

AI 개론 정기평가 대비 예상 문제

홍승은

1. 다음 연산의 결과로 맞는 것을 고르세요.

```
a = torch.tensor([[1, 2], [3, 4]])  
b = torch.tensor([10, 100])  
c = a + b
```

- a. 오류 발생 (차원 불일치)
- b. `[[11, 12], [103, 104]]`
- c. `[[11, 102], [13, 104]]`
- d. `[[11, 102], [13, 104]]`

해설)

두 텐서의 차원이 다르므로 **브로드캐스팅**이 적용되어 연산된다.

`a.shape = (2, 2)`

`b.shape = (2,)` → 브로드캐스팅으로 각 행에 `[10, 100]`이 더해짐

`[[1 + 10, 2 + 100],`

`[3 + 10, 4 + 100]]`

`= [[11, 102], [13, 104]]`

2. `x = torch.randn(4, 3, 2)`로 정의되어 있을 때, 다음 중 PyTorch의 텐서 차원 변환을 올바르게 수행하는 코드를 고르세요. (단, `.view()`는 메모리 연속성이 있다고 가정함)

- a. `x.view(3, 4, 1)`
- b. `x.reshape(2, 6, 2)`
- c. `x.view(-1, 5)` → -1은 자동 계산이지만, 5로 나눠 떨어지지 않아 에러 발생
- d. `x.view(2, 6, 2)`
- e. `x.reshape(4, -1)`
- f. `x.reshape(2, 3, 4)`

참고) `.reshape()` → 원소 수만 맞으면 대부분 가능 (내부 재배열 허용)

`.view()` → 원소 수 + 메모리 연속성까지 필요 (더 엄격함)

⇒ `x.view(2, 3, 4)`의 경우 메모리가 연속적일 때만 가능. 연속되지 않으면 에러 발생

- g. `x.view(24)`

3. 기울기 소실(Vanishing Gradient) 문제를 완화할 수 있는 방법으로 적절하지 않은 것은?

- a. ReLU 사용(기울기 0~1 사이 포화 영역 제거)
- b. tanh 사용
- c. Batch Normalization(입력 분포 정규화 → 학습 안정화)
- d. He 초기화(ReLU에 적합한 가중치 초기화 → 기울기 안정화)

4. 다음은 경사 하강법에서 파라미터를 업데이트하는 함수입니다. 빈칸에 들어갈 코드를 작성하세요.

```
def update(w, grad, lr):  
    with torch.no_grad():  
        w = w - lr * grad  
    return w
```

해설) 5 차시 – 경사하강법 수식 참고

▶ 경사 하강법의 단계

1. 초기값 설정: 파라미터를 임의로 초기화
2. 기울기 계산: 현재 위치에서 함수의 기울기(미분값)를 계산
3. 파라미터 업데이트: 계산된 기울기의 반대 방향으로 파라미터를 업데이트

$$\theta^{n+1} = \theta^n - \alpha \nabla_{\theta} J(\theta^n)$$

θ : 파라미터 (예: 모델의 가중치)

α : 학습률 (learning rate), 기울기만큼 얼마나 이동할지 결정하는 값

$\nabla_{\theta} J(\theta)$: 비용 함수 $J(\theta)$ 의 기울기

4. 반복: 기울기를 계산하고 파라미터를 업데이트하는 과정을 반복

5. CNN 합성곱층의 파라미터 수를 계산하시오.

```
nn.Conv2d(in_channels=3, out_channels=16, kernel_size=5, bias=True)
```

파라미터 수 = $16 \times (3 \times 5 \times 5) + 16 = 1216$

해설:

파라미터 수 = $(\text{out_channels}) \times (\text{in_channels} \times \text{kernel_height} \times \text{kernel_width}) + \text{bias}$

$\text{bias} = \text{True} \rightarrow$ output channel 당 bias 1 개씩 존재

6. (1) 다음 CNN 모델에서 각 계층의 출력 크기를 계산하여 첫 번째 빈칸을 채우시오.
(입력 크기: 1x28x28, 배치 크기 생략)

(2) CNN forward 메서드를 완성하시오.

```
class CNN(nn.Module):
    def __init__(self):
        super().__init__()

        # 입력 채널 1 개, 출력 채널 16 개, 3x3 커널, 패딩 1 → 출력 크기 유지 (28x28)
        self.conv1 = nn.Conv2d(1, 16, kernel_size=3, padding=1)

        # 2x2 MaxPooling → 출력 크기 절반으로 줄임
        self.pool = nn.MaxPool2d(2, 2)

        # 입력 채널 16 개, 출력 채널 32 개, 3x3 커널, 패딩 1 → 출력 크기 유지 (14x14)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, padding=1)

        # conv2 → pool2 이후 출력 크기: [batch, 32, 7, 7] → flatten 후 32×7×7 = 1568
        self.fc1 = nn.Linear(1568, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        # 첫 번째 Conv → ReLU → MaxPool → 출력 크기: [batch, 16, 14, 14]
        x = self.pool(F.relu(self.conv1(x)))

        # 두 번째 Conv → ReLU → MaxPool → 출력 크기: [batch, 32, 7, 7]
        x = self.pool(F.relu(self.conv2(x)))

        # 4 차원 Feature Map 을 2 차원으로 flatten: [batch, 32*7*7] = [batch, 1568]
        x = torch.flatten(x, 1) # x = x.view(x.size(0), -1)

        # Fully Connected Layer 1 → ReLU
        x = F.relu(self.fc1(x))

        # 출력층 (Softmax 는 CrossEntropyLoss 에서 자동 적용됨)
        x = self.fc2(x)

        return x
```

7. 다음 CNN 연산 흐름 중 올바른 것을 모두 고르시오.
- a. Conv → ReLU → Pool → Flatten → FC 순으로 구성된다.
 - b. Stride 가 클수록 출력 크기는 작아진다.
 - c. Pooling 은 특징 맵 크기를 줄이면서 중요한 정보를 유지한다.
 - d. Padding='valid'를 사용하면 출력 크기가 입력보다 커진다.
→ 패딩 없이 valid 한 영역만 연산하므로 출력은 작아짐.
8. 다음 중 최적화 알고리즘과 그 특징에 대한 설명 중 올바른 것을 모두 고르시오.
- a. SGD + Momentum 은 진동을 줄이고 더 빠르게 수렴할 수 있다.
 - b. Adam 은 RMSProp 과 Momentum 을 결합한 형태이다.
 - c. Adagrad 은 학습 후반에 학습률이 커져 안정적인 수렴을 유도한다.
→ Adagrad 은 학습이 진행될수록 학습률이 작아짐
 - d. NAG 는 예측된 위치에서의 기울기를 계산하여 더 빠른 수렴을 유도한다.
9. 다음 중 과적합(Overfitting)을 방지하거나 완화하는 방법으로 올바른 것을 모두 고르시오.
- a. Dropout 은 일부 뉴런을 무작위로 비활성화하여 모델의 일반화를 돕는다.
 - b. Weight Decay 는 가중치의 크기를 줄이는 정규화 기법이다.
 - c. Batch Normalization 은 모델 파라미터 수를 줄여 과적합을 방지한다.
→ 파라미터 수 감소와는 관련 없음. Internal Covariate Shift 를 줄임.
학습 도중 각 층의 출력 분포가 너무 많이 바뀌지 않도록 도와주는 역할.
 - d. Early Stopping 은 검증 손실이 더 이상 줄어들지 않으면 학습을 중단한다.
10. Confusion Matrix 에서 다음 정의 중 올바른 것을 고르시오.
- a. FP 는 실제 양성을 양성으로 예측한 경우
 - b. FN 은 실제 음성을 양성으로 예측한 경우
 - c. TP 는 실제 음성을 음성으로 예측한 경우
 - d. FN 은 실제 양성을 음성으로 잘못 예측한 경우

11. Residual Block 의 수식 표현으로 올바른 것은?

- a. $F(x) = W_2(W_1x + b_1) + b_2$
- b. $y = F(x)$
- c. $y = F(x) + x$
- d. $y = \text{ReLU}(Wx + b)$

12. 아래 코드 실행 시 출력 feature map 의 채널 수는?

```
class SimpleNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.block = nn.Sequential(
            # 입력: [batch_size, 3, 64, 64]
            # 출력: [batch_size, 16, 64, 64] (3 채널 → 16 채널, padding=1 → 크기 유지)
            nn.Conv2d(3, 16, kernel_size=3, padding=1),
            nn.ReLU(),

            # 입력: [batch_size, 16, 64, 64]
            # 출력: [batch_size, 32, 64, 64] (16 채널 → 32 채널, 크기 유지)
            nn.Conv2d(16, 32, kernel_size=3, padding=1),
            nn.ReLU()
        )

    def forward(self, x):
        return self.block(x)

# 입력 이미지: 1 개, 채널 3, 크기 64x64
x = torch.randn(1, 3, 64, 64)

model = SimpleNet()

# 출력: [1, 32, 64, 64] → 출력 feature map 의 채널 수는 32
out = model(x)
print(out.shape) # 출력: torch.Size([1, 32, 64, 64])
```

정답: 32

13. 다음은 torchvision.models.resnet18(pretrained=True) 모델의 요약 정보 일부이다. 이 모델을 전이 학습(Transfer Learning)에 사용하여, 10 개 클래스를 분류하는 새로운 작업에 적용하고자 한다. 아래 model 구조를 참고하여 출력층을 수정하는 코드를 직접 작성하시오.

모델 구조 요약(일부)

```
(...  
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))  
  (fc): Linear(in_features=512, out_features=1000, bias=True)  
)
```

```
import torchvision.models as models  
import torch.nn as nn  
  
model = models.resnet18(pretrained=True)  
  
# 출력층 교체  
model.fc = nn.Linear(in_features=512, out_features=10)
```

14. 다음은 사전 학습된 VGG16 모델을 기반으로 일부 레이어만 학습 대상에서 제외하고,
출력층만 학습 가능한 상태로 남기는 코드의 일부이다. 코드의 의도를 고려하여,
빈칸을 적절히 채우시오.

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

model = models.vgg16(pretrained=True)

# 특징 추출기(feature extractor)에 해당하는 파라미터들을 업데이트하지 않도록
설정
for param in model.features.parameters():
    param.requires_grad = False      # freezing

model.classifier[6] = nn.Linear(4096, 5)
```

15. 다음 중 Fine-Tuning 에 대한 설명으로 옳은 것을 모두 고르시오.

- a. 기존 모델의 전체 가중치를 고정한 채 새로운 데이터에 적응시킨다.
→ Feature Extraction
- b. 새로운 태스크에 맞게 일부 또는 전체 가중치를 다시 학습시킨다.
- c. 출력층만 재학습하며 나머지는 고정한다.
→ Feature Extraction
- d. 데이터의 특성이 기존과 많이 다를 때 적절하다.

16. 다음 중 AlexNet 과 VGGNet 의 구조적 차이점을 가장 올바르게 설명한 것은?

- a. AlexNet 은 정보 손실을 줄이기 위해 1x1 커널을 주요 구조로 사용하고, VGGNet 은 이를 사용하지 않는다.

→ 1x1 커널은 Inception, ResNet 등에서 주로 사용. AlexNet 은 11x11, 5x5 커널 사용.

- b. VGGNet 은 파라미터 수를 줄이기 위해 Dropout 을 다수의 합성곱 층에 적용한다.

→ VGGNet 은 주로 Fully Connected Layer 에 Dropout 을 적용. Conv 층에는 잘 사용되지 않음.

- c. AlexNet 은 큰 커널(예: 11x11, 5x5)을 사용하여 연산 깊이를 줄이고, VGGNet 은 3x3 커널을 여러 번 반복하여 깊이를 증가시킨다.

- d. VGGNet 은 Inception 모듈을 통해 여러 크기의 커널 연산을 병렬로 수행한다.

→ Inception 은 GoogleNet 계열의 구조. VGGNet 은 단순한 Conv+ReLU 반복 구조.

17. LSTM 의 게이트 중, 새로운 정보를 얼마나 반영할지 결정하는 게이트는?

- a. Forget Gate (과거 정보를 얼마나 잊을지 결정)
- b. Output Gate (셀 상태에 따라 최종 출력에서 output 으로 나갈 정보를 선별)
- c. Input Gate
- d. Reset Gate (LSTM 이 아니라 GRU 에서 사용됨. 헛갈리기 쉬우니 주의!!)

18. 아래는 LSTM 을 이용한 시퀀스 모델링 코드이다. 출력되는 h_n 텐서의 shape 은?

```
rnn = nn.LSTM(input_size=64, hidden_size=128, num_layers=2, batch_first=True)
x = torch.randn(32, 20, 64) # [batch_size, seq_len, input_size]
output, (h_n, c_n) = rnn(x)
print(h_n.shape)
```

정답: torch.Size([2, 32, 128])

[num_layers, batch_size, hidden_size]

19. 다음 중 RNN → LSTM → GRU 순으로 발전된 이유를 가장 잘 설명한 것은?

- a. 계산 속도를 줄이기 위해 hidden layer 를 제거했다
- b. 장기 의존성 문제를 해결하고, 파라미터 수를 줄이며 효율을 높였다
- c. 학습 속도를 줄이기 위해 activation 함수를 제거했다
- d. Attention 구조를 도입해 순환구조를 유지했다

20. 다음 보기 중 Faster R-CNN 의 처리 흐름을 올바른 순서대로 나열하시오.

- a. RoI Pooling 을 통해 고정된 크기의 feature vector 를 얻는다.
- b. CNN 을 이용해 입력 이미지에서 feature map 을 추출한다.
- c. Fully Connected Layer 를 통해 각 Region 을 분류하고 위치를 회귀한다.
- d. Region Proposal Network(RPN)를 통해 object 후보 영역을 생성한다.

정답: b → d → a → c