CV DEEP LEARNING WITH PYTORCH

PART

TRANSFER LEARNING

TRANSFER LEARNING

◆ 전이학습이란?

- 합성곱 신경망 기반 딥러닝 모델을 훈련시키려면 **많은 양의 데이터 필요**
- **충분히 큰 데이터셋을 얻는 것은 쉽지 않음**

해결방안

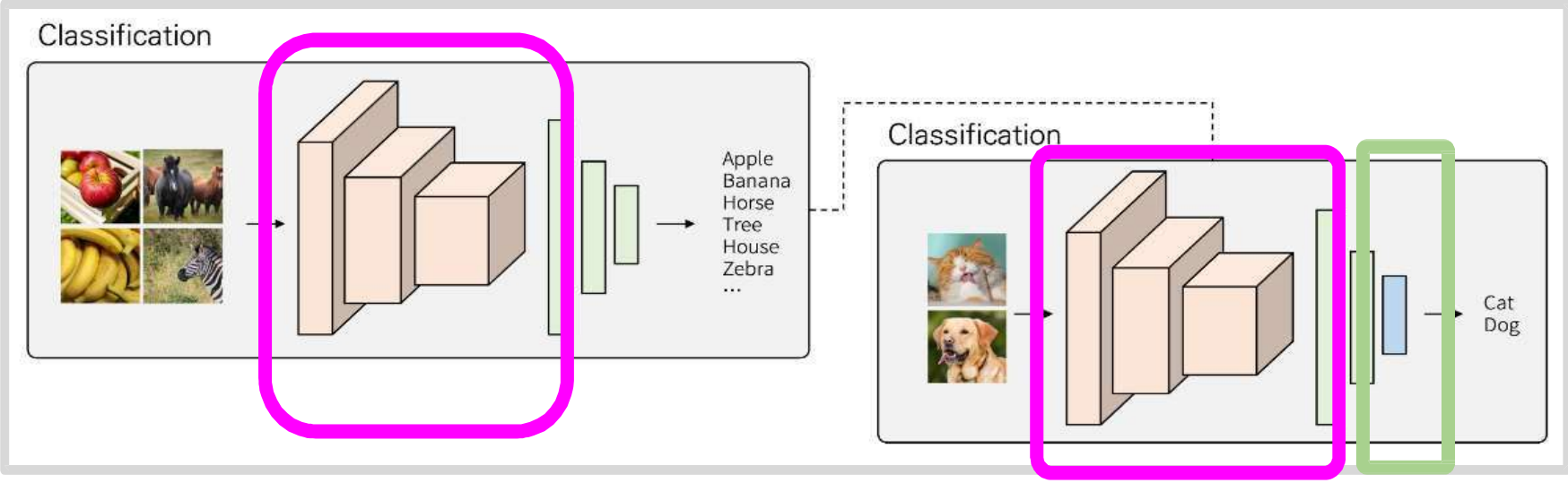
전이학습

- ImageNet 같은 **아주 큰 데이터셋을 사용하여 훈련된 모델**의 가중치 사용
- **기존 학습된 모델이 축적한 지식 활용**, 새로운 문제 더 빠르고 효과적으로 접근
- 커스텀 프로젝트에 맞도록 보정하여 사용
- 장점 : 데이터 부족 문제 해결 / 학습 시간 단축 / 성능 향상

TRANSFER LEARNING

◆ 전이학습이란?

- 특정 대상을 예측하도록 **훈련된 모델을 다른 대상에 사용**하는 것



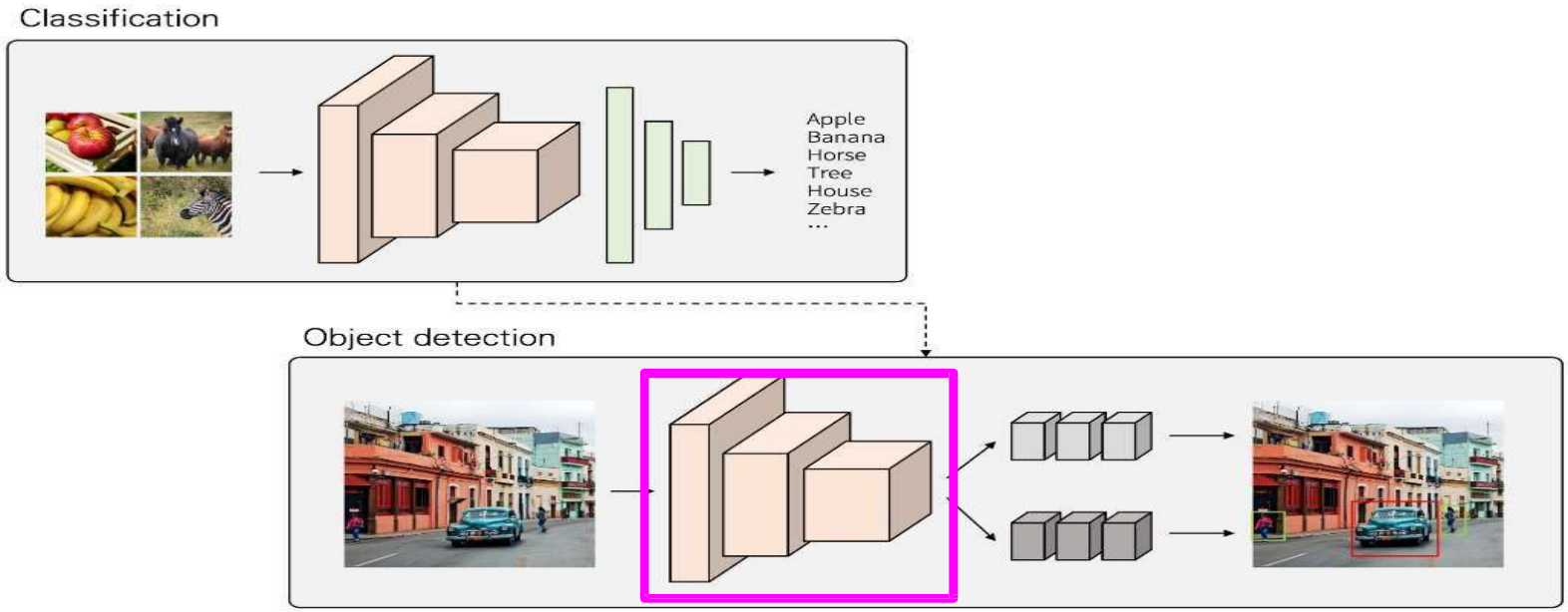
큰 데이터셋 학습 모델

작은 데이터셋 학습 모델

TRANSFER LEARNING

◆ 전이학습이란?

- 훈련된 모델을 다른 작업에 적용하는 학습



TRANSFER LEARNING

◆ 전이학습 방법

▪ 특성추출(Feature Extractor)

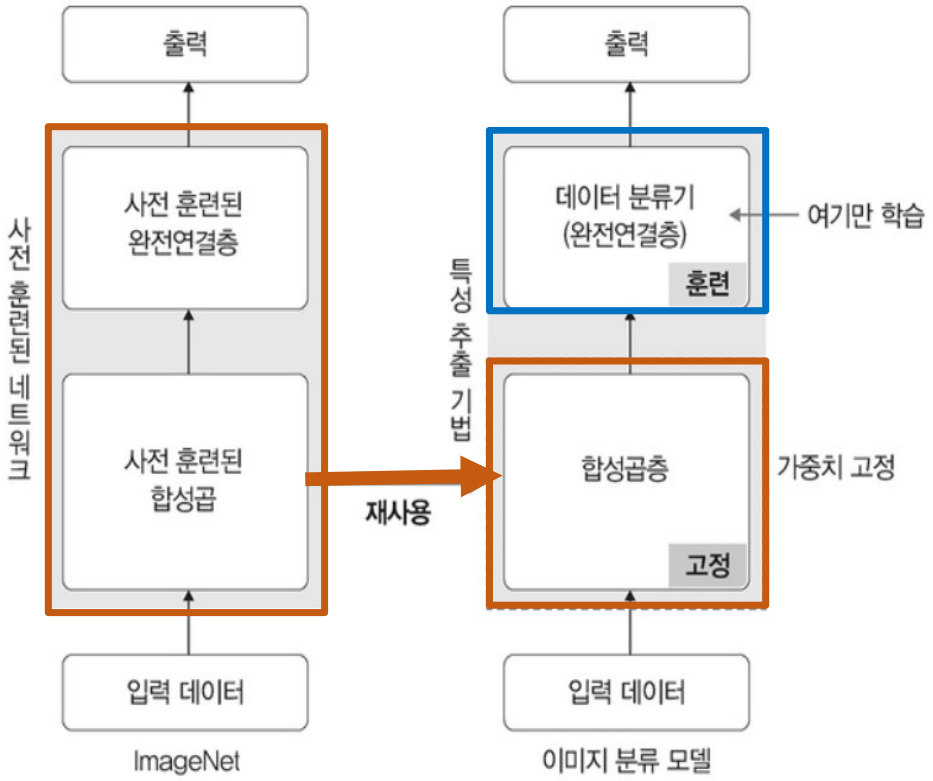
- ImageNet 데이터셋으로 **사전 훈련된 모델 가져와 마지막 완전연결층 부분만 새로 생성**
- 학습 시 마지막 **완전연결층만 학습, 나머지 계층들은 학습되지 않도록** 함
- 구성
  - 합성곱층: 합성곱층과 풀링층으로 구성
  - 데이터 분류기(완전연결층): 추출된 특성 입력받아 최종 이미지에 대한 클래스 분류

사전 훈련된 네트워크의 합성곱층(가중치 고정)에 새로운 데이터를 통과
그 출력을 데이터 분류기에서 훈련

TRANSFER LEARNING

◆ 전이학습 방법

- **특성추출(Feature Extractor)**



TRANSFER LEARNING

◆ 모델 역할 분류

- **Backborn Network**
  - 입력 **이미지의 feature map을 추출**시켜주는 **부분**
  - 사전학습된 모델로 **base model**이라고도 함
  - 대표 : VGG16, ResNet50, Xception, InceptionV3, MobileNet

- **Nect**
  - Backbone과 Head를 연결
  - **feature map을 refinement(정제), reconfiguration(재구성)**
  - 대표 : FPN, PAN, BiFPN, NAS-FPN

TRANSFER LEARNING

◆ 모델 역할 분류

- **Head**
  - Backbone에서 추출한 **feature map의 location 및 classification 작업** 수행
  - **하나의 Image에서 여러 객체를 효과적으로 detect** 하는 부분
  - 대표 : [1-Stage] YOLO, SSD, [2-Stage] Faster R-CNN, R-FCN

TRANSFER LEARNING

◆ 사용 이유 및 장점

- **학습 빠르게 수행**
  - 이미 입력되는 데이터에 대해 특징을 효율적으로 추출
  - 학습할 데이터에 대해 특징 추출하기 위한 별도 학습 필요 없음
- **작은 데이터셋 학습 시 오버피팅 예방**
  - 전이 학습을 이용해 마지막 레이어만 학습하여, 학습할 가중치 수가 줄어 과한 학습이 이루어지지 않게 할 수 있음

TRANSFER LEARNING

◆ 적용조건

- 사전 학습에 **사용한 데이터와** 새로운 **데이터가 비슷한 형태**
- 새로운 데이터보다 **많은 데이터로 사전 학습 수행**되어야 함

TRANSFER LEARNING

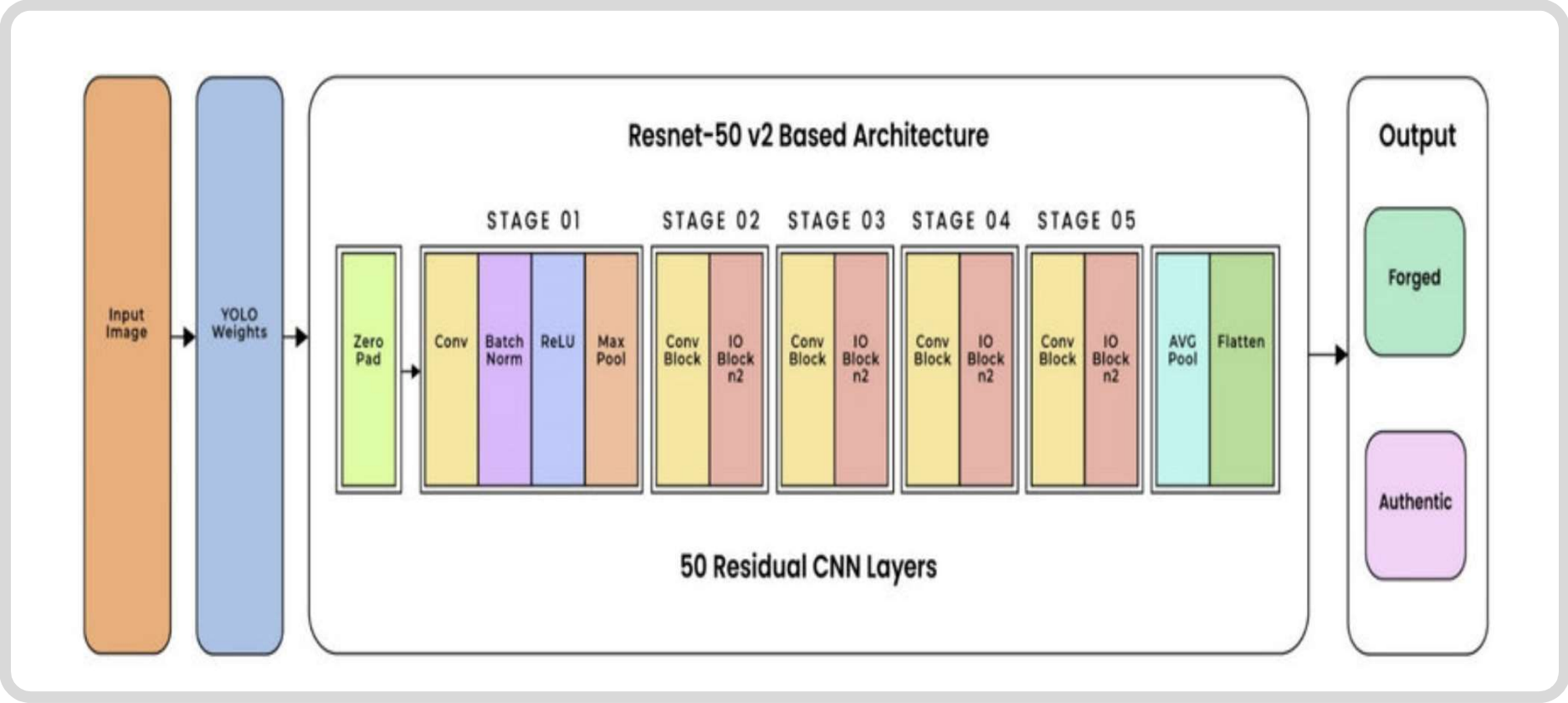
◆ Backborn : Resnet-50v2

- **50개 계층으로 구성된 컨벌루션 신경망**
- ResNet은 모델이 깊어질수록 최적화에서 멀어지는 것에 집중
  - 레이어가 깊어질수록 곱해지는 미분값들이 증가함
  - CNN 자체의 성능이 떨어지게 됨 ➔ **Residual Connection 기법** 해결

  - **Residual Connection** 란?
    layer와 layer 사이 건너뛰는 일종의 지름길 해당하는 연결

TRANSFER LEARNING

◆ Backborn : Resnet-50v2



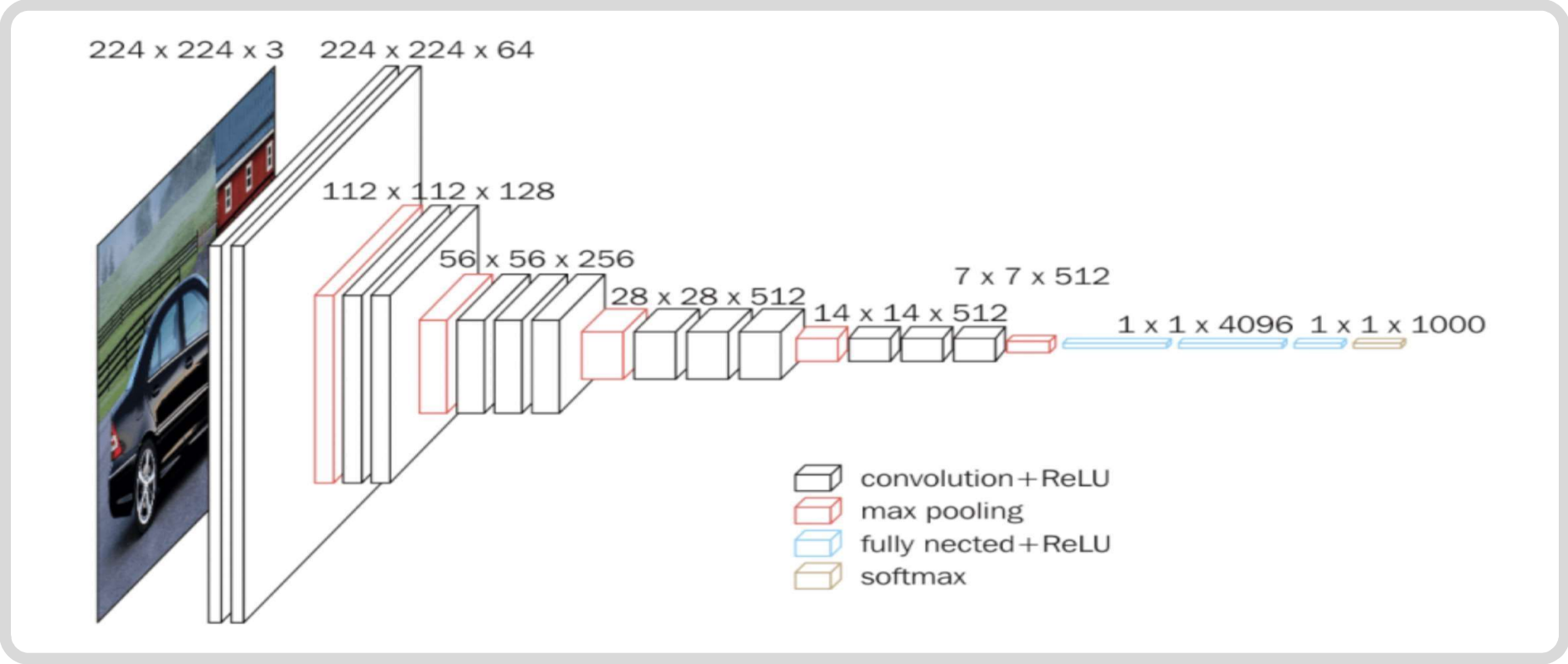
TRANSFER LEARNING

◆ Backborn : VGG16

- VGGNet-N(Very Deep Convolutional network for large-scale image recognition)
- 옥스포드 대학의 연구팀 VGG에 의해 개발된 모델
- **2014년 이미지넷 이미지 인식 대회에서 준우승**을 한 모델
- **16개 또는 19개의 Layer로 구성**된 모델
- **쉬운 구조와 좋은 성능 덕분에 우승 거둔 GoogLeNet(22Layer)보다 더 인기**

TRANSFER LEARNING

◆ Backborn : VGG16



TRANSFER LEARNING

◆ Fine-Tuning이란

- 전이 학습의 한 형태, **사전 훈련된 모델을 특정 작업/데이터셋 맞춰 추가 조정**
- 모델을 **특정 작업에 더욱 정밀하게** 맞추는 데 중점

- **방법 1 - 초기 가중치 활용**
  - ➢ 사전 훈련된 모델의 가중치를 시작점으로 사용
  - ➢ 이미 많은 일반적인 특징들을 학습한 상태, 이를 기반으로
  - ➢ 새로운 데이터에 대한 학습을 보다 효율적으로 진행

TRANSFER LEARNING

◆ Fine-Tuning이란

- **방법 2 - 조정과 재학습**
  - ➢ 모델 일부 또는 전체를 새로운 데이터셋에 맞게 조정/ 재학습
  - ➢ 특정 태스크에 더욱 최적화

- **방법 3 - 학습률 조절**
  - ➢ 학습률 낮추어 진행
  - ➢ 사전 훈련된 모델을 보존하면서 새로운 데이터에 맞춰 조정
  - ➢ 너무 높은 학습률은 사전 학습된 유용한 특징들을 손상

TRANSFER LEARNING

◆ 전이학습

- **ResNet18**

```
import torch
import torchvision.transforms as transforms # 데이터 전처리 패키지
import torchvision.models as models         # 다양한 모델 패키지
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader

import matplotlib.pyplot as plt

### ===> GPU 연산 확인
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(f'{device} is available.')
```

TRANSFER LEARNING

◆ 전이학습

- **ResNet18**

```
### ===> 사전학습된 모델 로딩
resnet18 = models.resnet18().to(device)

### ===> 사전 훈련된 모델의 파라미터 학습 유무 설정 함수
def set_parameter_requires_grad(model, feature_extracting = True):
    if feature_extracting:
        for param in model.parameters():
            param.requires_grad = False ## 학습 불가

set_parameter_requires_grad(resnet18)
```

TRANSFER LEARNING

◆ 전이학습

- **ResNet18**

```
### ===>  ResNet18에 완전연결층 추가
resnet18.fc = nn.Linear(512,2) # 2는 클래스가 두 개라는 의미


### ===> 모델의 파라미터 값 확인
for name, param in resnet18.named_parameters():
    if param.requires_grad:
        print(name, param.data)
```

TRANSFER LEARNING

◆ 전이학습

- **ResNet18**

```
# 모델 인스턴스 생성
model = models.resnet18(weights=ResNet50_Weights.DEFAULT)

# 모델의 합성곱층 가중치 고정
for param in model.parameters():
    param.requres_grad = False

model.fc = torch.nn.Linear(512,2)
for param in model.fc.parameters(): # 완전연결층은 학습
    param.requires_grad = True

optimizer = torch.optim.Adam( model.fc.parameters() )
cost = torch.nn.CrossEntropyLoss() # 손실 함수 정의
print(model)
```