라미터 조정하는 기법
- CPU보단 GPU 사용

♥ 그림 5-44 미세 조정 기법



## 📌 미세조정기법 사용 전략

- 데이터셋 大, 사전 훈련된 모델과 유사성 ↓

- 모델 전체 재학습
- 데이터셋 大, 사전 훈련된 모델과 유사성 
  - 합성곱층의 뒷부분(완전 연결층과 가까운 부분) & 데이터 분류기 재학습
  - 전체를 학습시키는 것보단 강한 특징이 나타나는 부분만 새로 학습
- 데이터셋 小, 사전 훈련된 모델과 유사성 
  - 합성곱층 일부 + 데이터 분류기 학습
  - 데이터가 적어서 일부계층에 미세 조정기법 적용해도 효과 없을 수 있음, 계층 범위는 적당히 설정
- 데이터셋 小, 사전 훈련된 모델과 유사성 
  - 데이터 분류기(완전 연결층)만 학습
  - 많은 계층에 미세조정 기법 적용 시 과적합 발생 가능
- 파라미터에 큰 변화를 주면 과적합 문제 발생 가능, 정교하고 미세하게 파라미터 업데이트할것

## 5.4 설명 가능한 CNN

- CNN - blackbox 모델
- CNN으로 얻은 결과의 신뢰성을 높이기 위해 처리 과정 시각화
- 필터에 대한 시각화, 특성맵에 대한 시각화가 있음

### 5.4.1 특성 맵 시각화

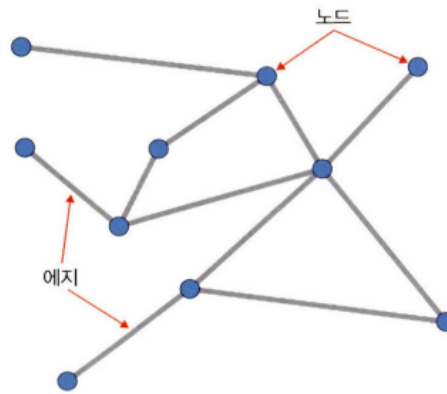
**\*\* 예제 부분은 제출한 .ipynb파일에 주석+마크다운으로 정리**

## 5.5 그래프 합성곱 네트워크

- 그래프 데이터를 위한 신경망

### 5.5.1 그래프란

▼ 그림 5-50 그래프



- 방향성이 있거나 없는 엣지로 연결된 노드의 집합
- 노드, 엣지 : 풀고자 하는 문제에 대한 도메인 지식/직관 등으로 구성
  - 노드 : 원소들
  - 엣지 : 결합방법 cf. single, double, triple, aromatic

## 5.5.2 그래프 신경망

- 그래프 구조에서 사용하는 신경망

### 📌 그래프 데이터에 대한 표현 2단계

▼ 그림 5-51 특성 행렬<sup>5</sup>



#### 1. 인접행렬 ( adjacency matrix )

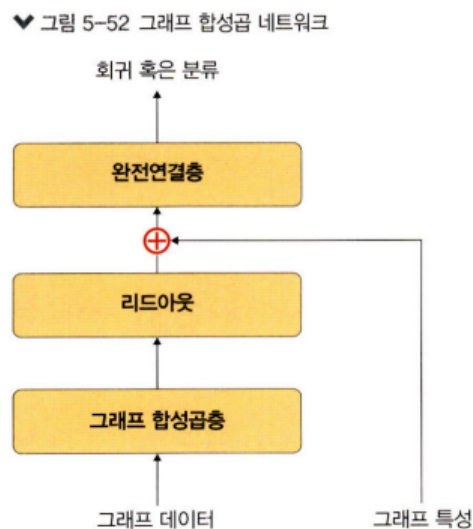
- 노드  $n$ 개  $\rightarrow n \times n$  행렬
- 인접 행렬 내의 값은  $A_{ij}$ 가  $i$ 와  $j$ 의 관련성 여부를 만족하는지 여부
- 컴퓨터가 이해하기 쉽게 그래프로 표현하는 과정

#### 2. 특성 행렬 ( feature matrix )

- 인접 행렬만으로는 특성 파악 어려움, 단위 행렬 적용
- 각 입력 데이터에서 이용할 특성 선택
- 각 행 = 선택된 특성에 대해 각 노드가 갖는 값
- → 그래프 특성 추출

### 5.5.3 그래프 합성곱 네트워크

- Graph Convolutional Network, GCN
- 이미지에 대한 합성곱을 그래프 데이터로 확장



- 리드아웃(readout) : 특성 행렬을 하나의 벡터로 변환하는 함수  
→ 모든 노드의 특성 벡터에 대해 평균을 구하고 그래프 전체를 표현하는 하나의 벡터 생성
- ★ 그래프 합성곱층
  - 이 층을 거친 그래프 형태의 데이터는 행렬로 변환돼 딥러닝 알고리즘을 적용할 수 있게 됨
- GCN의 활용처
  - SNS에서 관계 네트워크
  - 학술 연구에서 인용 네트워크
  - 3D Mesh