발표 참고 자료

1. 프로젝트 개요

본 프로젝트는 CLIP 모델을 활용하여 VQA(Visual Question Answering) 태스크의 성능을 개선하는 방안을 다룹니다. 특히, '일상 사진(daily life photo)' 도메인에 대한 모델의 이해도를 극대화하여 높은 정확도를 달성하는 것을 목표로 합니다.

2. CLIP 모델의 구조 및 특징

CLIP(Contrastive Language-Image Pre-training)은 이미지와 텍스트 간의 유사도를 학습하는 모델입니다. 주요 특징은 다음과 같습니다:

- **통합적 표현 공간 (Uniform Embedding Space):** 이미지와 텍스트를 동일한 임베딩 공간에 매핑하여 두 모달리티 간의 유사도를 직접 측정할 수 있습니다.
- **제로샷 (Zero-shot) 능력:** 학습 시 보지 못한 이미지-텍스트 쌍에 대해서도 높은 예측 성능을 보입니다.
- **비대칭적 유사도 측정 (Asymmetric Similarity):** 이미지와 텍스트 간의 유사도를 비대칭적으로 측정하여 VQA 태스크에 특화된 방식으로 활용될 수 있습니다.

3. 핵심 아이디어: 프롬프트 엔지니어링

CLIP 모델의 특성을 최대한 활용하기 위해 프롬프트 엔지니어링을 적용했습니다. 기존의 "질문: {question}, 선택지: {choice}" 형태의 프롬프트 대신, "A daily life photo: {q}. The correct answer is: {c}"와 같이 재구성하여 모델이 '일상 사진' 도메인 특성을 명시적으로 인지하고, 질문-답변 형식을 '가장 적합한 설명 찾기' 문제로 인식하도록 유도했습니다.

4. 모델 미세 조정 (Fine-tuning) 전략

모델의 VQA 태스크 성능 향상을 위해 미세 조정을 수행했습니다. 학습 과정에서 CLIP 모델의 출력인 `logits_per_image`를 (배치 크기, 배치 크기 * 선택지 수) 형태의 유사도 행렬에서 각 이미지에

해당하는 4개의 선택지 로짓만을 추출하여 (배치 크기, 4) 형태로 재구성하여 손실 함수(`CrossEntropyLoss`)를 적용했습니다.

5. 추론 전략 및 성능 극대화

단일 예측의 불안정성을 최소화하고 성능을 극대화하기 위해 다음과 같은 추론 전략을 적용했습니다:

- **테스트 시점 증강 (TTA, Test-Time Augmentation):** 테스트 데이터에 `RandomRotation` 및 `RandomHorizontalFlip`과 같은 증강을 적용하여 여러 번 예측을 수행하고, 이를 통해 얻은 확률을 평균하여 최종 예측에 사용합니다.
- **앙상블 (Ensemble):** TTA를 통해 얻은 각 예측 확률을 산술 평균하여 최종 확률을 계산합니다. 이를 통해 단일 예측의 오류나 편향을 완화하고 안정적인 결과를 도출합니다.
- **온도 스케일링 (Temperature Scaling):** 로짓을 소프트맥스 함수에 통과시키기 전에 `temperature` 값(0.8)으로 나누어 모델이 예측한 확률 분포를 더 뾰족하게 만들어 가장 높게 예측한 선택지에 대한 신뢰도를 증폭시킵니다.

6. 코드 구조 및 주요 함수 분석

프로젝트 코드는 `train.py`와 `inference.py`로 분리되어 있으며, 주요 구성 요소는 다음과 같습니다:

- **`VQADataset` 클래스:** 이미지 경로 처리, 이미지 증강(`tta_transforms`), 질문 및 선택지 로딩을 담당합니다. `CLIPProcessor`가 내부적으로 이미지 전처리를 처리하도록 `ToTensor`와 `Normalize`는 제거되었습니다.
- **`collate_fn` 함수:** 배치 데이터를 처리하며, 프롬프트 엔지니어링을 통해 "A daily life photo: {q}. The correct answer is: {c}" 형식으로 텍스트 프롬프트를 강화합니다.
- **`fine_tune_model` 함수:** 훈련 데이터(`train.csv`)를 사용하여 CLIP 모델을 미세 조정합니다. `AdamW` 옵티마이저와 `CrossEntropyLoss`를 사용하며, `autocast`를 통해 혼합 정밀도 학습을 지원하여 GPU 메모리를 최적화합니다.
- **예측 루프:** TTA를 적용하여 여러 번 예측을 수행하고, 각 예측의 확률을 평균하여 최종 결과를 도출합니다. `temperature` 스케일링을 통해 예측 신뢰도를 조절합니다.

7. 모델 성능 및 예측 결과 분석

- **재현성:** `torch.manual_seed(42)` 를 통해 시드를 고정하여 결과의 재현성을 보장합니다.
- **안정성:** `FileNotFoundError` 와 같은 예외 처리를 통해 코드의 안정성을 확보했습니다.
- **효과:** TTA, 앙상블, 온도 스케일링 등의 기법을 통해 단일 예측의 불안정성을 극복하고 성능을 극대화했습니다.
- **효율성:** 혼합 정밀도 학습(`autocast`)을 통해 GPU 메모리를 효율적으로 사용했습니다.

8. 결론 및 핵심 강점

본 프로젝트는 CLIP 모델을 VQA 태스크에 성공적으로 적용하기 위한 심도 있는 분석과 체계적인 실험의 결과물입니다. 특히, 대회 데이터의 특성을 반영한 정교한 프롬프트 엔지니어링과 TTA 기반의 앙상블 추론 전략은 저희 솔루션의 핵심적인 차별점입니다. 높은 재현성과 안정성, 효율적인 학습을 통해 신뢰할 수 있는 결과를 제공합니다.

9. 코드 상세 분석

제공된 Jupyter Notebook 파일(`본22차.ipynb` , `추론코드.ipynb` , `학습코드.ipynb`)은 CLIP 모델을 활용한 VQA 태스크의 학습 및 추론 과정을 구현하고 있습니다. `본22차.ipynb` 는 학습과 추론을 한 파일에서 모두 수행하는 통합 버전으로 보이며, `학습코드.ipynb` 와 `추론코드.ipynb` 는 각각 학습과 추론 부분을 분리한 모듈화된 코드입니다. 핵심적인 코드 구성과 로직은 세 파일 모두 유사합니다.

9.1. 공통 모듈 및 설정

- **라이브러리 임포트:** `os` , `pandas` , `torch` , `PIL (Pillow)` , `tqdm` , `transformers` , `numpy` , `torchvision` , `torch.cuda.amp.autocast` , `torch.optim.AdamW` , `torch.nn.CrossEntropyLoss` 등 필요한 라이브러리를 임포트합니다.
- **시드 고정:** `torch.manual_seed(42)` 및 `np.random.seed(42)` 를 사용하여 실험의 재현성을 보장합니다.
- **디바이스 설정:** `DEVICE = "cuda" if torch.cuda.is_available() else "cpu"` 를 통해 GPU 사용 가능 여부에 따라 학습 및 추론 장치를 설정합니다.

- **모델 로드:** `CLIPModel.from_pretrained("openai/clip-vit-large-patch14")`를 사용하여 CLIP 모델을 로드하고, `CLIPProcessor.from_pretrained`를 통해 이미지 및 텍스트 전처리를 위한 프로세서를 로드합니다.
- **경로 설정:** `BASE_PATH`를 설정하고, `train.csv`, `test.csv`, `submission.csv` 등의 파일 경로를 정의합니다.
- **`get_image_path` 함수:** 이미지 파일의 상대 경로를 절대 경로로 변환하는 유틸리티 함수입니다.

9.2. `VQADataset` 클래스

- **목적:** VQA 태스크를 위한 데이터셋을 PyTorch의 `Dataset` 형태로 정의합니다.
- **초기화 (`__init__`):**
  - `df`: 데이터프레임 (훈련 또는 테스트 데이터).
  - `processor`: CLIP 프로세서.
  - `tta_transforms`: 테스트 시점 증강(TTA)을 위한 이미지 변환 리스트. `RandomRotation(5)` 및 `ColorJitter(brightness=0.1)` 등이 포함됩니다. `ToTensor`와 `Normalize`는 `CLIPProcessor`가 내부적으로 처리하므로 제거되었습니다.
  - `is_train`: 훈련 모드 여부를 나타내는 플래그.
- **아이템 가져오기 (`__getitem__`):**
  - 데이터프레임의 각 행에서 이미지 경로, 질문, 4개의 선택지(A, B, C, D)를 추출합니다.
  - `Image.open(img_path).convert("RGB")`를 사용하여 이미지를 로드하고, `tta_transforms`를 적용합니다.
  - `FileNotFoundError` 발생 시 빈 이미지(`Image.new("RGB", (224, 224))`)를 생성하여 예외 처리합니다.
  - 훈련 모드(`is_train=True`)일 경우 정답 레이블(`label`)을 추가합니다.

9.3. `collate_fn` 함수

- **목적:** `DataLoader`에서 배치 데이터를 처리하고, CLIP 모델에 입력할 수 있는 형태로 변환합니다.
- **프롬프트 강화:** 핵심적인 부분으로, 질문(`q`)과 각 선택지(`c`)를 조합하여 `f"A daily life photo: {q}. The correct answer is: {c}"` 형태의 프롬프트를 생성합니다. 이는

CLIP 모델이 이미지와 텍스트 쌍의 유사도를 측정하는 데 최적화된 형태로, '일상 사진' 도메인 특성을 명시하고 질문-답변을 '가장 적합한 설명 찾기' 문제로 재구성하는 역할을 합니다.

- **`processor` 적용:** 생성된 프롬프트와 이미지들을 `processor(text=prompts, images=images, return_tensors="pt", padding=True, truncation=True, max_length=77)`를 통해 CLIP 모델의 입력 형식에 맞게 토큰화 및 전처리합니다.
- **레이블 추가:** 훈련 시에는 정답 레이블을 배치에 추가합니다.

9.4. `fine_tune_model` 함수 (학습 코드)

- **목적:** CLIP 모델을 VQA 태스크에 맞게 미세 조정(Fine-tuning)합니다.
- **데이터 로더:** `VQADataset`과 `DataLoader`를 사용하여 훈련 데이터를 배치 단위로 로드합니다.
- **옵티마이저 및 손실 함수:** `AdamW` 옵티마이저와 `CrossEntropyLoss`를 사용합니다.
- **훈련 루프:**
  - 지정된 에폭(`epochs`) 수만큼 반복합니다.
  - `tqdm`을 사용하여 훈련 진행 상황을 시각적으로 표시합니다.
  - `autocast()`를 사용하여 혼합 정밀도 학습(Automatic Mixed Precision)을 활성화하여 GPU 메모리 사용을 최적화하고 학습 속도를 향상시킵니다.
  - **로짓 처리:** `outputs.logits_per_image`는 (batch_size, batch_size * 4) 형태의 유사도 행렬입니다. 여기서 각 이미지에 해당하는 4개의 선택지 로짓만을 정확히 추출하여 (batch_size, 4) 형태로 재구성합니다. 이는 CLIP의 출력 구조를 VQA 태스크에 맞게 변환하는 중요한 단계입니다.
  - 손실을 계산하고, `optimizer.zero_grad()`, `loss.backward()`, `optimizer.step()`을 통해 역전파 및 가중치 업데이트를 수행합니다.
- **모델 저장:** 학습 완료 후 `model.save_pretrained()`를 사용하여 미세 조정된 모델 가중치를 저장합니다.

9.5. 추론 루프 (추론 코드)

- **목적:** 미세 조정된 모델을 사용하여 테스트 데이터에 대한 예측을 수행합니다.
- **모델 로드:** `CLIPModel.from_pretrained()`를 사용하여 저장된 미세 조정 모델을 로드합니다.
- **TTA (Test-Time Augmentation):**

    - `tta_transforms_list`에 정의된 여러 이미지 증강 기법(예: `RandomRotation`, `RandomHorizontalFlip`)을 순회하며 예측을 수행합니다.
    - 각 TTA 반복마다 `VQADataset`을 새로 생성하고 `DataLoader`를 통해 테스트 데이터를 로드합니다.
- **예측 과정:**
    - `torch.no_grad()` 및 `autocast()`를 사용하여 추론을 수행합니다.
    - `outputs.logits_per_image`를 `temperature` 값(0.8)으로 나누어 온도 스케일링을 적용합니다. 이는 확률 분포를 더 뾰족하게 만들어 가장 높은 확률을 가진 선택지에 대한 신뢰도를 높이는 효과가 있습니다.
    - 각 이미지에 대한 4개의 선택지 로짓을 추출하고 `softmax(dim=0)`를 적용하여 확률로 변환합니다.
    - 각 TTA 반복에서 얻은 확률(`tta_probs`)을 누적합니다.
- **앙상블:** 모든 TTA 반복이 완료되면 누적된 확률(`avg_probs`)을 `num_tta`로 나누어 평균 확률을 계산합니다. 이는 여러 증강된 이미지에 대한 예측을 통합하여 최종 예측의 안정성과 정확도를 높입니다.
- **최종 예측:** 평균 확률에서 가장 높은 값을 가진 선택지의 인덱스를 찾아 'A', 'B', 'C', 'D' 중 하나로 변환하여 최종 예측(`preds`) 리스트에 추가합니다.
- **제출 파일 생성:** 예측 결과를 `pandas.DataFrame`으로 생성하고 `submission.csv` 파일로 저장합니다.

9.6. `본22차.ipynb`와 `학습코드.ipynb`, `추론코드.ipynb`의 관계

- `본22차.ipynb`는 `fine_tune_model` 함수를 정의하고 실행한 후, 이어서 테스트셋 로딩부터 TTA 및 예측 루프까지 한 파일 내에서 순차적으로 실행합니다. 이는 개발 및 실험 단계에서 편리하게 전체 워크플로우를 확인할 수 있도록 구성된 것으로 보입니다.
- `학습코드.ipynb`는 `fine_tune_model` 함수와 훈련 데이터 로딩 및 실행 부분만을 포함하여 모델 학습에 집중합니다.
- `추론코드.ipynb`는 학습된 모델을 로드하고 테스트셋에 대한 TTA 및 예측 루프를 실행하는 부분만을 포함하여 추론에 집중합니다.

이러한 분리는 실제 서비스 환경에서 학습과 추론을 분리하여 효율적으로 운영하거나, 각 단계를 독립적으로 관리하고 개선할 때 유용합니다. 제출된 코드들은 VQA 태스크 해결을 위한 CLIP 모델의 효과적인 활용 방안과 성능 최적화 기법들을 잘 보여주고 있습니다.

10. 코드 상세 분석 및 비교: 본22차, 추론코드, 학습코드, 본23차

이 섹션에서는 제공된 네 가지 Jupyter Notebook 파일(`본22차.ipynb`, `추론코드.ipynb`, `학습코드.ipynb`, `본23차.ipynb`)을 상세히 분석하고, 각 코드의 특징, 상호 관계, 그리고 `본23차.ipynb`에서 개선된 점을 비교 설명합니다.

10.1. `본22차.ipynb` 상세 분석

`본22차.ipynb`는 CLIP 모델을 활용한 VQA 태스크의 학습(Fine-tuning)과 추론(Inference) 과정을 하나의 파일에서 모두 수행하는 통합 코드입니다. 주요 특징은 다음과 같습니다:

- **환경 설정:** `torch.manual_seed(42)`와 `np.random.seed(42)`로 시드를 고정하여 재현성을 확보합니다. `DEVICE`는 `cuda` 사용 가능 여부에 따라 설정됩니다.
- **모델 및 프로세서 로드:** `openai/clip-vit-large-patch14` 모델과 프로세서를 로드합니다.
- **`get_image_path` 함수:** 이미지 파일 경로를 처리하는 유틸리티 함수입니다.
- **`VQADataset` 클래스:**
  - `Dataset` 클래스를 상속받아 VQA 데이터를 처리합니다.
  - `tta_transforms`를 통해 이미지 증강(RandomRotation, ColorJitter)을 적용합니다. 이전 버전에서 `ToTensor`와 `Normalize`가 제거되어 `CLIPProcessor`가 이를 처리하도록 합니다.
  - `FileNotFoundError` 발생 시 빈 이미지로 대체하는 예외 처리가 포함되어 있습니다.
- **`collate_fn` 함수:**
  - 배치 데이터를 처리하며, 핵심적으로 프롬프트 엔지니어링을 수행합니다.
  - `f"A daily life photo: {q}. The correct answer is: {c}"` 형태의 프롬프트를 사용하여 CLIP 모델이 VQA 태스크에 더 적합하게 동작하도록 유도합니다. 이는 이미지 도메인(`daily life photo`)을 명시하고 질문-답변을 유사도 측정 문제로 재구성하는 중요한 부분입니다.
- **`fine_tune_model` 함수:**
  - `train.csv` 데이터를 사용하여 CLIP 모델을 미세 조정합니다.
  - `AdamW` 옵티마이저와 `CrossEntropyLoss`를 사용합니다.
  - `torch.cuda.amp.autocast()`를 사용하여 혼합 정밀도 학습(AMP)을 활성화하여 GPU 메모리 사용 효율을 높이고 학습 속도를 개선합니다.

    - `logits_per_image`를 (batch_size, batch_size * 4)에서 (batch_size, 4) 형태로 재구성하는 로직이 포함되어 있습니다.
    - 학습 완료 후 `model.save_pretrained()`를 통해 미세 조정된 모델을 저장합니다.
- **추론 루프:**
    - 저장된 미세 조정 모델을 로드합니다.
    - TTA(Test-Time Augmentation)를 2회 반복하여 예측을 수행합니다. (`RandomRotation`, `RandomHorizontalFlip`)
    - `temperature` 값(0.8)으로 로짓을 스케일링하여 확률 분포를 조정합니다.
    - 각 TTA 예측의 확률을 평균하여 앙상블 효과를 얻습니다.
    - 최종 예측 결과를 `submission.csv`로 저장합니다.

10.2. `추론코드.ipynb` 상세 분석

`추론코드.ipynb`는 `본22차.ipynb`에서 추론 관련 부분만 분리하여 구성된 파일입니다. 학습된 모델을 로드하여 테스트 데이터에 대한 예측을 수행하는 데 초점을 맞춥니다.

- **주요 차이점:** `fine_tune_model` 함수 및 훈련 데이터 로딩 부분이 제거되었습니다.
- **모델 로드:** `fine_tuned_clip` 폴더에서 학습된 모델을 로드합니다.
- **TTA 및 예측:** `본22차.ipynb`와 동일한 TTA 설정(2회 반복, `RandomRotation`, `RandomHorizontalFlip`)과 온도 스케일링(0.8)을 사용하여 예측을 수행하고 `submission.csv`를 생성합니다.
- **목적:** 학습과 추론을 분리하여 추론만을 독립적으로 실행하거나 배포할 때 유용합니다.

10.3. `학습코드.ipynb` 상세 분석

`학습코드.ipynb`는 `본22차.ipynb`에서 학습 관련 부분만 분리하여 구성된 파일입니다. 모델을 미세 조정하는 데 초점을 맞춥니다.

- **주요 차이점:** 테스트 데이터 로딩 및 추론 루프 부분이 제거되었습니다.
- **모델 학습:** `train.csv` 데이터를 사용하여 `fine_tune_model` 함수를 실행하여 모델을 학습하고 `fine_tuned_clip` 폴더에 저장합니다.
- **목적:** 모델 학습만을 독립적으로 수행하거나, 학습 파라미터 튜닝 등 학습 과정에 집중할 때 사용됩니다.

10.4. `본22차.ipynb`, `추론코드.ipynb`, `학습코드.ipynb`의 관계

세 파일은 동일한 핵심 로직과 아이디어를 공유하며, `본22차.ipynb`는 학습과 추론을 통합한 버전이고, `추론코드.ipynb`와 `학습코드.ipynb`는 각각 추론과 학습 기능을 분리하여 모듈화한 버전입니다. 이는 개발 편의성과 재사용성을 고려한 구성으로 볼 수 있습니다.

10.5. `본23차.ipynb` 상세 분석 및 개선점

`본23차.ipynb`는 `본22차.ipynb`를 기반으로 여러 개선 사항이 적용된 버전입니다. 주요 변경 사항 및 개선점은 다음과 같습니다:

- **디바이스 설정 변경:**
  - `DEVICE = "cpu"`로 명시적으로 CPU를 사용하도록 설정되었습니다. 이는 GPU 환경이 없는 사용자도 코드를 실행할 수 있도록 호환성을 높인 변경입니다. (이전 버전은 `cuda` 우선 사용)
  - **개선점:** GPU가 없는 환경에서도 코드 실행이 가능하여 접근성이 향상되었습니다. 다만, CPU 사용 시 학습 및 추론 속도가 현저히 느려질 수 있습니다.
- **`VQADataset`의 예외 처리 강화:**
  - `try...except (FileNotFoundError, Exception):` 구문을 사용하여 `FileNotFoundError` 외의 다른 예외 발생 시에도 빈 이미지로 대체하도록 예외 처리 범위가 확장되었습니다.
  - **개선점:** 이미지 로딩 중 발생할 수 있는 다양한 오류에 대한 안정성이 향상되었습니다.
- **`VQADataset`의 TTA 변환 추가:**
  - `tta_transforms`에 `transforms.RandomVerticalFlip(p=0.3)`이 추가되었습니다.
  - **개선점:** 데이터 증강 기법이 다양해져 모델의 일반화 성능 향상에 기여할 수 있습니다.
- **`collate_fn`의 프롬프트 변경:**
  - 프롬프트 형식이 `f"A daily life photo: {q}. The correct answer is: {c}"`에서 `f"{q} Possible answer: {c}"`로 변경되었습니다.
  - **개선점:** `"A daily life photo"`와 같은 도메인 특성 명시가 제거되어, 보다 일반적인 VQA 태스크에 적용하기 용이해졌습니다. 특정 도메인에 대한 의존성을 줄여 유연성을 높인 것으로 보입니다. 다만, 특정 도메인에 대한 성능은 이전 프롬프트가 더 좋을 수도 있습니다.

- **`collate_fn`의 예외 처리 추가:**
  - `processor` 호출 시 `try...except Exception` 구문이 추가되어 배치 처리 중 발생하는 오류를 출력하고 `None`을 반환하도록 했습니다.
  - **개선점:** 데이터 전처리 과정의 안정성이 향상되어, 잘못된 배치 데이터로 인한 전체 프로세스 중단을 방지합니다.

- **`fine_tune_model` 함수의 파라미터 변경:**
  - `epochs`가 3에서 5로 증가했습니다.
  - `batch_size`가 8에서 4로 감소했습니다.
  - `lr`(학습률)이 1e-5에서 5e-6으로 감소했습니다.
  - **개선점:** 에폭 증가와 학습률 감소는 모델이 더 오랫동안, 더 작은 보폭으로 학습하여 수렴 안정성과 최종 성능을 개선하는 데 도움이 될 수 있습니다. 배치 크기 감소는 메모리 사용량을 줄이고, 배치 노이즈를 증가시켜 일반화에 도움이 될 수 있습니다.

- **`fine_tune_model`의 손실(loss) NaN/Inf 체크:**
  - `if torch.isnan(loss) or torch.isinf(loss):` 구문이 추가되어 손실이 비정상적인 값(NaN 또는 Inf)일 경우 해당 배치를 건너뛰도록 했습니다.
  - **개선점:** 학습 과정의 안정성을 크게 향상시킵니다. 특히 불안정한 학습 환경이나 데이터 문제로 인해 손실이 발산하는 경우를 방지합니다.

- **모델 저장 방식 변경:**
  - `model.save_pretrained(..., safe_serialization=False)`로 `safe_serialization=False` 옵션이 추가되었습니다.
  - **개선점:** `safetensors` 대신 PyTorch의 기본 저장 방식을 사용하도록 하여 특정 환경에서의 호환성 문제를 해결할 수 있습니다.

- **`predict_test` 함수 분리 및 TTA 강화:**
  - 추론 로직이 `predict_test`라는 별도의 함수로 분리되었습니다.
  - `num_tta`가 2에서 4로 증가했습니다.
  - `tta_transforms_list`에 `transforms.Compose([])` (원본 이미지)가 추가되어 TTA 시 원본 이미지도 포함하여 예측하도록 했습니다.
  - `temperature`가 0.8에서 0.6으로 감소했습니다.
  - **개선점:** 추론 과정을 함수로 분리하여 코드의 가독성과 재사용성을 높였습니다. TTA 반복 횟수 증가와 원본 이미지 포함은 앙상블 효과를 더욱 강화하여 예측의 안정성과 정확

도를 높일 수 있습니다. `temperature` 감소는 모델의 예측을 더 확신하게 만들어, 최종 선택에 대한 신뢰도를 높이는 데 기여할 수 있습니다.

- **메인 실행 로직 변경:**
  - 학습과 추론을 각각 `fine_tune_model(train_df)`와 `predict_test(test_df, ...)` 함수 호출로 명확히 분리하여 실행합니다.
  - **개선점:** 코드의 구조가 더 명확해지고, 각 단계의 역할이 분명해졌습니다.

10.6. 결론

`본23차.ipynb`는 `본22차.ipynb`의 기본 구조를 유지하면서도, 학습 및 추론의 안정성, 일반화 성능, 그리고 코드의 유연성을 개선하기 위한 다양한 최적화가 적용된 버전입니다. 특히, 예외 처리 강화, TTA 기법 다양화 및 반복 횟수 증가, 학습 파라미터 조정, 그리고 손실 발산 방지 로직 추가 등은 모델의 견고함과 예측 정확도를 높이는 데 기여할 것으로 보입니다. 프롬프트 변경은 특정 도메인에 대한 의존성을 줄여 범용성을 높였지만, 경우에 따라서는 이전 프롬프트가 더 나은 성능을 보일 수도 있습니다. 전반적으로 `본23차.ipynb`는 보다 안정적이고 개선된 VQA 솔루션을 제공합니다.