과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

➢Intro

•학습열기

•학습목표

➢학습하기

1. CNN 구현 (후반부)
2. CNN 학습
3. CNN 평가

➢적용하기

➢Outro

•문제풀기

- 기계가 스스로 배우는 뇌를 가질 수 있을까?

  인간의 뇌는 경험을 통해 배웁니다. 어린아이가 사과를 여러 번 본 후 '이건 사과야!'라고 인식하는 것처럼, 인공지능도 반복적인 데이터를 통해 무엇이 무엇인지를 알아갑니다. CNN(합성곱 신경망)은 이미지 인식 분야에서 인간 시각 시스템을 모방하여 '기계의 눈'을 갖게 해주는 기술입니다. 여러분은 SNS에서 친구 얼굴을 자동으로 태그하거나, 고양이 사진을 자동으로 분류하는 경험을 해보셨을 겁니다. 이런 기능들 대부분이 CNN으로 구현됩니다. 이 강의에서는 단순히 CNN 구조를 배우는 것을 넘어, 손실 함수와 옵티마이저의 설정부터 학습 정확도 시각화, 실제 이미지 예측까지의 전 과정을 직접 실습하게 됩니다. 그렇다면 우리가 만든 이 '인공 두뇌'는 어느 수준까지 정답을 맞히고, 어느 순간 실수하게 될까요? CNN이 학습하면서 '사람처럼 성장'하는 과정을 함께 들여다봅시다.

① 본 학습 내용으로 들어가기 전, 학습 주제의 흥미를 이끌 만한 도입부의 내용이 있다면 제시해주세요.

② ex. 관련 뉴스기사, 실생활과 관련된 이야기 등

③ 저작권 침해가 되지 않도록 내용을 구성해 주세요.

④ 출처가 있을 경우 반드시 작성해 주세요.

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

◈ 학습목표

1. 전체 CNN 모델을 완성할 수 있다.
2. 손실 함수와 옵티마이저를 설정하여 모델을 학습할 수 있다.
3. 테스트 데이터에서 성능을 평가하고 결과를 분석할 수 있다.

◈ 학습내용

1. CNN 구현 (후반부)
2. CNN 학습
3. CNN 평가

① 학습내용과 학습목표는 강의계획서와 일치해야 하며, 필요시 강의계획서를 수정할 수 있습니다.

② **학습목표**

✓ 각 레슨에 맞는 학습목표를 2~3개 작성해 주세요.

③ **학습내용**

✓ 1회차 당 25분 분량이 되도록 2~3개 레슨으로 구성해주세요.

✓ 학습내용과 레슨명은 일치해야 합니다.

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- ➢ **Intro**
  - •학습열기
  - •학습목표
- ➢ **학습하기**
  1. CNN 구현 (후반부)
  2. CNN 학습
  3. CNN 평가
- ➢ **적용하기**
- ➢ **Outro**
  - •문제풀기

간지

CNN 구현 (후반부)

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 전체 CNN 모델 준비

  모델 생성해서 GPU 또는 CPU에서 실행 준비

- 전체 CNN 모델 구조 출력

  torchsummary.summary 함수

  summary를 쓰기 위해선 from torchsummary import summary 임포트

  net 모델의 전체 레이어 구조, 출력 크기, 파라미터 수 등을 요약해서 출력

  (3, 32, 32)는 입력 이미지 CIFAR-10의 채널 수와 크기(RGB 이미지 32 x 32)를 의미

```
# 3. 모델 준비 및 구조 출력
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
net = SimpleCNN().to(device)
summary(net, (3, 32, 32))
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- CNN 구조 요약 결과

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 32, 32]             896
       BatchNorm2d-2           [-1, 32, 32, 32]              64
              ReLU-3           [-1, 32, 32, 32]               0
         MaxPool2d-4           [-1, 32, 16, 16]               0
            Conv2d-5           [-1, 64, 16, 16]          18,496
       BatchNorm2d-6           [-1, 64, 16, 16]             128
              ReLU-7           [-1, 64, 16, 16]               0
         MaxPool2d-8             [-1, 64, 8, 8]               0
            Conv2d-9            [-1, 128, 8, 8]          73,856
      BatchNorm2d-10            [-1, 128, 8, 8]             256
             ReLU-11            [-1, 128, 8, 8]               0
        MaxPool2d-12            [-1, 128, 4, 4]               0
           Linear-13                  [-1, 10]          20,490
================================================================
Total params: 114,186
Trainable params: 114,186
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 1.42
Params size (MB): 0.44
Estimated Total Size (MB): 1.87
----------------------------------------------------------------
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- ➢ **Intro**
  - •학습열기
  - •학습목표
- ➢ **학습하기**
  1. CNN 구현 (후반부)
  2. CNN 학습
  3. CNN 평가
- ➢ **적용하기**
- ➢ **Outro**
  - •문제풀기

간지

CNN 학습

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 손실 함수와 옵티마이저를 설정
- 학습 중에 성능을 기록할 변수들을 초기화

```
# 4. 학습을 위한 손실함수와 옵티마이저 설정
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

# 학습 과정 시, 손실 값과 정확도를 저장할 변수 초기화
train_losses = []
test_accuracies = []
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 손실 함수 설정

  nn.CrossEntropyLoss()는 분류(classification) 문제에서 자주 쓰는 손실 함수

  출력 값이 확률 형태가 아니더라도 사용 가능

```
# 4. 학습을 위한 손실함수와 옵티마이저 설정
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

# 학습 과정 시, 손실 값과 정확도를 저장할 변수 초기화
train_losses = []
test_accuracies = []
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 옵티마이저 설정

  optim.Adam은 대표적인 경사 하강법 기반 옵티마이저 중 하나

  SGD보다 빠르고 튜닝도 간편

  net.parameters():

  학습 대상인 모델의 모든 파라미터(가중치, 편향 등)를 넘겨 줌

  lr=0.001:

  learning rate로, 모델이 가중치를 얼마나 크게 조정할지를 의미

  너무 크면 발산하고, 너무 작으면 느려지니 보통은 0.001에서 시작해서 튜닝

```
# 4. 학습을 위한 손실함수와 옵티마이저 설정
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

# 학습 과정 시, 손실 값과 정확도를 저장할 변수 초기화
train_losses = []
test_accuracies = []
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- CNN 학습의 기본 구조

에폭(epoch)마다 학습 데이터를 반복하면서 손실을 계산하고, 모델을 업데이트

```
# 5. 학습 루프
epochs = 10 # 총 10번의 epoch 동안 반복
for epoch in range(epochs):
    net.train()  # 모델을 학습 모드로 설정 (Dropout, BatchNorm 등 활성화)
    running_loss = 0.0  # epoch 동안 손실값을 누적할 변수 초기화

    # 훈련 데이터(trainloader)에서 배치 단위로 반복
    for i, (inputs, labels) in enumerate(trainloader):
        # 입력 데이터와 정답 레이블을 GPU 또는 CPU로 이동
        inputs, labels = inputs.to(device), labels.to(device)

        optimizer.zero_grad()      # 이전에 계산된 gradient를 초기화
        outputs = net(inputs)      # 모델에 입력을 넣고 예측값 출력
        loss = criterion(outputs, labels)  # 손실(loss) 계산
        loss.backward()            # 역전파로 gradient 계산
        optimizer.step()           # gradient를 사용해 모델 파라미터 업데이트
        running_loss += loss.item()  # 배치 손실값을 누적 (float 값)
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 데이터 시각화를 위해 모델 학습 과정에서의 평균 손실 값 저장

  에폭(epoch)마다 학습 손실(loss)의 평균값을 계산해 avg_loss에 저장

  avg_loss를 train_losses 리스트에 계속 누적

```
# 5. 학습 루프
epochs = 10 # 총 10번의 epoch 동안 반복
for epoch in range(epochs):
    net.train()  # 모델을 학습 모드로 설정 (Dropout, BatchNorm 등 활성화)
    running_loss = 0.0  # epoch 동안 손실값을 누적할 변수 초기화

    # 훈련 데이터(trainloader)에서 배치 단위로 반복
    for i, (inputs, labels) in enumerate(trainloader):
        # 입력 데이터와 정답 레이블을 GPU 또는 CPU로 이동
        inputs, labels = inputs.to(device), labels.to(device)

        …
        running_loss += loss.item()  # 배치 손실값을 누적 (float 값)

    # epoch별 평균 손실값 계산
    avg_loss = running_loss / len(trainloader)
    train_losses.append(avg_loss)  # 그래프 시각화 등을 위한 손실 기록
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 데이터 시각화를 위해 모델 학습 과정에서의 테스트 데이터의 정확도 저장

  epoch가 끝난 후 테스트 정확도(accuracy)를 계산하고, 이를 test_accuracies에 저장

```
# 5. 학습 루프
epochs = 10 # 총 10번의 epoch 동안 반복
for epoch in range(epochs):
    …
    # 평가 모드로 전환 (Dropout, BatchNorm 등 비활성화)
    net.eval()
    correct = 0   # 맞힌 샘플 수
    total = 0     # 전체 샘플 수

    # 테스트 시에는 gradient 계산 필요 없음 → 연산 효율 높임
    with torch.no_grad():
        # 테스트 데이터셋에서 배치 단위로 평가
        for inputs, labels in testloader:
            …
            total += labels.size(0)   # 총 정답 수 누적
            correct += (predicted == labels).sum().item()   # 맞힌 개수 누적

    # 정확도(%) 계산
    accuracy = 100 * correct / total
    test_accuracies.append(accuracy)   # 정확도 기록
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 테스트 데이터의 정확도 계산

학습된 모델을 테스트 데이터셋에 대해 정확도(accuracy) 기준으로 평가하는 부분

torch.no_grad()를 통해 불필요한 연산을 줄이고

모델이 얼마나 정답을 잘 맞추는지 accuracy에 퍼센트로 계산한 다음

test_accuracies 리스트에 저장

```
# 테스트 시에는 gradient 계산 필요 없음 → 연산 효율 높임
with torch.no_grad():
    # 테스트 데이터셋에서 배치 단위로 평가
    for inputs, labels in testloader:
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = net(inputs)  # 모델 예측
        _, predicted = torch.max(outputs, 1)  # 가장 높은 확률을 가진 클래스 선택
        total += labels.size(0)  # 총 정답 수 누적
        correct += (predicted == labels).sum().item()  # 맞힌 개수 누적

# 정확도(%) 계산
accuracy = 100 * correct / total
test_accuracies.append(accuracy)  # 정확도 기록
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- torch.max() 이해

  텐서(tensor)에서 최댓값을 찾는 함수

  하나의 값에서 최대값을 찾거나

  차원을 기준으로 여러 값 중 최대값과 해당 인덱스(index)를 함께 반환

  torch.max(x)

  텐서 전체에서 가장 큰 값을 반환

```
x = torch.tensor([1, 5, 3])
max_value = torch.max(x)
print(max_value)  # tensor(5)
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- torch.max(..., dim=1)

  차원을 기준으로 최댓값과 인덱스를 함께 반환

  dim=1: 행(row)마다 열(column) 기준으로 최댓값을 계산

  dim=0: 열마다 행 기준으로 계산

```
x = torch.tensor([
    [1, 5, 2],
    [8, 3, 7]
])
values, indices = torch.max(x, dim=1)
print(values)   # tensor([5, 8]) → 각 행의 최댓값
print(indices)  # tensor([1, 0]) → 각 행에서 최댓값이 있는 열 인덱스
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 분류 모델에서 예측 클래스 뽑기

분류 모델에서는 각 샘플에서 가장 높은 확률의 클래스를 고르는 데 사용

predicted: 각 행(샘플)에 대해 가장 큰 값을 가진 열 = 예측 클래스

_: 실제 max 값은 필요 없으니까 무시하기 위해 사용

```
import torch

# 예: 모델의 예측 결과 (batch_size=3, num_classes=4)
outputs = torch.tensor([
    [0.1, 0.2, 0.6, 0.1],   # 샘플 0 → class 2
    [0.8, 0.1, 0.05, 0.05], # 샘플 1 → class 0
    [0.3, 0.4, 0.1, 0.2]    # 샘플 2 → class 1
])
# 각 행(샘플)에서 가장 큰 값의 인덱스를 가져옴
_, predicted = torch.max(outputs, 1)

print("Outputs:\n", outputs)
print("Predicted class indices:", predicted)
print("variable _:", _)
```

```
Outputs: tensor([[0.1000, 0.2000, 0.6000, 0.1000],
                 [0.8000, 0.1000, 0.0500, 0.0500],
                 [0.3000, 0.4000, 0.1000, 0.2000]])
Predicted class indices: tensor([2, 0, 1])
variable _: tensor([0.6000, 0.8000, 0.4000])
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 매 에폭마다 평균 손실 값과 정확도를 출력

```
# 5. 학습 루프
epochs = 10 # 총 10번의 epoch 동안 반복
for epoch in range(epochs):

    …

    # epoch 결과 출력
    print(f'[Epoch {epoch+1}] Loss: {avg_loss:.3f} | Test Accuracy: {accuracy:.2f}%')
```

```
[Epoch 1] Loss: 1.387 | Test Accuracy: 61.53%
[Epoch 2] Loss: 1.030 | Test Accuracy: 66.44%
[Epoch 3] Loss: 0.912 | Test Accuracy: 68.67%
[Epoch 4] Loss: 0.847 | Test Accuracy: 73.22%
[Epoch 5] Loss: 0.801 | Test Accuracy: 74.53%
[Epoch 6] Loss: 0.755 | Test Accuracy: 74.22%
[Epoch 7] Loss: 0.721 | Test Accuracy: 72.89%
[Epoch 8] Loss: 0.691 | Test Accuracy: 75.98%
[Epoch 9] Loss: 0.666 | Test Accuracy: 78.51%
[Epoch 10] Loss: 0.653 | Test Accuracy: 79.43%
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- ➢ **Intro**
  - •학습열기
  - •학습목표
- ➢ **학습하기**
  1. CNN 구현 (후반부)
  2. CNN 학습
  3. CNN 평가
- ➢ **적용하기**
- ➢ **Outro**
  - •문제풀기

간지

CNN 평가

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 평가 준비

  모델을 평가 모드로 설정

  정확도 계산을 위한 변수 초기화

  결과 시각화를 위한 리스트 초기화

```
# 6. 테스트 정확도 및 예측 수집
# 모델을 평가 모드로 설정 (Dropout, BatchNorm 비활성화)
net.eval()

# 정확도 계산을 위한 변수 초기화
correct = 0  # 맞힌 개수
total = 0    # 전체 테스트 샘플 수

# 결과 시각화를 위한 리스트들
all_preds = []   # 예측된 레이블 저장용
all_images = []  # 이미지 데이터 저장용
all_labels = []  # 실제 정답 저장용
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 테스트 데이터로 정확도를 위한 계산

  전체 수와 맞힌 수 누적

```
# gradient 계산은 하지 않도록 설정 → 속도 ↑, 메모리 ↓
with torch.no_grad():
    # 테스트 데이터셋을 배치 단위로 평가
    for inputs, labels in testloader:
        # 입력과 레이블을 CPU 또는 GPU로 이동
        inputs, labels = inputs.to(device), labels.to(device)

        # 모델에 입력을 넣고 출력(예측값) 계산
        outputs = net(inputs)

        # 가장 높은 확률값을 갖는 클래스 인덱스를 예측값으로 선택
        _, predicted = torch.max(outputs, 1)
        # 전체 수와 맞힌 수 누적
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

        # 결과 저장 (CPU로 이동한 뒤 numpy나 텐서로 저장)
        all_preds.extend(predicted.cpu().numpy())
        all_images.extend(inputs.cpu())
        all_labels.extend(labels.cpu())
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 정확도 출력

```
# 최종 정확도 출력
print(f'Test Accuracy: {100 * correct / total:.2f}%')
```

```
Test Accuracy: 79.43%
```

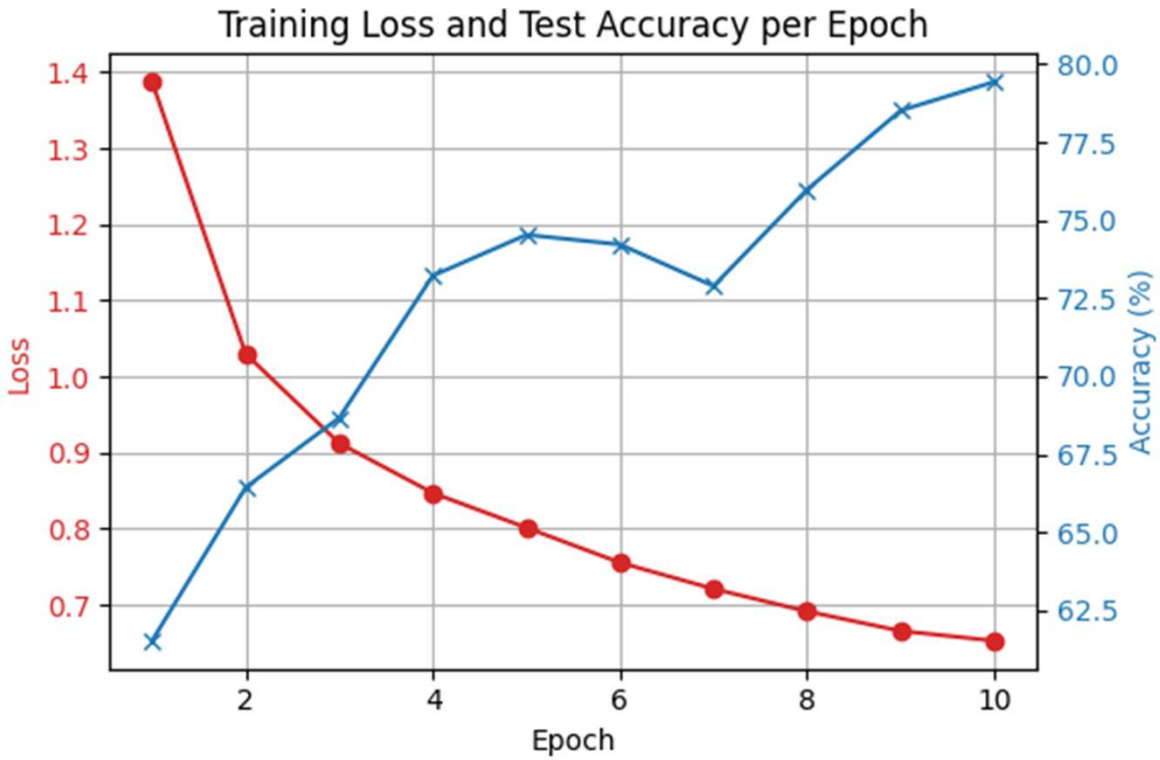
용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 손실과 정확도를 함께 시각화

  학습 과정에서 저장된 train_losses, test_accuracies를 사용



용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 실제 이미지와 예측 값 출력

이미지 복원 및 시각화 함수 정의

```
# 7. 이미지 복원 및 시각화 함수 정의
def imshow(img):
    # CIFAR-10 데이터셋에 적용된 정규화를 복원 (역정규화)
    mean = torch.tensor([0.4914, 0.4822, 0.4465]).view(3, 1, 1)
    std = torch.tensor([0.2023, 0.1994, 0.2010]).view(3, 1, 1)
    img = img * std + mean  # 정규화 반대로 되돌리기

    # 텐서를 numpy로 변환하고 픽셀 값 범위를 0~1로 제한
    img = img.numpy()
    img = np.clip(img, 0, 1)

    # (채널, 높이, 너비) → (높이, 너비, 채널) 로 차원 전환 후 이미지 출력
    plt.imshow(np.transpose(img, (1, 2, 0)))
    plt.axis('off')  # 축은 생략
```

용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 실제 이미지와 예측 값 출력 코드

```
# 8. 예측 결과 시각화 (앞에서 가져온 이미지 중 10개만)
fig = plt.figure(figsize=(6, 4))  # 전체 출력 이미지 크기 설정

# 예측을 점검을 시작 첨자 설정, 0, 10, 20, ... 등 설저
start_idx = 20
for idx in range(10):
    # 2행 5열의 서브플롯에 이미지 추가
    ax = fig.add_subplot(2, 5, idx+1, xticks=[], yticks=[])

    # idx번째 이미지 출력
    idx = idx + start_idx
    imshow(all_images[idx])

    # 예측 라벨을 이미지 상단에 표시
    ax.set_title(
        f"{classes[all_preds[idx]]}",
        # 정답이면 초록, 오답이면 빨강
        color=('green' if all_preds[idx]==all_labels[idx] else 'red')
    )

# 전체 타이틀 및 레이아웃 정리
plt.suptitle("CIFAR-10 Test Predictions")
plt.tight_layout()
plt.show()
```

용어설명

내레이션

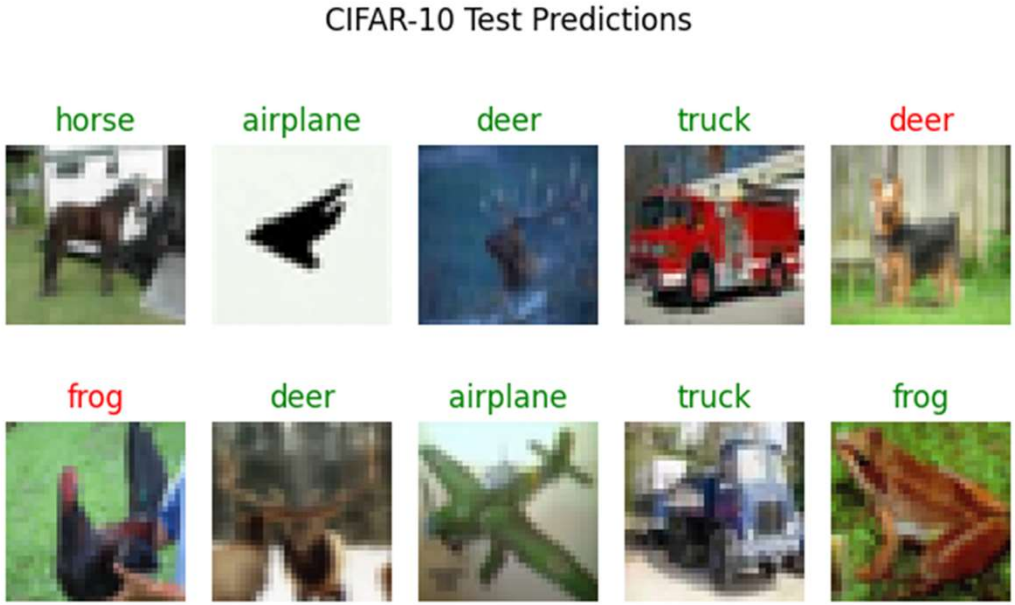
과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

- 실제 이미지와 예측 값 출력 결과

  붉은색은 잘못된 예측 결과

  정답이 dog인데 deer로 예측

  정답이 bird인데 frog로 예측



용어설명

내레이션

과정명	PyTorch로 배우는 머신러닝 알고리즘	회차명	9	화면설명

➢ **Intro**

- 학습열기
- 학습목표

➢ **학습하기**

1. CNN 구현 (후반부)
2. CNN 학습
3. CNN 평가

➢ **적용하기**

➢ **Outro**

- 문제풀기

- 모델을 학습한 후, 테스트 데이터의 정확도를 평가하고 이를 시각화하는 과정은 CNN 모델의 성능을 해석하는 데 매우 중요합니다. 학습 중 저장한 손실 값 (train_losses)과 정확도(test_accuracies)를 어떻게 활용하면 효과적인 모델 평가와 개선이 가능한지 설명해 주세요. 또한, 잘못된 예측 사례 시각화가 왜 중요한지 구체적인 예시와 함께 작성해 보세요."

  1. train_losses와 test_accuracies는 각각 학습 과정에서의 손실 감소와 정확도 향상을 시각적으로 보여주는 지표입니다. 이를 에폭별로 그래프로 나타내면 과적합(overfitting) 여부나 학습 안정성을 쉽게 판단할 수 있습니다.
  2. 예를 들어, 손실은 감소하는데 정확도가 떨어진다면 모델이 학습 데이터를 외워버렸을 가능성이 있으므로 조기 종료(Early Stopping)나 정규화(Regularization) 기법을 도입할 수 있습니다.
  3. 또한, 잘못된 예측 이미지를 시각화하면 어떤 클래스 간 혼동이 있는지 식별할 수 있습니다. 예컨대, dog 이미지를 deer로 잘못 예측한 경우, 두 이미지 간 특징이 겹치거나 학습 데이터가 불균형했을 가능성이 있습니다.
  4. 따라서 정확도 시각화 + 예측 결과 시각화 = 모델 디버깅과 개선의 출발점이 됩니다.

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.

용어설명

내레이션