

+  
o •

# AI 활용 특강

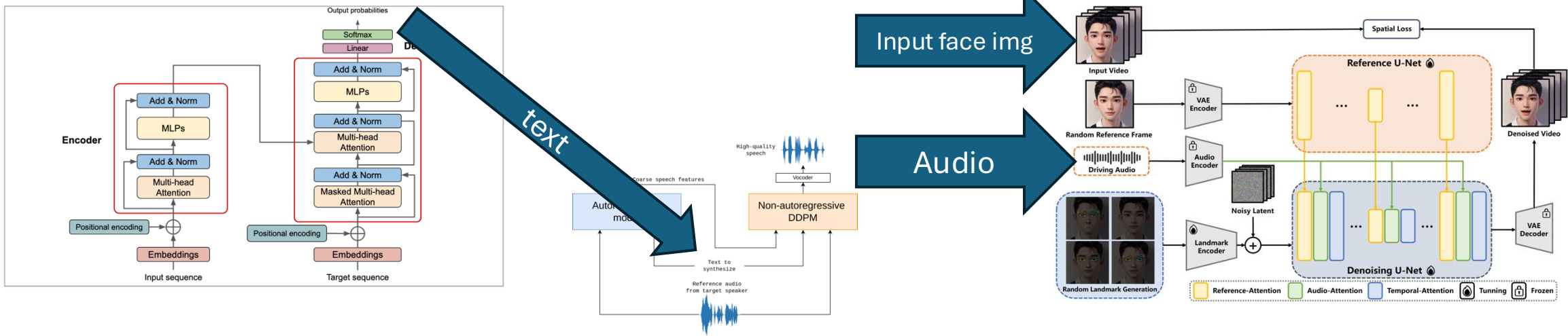
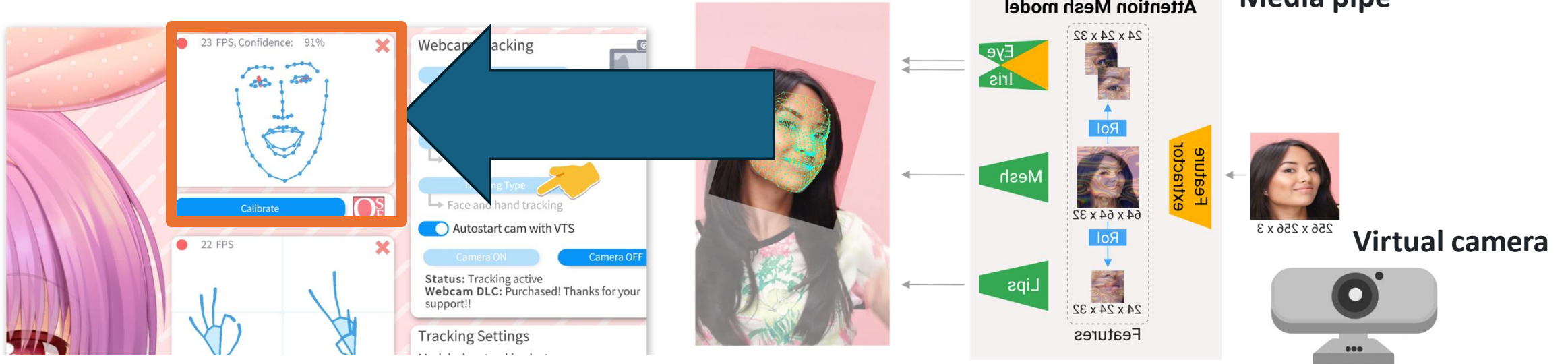
## LLM for AI ~~Vtuber~~(2)

+  
• o

구현해야  
하는 것들

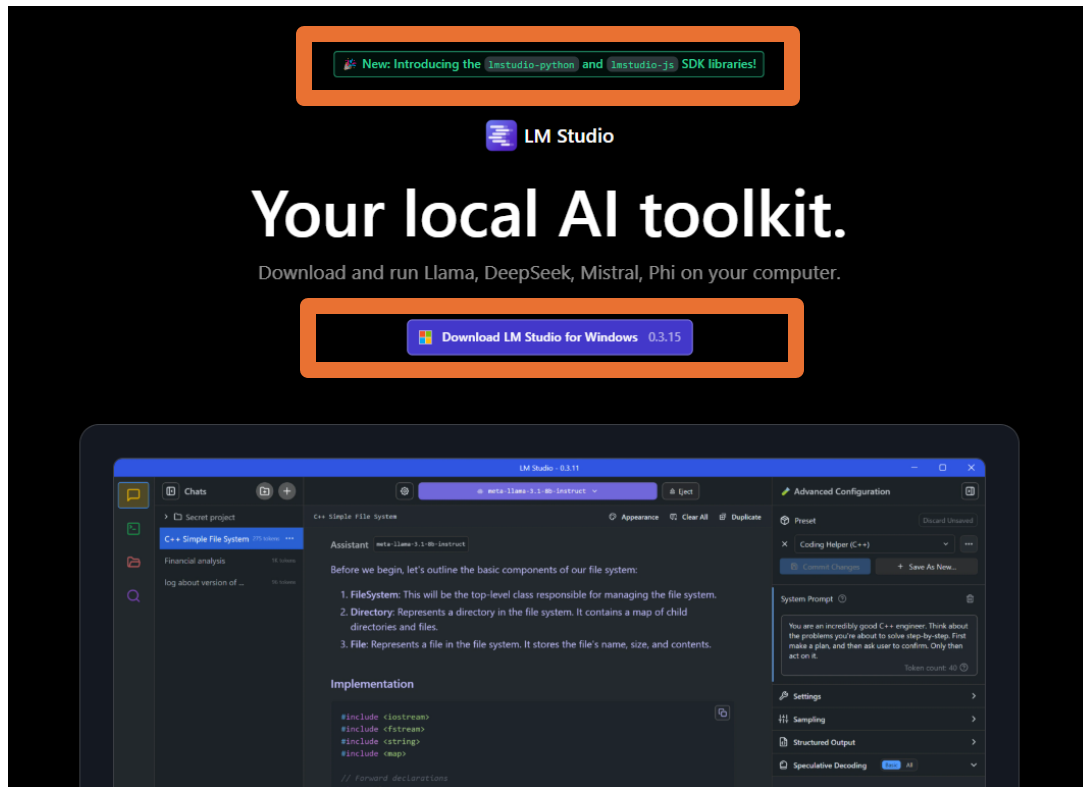
---

# Develop AI Vtuber target



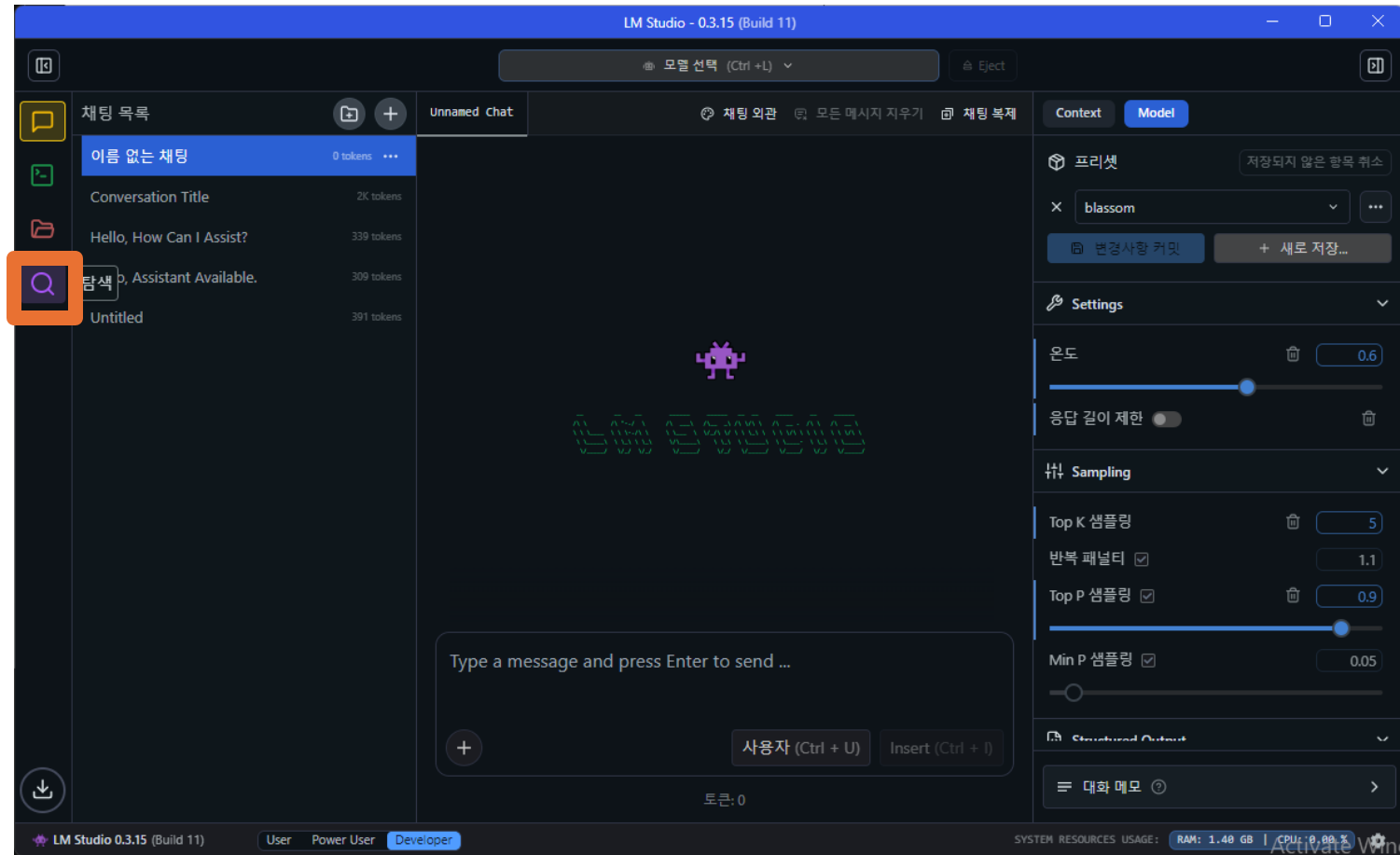
실습해보기  
버튜버 만들기 위해 필요한  
기능 구동해보기

# Running LLM for GUI Envorment

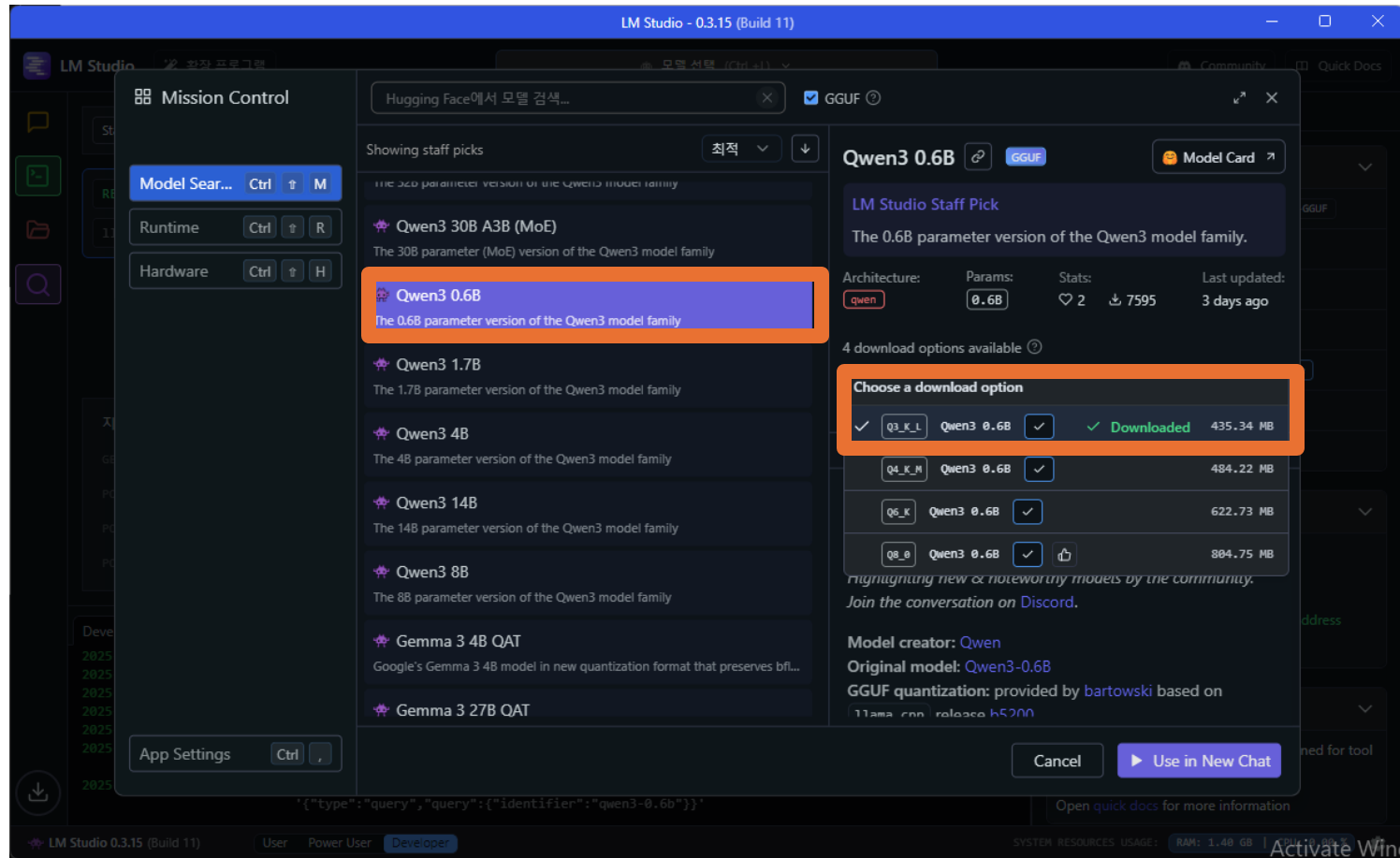


- GUI 환경에서 LLM On Device 구동 가능.
- (진짜 따끈따끈한)최근 python API 쉽게 불러 올 수 있게 만들어져 있음.
- API 관련 문서는 [여기로](#)

# Download Model

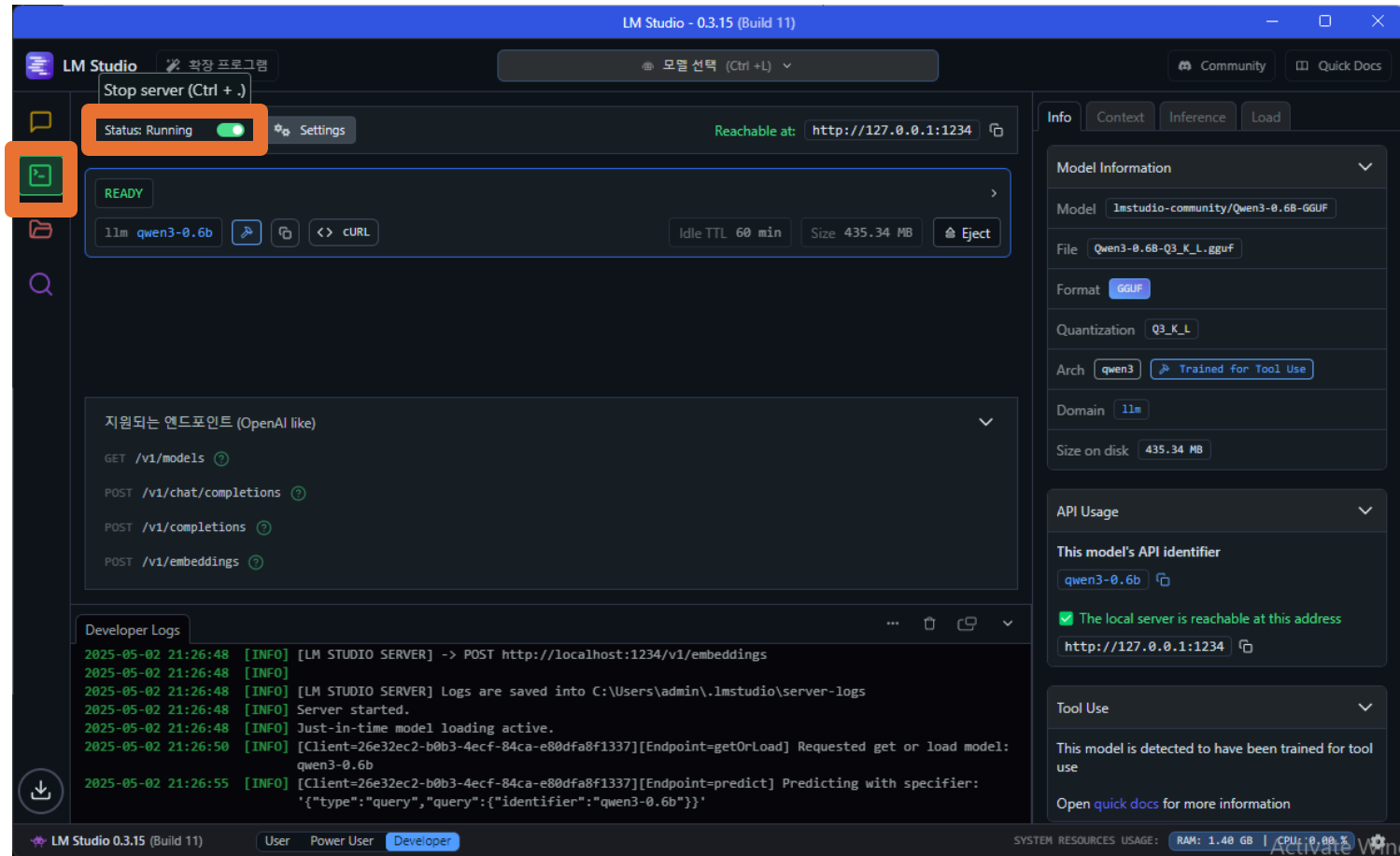


# 경량형 LLM QWEN 3.0 0.6B



\* 모델파일 자체적으로 500MB도 안한다!....ㅎㄷㄷ

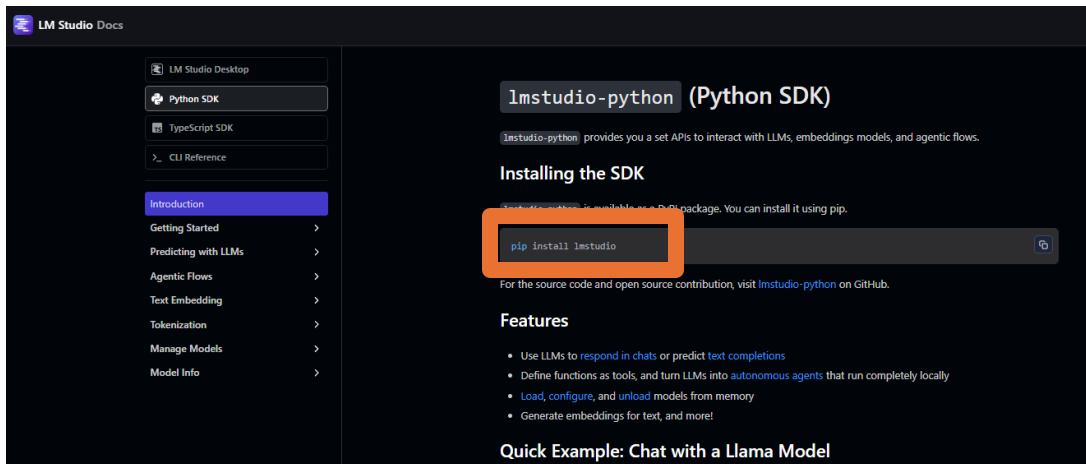
# LM Studio API 실행하기





# LM Studio API 파이썬에서 호출

- Pip install lmstudio 설치 ㄱ

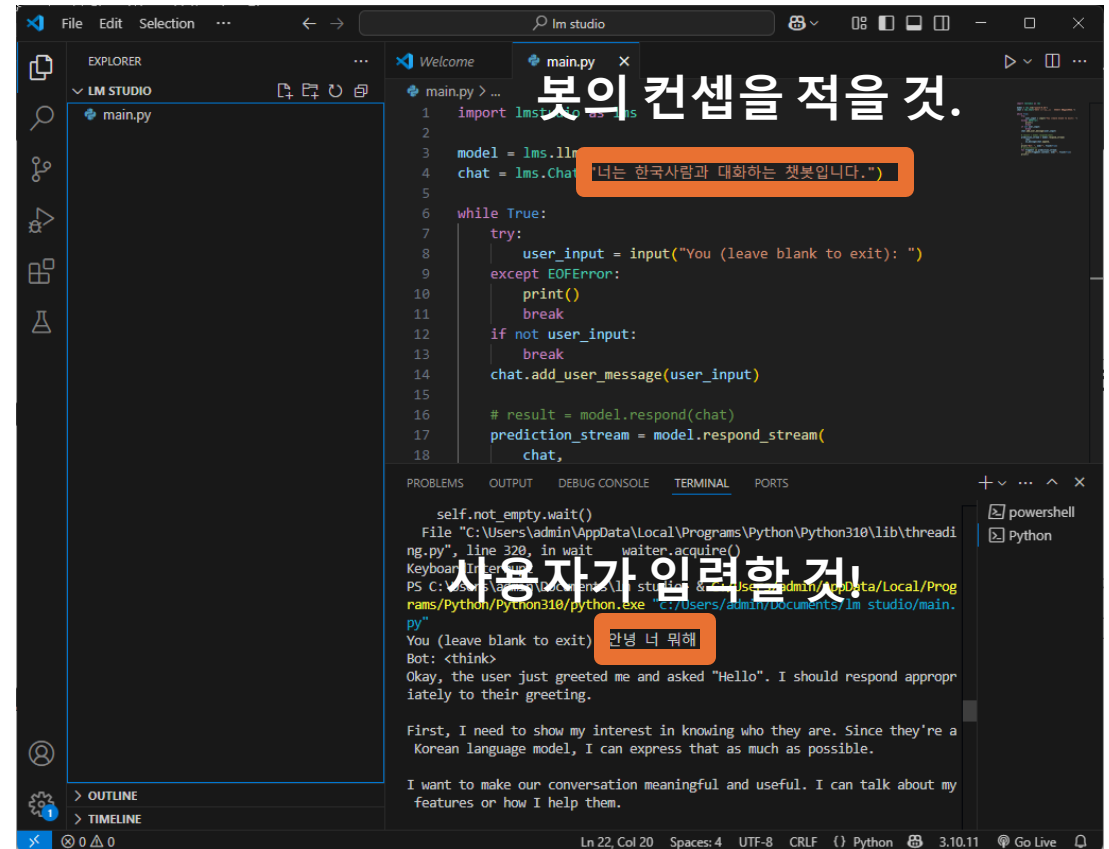


# LM Studio example Chat for Python

## Example: Multi-turn Chat

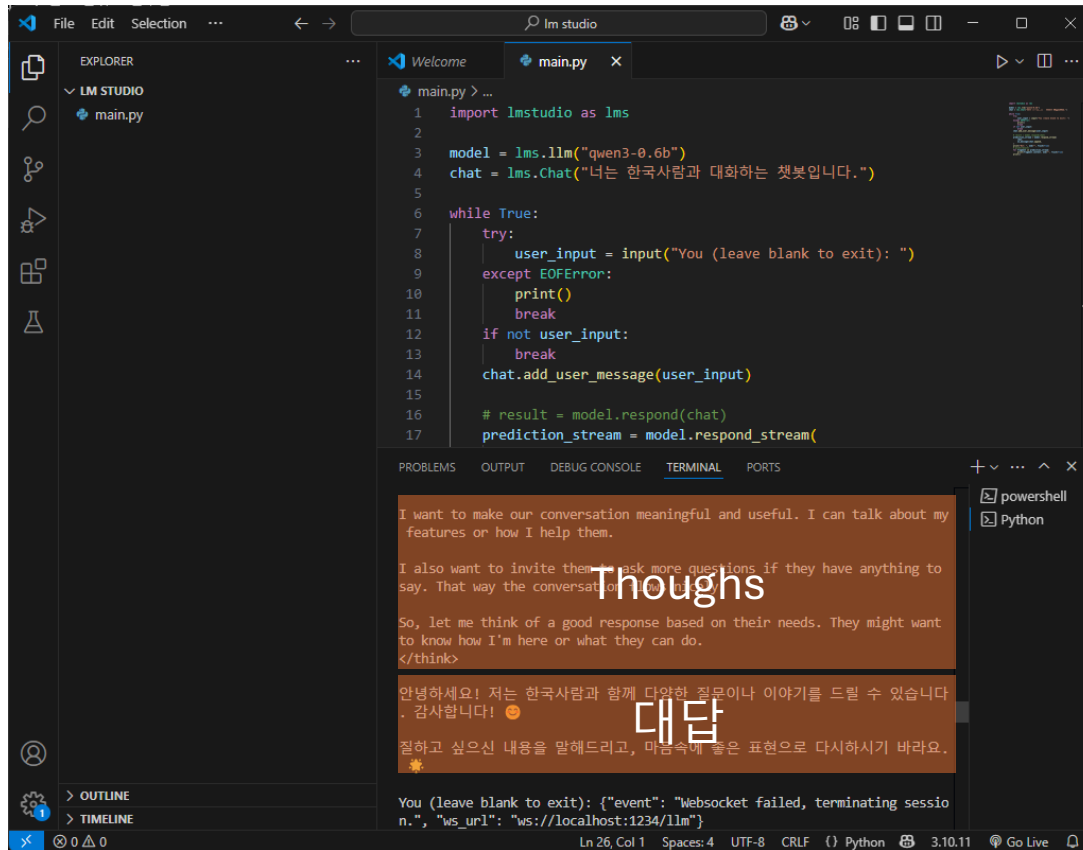
chatbot.py

```
1 import lmstudio as lms
2
3 model = lms.llm()
4 chat = lms.Chat("You are a task focused AI assistant")
5
6 while True:
7     try:
8         user_input = input("You (leave blank to exit): ")
9     except EOFError:
10         print()
11         break
12     if not user_input:
13         break
14     chat.add_user_message(user_input)
15     prediction_stream = model.respond_stream(
16         chat,
17         on_message=chat.append,
18     )
19     print("Bot: ", end="", flush=True)
20     for fragment in prediction_stream:
21         print(fragment.content, end="", flush=True)
22     print()
```



<https://lmstudio.ai/docs/python/llm-prediction/chat-completion>

# 결과 확인할 것



The screenshot shows a VS Code editor with a Python file named `main.py` open. The code imports `lmstudio` and creates a chatbot instance. It then enters a loop where it takes user input and sends it to the chatbot. The chatbot's response is printed to the terminal. The terminal output shows the chatbot's response, which includes a thinking phase and a final answer.

```
1 import lmstudio as lms
2
3 model = lms.llm("qwen3-0.6b")
4 chat = lms.Chat("너는 한국사람과 대화하는 챗봇입니다.")
5
6 while True:
7     try:
8         user_input = input("You (leave blank to exit): ")
9     except EOFError:
10        print()
11        break
12    if not user_input:
13        break
14    chat.add_user_message(user_input)
15
16    # result = model.respond(chat)
17    prediction_stream = model.respond_stream(
```

The terminal output shows the chatbot's response, which includes a thinking phase and a final answer. The response is displayed in a scrollable area with a dark background and light text. The text is as follows:

```
I want to make our conversation meaningful and useful. I can talk about my
features or how I help them.

I also want to invite them to ask more questions if they have anything to
say. That way the conversation can be more interesting.

So, let me think of a good response based on their needs. They might want
to know how I'm here or what they can do.
</think>

안녕하세요! 저는 한국사람과 함께 다양한 질문이나 이야기를 드릴 수 있습니다
. 감사합니다! 🌟

철하고 싶으신 내용을 알려드리고, 마음속에 좋은 표현으로 다시하시기 바라요.
🌟
```

The response is displayed in a scrollable area with a dark background and light text. The text is as follows:

```
You (leave blank to exit): {"event": "Websocket failed, terminating sessio
n.", "ws_url": "ws://localhost:1234/llm"}
```

- `</think>` 이후로 답변을 추출할 필요가 있음.
- 실제 String으로 되어 있는 값을 추출할 필요 있음.

# 실시간 값 확인하면서 텍스트 추출

```
import lmstudio as lms
from emoji import core

model = lms.llm("qwen3-0.6b")
chat = lms.Chat("너는 한국사람과 대화하는 챗봇입니다.")

while True:
    try:
        user_input = input("You (leave blank to exit): ")
    except EOFError:
        print()
        break
    if not user_input:
        break
    chat.add_user_message(user_input)

    prediction_stream = model.respond_stream(
        chat,
        on_message=chat.append,
    )
    print("Bot: ", end="", flush=True)

    for fragment in prediction_stream:
        print(fragment.content, end="", flush=True)
        print()

    bot_result = core.replace_emoji(prediction_stream.result().content.split('</think>')[1].strip())
    print(bot_result)
```

- 이모티콘이 있으니 제거해봅시다!
- pip install emoji 설치 ㄱㄱ

```
You (leave blank to exit): 안녕
Bot: <think>
Okay, the user just said "안녕" which translates to "Hello" in English. Let me think about how to respond appropriately.

Also, maybe I can offer some help or assistance to make the conversation more friendly. That way both parties feel comfortable and connected.

Let me check if there's anything else they might need. Maybe ask them about their current situation or interests in case we need further information.

I should keep my tone warm and welcoming, making sure the conversation flows naturally. Let me make sure I'm responding appropriately without adding any unnecessary information.
</think>

안녕하세요! 저는 항상 신뢰하고 건강하게 대화하며 도움을 드릴 수 있습니다. 어떤 주제나 관심 분야에 대해 이야기해주세요. 🌟
안녕하세요! 저는 항상 신뢰하고 건강하게 대화하며 도움을 드릴 수 있습니다. 어떤 주제나 관심 분야에 대해 이야기해주세요.
You (leave blank to exit):
```

# Text to Speech sample record mp3 download



[HOME](#)

성우녹음

게임사운드

AI 보이스데이터터

## 성우샘플

검색어를 적고 엔터를 눌러주세요

Search...

☐ 한국어

영어

☐ 중국어☐ 일본어

기타언어

☐ 남자☐ 여자

▶ 일반\_최윤희-나레01

Download

▶ 일본남 타스쿠 010

Download

▶ 일반\_김민희\_N6

Download

▶ KBS\_정해은\_세아\_o5

Download

▶ KBS\_서지연\_cvo\_Dorka\_skillo220

Download

▶ 일반\_문우리\_게임\_귀여운 여아

Download

▶ EBS\_오민혁\_40대다혈질남자

Download

<https://studiomonkey.co.kr/voicesample/> | 예제 음성 파일은 일반\_한호진\_강현\_03 임. 아무거나 해도 ㄱㅏ음.  
위 해당 링크로는 mp3로 제공함. Wav 파일로 변환해서 구동해야할 것!

# Text to Speech – OuteTTS install

- llama.cpp Build, llama.cpp Python 설치해야한다!

## Installation

### OuteTTS Installation Guide

OuteTTS now installs the llama.cpp Python bindings by default. Therefore, you must specify the installation based on your hardware. For more detailed instructions on building llama.cpp, refer to the following resources: [llama.cpp Build](#) and [llama.cpp Python](#)

**Pip:**

- ▼ Transformers + llama.cpp CPU

```
pip install outetts --upgrade
```

- ▶ Transformers + llama.cpp CUDA (NVIDIA GPUs)
- ▶ Transformers + llama.cpp ROCm/HIP (AMD GPUs)
- ▶ Transformers + llama.cpp Vulkan (Cross-platform GPU)
- ▶ Transformers + llama.cpp Metal (Apple Silicon/Mac)

## Usage

<https://github.com/edwko/OuteTTS?tab=readme-ov-file#installation> | 각자 환경 맞추어서 설치할 것!

# llama.cpp install

## Build llama.cpp locally

### To get the Code:

```
git clone https://github.com/ggml-org/llama.cpp
cd llama.cpp
```

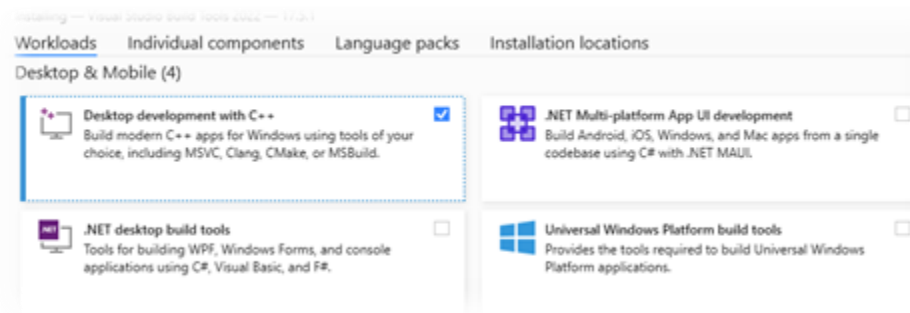
The following sections describe how to build with different backends and options.

### CPU Build

Build llama.cpp using CMake :

```
cmake -B build
cmake --build build --config Release
```

```
MINGW64/c/Users/admin/Downloads/llama.cpp
Created wheel for unidic-lite: filename=unidic_lite-1.0.8-py3-none-any.whl size=47658929 sha256=bca54
c36d6408d17b0b0b899a2eb2769ffb5d6740ba659ed85de57c7ef3121
Stored in directory: c:\users\admin\appdata\local\pip\cache\wheels\89\e8\68\f9ac36b8cc6c8b3c96888cd57
434abed96595d444f42243853
Successfully built llama-cpp-python openai-whisper argbind encodec julius randomname fire unidic-lite
Installing collected packages: wcwidth, unidic-lite, pure-eval, mpmath, mecab-python3, win32-setctime,
urllib3, typeguard, traitlets, threadpoolctl, termcolor, tensorboard-data-server, sympy, six, safetens
rs, regex, pyyaml, pygments, pygame, pycparser, protobuf, prompt_toolkit, polars, platformdi
rs, pillow, parso, packaging, numpy, networkx, natsort, msgpack, more-itertools, mdurl, MarkupSafe, mar
kdown2, markdown, llvmlite, kiwisolver, joblib, importlib-resources, grpcio, future, fsspec, fonttools,
filelock, ffmpeg, executing, einops, docstring-parser, diskcache, decorator, cycler, colorama, charset
-normalizer, audioread, asttokens, absl-py, werkzeug, uroman, tqdm, stack_data, soxr, scipy, requests, p
ython-dateutil, numba, matplotlib-inline, markdown-it-py, loguru, lazy_loader, jinja2, jedi, inflect, f
latten-dict, fire, contourpy, cffi, argbind, torch, tiktoken, tensorboard, soundfile, sounddevice, scik
it-learn, rich, randomname, pystoi, pyloudnorm, pooch, matplotlib, llama-cpp-python, ipython, huggingfa
ce-hub, torchvision, torchaudio, tokenizers, openai-whisper, librosa, julius, transformers, torch-stoi,
encodec, describe-audiotools, describe-audio-codec, outetts
Successfully installed MarkupSafe-3.0.2 absl-py-2.2.2 argbind-0.3.9 asttokens-3.0.0 audioread-3.0.1 cff
i-1.17.1 charset-normalizer-3.4.2 colorama-0.4.6 contourpy-1.3.2 cyclr-0.12.1 decorator-5.2.1 describe
-audio-codec-1.0.0 describe-audiotools-0.7.2 diskcache-5.6.3 docstring-parser-0.16 einops-0.8.1 encodec
-0.1.1 executing-2.2.0 ffmpeg-0.5.0 filelock-3.18.0 fire-0.7.0 flatten-dict-0.4.2 fonttools-4.57.0 fsspe
c-2025.3.2 future-1.0.0 grpcio-1.71.0 huggingface-hub-0.30.2 importlib-resources-6.5.2 inflect-7.5.0 ip
ython-8.36.0 jedi-0.19.2 jinja2-3.1.6 joblib-1.4.2 julius-0.2.7 kiwisolver-1.4.8 lazy_loader-0.4 libros
a-0.11.0 llama-cpp-python-0.3.8 llvmlite-0.44.0 loguru-0.7.3 markdown-3.8 markdown-it-py-3.0.0 markdown
-2.5.3 matplotlib-3.10.1 matplotlib-inline-0.1.7 mdurl-0.1.2 mecab-python3-1.0.10 more-itertools-10.7.
0 mpmath-1.3.0 msgpack-1.1.0 natsort-8.4.0 networkx-3.4.2 numba-0.61.2 numpy-2.2.5 openai-whisper-20240
930 outetts-0.4.1 packaging-25.0 parso-0.8.4 pillow-11.2.1 platformdirs-4.3.7 polars-1.29.0 pooch-1.8.2
prompt_toolkit-3.0.51 protobuf-3.19.6 pure-eval-0.2.3 pycparser-2.22 pygame-2.6.1 pygments-2.19.1 pylo
udnorm-0.1.1 pyparsing-3.2.3 pystoi-0.4.1 python-dateutil-2.9.0.post0 pyyaml-6.0.2 randomname-0.2.1 reg
ex-2024.11.6 requests-2.32.3 rich-14.0.0 safetensors-0.5.3 scikit-learn-1.6.1 scipy-1.15.2 six-1.17.0 s
ounddevice-0.5.1 soundfile-0.13.1 soxr-0.5.0 post1 stack_data-0.6.3 sympy-1.14.0 tensorboard-2.19.0 ten
sorboard-data-server-0.7.2 termcolor-3.1.0 threadpoolctl-3.6.0 tiktoken-0.9.0 tokenizers-0.21.1 torch-2
.7.0 torch-stoi-0.2.3 torchaudio-2.7.0 torchvision-0.22.0 tqdm-4.67.1 traitlets-5.14.3 transformers-4.4
8.3 typeguard-4.4.2 unidic-lite-1.0.8 urllib3-2.4.0 uroman-1.3.1.1 wcwidth-0.2.13 werkzeug-3.1.3 win32-
setctime-1.2.0
admin@DESKTOP-GLT2ICF MINGW64 ~/Downloads/llama.cpp (master)
$
```



**중요!** cmake 설치 필요한 학생(윈도우 기준) : <https://ndb796.tistory.com/365> 들어가서 설치  
Visual Studio 2022 Community 데스크탑 C++ 설치할 것! 여기에서 대부분 많이 막힐거다 ㅋㅋ..

# Llama cpp python install

## Installation Configuration

`llama.cpp` supports a number of hardware acceleration backends to speed up inference as well as backend specific options. See the [llama.cpp README](#) for a full list.

All `llama.cpp` cmake build options can be set via the `CMAKE_ARGS` environment variable or via the `--config-settings / -C` cli flag during installation.

### ▼ Environment Variables

```
# Linux and Mac
CMAKE_ARGS="-DGGML_BLAS=ON -DGGML_BLAS_VENDOR=OpenBLAS" \
pip install llama-cpp-python
```

```
# Windows
$env:CMAKE_ARGS = "-DGGML_BLAS=ON -DGGML_BLAS_VENDOR=OpenBLAS"
pip install llama-cpp-python
```



# OuteTTS install

```
PS C:\Users\admin\Documents\lm studio> pip install outetts --upgrade
Requirement already satisfied: outetts in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (0.4.1)
Requirement already satisfied: llama-cpp-python==0.3.8 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from outetts) (0.3.8)
Requirement already satisfied: torch in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from outetts) (2.7.0)
Requirement already satisfied: torchvision in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from outetts) (0.22.0)
Requirement already satisfied: torchaudio in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from outetts) (2.7.0)
Requirement already satisfied: scipy in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from outetts) (1.15.2)
Requirement already satisfied: einops in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from outetts) (0.8.1)
Requirement already satisfied: pyyaml in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from outetts) (6.0.2)
Requirement already satisfied: huggingface-hub in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from outetts) (0.30.2)
Requirement already satisfied: encodec in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from outetts) (0.1.1)
Requirement already satisfied: matplotlib in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from outetts) (3.10.1)
Requirement already satisfied: transformers==4.48.3 in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from outetts) (4.48.3)
Requirement already satisfied: soundfile in c:\users\admin\appdata\local\programs\python\python310\lib\site-packages (from outetts) (0.13.1)
```

# OuteTTS 목소리 추출 & 생성

```
import outetts

# Initialize the interface
interface = outetts.Interface(
    config=outetts.ModelConfig.auto_config(
        model=outetts.Models.VERSION_1_0_SIZE_1B,
        # For llama.cpp backend
        backend=outetts.Backend.LLAMACPP,
        quantization=outetts.LlamaCppQuantization.Q2_K # 양자화 모델, 작은걸로 함
    )
)

speaker = interface.create_speaker("audio.wav") # 생성할 목소리 원본파일 넣기
interface.save_speaker(speaker, "speaker.json")
```

```
import outetts

# Initialize the interface
interface = outetts.Interface(
    config=outetts.ModelConfig.auto_config(
        model=outetts.Models.VERSION_1_0_SIZE_1B,
        # For llama.cpp backend
        backend=outetts.Backend.LLAMACPP,
        quantization=outetts.LlamaCppQuantization.Q2_K
        # For transformers backend
        # backend=outetts.Backend.HF,
    )
)

speaker = interface.load_speaker("speaker.json")




# Generate speech
output = interface.generate(
    config=outetts.GenerationConfig(
        text="텍스트 음성 생성할 내용 작성",
        generation_type=outetts.GenerationType.CHUNKED,
        speaker=speaker,
        sampler_config=outetts.SamplerConfig(
            temperature=0.6
        ),
    )
)

# Save to file
output.save("output.wav")
```

<https://huggingface.co/OuteAI/Llama-OuteTTS-1.0-1B-GGUF#quick-start-guide>

# 설정하는 것이 오래걸리고 실행 속도 느리다면?

- 일반적인 TTS 사용하는 것으로  
해봅시다.

 hexgrad / **Kokoro-82M**   like 4.2k



Text-to-Speech



English

doi:10.57967/hf/4329



arxiv:2306.07691



arxiv:2203.02395



Model card



Files and versions

**Kokoro** is an open-weight TTS model with 82 million parameters. Despite its lightweight architecture, it delivers comparable quality to larger models while being significantly faster and more cost-efficient. With Apache-licensed weights, Kokoro can be deployed anywhere from production environments to personal projects.



0:00



-0:20



GitHub: <https://github.com/hexgrad/kokoro>



Demo: <https://hf.co/spaces/hexgrad/Kokoro-TTS>

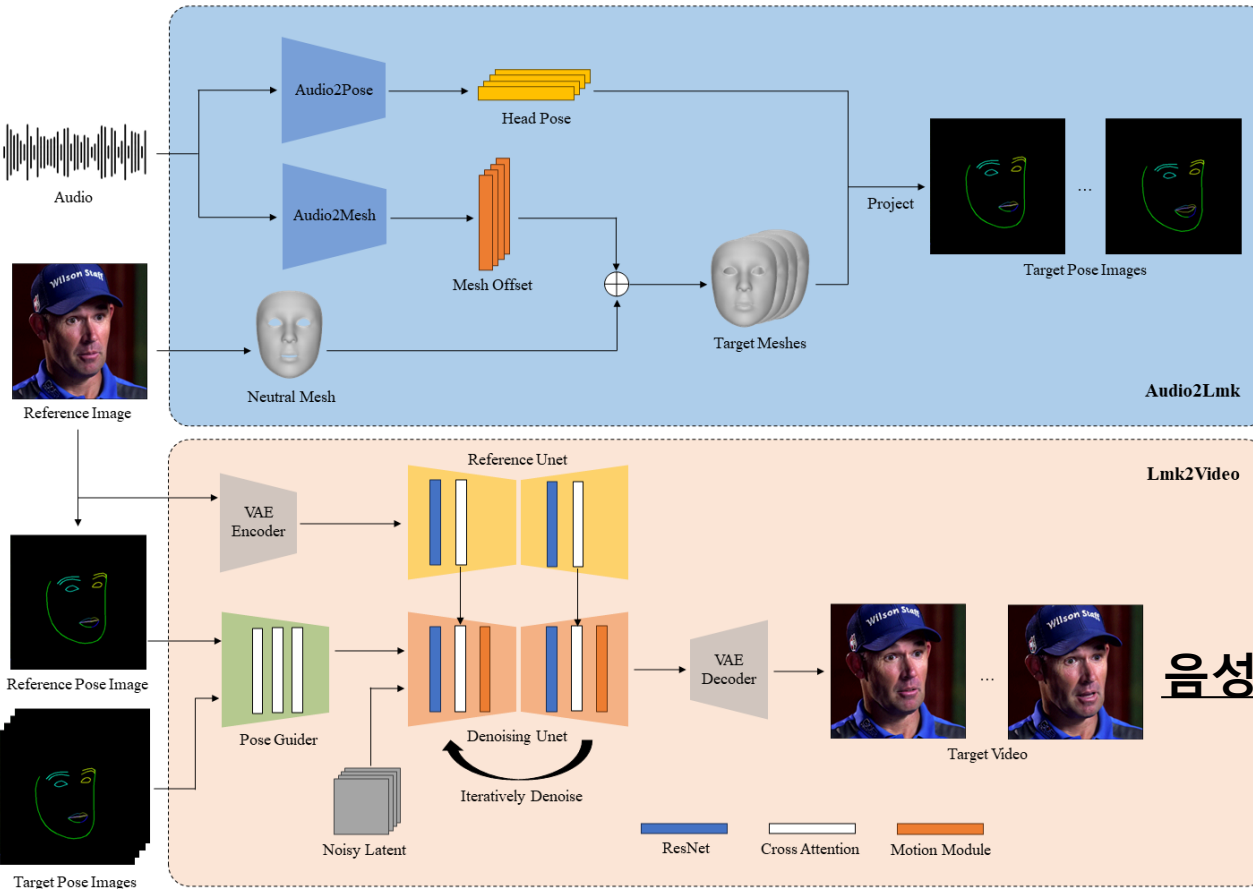
# 실습과제

## Audio Driven 기능 실행해보기

# AniPortrait: Audio-Driven Synthesis of Photorealistic Portrait Animations

내가 말해야하는 음성(TTS 결과)

얼굴 움직여야하는 대상 이미지

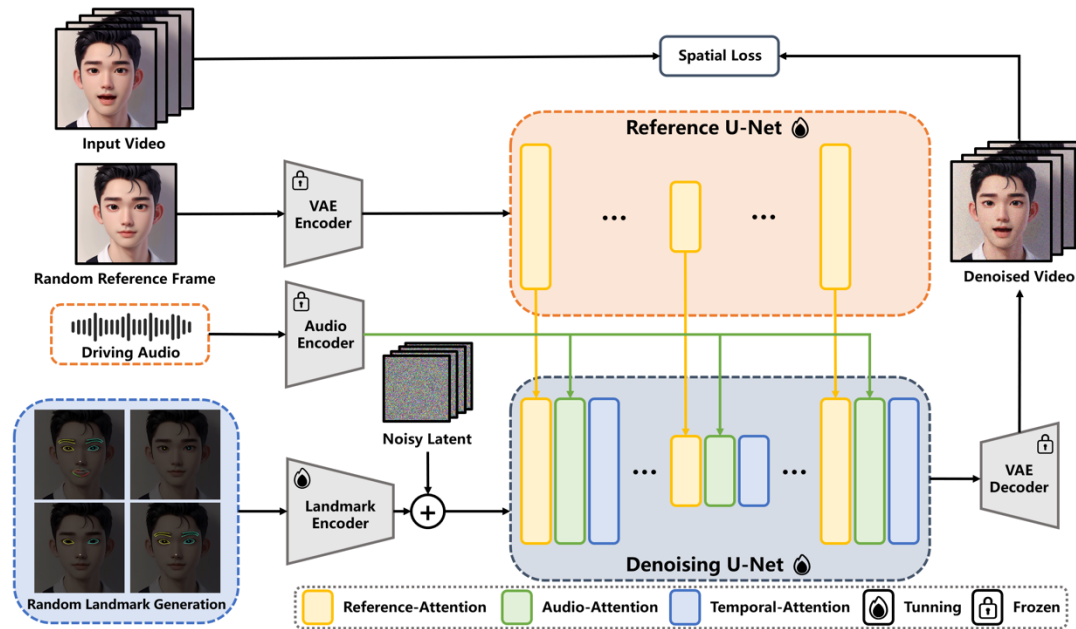


음성에 맞는 얼굴 생성

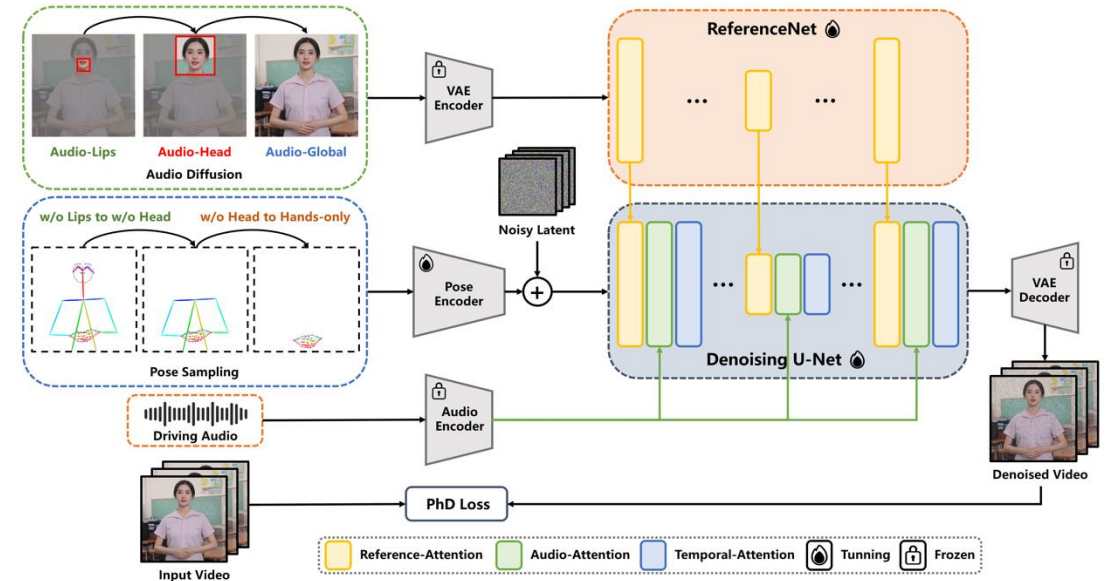
이미지 생성 기술은 Stable Diffusion 이라는 것을 확인할 수 있음..! (SD1.5 version)

# EchoMimic: Lifelike Audio-Driven Portrait Animations through Editable Landmark Conditioning

v1



v2



# 이러한 것들을 구동하기 위해서 필요한 것?

- ~~좋은 컴퓨터 부품과 API 호출할 돈만 준비되면 가능하다!~~
- 요구 사양을 낮추고 싶다면 양자화 모델이나 크기가 작은 모델로 사용하면 된다!

# Real-time video audio api

- <https://x.com/i/status/1915808257958154284>



# TTS 음질 좋게 하고 싶다면?

**Nari Labs: Dia Examples**

Comparison between [Dia-1.6B \(ours\)](#), [ElevenLabs Studio](#), and [Sesame CSM-1B](#). Plus fun examples (including audio prompt use).

**Standard Usage**

Input script

```
[S1] Dia is an open weights text to dialogue model.  
[S2] You get full control over scripts and voices.  
[S1] Wow. Amazing. (laughs)  
[S2] Try it now on Git hub or Hugging Face.
```

**Dia-1.6B (ours)**

▶ 0:00 / 0:00

**ElevenLabs Studio**

▶ 0:00 / 0:00

**Sesame CSM-1B**

▶ 0:00 / 0:00

Note that ElevenLabs and Sesame models do not have the ability to transcribe laughter tags into speech. We replace (laughs) with haha. Also, Dia is not fine-tuned on a specific voice. It will generate random voices unless you add audio prompts, or fix the seed.

- <https://yummy-fir-7a4.notion.site/dia>

- ~~물론 모델 크기가 10GB인걸로...~~