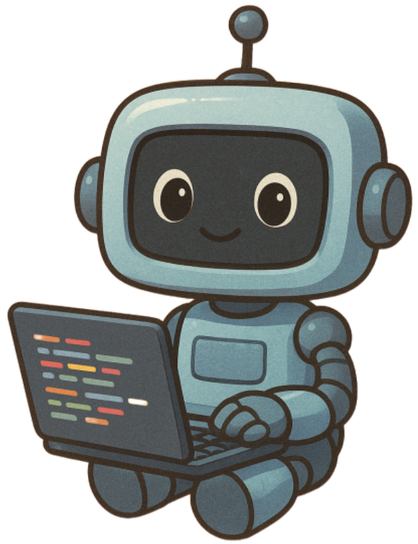


SSAFY AI 챌린지

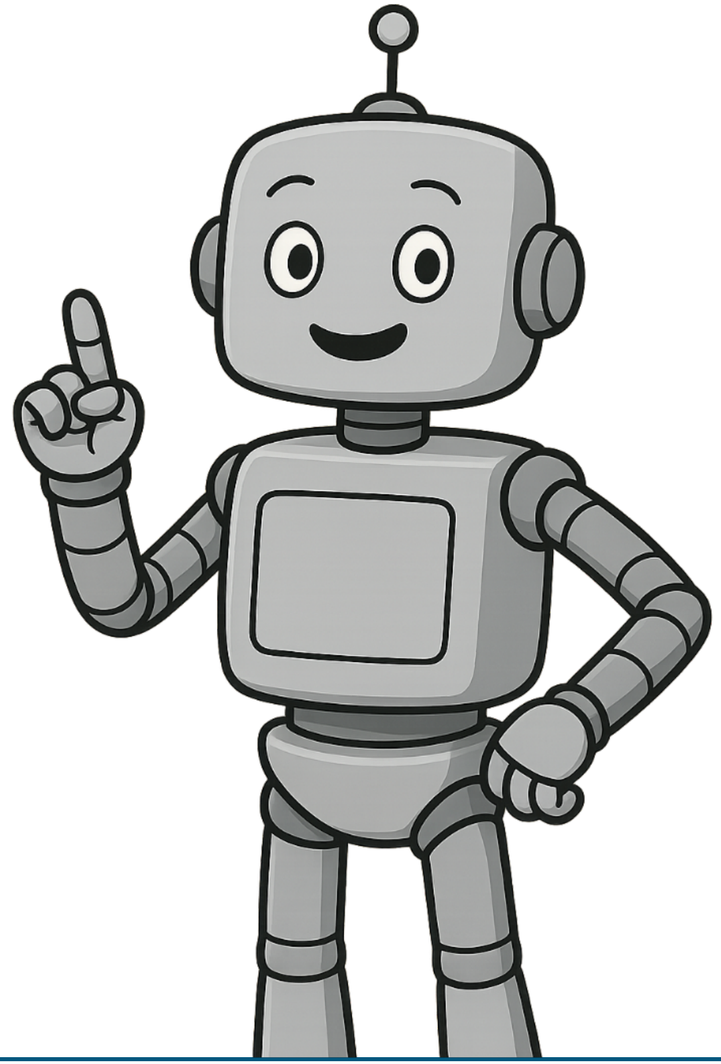
상미나이

김민수 양새하 한상민 황가연

CONTENTS

submissions

submission12.csv Complete · A044_한상민_1441020 · 15h ago	0.93106	☐
12-submission.csv Complete · A044_황가연_1440914 · 18h ago	0.26183	☐
11-submission.csv Complete · A044_황가연_1440914 · 2d ago	0.94135	☐
10-submission.csv Complete · A044_양새하_1444879 · 3d ago	0.86625	☐
submission.csv Complete · A044_황가연_1440914 · 3d ago	0.87860	☐
8 - submission.csv Complete · A044_한상민_1441020 · 3d ago	0.81224	☐
7 - submission.csv Complete · A044_김민수_1445638 · 3d ago	0.86162	☐
5-submission(5e-3).csv Complete · A044_양새하_1444879 · 3d ago · 5e-3	0.24176	☐
5- submission.csv Complete · A044_한상민_1441020 · 3d ago	0.79938	☐
4 - submission.csv Complete · A044_김민수_1445638 · 3d ago	0.86162	☐
submission.csv Complete · A044_한상민_1441020 · 3d ago	0.81224	☐
submission.csv Complete · A044_김민수_1445638 · 3d ago	0.76234	☐
submission.csv Complete · A044_한상민_1441020 · 3d ago	0.76028	☐



변경 사항

사전 학습 모델 정의

모델, processor

데이터 로더

fine-tuning

사전 학습 모델 정의

before

```
# 사전 학습 모델 정의
MODEL_ID = "Qwen/Qwen2.5-VL-3B-Instruct"
IMAGE_SIZE = 384
MAX_NEW_TOKENS = 8
SEED = 42
random.seed(SEED); torch.manual_seed(SEED); torch.cuda.manual_seed_all(SEED)

# 데이터셋 로드
train_df = pd.read_csv("/content/train.csv")
test_df  = pd.read_csv("/content/test.csv")

# 학습데이터 200개만 추출
train_df = train_df.sample(n=200, random_state=SEED).reset_index(drop=True)
```

after

```
# 사전 학습 모델 정의
MODEL_ID = "Qwen/Qwen2.5-VL-7B-Instruct"
IMAGE_SIZE = 448
MAX_NEW_TOKENS = 8
SEED = 42
random.seed(SEED); torch.manual_seed(SEED); torch.cuda.manual_seed_all(SEED)

# 데이터셋 로드
train_df = pd.read_csv("/home/team020/train.csv")
test_df  = pd.read_csv("/home/team020/test.csv")
```

Qwen/Qwen2.5-VL-3B-Instruct

↓

Qwen/Qwen2.5-VL-7B-Instruct

IMAGE_SIZE = 384

↓

IMAGE_SIZE = 448

학습 데이터 개수 : 200개

↓

학습 데이터 개수 : 전체

모델, processor

before

```
# 양자화
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.float16,
)

# 프로세서
processor = AutoProcessor.from_pretrained(
    MODEL_ID,
    min_pixels=IMAGE_SIZE*IMAGE_SIZE,
    max_pixels=IMAGE_SIZE*IMAGE_SIZE,
    trust_remote_code=True,
)

# 사전학습 모델
base_model = AutoModelForVision2Seq.from_pretrained(
    MODEL_ID,
    quantization_config=bnb_config,
    device_map="auto",
    trust_remote_code=True,
)

# 양자화 모델로 로드
base_model = prepare_model_for_kbit_training(base_model)
base_model.gradient_checkpointing_enable()

# LoRA 세팅
lora_config = LoraConfig(
    r=8,
    lora_alpha=16,
    lora_dropout=0.05,
    bias="none",
    target_modules=["q_proj","k_proj","v_proj","o_proj","gate_proj","up_proj","down_proj"],
    task_type="CAUSAL_LM",
)
```

after

```
# ----------------------------
# 프로세서
# ----------------------------
processor = AutoProcessor.from_pretrained(
    MODEL_ID,
    min_pixels=IMAGE_SIZE*IMAGE_SIZE,
    max_pixels=IMAGE_SIZE*IMAGE_SIZE,
    trust_remote_code=True,
)

# ----------------------------
# 사전학습 모델 로드 (bf16 full precision)
# ----------------------------
base_model = AutoModelForVision2Seq.from_pretrained(
    MODEL_ID,
    torch_dtype=torch.bfloat16,    # bf16 full precision
    device_map="auto",
    trust_remote_code=True,
)

# Gradient checkpointing 켜서 VRAM 절약
base_model.gradient_checkpointing_enable()

# ----------------------------
# LoRA 세팅
# ----------------------------
lora_config = LoraConfig(
    r=32,
    lora_alpha=64,
    lora_dropout=0.05,
    bias="none",
    target_modules=["q_proj","k_proj","v_proj","o_proj","gate_proj","up_proj","down_proj"],
    task_type="CAUSAL_LM",
)
```

모델, processor

after

```python
# ----------------------------
# 프로세서
# ----------------------------
processor = AutoProcessor.from_pretrained(
    MODEL_ID,
    min_pixels=IMAGE_SIZE*IMAGE_SIZE,
    max_pixels=IMAGE_SIZE*IMAGE_SIZE,
    trust_remote_code=True,
)

# ----------------------------
# 사전학습 모델 로드 (bf16 full precision)
# ----------------------------
base_model = AutoModelForVision2Seq.from_pretrained(
    MODEL_ID,
    torch_dtype=torch.bfloat16,    # bf16 full precision
    device_map="auto",
    trust_remote_code=True,
)

# Gradient checkpointing 켜서 VRAM 절약
base_model.gradient_checkpointing_enable()

# ----------------------------
# LoRA 세팅
# ----------------------------
lora_config = LoraConfig(
    r=32,
    lora_alpha=64,
    lora_dropout=0.05,
    bias="none",
    target_modules=["q_proj","k_proj","v_proj","o_proj","gate_proj","up_proj","down_proj"],
    task_type="CAUSAL_LM",
)
```

항목	기존	변경 후
양자화 설정	BitsAndBytes Config + 4bit	삭제
kbit 학습 준비	prepare_model_ for_kbit_training	삭제
dtype	4bit float16	bf16 full precision
gradient checkpointing	있음	그대로 유지
LoRA	r = 8 lora_alpha = 16	r = 32 lora_alpha = 64

데이터 로더

before

```
# 검증용 데이터 분리
split = int(len(train_df)*0.9)
train_subset, valid_subset = train_df.iloc[:split], train_df.iloc[split:]

# VQAMCDataset 형태로 변환
train_ds = VQAMCDataset(train_subset, processor, train=True)
valid_ds = VQAMCDataset(valid_subset, processor, train=True)

# 데이터로더
train_loader = DataLoader(train_ds, batch_size=1, shuffle=True, collate_fn=DataCollator(processor, True), num_workers=0)
valid_loader = DataLoader(valid_ds, batch_size=1, shuffle=False, collate_fn=DataCollator(processor, True), num_workers=0)
```

after

```
# 검증용 데이터 분리
split = int(len(train_df)*0.9)
train_subset, valid_subset = train_df.iloc[:split], train_df.iloc[split:]

# VQAMCDataset 형태로 변환
train_ds = VQAMCDataset(train_subset, processor, train=True)
valid_ds = VQAMCDataset(valid_subset, processor, train=True)

# 데이터로더
train_loader = DataLoader(train_ds, batch_size=2, shuffle=True, collate_fn=DataCollator(processor, True), num_workers=0)
valid_loader = DataLoader(valid_ds, batch_size=2, shuffle=False, collate_fn=DataCollator(processor, True), num_workers=0)
```

fine-tuning

before

```
# 옵티마이저, 학습 스케줄러
optimizer = torch.optim.AdamW(model.parameters(), lr=1e-4)
num_training_steps = 1 * math.ceil(len(train_loader)/GRAD_ACCUM)
scheduler = get_linear_schedule_with_warmup(optimizer, int(num_training_steps*0.03), num_training_steps)

# 학습 루프
global_step = 0
for epoch in range(1):
    running = 0.0
    progress_bar = tqdm(train_loader, desc=f"Epoch {epoch+1} [train]", unit="batch")
    for step, batch in enumerate(progress_bar, start=1):
        batch = {k:v.to(device) for k,v in batch.items()}
```

after

```
# 옵티마이저, 학습 스케줄러
optimizer = torch.optim.AdamW(model.parameters(), lr=2e-4)
num_training_steps = 5 * math.ceil(len(train_loader)/GRAD_ACCUM)
scheduler = get_linear_schedule_with_warmup(optimizer, int(num_training_steps*0.03), num_training_steps)

# 학습 루프
global_step = 0
for epoch in range(5):
    running = 0.0
    progress_bar = tqdm(train_loader, desc=f"Epoch {epoch+1} [train]", unit="batch")
    for step, batch in enumerate(progress_bar, start=1):
        batch = {k:v.to(device) for k,v in batch.items()}
```

학습률 (lr)

1e-4 -> 2e-4

epoch

1 -> 5

소감

김민수

AI 공부를 충분히 하지 못한 상태에서 챌린지를 하게 되어 많은 어려움이 있었으나 LLM을 사용하여 코드를 이해하고 사전학습 모델 사용법을 배울 수 있었습니다

황가연

모델 구조와 세부 하이퍼파라미터의 작은 변화만으로도 정확도에 큰 영향을 미친다는 것을 깨달았습니다
시간이 부족해 데이터 정제 과정을 충분히 수행하지 못한 점이 아쉽습니다

한상민

GPU의 중요성을 직접 체감하는 계기가 되었습니다
모델 선정도 중요하다는 것을 알 수 있었습니다

양새하

UI로 표현된 AI 모델링 툴을 여럿 사용해봤지만
직접 코드를 수정하며 AI 모델링을 해본것은 처음이었습니다

코드를 완벽하게 이해하지 못한 상황에서
잘못 수정했다가는 망칠 것 같다는 걱정에
소심하게 수치 조정 정도만 진행하였지만,

주변에서 ChatGPT 를 활용하여 코드를 엎는 것을 보고
용기를 얻어 점점 더 과감하게 코드 수정을 할 수 있었습니다

발표자료 준비를 하며, 이제서야 코드를 이해하게 되어,
SSH 서버 사용 시간이 조금 더 주어졌더라면,
더 제대로 학습시켜보고 싶다는 욕심이 생겼습니다

Thank you