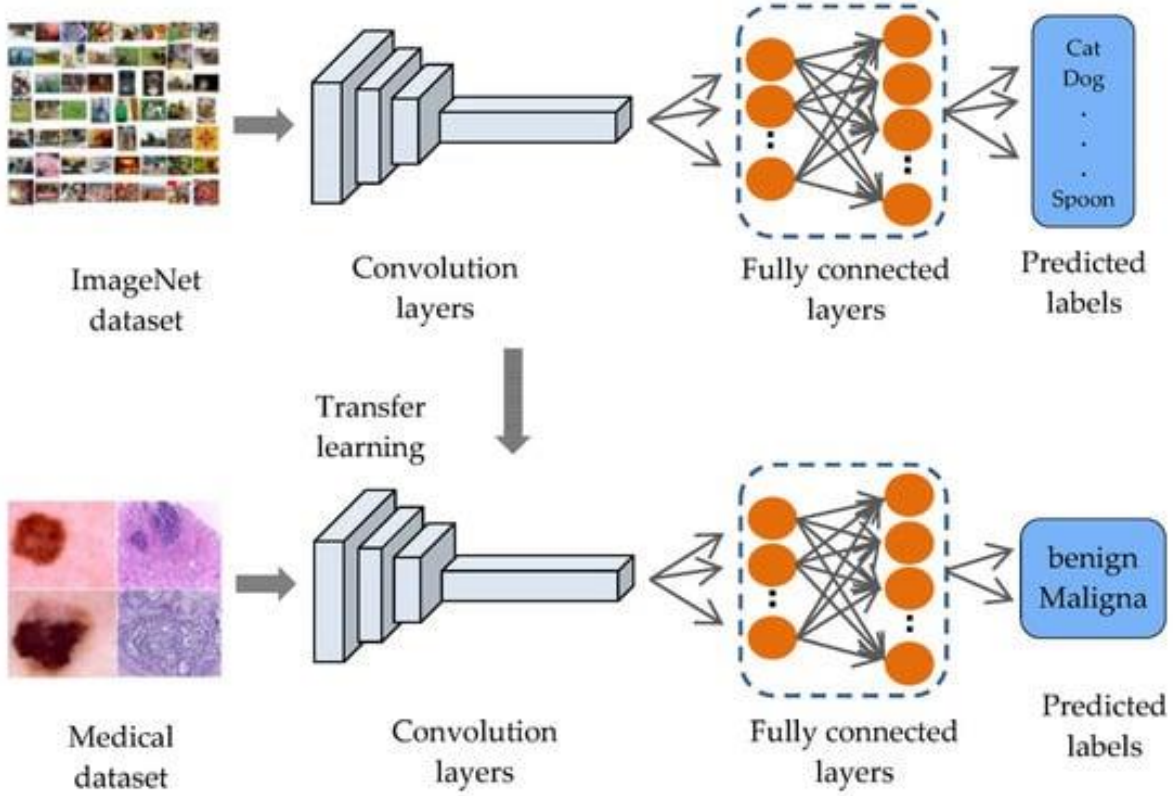
AI 실전 3주차

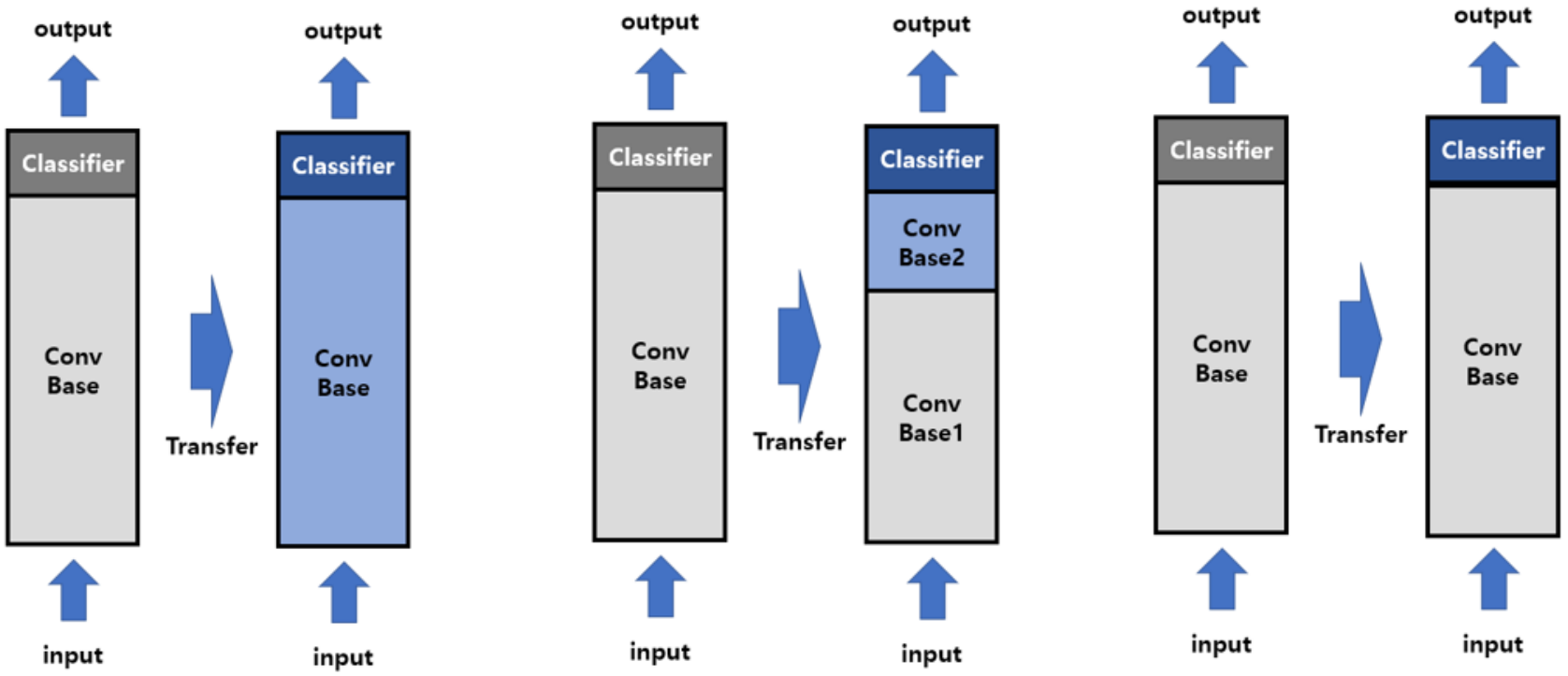
이미 학습이 되어 있는 모델을 학습하여 이용하기

전이 학습

전이 학습


Cat
Dog
Spoon
ImageNet dataset
Convolution layers
Fully connected layers
Predicted labels
Transfer learning
benign
Maligna
Medical dataset
Convolution layers
Fully connected layers
Predicted labels

전이 학습 적용



Strategy 1
: 모델 전체를 학습하는 경우

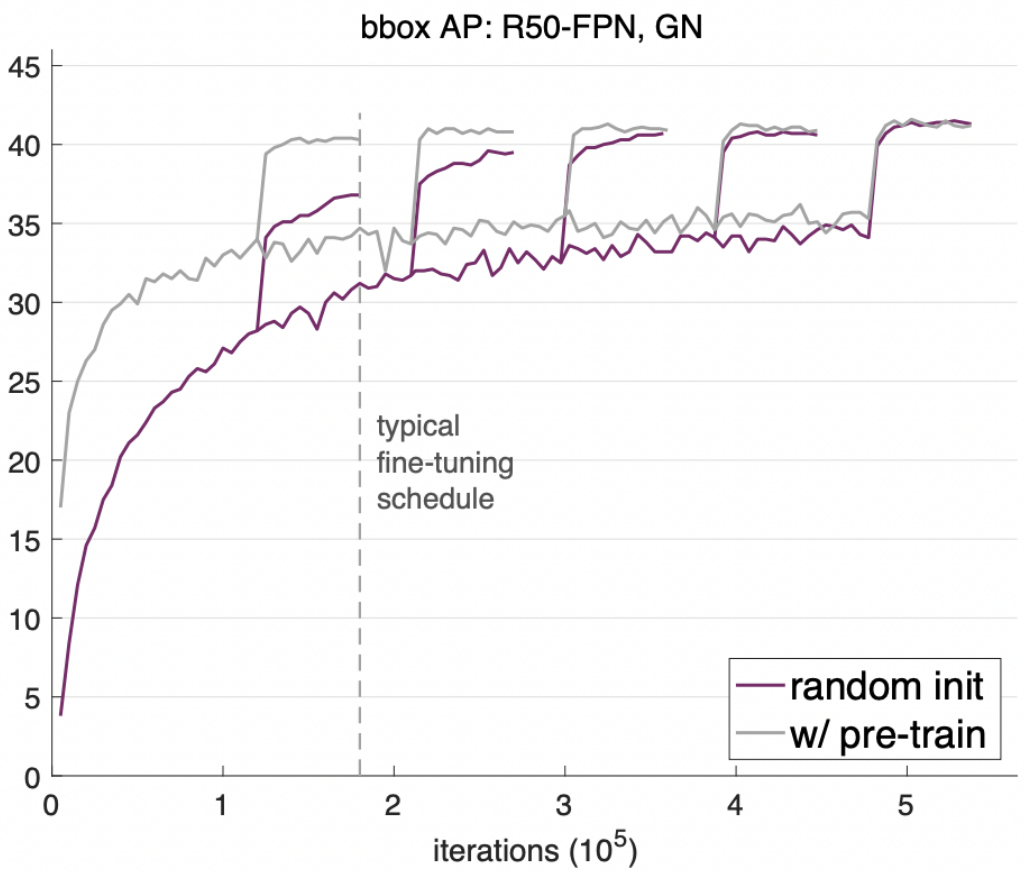
Strategy 2
: 일부만 학습시키는 경우

Strategy 3
: 학습 과정 없이 사용하는 경우

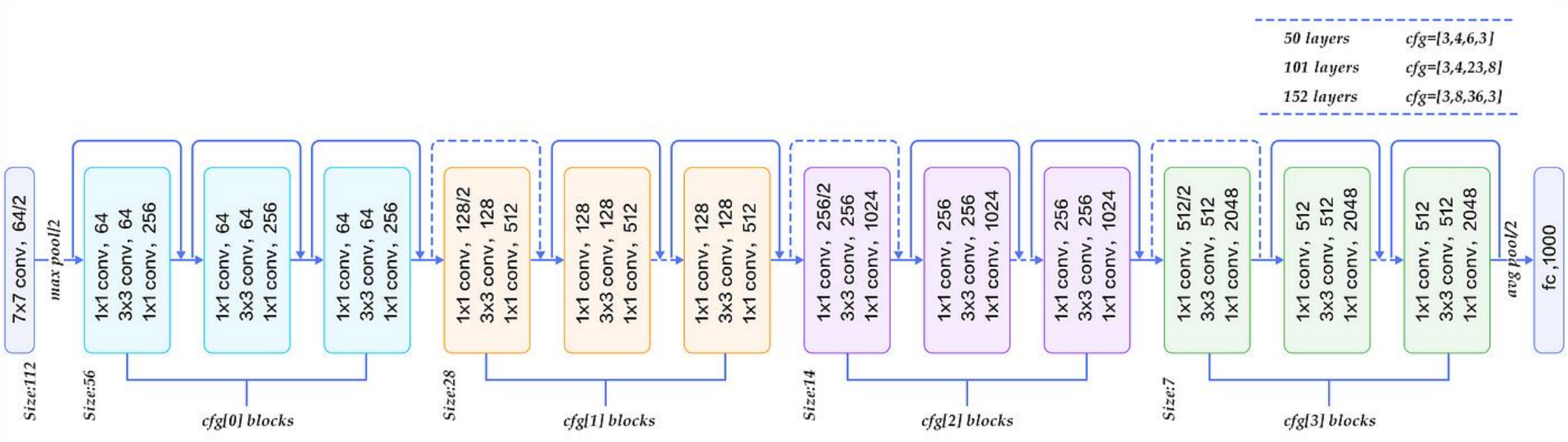
최근 파인튜닝 적용

- P-tuning
- Prefix tuning
- Pormpt tuning
- LoRA
- …

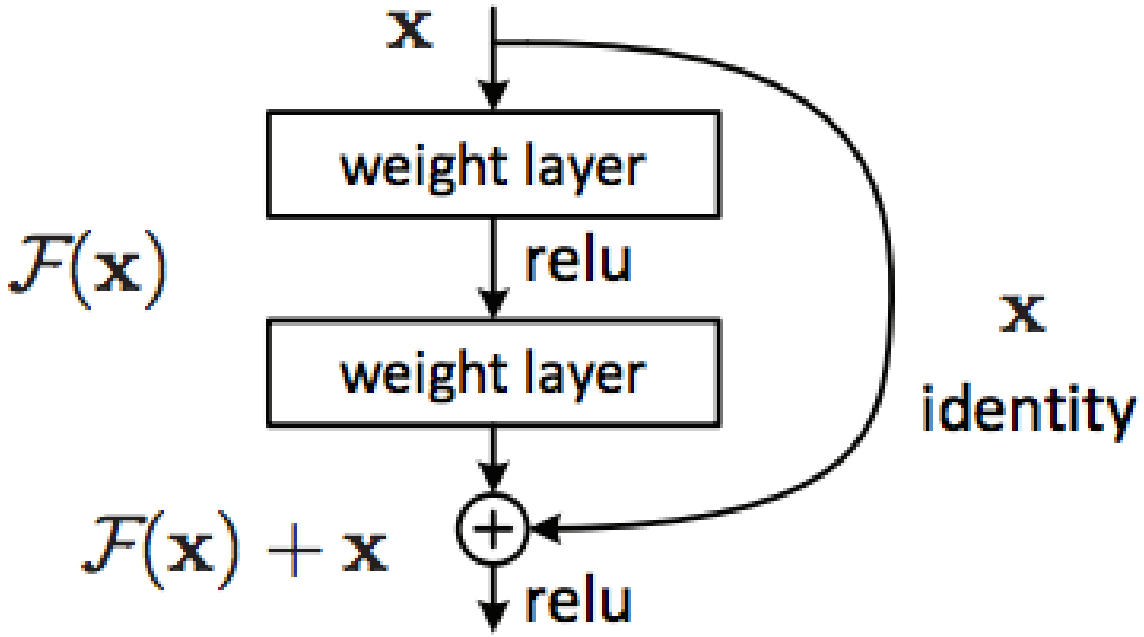
Q. 굳이 학습된 모델 안써도 되잖..?



resnet50


7x7 conv, 64/2
max pool/2
Size:112
1x1 conv, 64
3x3 conv, 64
1x1 conv, 256
Size:56
cfg[0] blocks
1x1 conv, 128/2
3x3 conv, 128
1x1 conv, 512
1x1 conv, 128
3x3 conv, 128
1x1 conv, 512
Size:28
cfg[1] blocks
1x1 conv, 256/2
3x3 conv, 256
1x1 conv, 1024
1x1 conv, 256
3x3 conv, 256
1x1 conv, 1024
Size:14
cfg[2] blocks
1x1 conv, 512/2
3x3 conv, 512
1x1 conv, 2048
1x1 conv, 512
3x3 conv, 512
1x1 conv, 2048
Size:7
cfg[3] blocks
avg pool/2
fc ,1000
50 layers cfg=[3,4,6,3]
101 layers cfg=[3,4,23,8]
152 layers cfg=[3,8,36,3]

Resnet 적용한 기술 (skip connections)



필요한 라이브러리 불러오기

```
import torchvision.transforms as transforms
import torchvision.datasets as datasets
from torch.utils.data import DataLoader
```

학습할 데이터 변환

```
# 데이터 변환 정의
transform = transforms.Compose([
transforms.Resize((224, 224)),
transforms.ToTensor(),
])
```

다운로드 안되는 학생을 위한 SSL 등록

```
# pip install certifi
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
```

0. Download CIFAR10 Dataset

```
train_dataset = datasets.CIFAR10(root='./data', train=True,
transform=transform, download=True)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
```

Load resnet50 model

```
import torch
import torch.nn as nn
import torchvision.models as models

# 모델 불러오기
model = models.resnet50(pretrained=True)
for param in model.parameters():
    param.requires_grad = False
```

OUTPUT Layer 데이터 맞춰서 변경

```
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs,10)
```

모델 학습 함수 생성

```
def train_model(model, dataloader, criterion, optimizer, num_epochs=10, device='cuda'):
    model. to (device)
    model. train()
    for epoch in range(num_epochs) :
        running_loss = 0.0
        for inputs, labels in dataloader:
            inputs, labels = inputs. to(device), labels. to(device)
            optimizer.zero_grad()
            outputs = model (inputs)
            loss = criterion (outputs, labels)
            loss. backward()
            optimizer.step ()
            running_loss += loss. item()
            print(f'Epoch {epoch+1}/{num_epochs}, Loss:{running_loss/len(dataloader)}')
            print( 'Training complete')
```

4. 모델 학습 함수

```python
def train(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)

        # 예측 오류 계산
        pred = model(X)
        loss = loss_fn(pred, y)

        # 역전파
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

        if batch % 100 == 0:
            loss, current = loss.item(), (batch + 1) * len(X)
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")
```

모델 학습

```
import torch.optim as optim

# 학습 설정
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.fc.parameters(),lr=0.001)
# 모델 학습
train_model(model,train_loader, criterion, optimizer, num_epochs=5,device='cpu')
```