전이 학습 (Transfer Learning)

- **정의**: 전이 학습은 대규모 데이터셋(예: ImageNet)으로 사전 훈련된 모델을 활용하여, 새로운 과제에 맞게 보정하여 사용하는 방법입니다.
- **장점**: 사전 훈련된 모델을 활용함으로써, 적은 양의 데이터로도 높은 성능을 낼 수 있습니다.

전이 학습 기법

- **특성 추출 (Feature Extraction)**:
    - 사전 훈련된 모델의 마지막 완전 연결층만 학습시키고, 나머지 층들은 고정시킵니다.
    - 예를 들어, ImageNet으로 사전 훈련된 모델을 사용하고, 완전 연결층을 새로운 데이터셋에 맞게 학습시킵니다.
- **미세 조정 (Fine-tuning)**:
    - 사전 훈련된 모델의 마지막 몇 개 층을 학습시켜서 새로운 과제에 맞게 조금 더 세부적인 조정을 가합니다.

사전 훈련된 모델 예시

- Xception
- Inception V3
- ResNet50
- VGG16, VGG19
- MobileNet

전처리 기법

- Resize: 이미지 크기를 지정된 크기(256x256)로 조정합니다.
- RandomResizedCrop: 랜덤한 비율로 자르고 다시 크기 조정.
- RandomHorizontalFlip: 이미지를 수평으로 랜덤하게 뒤집음.
- ToTensor: 이미지를 텐서로 변환하여 학습에 사용.

OpenCV

- **정의**: OpenCV는 Open Source Computer Vision Library의 약자로, 이미지와 비디오 처리를

위한 라이브러리입니다.

- OpenCV-Python 라이브러리를 설치하면 파이썬에서 OpenCV 기능을 사용할 수 있습니다.

반복자 (Iterator)

- **iter()와 next()**:
    - iter()는 전달된 데이터의 반복자를 반환.
    - next()는 그 반복자에서 다음 값을 반환.
    - 예시: train_loader에서 데이터를 순차적으로 꺼내서 samples와 labels로 저장.
    - 반복자를 사용해 train_loader에서 데이터를 하나씩 꺼내올 수 있음.

np.transpose() 함수

- 행렬의 차원을 바꾸기 위해 사용.
- 일반적으로 행렬 내적을 할 때 사용하며, 행과 열의 크기가 맞지 않을 경우 np.transpose() 또는 np.reshape()로 조정 가능.
- 예시: samples.shape 출력 시 (32, 3, 224, 224) 형태의 데이터가 나오며, 이를 np.transpose(samples[1].numpy(), (1, 2, 0))로 변환해 (224, 224, 3) 형태로 바꿈.

ResNet18 설명

- **ResNet18**: 50개의 계층으로 구성된 합성곱 신경망.
    - ImageNet 데이터베이스의 100만 개 이상의 이미지로 훈련된 사전 훈련 모델 제공.
    - 전이 학습에 사용되며, 입력 제한이 큼.
    - 충분한 메모리(RAM)가 없으면 학습 속도가 느릴 수 있음.

사전 훈련된 모델의 파라미터 고정

- 특정 계층의 파라미터를 고정하고 학습하지 않도록 설정 가능.
- 예시: set_parameter_requires_grad() 함수를 사용하여 파라미터의 requires_grad 속성을 False로 설정.
- 합성곱 층과 풀링 층의 파라미터를 고정하여 일부만 학습 가능.

requires_grad = False 설명

- **requires_grad = False**는 역전파 중에 파라미터가 업데이트되지 않음을 의미.

- 사전 훈련된 모델의 일부를 고정하고, 나머지를 학습할 때 사용.
- 합성곱 층과 풀링 층에 적용하여 모델의 일부를 고정.

파라미터 값

- **파라미터(Parameters)**는 weight와 bias로 구성됨.

파일 가져오기

- **pth 확장자 파일**: .pth는 모델 훈련 후 생성된 파일.
- 디렉터리에서 특정 확장자를 가진 파일만 가져오려면 사용.

torch.max() 함수

- 주어진 텐서 배열에서 최대값을 포함하는 **인덱스 반환**.
    - **예시**: y_pred = [[0.2, 0.7, 0.8, 0.4]]에서 torch.max(y_pred.data, 1)을 적용하면 최대값은 0.8, 그 인덱스는 2이므로 반환 값은 2.

preds.eq(labels)

- preds 배열과 labels 배열이 **일치하는지 검증**.
- 뒤에 사용된 sum()은 예측 결과와 정답(label)이 일치하는 개수를 출력.

모델 평가 및 정확도 측정

- **모델 평가 함수**를 테스트 데이터에 적용하여 **성능(정확도)**을 측정.
- 사전 훈련된 모델을 사용하면 직접 네트워크를 구현하고 최적의 파라미터 값을 찾는 데 걸리는 시간을 크게 줄일 수 있음.

훈련 및 테스트 데이터 정확도 시각화

- **정확도 그래프**: 훈련과 테스트 데이터에 대해 에포크가 진행될 때마다 정확도를 시각적으로 확인 가능.
- 에포크가 많아질수록 훈련과 테스트 데이터 모두 100%에 가까워짐.

tensor.clone()와 tensor.detach()

- **tensor.clone()**: 기존 텐서의 내용을 복사한 새로운 텐서 생성. 계산 그래프에 계속 포함됨.
- **tensor.detach()**: 기존 텐서를 공유하되 계산 그래프에 포함되지 않음.
- **tensor.clone().detach()**: 새롭게 할당되지만 계산 그래프에 포함되지 않음.

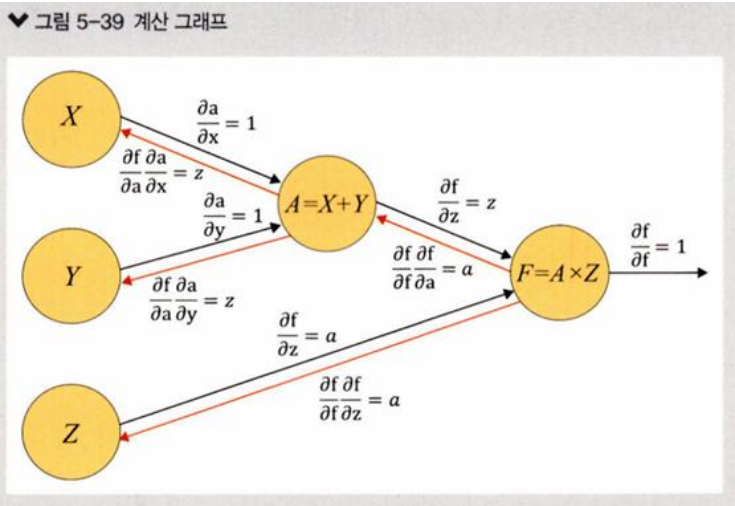
❯ 표 5-2 tensor.clone(), tensor.detach(), tensor.clone().detach()의 비교

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

계산 그래프 (Computational Graph)

- 계산 과정을 그래프로 나타낸 것. **노드(Node)**와 **엣지(Edge)**로 구성.
- 계산 그래프를 사용하는 이유:
    1. **국소적 계산**이 가능함. 일부 값이 변경되면 나머지 계산을 그대로 유지한 채 변경된 값만 다시 계산.
    2. **역전파**를 통한 미분 계산이 편리함. **연쇄 법칙(Chain Rule)**을 사용하여 미분을 빠르고 간편하게 계산 가능.

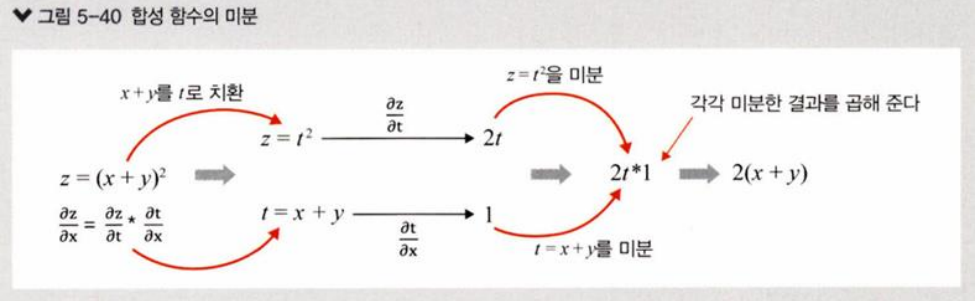
❯ 그림 5-39 계산 그래프



연쇄 법칙 (Chain Rule)

- 두 개 이상의 함수가 결합된 **합성 함수**의 미분을 계산하는 법칙.

❯ 그림 5-40 합성 함수의 미분



clip() 함수

- clip() 함수는 주어진 입력 값이 특정 범위를 벗어날 경우, 값을 해당 범위 내로 제한시키기 위해 사용됩니다.
- 예시로 image.clip(0, 1)은 이미지 데이터 값을 0과 1 사이로 제한하는 것입니다. 즉, 이미지 픽셀 값이 0보다 작으면 0으로, 1보다 크면 1로 변경합니다.

add_subplot() 함수

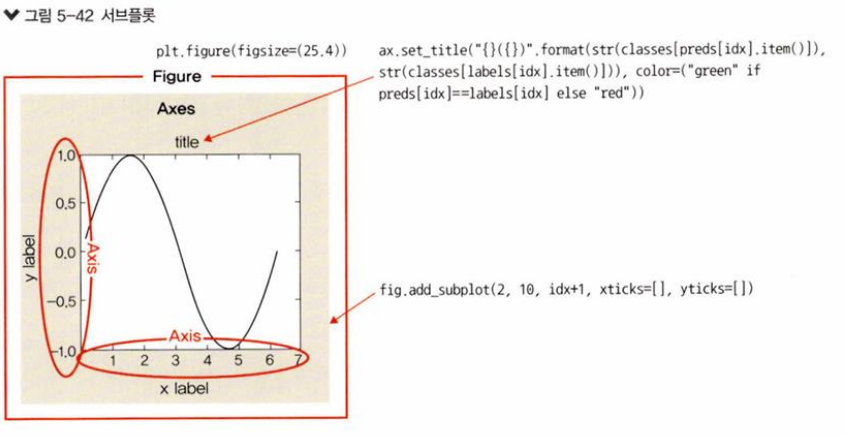
- add_subplot() 함수는 여러 개의 이미지를 한 화면에 배치하기 위해 사용됩니다.
- 파라미터는 다음과 같습니다:
    - 첫 번째 파라미터: 행(row)의 수
    - 두 번째 파라미터: 열(column)의 수
    - 세 번째 파라미터: 인덱스 (이미지가 출력될 위치)
    - xticks, yticks: x축과 y축의 눈금을 제거하기 위해 사용.

클래스 분류 결과

- 예측된 값이 0이면 "고양이", 1이면 "개"로 출력됩니다. 이때 classes[preds[idx].item()] 코드를 사용하여 예측 값을 확인합니다.

서브플롯 위치 조정

- Figure 안에서 서브플롯(subplot)의 위치를 조정할 수 있습니다. left, bottom, right, top 파라미터를 사용해 위치를 변경합니다.
- hspace, wspace는 서브플롯 간의 간격을 조정합니다 (가로와 세로 간격 조절).



미세 조정(fine-tuning) 기법

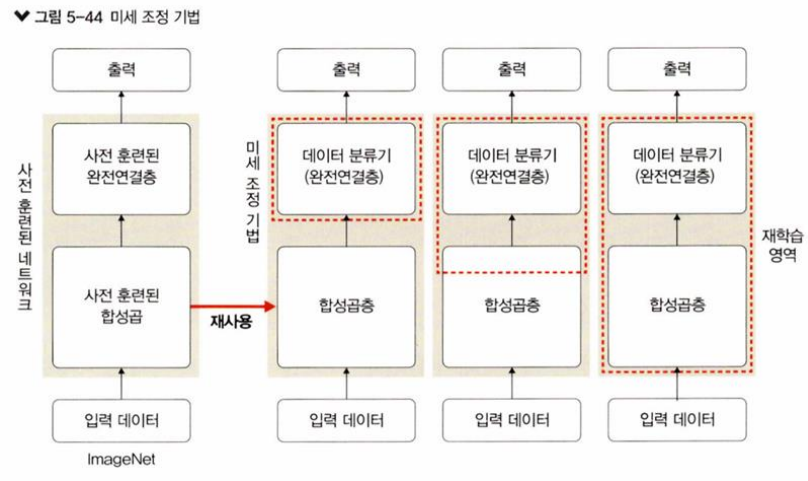
- 미세 조정 기법은 사전 학습된 모델의 일부 가중치를 업데이트하여 모델을 재학습시키는 방식입니다. 주로 사전 학습된 모델의 특성이 목표 데이터와 다를 경우 사용됩니다.
- 예를 들어, ImageNet 데이터셋으로 학습된 모델이 다른 이미지 데이터셋(예: 전자상거래 이미지)에서 잘 작동하지 않는다면, 미세 조정을 통해 특성을 다시 추출할 수 있습니다.

미세 조정 적용 전략:

- **데이터셋이 크고, 사전 학습된 모델과 유사성이 작을 경우**: 모델 전체를 재학습시킵니다.
- **데이터셋이 크고, 유사성이 큰 경우**: 합성곱 층 중 뒷부분(완전 연결 층에 가까운 부분)과 데이터 분류기만 학습시킵니다.
- **데이터셋이 작고, 유사성이 작을 경우**: 합성곱 층의 일부와 데이터 분류기를 학습시킵니다. 데이터가 적어 과적합 위험이 있기 때문에 조심해야 합니다.
- **데이터셋이 작고, 유사성이 큰 경우**: 데이터 분류기만 학습시킵니다.

미세 조정의 필요성

- 미세 조정 기법은 사전 학습된 네트워크를 분석하려는 데이터셋에 맞게 조정하는 과정입니다.
- 많은 연산량이 필요하기 때문에 CPU보다 GPU를 사용하는 것이 좋습니다.



설명 가능한 CNN (Explainable CNN)

설명 가능한 CNN은 딥러닝 처리 결과를 사람이 이해할 수 있는 방식으로 설명하는 기술입니다. CNN은 내부에서 어떻게 동작하는지 알기 어려운 블랙박스(Blackbox) 구조입니다. 이러한 블랙박스 문제로 인해 CNN의 결과에 대한 신뢰성이 떨어질 수 있습니다. 이를 해결하기 위해 CNN의 처리 과정을 시각화하여, 결과를 더 신뢰할 수 있도록 해야 합니다.

CNN 시각화 방법

CNN의 시각화는 CNN을 구성하는 각 계층에서 입력된 이미지가 어떻게 처리되고, 학습되는지 보여줍니다. 여기서 주로 다루는 시각화 방법은 **특성 맵 시각화**입니다.

특성 맵 시각화

특성 맵(Feature Map, 활성화 맵)은 입력 이미지나 다른 특성 맵처럼 필터를 적용한 결과입니다. 즉, CNN이 입력 이미지에서 어떤 특성을 감지하는지 시각화하여 이해할 수 있습니다.

특성 맵 시각화는 CNN의 처리 과정을 명확하게 보여주기 때문에, 결과의 신뢰성을 높이는 데 중요한 역할을 합니다.

로그 소프트맥스 (LogSoftmax)

LogSoftmax는 신경망 말단에서 출력된 값들을 확률로 해석하기 위해 소프트맥스 함수에 로그 값을 취한 연산입니다. 일반 소프트맥스는 기울기 소멸 문제에 취약하기 때문에, 로그 소프트맥스를 사용하는 것입니다.

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1} e^{x_j}}$$

$$\text{logsoftmax} = \log\left(\frac{e^{x_i}}{\sum_{j=1} e^{x_j}}\right)$$

$$= x_i - \log\left(\sum_{j=1} e^{x_j}\right)$$

PyTorch를 사용한 CNN 구현

> **Hook 기능**: PyTorch에서는 각 계층의 출력값을 register_forward_hook 함수를 사용해 쉽게 추적할 수 있습니다. 이를 통해 순전파(forward propagation) 중에 각 계층의 입력과 출력 값을 확인할 수 있습니다. 이를 통해 CNN 내부의 특성 맵을 쉽게 확인할 수 있습니다.

코드에서는 각 텐서의 기울기(gradient)를 확인할 수 있습니다. PyTorch는 중간 계산 결과에 대해 기울기 값을 자동으로 저장하지 않지만, Hook을 사용하여 중간 값들을 확인할 수 있습니다.

이미지 처리 예제

이미지 크기를 변경하기 위해 cv2.resize 함수를 사용합니다.
첫 번째 매개변수는 이미지 파일을, 두 번째는 변경할 이미지 크기 (너비, 높이)를 지정합니다. 세 번째 매개변수는 보간법(interpolation)을 설정합니다.

- 보간법은 이미지 크기를 변경할 때 새 픽셀 값을 추정하여 부여하는 방식입니다. 픽셀 데이터를 압축하거나 새 픽셀을 생성하는 데 사용됩니다.

또한, 이미지 데이터를 텐서로 변환할 때 ToTensor와 unsqueeze(0)를 사용하여 이미지 데이터를 1차원 텐서로 변경합니다.

그래프 신경망

그래프 구조에서 사용되는 신경망을 의미합니다.

1단계 : 인접행렬

2단계 : 특성행렬

합성곱 네트워크 구조