

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
Intro		<ul style="list-style-type: none"><li>내 손안의 AI 눈, 이미지 분류의 마법</li></ul> <p>우리는 매일 수많은 이미지를 접하고, 그 안에서 의미 있는 정보를 찾아냅니다. 사람에게는 너무나 자연스러운 이 과정이, 컴퓨터에게는 엄청나게 복잡한 과제였습니다. 하지만 이제는 인공지능이 마치 사람처럼 이미지를 '보고' '이해'하고 '분류'할 수 있게 되었습니다.오늘 우리는 이러한 이미지 분류의 핵심 기술인 CNN(Convolutional Neural Network), 즉 합성곱 신경망에 대해 알아보는 시간을 가질 것입니다. 특히, 파이토치(PyTorch)라는 강력한 딥러닝 프레임워크를 활용하여, 컴퓨터가 어떻게 수많은 컬러 이미지를 정확하게 분류할 수 있는지 그 원리와 실제 구현 방법을 깊이 있게 탐구해 볼 예정입니다. 데이터를 어떻게 준비하고, 모델은 어떻게 설계하며, 학습 과정에서 발생할 수 있는 문제들을 어떻게 해결하는지 등, 마치 여러분이 인공지능 모델에게 '눈'을 만들어주는 과정이라고 생각하시면 흥미를 느끼실 겁니다. 여러분의 손끝에서 인공지능이 이미지를 인식하고 분류하는 마법 같은 경험을 직접 구현해 볼 준비가 되셨나요? 함께 그 비밀을 파헤쳐 봅시다!</p>			<ul style="list-style-type: none"><li>본 학습 내용으로 들어가기 전, 학습 주제의 흥미를 이끌 만한 도입부의 내용이 있다면 제시해주세요.</li><li>ex. 관련 뉴스기사, 실생활과 관련된 이야기 등</li><li>저작권 침해가 되지 않도록 내용을 구성해 주세요.</li><li>출처가 있을 경우 반드시 작성해 주세요.</li></ul>
학습열기					
학습목표					
학습하기					
1. CIFAR-10 데이터 전처리					
2. CNN 구현 (전반부)					
적용하기					
Outro					
문제풀기					
내레이션		3			

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명	
과정명	➤Intro	<div>◆ 학습목표</div> <div>1. 데이터 증강 및 정규화 기법을 적용할 수 있다.</div> <div>2. 합성곱 층과 풀링 층을 활용하여 모델을 작성할 수 있다.</div> <div>◆ 학습내용</div> <div>1. CIFAR-10 데이터 전처리</div> <div>2. CNN 구현 (전반부)</div>			① 학습내용과 학습목표는 강의계획서와 일치해야 하며, 필요시 강의계획서를 수정할 수 있습니다.	
	•학습열기				② 학습목표	✓ 각 레슨에 맞는 학습 목표를 2~3개 작성해 주세요.
	•학습목표				③ 학습내용	✓ 1회차 당 25분 분량이 되도록 2~3개 레슨으로 구성해주세요.
➤ 학습하기	1. CIFAR-10 데이터 전처리				✓ 학습내용과 레슨명은 일치해야 합니다.	
	2. CNN 구현 (전반부)					
➤ 적용하기						
➤ Outro						
	•문제풀기					
내레이션	4					

과정명		PyTorch로 배우는 머신러닝 알고리즘		회차명	8	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. CIFAR-10 데이터 전처리</div> <div>2. CNN 구현 (전반부)</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div><div>간지</div><div>CIFAR-10 데이터 전처리</div></div>					
	<div>용어설명</div>					
내레이션	5					

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
<div> <div>➤Intro</div> <div> <div>•학습열기</div> <div>•학습목표</div> </div> <div>➤학습하기</div> <div> <div>1.CIFAR-10 데이터 전처리</div> <div>2.CNN 구현 (전반부)</div> </div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div> </div>	<div> <div>• 데이터 전처리와 CNN 구현</div> <div>필요한 라이브러리 메모리 로드</div> </div> <div> <pre> import torch import torch.nn as nn import torch.optim as optim import torchvision import torchvision.transforms as transforms import matplotlib.pyplot as plt import numpy as np from torchsummary import summary </pre> </div>				
					용어설명
내레이션					6

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. CIFAR-10 데이터 전처리</div> <div>2. CNN 구현 (전반부)</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• 데이터 전처리 구성 함수: transforms.Compose([...])</div> <div>전처리 모듈에서 여러 개의 변환 작업(transforms)을 순서대로 묶어주는 함수</div> <div>전처리 파이프라인 전체를 구성: 여러 단계의 이미지 처리 로직이 순차적으로 수행</div> <div>리스트 형태로 묶어서 하나의 통합 변환 함수처럼 작동하게 만드는 도구</div> <div>아직 데이터가 없어도, 앞으로 들어올 데이터를 어떻게 전처리할지 미리 정의해두는 전처리 파이프라인을 만드는 방법으로, 일종의 설계도이자 가공기계 준비 작업</div> <div>실제 데이터는 나중에 DataLoader를 통해 들어와서 이 파이프라인을 거치게 됨</div> <div># 1. 데이터 전처리</div> <div>transform_train = transforms.Compose([     transforms.RandomHorizontalFlip(), # 데이터 증강     transforms.RandomCrop(32, padding=4), # 데이터 증강     transforms.ToTensor(), # 데이터 스케일 조정     transforms.Normalize((0.4914, 0.4822, 0.4465), # 데이터 정규화                           (0.2023, 0.1994, 0.2010)) ])  trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform_train)</div>			용어설명

내레이션

7

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
➤Intro •학습열기 •학습목표 ➤학습하기 1. CIFAR-10 데이터 전처리 2. CNN 구현 (전반부)  ➤적용하기 ➤Outro •문제풀기	• 이미지 데이터를 학습용(train)과 테스트용(test)으로 전처리 설계 이미지를 모델이 잘 학습하도록 변환하는 표준 절차 transforms.Compose를 사용 학습용은 데이터 증강(data augmentation)을 수행 테스트용은 정규화만 수행  # 1. 데이터 전처리 transform_train = transforms.Compose([ transforms.RandomHorizontalFlip(), # 데이터 증강 transforms.RandomCrop(32, padding=4), # 데이터 증강 transforms.ToTensor(), # 데이터 스케일 조정 transforms.Normalize((0.4914, 0.4822, 0.4465), # 데이터 정규화 (0.2023, 0.1994, 0.2010)) ])  transform_test = transforms.Compose([ transforms.ToTensor(), transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)) ])				
					용어설명
내레이션	8				

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
➤Intro <ul style="list-style-type: none"><li>•학습열기</li><li>•학습목표</li></ul> ➤학습하기 <div>1. CIFAR-10 데이터 전처리</div> <div>2. CNN 구현 (전반부)</div>	<ul style="list-style-type: none"><li>• 데이터 전처리 설계 구현 1</li></ul> <pre>transform_train = transforms.Compose([...])</pre> <p>학습 데이터에 대해 순차적으로 적용할 여러 전처리 작업을 설계</p> <p>transforms.RandomHorizontalFlip(): 50% 확률로 이미지를 좌우로 뒤집음</p> <p>데이터 다양성 증가 → 과적합 방지에 도움</p> <p>transforms.RandomCrop(32, padding=4): 원래 이미지 주변에 4픽셀 패딩(0으로 채움) 추가</p> <p>그 후 무작위로 32×32 크기로 잘라(crop) 냄</p> <p>이미지를 살짝 흔들어서 학습시킴 → 위치 변화에 대한 일반화 능력 향상</p> <pre># 1. 데이터 전처리 transform_train = transforms.Compose([     transforms.RandomHorizontalFlip(), # 데이터 증강     transforms.RandomCrop(32, padding=4), # 데이터 증강     transforms.ToTensor(), # 데이터 스케일 조정     ... ])</pre>			용어설명
				➤적용하기
➤Outro <ul style="list-style-type: none"><li>•문제풀기</li></ul>				

내레이션

9

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
과정명	➤Intro	•학습열기 transform_train = transforms.Compose([...]) 학습 데이터에 대해 순차적으로 적용할 여러 전처리 작업을 설계 transforms.ToTensor() 이미지나 NumPy 배열을 PyTorch Tensor로 변환 [0, 255] 범위의 픽셀 값을 [0.0, 1.0] 범위로 스케일링  # 1. 데이터 전처리 transform_train = transforms.Compose([ transforms.RandomHorizontalFlip(), # 데이터 증강 transforms.RandomCrop(32, padding=4), # 데이터 증강 transforms.ToTensor(), # 데이터 스케일 조정 ... ])			
	•학습목표				
	➤학습하기				
	1. CIFAR-10 데이터 전처리				
	2. CNN 구현 (전반부)				
	➤적용하기				용어설명
	➤Outro				
	•문제풀기				
내레이션					
10					



과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명												
➤Intro •학습열기 •학습목표 ➤학습하기 1. CIFAR-10 데이터 전처리 2. CNN 구현 (전반부)  ➤적용하기 ➤Outro •문제풀기	<ul style="list-style-type: none"><li>데이터 정규화: <code>transforms.Normalize((0.4914, 0.4822, 0.4465), ...)</code> 각 채널(R, G, B)의 평균과 표준편차로 정규화(평균을 빼고 표준편차로 나눠서) CIFAR-10 데이터셋의 R, G, B 채널별 평균과 표준편차를 의미 이는 데이터를 모델에 넣기 전에 표준화(normalization)하여 학습이 더 안정되고 빠르게 진행되도록 돕는 핵심 값</li></ul> <table><tr><th>채널</th><th>평균 (mean)</th><th>표준편차 (std)</th></tr><tr><td>R</td><td>0.4914</td><td>0.2023</td></tr><tr><td>G</td><td>0.4822</td><td>0.1994</td></tr><tr><td>B</td><td>0.4465</td><td>0.2010</td></tr></table> <pre># 1. 데이터 전처리 transform_train = transforms.Compose([     transforms.RandomHorizontalFlip(),     transforms.RandomCrop(32, padding=4),     transforms.ToTensor(),     transforms.Normalize((0.4914, 0.4822, 0.4465),                           (0.2023, 0.1994, 0.2010)) ])</pre>				채널	평균 (mean)	표준편차 (std)	R	0.4914	0.2023	G	0.4822	0.1994	B	0.4465	0.2010	
	채널	평균 (mean)	표준편차 (std)														
	R	0.4914	0.2023														
G	0.4822	0.1994															
B	0.4465	0.2010															
				용어설명													
내레이션	11																

과정명		PyTorch로 배우는 머신러닝 알고리즘		회차명	8	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. CIFAR-10 데이터 전처리</div> <div>2. CNN 구현 (전반부)</div> <div>➤ 적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div>	<div>간지</div> <div>CNN 구현 (전반부)</div>					
		용어설명				
내레이션						12

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. CIFAR-10 데이터 전처리</div> <div>2. CNN 구현 (전반부)</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• CNN을 구성하는 합성곱 층 이해</div> <div>첫 층을 합성곱 → 배치 정규화 → ReLU 활성화 → 풀링 순서로 작동하게 구성</div> <div># 2. CNN 모델 정의</div> <div>class SimpleCNN(nn.Module):</div> <div>def __init__(self):</div> <div>super(SimpleCNN, self).__init__()</div> <div>self.layer1 = nn.Sequential(</div> <div># 채널 수, 출력 필터인 feature map 수, 커널(필터)의 크기, 패딩 픽셀 수</div> <div># 내부적으로 (채널 수 × feature map 수) 만큼의 커널(필터)를 생성</div> <div># nn.Conv2d(3, 32, 3, padding=1),</div> <div>nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1),</div> <div># 학습 중에 각 채널(feature map)의 평균과 분산을 정규화</div> <div># 학습 안정화, 수렴 속도 증가, 과적합 방지 효과</div> <div>nn.BatchNorm2d(32),</div> <div># 비선형 활성화 함수 (Rectified Linear Unit)</div> <div># 음수는 0, 양수는 그대로 통과</div> <div>nn.ReLU(),</div> <div># 2×2 크기의 창(window)을 사용해, 그 안에서 최댓값만 추출</div> <div># 32×32 → 16×16로 줄어듦</div> <div>nn.MaxPool2d(2))</div>			
				용어설명
내레이션	...13			

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
<div>➤Intro<ul style="list-style-type: none"><li>•학습열기</li><li>•학습목표</li></ul></div> <div>➤ 학습하기<ol style="list-style-type: none"><li>1. CIFAR-10 데이터 전처리</li><li>2. CNN 구현 (전반부)</li></ol></div>	<div>• 합성곱 구성 함수 <code>nn.Conv2d(...)</code> <code>in_channels=3</code>: 입력 이미지가 3채널 (예: RGB) <code>out_channels=32</code>: 출력 피쳐 맵(feature map) 수 = 32개 생성, 이것이 필터(커널) 수 <code>kernel_size=3</code>: 각 필터가 3×3 크기 <code>padding=1</code>: 출력 크기 유지용</div> <div># 채널 수, 출력 필터인 feature map 수, 커널(필터)의 크기, 패딩 픽셀 수 # 내부적으로 (채널 수 × feature map 수) 만큼의 커널(필터)를 생성 # <code>nn.Conv2d(3, 32, 3, padding=1),</code> <code>nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1)</code></div> <div><p>입력: 3 × H × W</p><p>커널 세트: 32 × (3 × 3)</p><p>합성곱: padding=1</p><p>출력: 32 × H × W</p></div>			
<div>➤적용하기</div> <div>➤Outro<ul style="list-style-type: none"><li>•문제풀기</li></ul></div>	<div>용어설명</div> <div>강환수, 직업 그림</div>			

내레이션	14	1
------	----	---

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. CIFAR-10 데이터 전처리</div> <div>2. CNN 구현 (전반부)</div> <div>➤ 적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div>	<div>• nn.Conv2d(in_channels=3, out_channels=2, kernel_size=3, padding=1)</div> <div><div><div><div>Input Volume (+pad 1) (7x7x3)</div><div><div>x[:, :, 0]</div><div><div>0 0 0 0 0 0 0</div><div>0 2 2 2 1 2 0</div><div>0 0 2 1 2 2 0</div></div></div><div><div>0 1 1 0 1 0 0</div><div>0 2 1 1 1 2 0</div><div>0 1 1 2 2 2 0</div></div><div><div>0 0 0 0 0 0 0</div><div>0 0 0 0 0 0 0</div><div>0 0 0 0 0 0 0</div></div></div><div><div>x[:, :, 1]</div><div><div>0 0 0 0 0 0 0</div><div>0 1 1 2 1 1 0</div><div>0 2 2 0 0 1 0</div></div></div><div><div>0 0 0 1 0 2 0</div><div>0 1 2 1 0 2 0</div><div>0 1 0 2 1 2 0</div></div><div><div>0 0 0 0 0 0 0</div><div>0 0 0 0 0 0 0</div><div>0 0 0 0 0 0 0</div></div></div><div><div>x[:, :, 2]</div><div><div>0 0 0 0 0 0 0</div><div>0 1 0 0 2 0 0</div><div>0 1 1 1 1 2 0</div></div></div><div><div>0 1 2 0 1 2 0</div><div>0 2 0 0 0 0 0</div><div>0 1 2 0 2 1 0</div></div><div><div>0 0 0 0 0 0 0</div><div>0 0 0 0 0 0 0</div><div>0 0 0 0 0 0 0</div></div></div> <div><div><div>Filter W0 (3x3x3)</div><div><div>w0[:, :, 0]</div><div><div>0 -1 -1</div><div>0 0 -1</div><div>0 1 1</div></div></div><div><div>0 0 1</div><div>0 -1 0</div><div>-1 0 1</div></div><div><div>-1 0 0</div><div>-1 1 1</div><div>0 0 0</div></div></div><div><div>Bias b0 (1x1x1)</div><div><div>b0[:, :, 0]</div><div>1</div></div></div></div> <div><div><div>Filter W1 (3x3x3)</div><div><div>w1[:, :, 0]</div><div><div>0 1 1</div><div>1 0 0</div><div>-1 -1 -1</div></div></div><div><div>0 -1 0</div><div>0 0 0</div><div>0 0 1</div></div><div><div>-1 1 1</div><div>-1 1 0</div><div>0 1 0</div></div></div><div><div>Bias b1 (1x1x1)</div><div><div>b1[:, :, 0]</div><div>0</div></div></div></div> <div><div><div>Output Volume (3x3x2)</div><div><div>o[:, :, 0]</div><div><div>3 1 0</div><div>8 -6 -1</div><div>1 -5 -4</div></div></div><div><div>2 -2 -3</div><div>4 0 1</div><div>5 0 1</div></div></div></div> <div>toggle movement</div>			
				용어설명
<a href="https://ai.stackexchange.com/questions/5769/in-a-cnn-does-each-new-filter-have-different-weights-for-each-input-channel-or">https://ai.stackexchange.com/questions/5769/in-a-cnn-does-each-new-filter-have-different-weights-for-each-input-channel-or</a>				

내레이션	15	1
------	----	---

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
<div> <div>➤Intro</div> <div> <div>•학습열기</div> <div>•학습목표</div> </div> <div>➤ 학습하기</div> <div> <div>1. CIFAR-10 데이터 전처리</div> <div>2. CNN 구현 (전반부)</div> </div> <div>➤ 적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div> </div>	<div> <ul style="list-style-type: none"> <li>필터 하나의 가중치(패러미터) 수 이해 <div>Conv2d는 필터 하나당 (입력 채널 수 × 필터 크기 × 필터 크기)의 가중치를 가짐</div> <div>여기선 필터 하나는 3채널 입력 각각에 대해 3×3 커널을 가짐</div> <div>3개의 3×3 커널 → 총 (3×3×3 = 27 + 1(bias)) 개의 가중치</div> <div>필터 수만큼의 편향(bias) 값이 있음</div> <div>이런 필터가 32개 있음 → 그래서 출력은 32채널로 구성됨</div> </li> <li>"입력의 R, G, B를 각각 3×3씩 보고, 그걸 하나의 숫자로 압축해서 피쳐 맵 하나를 만드는 작업을 32번 반복"하는 것</li> </ul> <div> <div>컨볼루션 파라미터 수 = 커널사이즈<sup>2</sup> × 커널수 × 채널(색상) + 커널수(bias)</div> </div> </div>				
					용어설명
내레이션	<div>16</div>				

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명												
과정명	➤Intro	<div>• 패러미터 수</div> <div>nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1),</div> <table><tr><th>항목</th><th>수</th></tr><tr><td>커널 크기</td><td>3×3</td></tr><tr><td>입력 채널 수</td><td>3</td></tr><tr><td>출력 채널 수</td><td>32</td></tr><tr><td>총 커널 수</td><td>3×32 = 96개</td></tr><tr><td>총 파라미터 수</td><td>96 × 3×3 + 32 (bias) = 864 + 32 = 896개</td></tr></table> <div>컨볼루션 파라미터 수 = 커널사이즈<sup>2</sup> * 커널수 * 채널(색상) + 커널수(bias)</div> <div>= (3 × 3) × 32 * 3 + 32</div> <div>= 896</div>			항목	수	커널 크기	3×3	입력 채널 수	3	출력 채널 수	32	총 커널 수	3×32 = 96개	총 파라미터 수	96 × 3×3 + 32 (bias) = 864 + 32 = 896개	
	항목				수												
	커널 크기				3×3												
	입력 채널 수				3												
출력 채널 수	32																
총 커널 수	3×32 = 96개																
총 파라미터 수	96 × 3×3 + 32 (bias) = 864 + 32 = 896개																
•학습열기																	
•학습목표																	
➤ 학습하기																	
1. CIFAR-10 데이터 전처리																	
2. CNN 구현 (전반부)																	
➤적용하기					용어설명												
➤Outro																	
•문제풀기																	
내레이션	17																

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
과정명	➤Intro	<ul style="list-style-type: none"><li>32개의 출력 채널(feature map)에 대해 각각 정규화(normalization)를 수행하는 계층 <code>nn.BatchNorm2d(32)</code> 학습 가능한 두 개의 파라미터(<math>\gamma</math>, <math>\beta</math>)를 각 채널마다 가지고 있어서, 학습 중 스케일과 이동을 조정 가능: <math>32 \times 2 = 64</math>개의 패러미터가 필요 즉, 단순 정규화가 아니라, 학습 가능한 정규화라고 보면 됨 신경망 학습 과정에서는 일반적으로 전체 데이터가 아닌 배치 단위로 학습이 진행 각 배치의 분포가 다르면 내부 (공변량 변화) 문제 발생 가능성 즉, 학습 과정에서 데이터 분포가 바뀌어 네트워크의 다음 계층이 학습하기 어려워 짐 일괄 정규화는 데이터를 평균이 0이고 분산이 1인 정규 분포로 강제로 되돌임 배치 정규화에서는 데이터 분포를 일관되게 만들고 , 다른 한편으로는 기울기가 사라지는 것을 방지</li></ul> <div># 학습 중에 각 채널 (feature map)의 평균과 분산을 정규화</div> <div># 학습 안정화, 수렴 속도 증가, 과적합 방지 효과</div> <code>nn.BatchNorm2d(32),</code>			
	•학습열기				
	•학습목표				
과정명	➤학습하기	<ul style="list-style-type: none"><li>32개의 출력 채널(feature map)에 대해 각각 정규화(normalization)를 수행하는 계층 <code>nn.BatchNorm2d(32)</code> 학습 가능한 두 개의 파라미터(<math>\gamma</math>, <math>\beta</math>)를 각 채널마다 가지고 있어서, 학습 중 스케일과 이동을 조정 가능: <math>32 \times 2 = 64</math>개의 패러미터가 필요 즉, 단순 정규화가 아니라, 학습 가능한 정규화라고 보면 됨 신경망 학습 과정에서는 일반적으로 전체 데이터가 아닌 배치 단위로 학습이 진행 각 배치의 분포가 다르면 내부 (공변량 변화) 문제 발생 가능성 즉, 학습 과정에서 데이터 분포가 바뀌어 네트워크의 다음 계층이 학습하기 어려워 짐 일괄 정규화는 데이터를 평균이 0이고 분산이 1인 정규 분포로 강제로 되돌임 배치 정규화에서는 데이터 분포를 일관되게 만들고 , 다른 한편으로는 기울기가 사라지는 것을 방지</li></ul> <div># 학습 중에 각 채널 (feature map)의 평균과 분산을 정규화</div> <div># 학습 안정화, 수렴 속도 증가, 과적합 방지 효과</div> <code>nn.BatchNorm2d(32),</code>			
	1. CIFAR-10 데이터 전처리				
	2. CNN 구현 (전반부)				
과정명	➤적용하기	<ul style="list-style-type: none"><li>32개의 출력 채널(feature map)에 대해 각각 정규화(normalization)를 수행하는 계층 <code>nn.BatchNorm2d(32)</code> 학습 가능한 두 개의 파라미터(<math>\gamma</math>, <math>\beta</math>)를 각 채널마다 가지고 있어서, 학습 중 스케일과 이동을 조정 가능: <math>32 \times 2 = 64</math>개의 패러미터가 필요 즉, 단순 정규화가 아니라, 학습 가능한 정규화라고 보면 됨 신경망 학습 과정에서는 일반적으로 전체 데이터가 아닌 배치 단위로 학습이 진행 각 배치의 분포가 다르면 내부 (공변량 변화) 문제 발생 가능성 즉, 학습 과정에서 데이터 분포가 바뀌어 네트워크의 다음 계층이 학습하기 어려워 짐 일괄 정규화는 데이터를 평균이 0이고 분산이 1인 정규 분포로 강제로 되돌임 배치 정규화에서는 데이터 분포를 일관되게 만들고 , 다른 한편으로는 기울기가 사라지는 것을 방지</li></ul> <div># 학습 중에 각 채널 (feature map)의 평균과 분산을 정규화</div> <div># 학습 안정화, 수렴 속도 증가, 과적합 방지 효과</div> <code>nn.BatchNorm2d(32),</code>			용어설명
	➤Outro				
	•문제풀기				
내레이션	18				



과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
과정명	➤Intro	• 풀링(pooling) 층 구성 피쳐 맵(feature map)의 공간 크기를 줄이기 위해 사용 최대 풀링(max pooling) 연산 입력 피쳐 맵에서 일정한 크기의 창(window)을 이동시키며 그 영역 안의 최댓값만 뽑는 연산 대표적인 downsampling 기법 불필요한 정보(작은 값) 버리고, 가장 두드러진 특징만 남김  # 2×2 크기의 창(window)을 사용해, 그 안에서 최댓값만 추출 # 32×32 → 16×16로 줄어듦 nn.MaxPool2d(2)			
	•학습열기 •학습목표 ➤ 학습하기 1. CIFAR-10 데이터 전처리 2. CNN 구현 (전반부)				
과정명	➤적용하기	•문제풀기			용어설명
	➤Outro				

내레이션

19


과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤학습하기</div> <div>1. CIFAR-10 데이터 전처리</div> <div>2. CNN 구현 (전반부)</div> <div>➤적용하기</div> <div>➤Outro</div> <div>•문제풀기</div>	<div>• CNN의 두 번째 합성곱 층: self.layer2</div> <div>32 채널 입력(feature map)을 받아 64 채널로 확장</div> <div>BatchNorm, ReLU, MaxPooling을 통해 정규화, 비선형 처리, 공간 축소를 수행</div> <div>결과적으로 특징(feature)의 추상화 수준이 더 깊어지고, 크기는 또 절반으로 줄어듦</div> <div>이미지 크기가 16×16에서 8×8로 줄어들고(풀링으로), 채널 수는 64개(Conv2D로)로 증가</div> <div># 2. CNN 모델 정의</div> <pre>class SimpleCNN(nn.Module):     def __init__(self):         super(SimpleCNN, self).__init__()         self.layer1 = nn.Sequential(             # 채널 수, 출력 필터인 feature map 수, 커널(필터)의 크기, 패딩 픽셀 수             # 내부적으로 (채널 수 × feature map 수) 만큼의 커널(필터)를 생성             # nn.Conv2d(3, 32, 3, padding=1),             nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1),             ...              self.layer2 = nn.Sequential(                 nn.Conv2d(32, 64, 3, padding=1),                 nn.BatchNorm2d(64),                 nn.ReLU(),                 nn.MaxPool2d(2))</pre>			
				용어설명

내레이션

20

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
➤Intro •학습열기 •학습목표 ➤ 학습하기 1. CIFAR-10 데이터 전처리 2. CNN 구현 (전반부)  ➤ 적용하기 ➤ Outro •문제풀기	• CNN 모델에서 세 번째 합성곱 층: self.layer3 64채널 입력을 받아 128채널로 확장 ReLU, 배치 정규화, 최대 풀링(MaxPooling)을 통해 특징을 더 고차원적으로 추상화 출력 공간 크기를 다시 절반(8×8 → 4×4)으로 줄이는 계층  <pre>self.layer3 = nn.Sequential(     nn.Conv2d(64, 128, 3, padding=1),     nn.BatchNorm2d(128),     nn.ReLU(),     # 8×8 → 4×4로 줄어듦     nn.MaxPool2d(2))</pre>				
					용어설명
내레이션	21				

과정명		PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
➤Intro •학습열기 •학습목표 ➤학습하기 1. CIFAR-10 데이터 전처리 2. CNN 구현 (전반부)  ➤적용하기 ➤Outro •문제풀기	• 마지막 완전 연결층(Fully Connected layer)  self.fc = nn.Linear(128 * 4 * 4, 10)  앞서 추출한 추상화된 특징들을 기반으로 분류 결정(score 계산)을 내리는 단계  CNN의 마지막 출력 feature map을 평탄화(flatten)한 후  4 x 4의 피쳐 맵이 128개이므로 입력이 128 x 4 x 4 = 2048  10개의 클래스(예: CIFAR-10)*에 대한 최종 분류를 수행  출력 차원 10: 분류하고자 하는 클래스 수를 의미  CIFAR-10이라면 10개 (비행기, 자동차, 고양이, 개, ...)  결과적으로, 이 계층은 2048차원 → 10차원으로 변환하는 선형 연산을 수행   self.fc = nn.Linear(128 * 4 * 4, 10)				
					용어설명
내레이션	22				

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
<div>➤Intro</div> <div>•학습열기</div> <div>•학습목표</div> <div>➤ 학습하기</div> <div>1. CIFAR-10 데이터 전처리</div> <div>2. CNN 구현 (전반부)</div> <div>➤ 적용하기</div> <div>➤ Outro</div> <div>•문제풀기</div>	<div>• CNN 전체 구조</div> <div></div>			
	<div>강환수, 직접 그림</div>			
				<div>용어설명</div>

강환수, 직접 그림

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	8	화면설명
<b>➤Intro</b> •학습열기 •학습목표  <b>➤학습하기</b> 1. CIFAR-10 데이터 전처리 2. CNN 구현 (전반부)	• 학습한 CNN 모델은 세 개의 합성곱 층과 하나의 완전 연결층으로 구성되어 있습니다. 이 모델을 이용하여 실제 의료 이미지(예: 흉부 X-ray)를 분류한다고 가정할 때, 기존 CIFAR-10 모델 구조를 그대로 사용하는 것이 적절한지 분석하고, 적절하지 않다면 어떻게 구조를 수정해야 하는지 구체적으로 서술해 주십시오.(모델 구조, 데이터 특성, 오버피팅/언더피팅 문제, 성능 개선 기법 등을 포함해 설명하십시오.) <ol style="list-style-type: none"> <li>적절하지 않은 이유 <ol style="list-style-type: none"> <li>CIFAR-10은 컬러(RGB) 이미지이며, 크기가 32x32로 작지만 의료 이미지는 보통 흑백(1채널)이고 크기가 큼니다. 따라서 in_channels=3으로 고정된 구조는 부적절합니다.</li> <li>의료 영상은 노이즈가 적고 특징이 미묘하여 더 깊은 네트워크 구조가 필요할 수 있습니다.</li> </ol> </li> <li>수정 제안 <ol style="list-style-type: none"> <li>in_channels=1로 수정하여 흑백 이미지 대응.</li> <li>합성곱 층을 5~6개로 늘려 더 많은 특징을 추출.</li> <li>Dropout, Regularization 등을 통해 과적합 방지.</li> <li>데이터 증강을 통해 훈련 데이터의 다양성을 확보.</li> </ol> </li> </ol>			① 학습 내용과 관련하여 실제 적용력을 높일 수 있는 문제, 혹은 주제를 작성해 주세요. ② ex. 사례 제시 후 전문가 의견, 실습과제, 응용 예시 시뮬레이션 등 ③ 저작권 침해가 되지 않도록 내용을 구성해 주세요. ④ 출처가 있을 경우 반드시 작성해 주세요.
<b>➤적용하기</b>  <b>➤Outro</b> •문제풀기				<b>용어설명</b>
내레이션				24