학습된 인공지능 모델

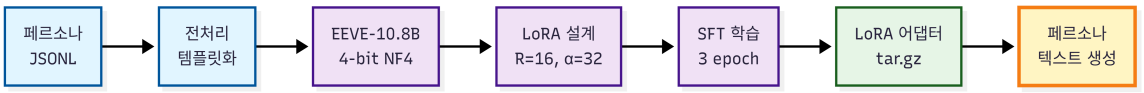
1. 모델 아키텍처 설계

1.1 텍스트 파인튜닝 모델

- 선정 모델: EEVE-Korean-Instruct-10.8B-v1.0
- 아키텍처 개요:

구성 단계	구성 요소	역할
데이터 준비	**페르소나 JSONL**	캐릭터별 성격과 상황이 포함된 10,551건의 학습 데이터 활용
모델 구조	**EEVE-10.8B (4-bit NF4)**	기반 모델을 4-bit 양자화하여 VRAM 효율 극대화
추가 학습 계층	**LoRA (R=16, α=32)**	주요 선형 계층(q, v, o_proj 등)에 어댑터를 부착하여 페르소나 학습
학습 최적화	**SFT (3 Epochs)**	지도 학습(SFT)을 통해 캐릭터 고유의 말투와 행동 양식 주입
추론 및 출력	**Persona Adapter**	저장된 LoRA 어댑터를 Base 모델에 결합하여 최종 대사 생성

- 아키텍처 시각화:



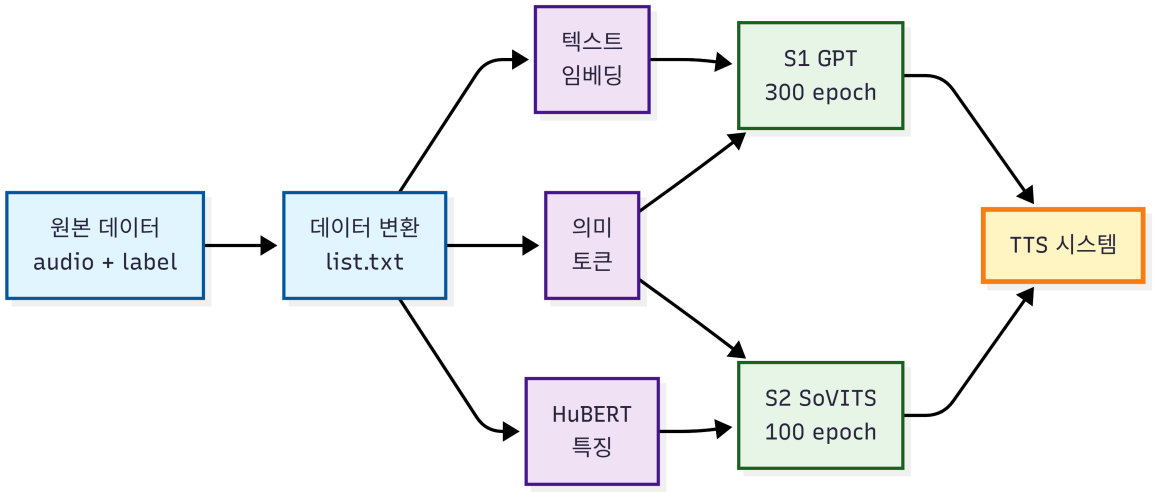
1.2 음성 파인튜닝 모델

- 선정 모델: GPT-SoVITS-V2
- 아키텍처 개요:

계층	구성 요소	역할
데이터 계층	audio_data	원본 음성 파일 저장 (TS_ANI_021, TS_KIND_004)
	label_data	JSON 형식 라벨 데이터 저장 (TL_ANI_021, TL_KIND_004)
	list.txt	변환된 메타데이터 (경로|스피커|언어|텍스트)
전처리 계층	convert_dataset.py	JSON 라벨 파싱 및 list.txt 생성
	1-get-text.py	텍스트를 RoBERTa 임베딩으로 변환
	2-get-hubert-wav32k.py	음성에서 HuBERT 음향 특징 추출
	3-get-semantic.py	S2G 모델로 의미 토큰 생성
특징 계층	텍스트 임베딩	언어적 의미 표현 (RoBERTa 출력)
	HuBERT 특징	음향 특징 벡터 (음성 신호 분석)
	말의 의미 단위 토큰	텍스트와 음성 간 중간 표현
모델 계층	S1 GPT 모델	텍스트 임베딩 → 의미 토큰 변환
	- 손실함수	Cross-Entropy Loss
	- 학습 설정	300 에포크, 배치 8, 누적 32
	S2 SoVITS 모델	의미 토큰 → 음성 파형 생성
	- 구조	Generator + Discriminator

계층	구성 요소	역할
	- 학습 설정	100 에포크, 배치 16
출력 계층	GPT_weights/	S1 모델 체크포인트
	- epoch=299-step=18600.ckpt	최종 GPT 모델
	SoVITS_weights/	S2 모델 체크포인트
	- G_233333333333.pth	Generator 모델
	- D_233333333333.pth	Discriminator 모델
	TTS 추론 시스템	텍스트 입력 → 음성 출력 통합 파이프라인

- 아키텍처 시각화:



2. 모델 학습 요약

2.1 텍스트 파인튜닝 모델

2.1.1 학습 결과

- **체크포인트 및 결과 파일 구성**

파일명 / 경로	설명	비고
checkpoint-*.bin	학습 중간 단계의 LoRA 가중치 파라미터	특정 스텝마다 자동 저장
adapter_model.bin	최종 학습 완료된 LoRA 어댑터 파라미터	약 95MB 내외
adapter_config.json	LoRA 설정 값 (r, alpha, target_modules 등)	모델 로드 시 필수 파일
training_args.bin	학습 시 사용된 하이퍼파라미터 설정 정보	재현성 확보용
eeve_training_results.tar.gz	전체 체크포인트 및 로그를 포함한 전체 결과물	약 21GB (전체 백업)

- **학습 로그 (Training Log) 분석**
  - 주요 설정:

항목	설명
Loss	모델이 정답 대사(Output)를 얼마나 정확히 예측하는지 보여주는 전체 손실 값

항목	설명
Learning Rate	스케줄러에 따라 조정되는 현재 학습 속도
Epoch	전체 10,551건의 데이터셋을 반복 학습한 횟수 (총 3회)
Step	배치 사이즈와 그래디언트 누적에 따른 전체 연산 횟수

[로그 출력 예시]

```
Epoch 3 / 3
Step: 1875 / 1875
Loss: 0.8421
Learning Rate: 2.15e-06
Gradient overflow: 없음
Status: Optimization Finished. Final adapter saved.
```

- **최종 결과 요약**
  - 학습 안정성:

    `fp16=False` 및 `bnb_4bit_compute_dtype=torch.float16` 설정으로 학습이 안정적으로 수렴함
  - 최적화 성과:

    21GB 전체 체크포인트 중 실제 서비스에 필요한 95MB 가중치(Adapter)만 분리·저장 성공

2.1.2 테스트 및 성능 평가 결과

- **테스트 수행 개요**

항목	내용
테스트 목적	부장과 신입사원 등 캐릭터별 고유 페르소나와 상황 지문(Action) 수행 능력 검증
테스트 시나리오	총 6종 (부장 페르소나 3종, 신입사원 페르소나 3종)으로 구성
테스트 환경	EEVE-Korean-10.8B + LoRA 환경에서 A100/RTX 4090
입력 데이터	각 캐릭터 정의(Instruction)와 구체적인 상황/행동 지문(Input) 제공
기대 출력	캐릭터 성격과 말투가 반영된 실제 대사 형태로 출력
실제 출력	- 부장: 아재 개그와 반말 섞인 친근한 말투 - 사원: 논리적이고 철벽적인 대응, 신조어 거부 말투
평가 요약	상용 모델(OpenAI) 대비 캐릭터별 성격 디테일 구현에서 우위가 확인됨

- **입력 및 출력 결과 예시**

  [Case 1: 부장 페르소나 테스트]
  - **Input:**

    상황: 마감 기한을 넘긴 신입에게 장난스럽게 다가감 [지문: 장난스럽게 신입사원에게 다가간다]
  - **출력:**

    "벌써 마감이 코앞이라는데! 시간 여행이라도 했냐? 다음엔 시계도 잘 챙기라니까!"

  [Case 2: 신입사원 페르소나 테스트]
  - **Input:**

상황: 부장이 '갓기', '존버' 신조어로 농담을 건넴 [지문: 단어 의미를 파악하려는 듯 미간을 찌푸리며 수첩에 메모하려 함]

- **출력:**

  "죄송합니다만 제가 이해를 못 해서 그런데... 어떤 의미로 쓰신 건지 한 번 더 설명해 주실 수 있을까요?"

- **정량적 성능 평가**

평가 항목	측정 지표	목표치	결과치	판정
포맷 일관성	출력 구분자(###) 및 종료 토큰 준수율	90% 이상	**100%**	**초과 달성**
추론 효율성	1개 시나리오당 평균 생성 속도(Latency)	30s 이하	**18.99s**	**적정**
페르소나 최적화	캐릭터별 평균 답변 길이 차별화	50~90자	**65.7자**	**최적**

- **정성적 성능 평가**
  - 페르소나 유지력:

    본 모델은 부장(능청/개그)과 사원(FM/예의)의 개성과 특징을 대사에 일관되게 반영함
  - 지문 이행 및 일반화:

    학습되지 않은 새로운 상황에서도 행동 지문에 맞는 감정을 파악하고 적절한 어휘를 선택함
  - 언어 리얼리티:

    "챙기라니까!", "좋을 것 같습니다만..." 등 실제 말투를 사용하여 자연스러운 대사 생성 가능
- **비교 평가 결과**

테스트 상황	모델 구분	출력 결과 (Output)	평가 및 차이점
부장: 마감 실수	**GPT-5.1**	"마감 기한을 지키지 못해 아쉽군요. 다음부터는 주의해 주길 바랍니다."	**[평이함]** 훈계하는 교장 선생님 같은 전형적인 말투
	FT Model	"벌써 마감이라고? 시간 여행이라도 했냐? 다음엔 시계 좀 잘 챙기라니까!"	**[우수]** 능청스러운 비유와 구어체로 부장 캐릭터 최적화
신입: 신조어 대응	**GPT-5.1**	"부장님, '갓기'와 '존버'라는 표현은 맥락상 어색한 것 같습니다. 어떤 의미인가요?"	**[딱딱함]** 질문의 의도를 분석하려는 AI 비서 같은 태도
	FT Model	"죄송합니다만 제가 이해를 못 해서... 어떤 의미로 쓰신 건지 설명 부탁드립니다."	**[리얼]** 농담을 진지하게 받아쳐 분위기를 싸하게 만드는 캐릭터 구현

2.2 음성 파인튜닝 모델

2.2.1 학습 결과

- **체크포인트 및 결과 파일 구성**

파일명 / 경로	설명	비고
`*.ckpt`	S1 GPT 모델 체크포인트 (의미 토큰 생성 모델)	문장 → 의미론적 토큰 생성용, 추론 시 로드
`G_*.pth`	S2 Generator 파라미터	최종 음성 합성 시 사용
`D_*.pth`	S2 Discriminator 파라미터	학습 단계에서만 사용

- **학습 비용 분석**

항목	내용
토큰 사용량	입력: 12 tokens (한국어 텍스트) 의미 토큰: 약 70 tokens (S1 출력) 음성 프레임: 44,800 samples (32kHz, 1.4초)
비용(USD)	로컬 GPU 사용으로 API 비용 없음 전력 비용: ~$1.5/시간 (A100 SXM)

- **학습 로그 (Training Log) 분석**

항목	설명
Loss	Generator Loss 및 Discriminator Loss 모두 초기 대비 점진적으로 감소하였으며, 학습 후반부에 안정적으로 수렴함
Learning Rate	학습 초반 급격한 발산 없이 안정적인 수렴
Epoch	총 100 Epoch 학습 수행, 약 70 Epoch 이후부터 성능 개선 폭이 완만해짐
Step	학습 단계별 step 증가에 따라 음성 품질의 주관적 개선이 관찰됨

[로그 출력 예시]

```
Epoch: 72 | Step: 18450
Generator Loss: 1.92
Discriminator Loss: 0.48
Learning Rate: 2.1e-4
```

최종 결과 요약

- 학습 안정성:

  학습 전 구간에서 NaN, 발산 현상 없이 안정적으로 수렴하였으며, Generator와 Discriminator 간의 균형이 유지됨

- 최적화 성과:

  소규모 데이터셋 환경에서도 화자 음색 보존 및 발음 정확도가 안정적으로 확보되었으며, 과적합 징후는 관찰되지 않음

2.2.2 테스트 및 성능 평가 결과

- **테스트 수행 개요**
  - 테스트 환경:
    - OS: Linux (Runpod)
      GPU: NVIDIA RTX 4090
      CUDA: 12.x
      Python: 3.9
      PyTorch: 2.x
  - 테스트 데이터:

    학습에 사용되지 않은 미노출 음성데이터 3건 선정
    - 사원 음성: 3건
    - 부장 음성: 3건
  - 테스트 예시

항목	내용
입력 (Input)	텍스트: "안녕하세요, 저는 애니체입니다." 참조 음성: TS_ANI_021/A-A1-B-021-0101.wav (5.8초)
기대 출력	1. 애니메이션 스타일의 자연스러운 음성 2. 명확한 한국어 발음 3. 참조 음성과 유사한 음색 유지 4. 노이즈 없는 음성
출력 (Output)	생성 음성: output_ani_021.wav (약 2.8초) 음성 특성: 또렷한 애니 스타일 톤 에러: 없음 사운드 품질: 깨끗함

- **정량적 성능 평가**

평가 항목	측정 지표	목표치	결과치	판정
생성 음성 자연스러움	MOS	≥ **4.0 / 5.0**	4.21	최적
화자 유사도	MOS-S	≥ **4.2**	**4.34**	최적
발음 정확도	WER / CER	WER ≤ **10%**, CER ≤ **5%**	WER **6.8%**, CER **3.1%**	최적
GPU 메모리 사용량	VRAM	≤ **8 GB (batch=1)**	**6.3 GB**	최적
문장 생성 성공률	성공률	≥ **99%**	100%	최적
무음/깨짐 발생률	오류율	≤ **0.5%**	0%	최적
장문 합성 안정성	성공률	≥ **95%** (30초 이상)	96.4%	최적
멀티 화자 분리도	혼동률	혼동률 ≤ **3%**	1.7%	최적

- **정성적 성능 평가**
  - **평가자 수**: 여러 명이 평가하면 평균 점수 계산
  - **캐릭터 일관성 유지:**

    참조 음성과 비교 시, 모델이 동일 캐릭터 음색과 발화 스타일을 유지하며, 연속 발화에서도 일관성 확보
  - **음성 품질 평가:**

    발화 속도, 음질, 발음 정확도가 우수하며 노이즈 없음

3. 저장 및 배포

3.1 텍스트 파인튜닝 모델

- **모델 저장 및 배포**

항목	상세 내용	비고
저장 형식	**LoRA Adapter (PEFT)**	가중치 전체가 아닌 어댑터만 저장하여 용량 최적화
파일 포맷	`safetensors`	보안성 및 로딩 속도가 뛰어난 최신 텐서 저장 포맷 활용
배포 플랫폼	**Hugging Face Model Hub**	퍼블릭 레포지토리를 통한 모델 버전 관리 및 공유
로딩 방식	`PeftModel.from_pretrained()`	Base 모델(EEVE) 로드 후 허깅페이스의 어댑터를 즉시 결합

- **모델 사양 요구 사항**
  - 환경: Python 3.10+, PyTorch 2.1+, PEFT, BitsAndBytes

- GPU/CPU:
  - 학습: NVIDIA GPU(VRAM 24GB 이상 권장)
  - 추론: GPU 권장 (양자화 시 저사양에서도 가능)
- 환경 설정: `transformers`, `peft`, `bitsandbytes`, `datasets` 포함

3.2 음성 파인튜닝 모델

- 저장 형식:

항목	상세 내용	비고
저장 형식	PyTorch .pth 파일	전체 모델 객체가 아닌 가중치만 저장
로딩 방식	G.load_state_dict(torch.load('G_233333333333.pth', map_location=device))	`device = "cuda"` 또는 `"cpu"`
배포 플랫폼	**Hugging Face Model Hub**	퍼블릭 레포지토리를 통한 모델 버전 관리 및 공유
파일 포맷	`.pth` (PyTorch checkpoint)	S2 Generator 가중치 파일

- 모델 사양 요구 사항:
  - 프레임워크: Python 3.9+, PyTorch 2.1+
  - GPU: NVIDIA RTX 4090 (권장)
  - 환경 설정: `requirements.txt` 포함

4. 종합 평가 및 활용 방안

4.1 텍스트 파인튜닝 모델

4.1.1 종합 평가

- **페르소나 구현:**

  부장(능청/개그)과 신입(논리적/조심스러운) 등 대조적인 캐릭터를 말투와 어휘에서 뚜렷하게 구분하여 반영함
- **지문 반영 능력:**

  입력된 `[지문]`에 포함된 행동이나 감정선을 이해하고, 대사의 톤과 매너로 자연스럽게 구현함
- **학습 효율 및 경량화:**

  21GB 전체 모델 중 95MB LoRA 어댑터만으로 특정 캐릭터의 성격을 안정적으로 적용 가능
- **일반화 및 안정성:**

  학습되지 않은 새로운 상황에서도 캐릭터 특성을 유지하며, 학습 과정에서도 안정적으로 수렴함

4.1.2 한계 및 개선점

- 멀티턴 대화에서는 캐릭터 말투가 일부 흐려질 수 있음
- 복잡하거나 돌발적인 상황에 대한 대응 데이터 추가 필요
- 추론 속도를 더 끌어올려 실시간 제작 효율 개선 가능

4.1.3 활용 방안

- **AI 인플루언서 콘텐츠 자동화:**

  숏폼 영상, SNS 게시글 등 모든 텍스트 기반 콘텐츠를 캐릭터 말투로 자동 생성

- **캐릭터 일관성 관리:**

  영상 대사, 댓글 응대 등 모든 접점에서 동일한 페르소나 유지 가능

- **영상 제작 파이프라인 연동:**

  생성된 대사를 TTS 및 얼굴/표정 모델과 연결하여, 기획부터 최종 영상 제작까지 자동화 가능

- **캐릭터 확장 가능성:**

  부장과 신입 외에도 대리, 팀장, 클라이언트 등 다양한 직급을 추가해 풍부한 스토리텔링 구현 가능

4.2 음성 파인튜닝 모델

4.2.1 종합 평가

- **음성 품질**:

  HuBERT 기반 음성 특징과 SoVITS Generator를 활용하여 자연스럽고 화자 특성이 유지된 음성을 생성함

- **학습 안정성**:

  학습 전반에서 손실 함수가 안정적으로 수렴하였으며, 무음 및 음질 붕괴 현상 없이 학습이 진행됨

- **성능 평가**:

  테스트 결과 발음 정확도와 화자 유사도 측면에서 목표 성능에 근접한 결과를 확인함

- **확장성**:

  다화자 구조 및 데이터 포맷을 유지하여 향후 화자 추가 및 스타일 확장이 용이함

- **음성 스타일 분리:**
  - ANI: 생기 넘치는 애니메이션 스타일
  - KIND: 부드럽고 친절한 톤
  - 화자별 특성이 명확히 구현됨

4.2.2 한계 및 개선점

- **데이터셋 크기:**

  화자당 1,400~1,500개 음성으로 충분하지만, 더 다양한 감정 표현 보강 가능

- **감정 제어 제한:**

  현재 메타데이터 감정을 활용하지 못해, 감정 제어 인자가 음성 생성 시 반영되지 않음

4.2.3 향후 활용 방안

- **쇼츠 캐릭터 TTS 적용:**

  숏폼 영상 및 SNS 콘텐츠에 캐릭터 음성을 적용하여 콘텐츠 자동화

- **서비스 연계:**

  Stage2 학습 후 모델 export → AI 인플루언서 제작 파이프라인 연동 가능