In-browser AI apps with Gradio and Transformers

FEDAY 2024

Yuichiro Tachibana, ML Developer Advocate @ 🤗

In-browser AI apps with Gradio and Transformers

tGPT 4o

Share

T

Generate an image of a futuristic city at sunset, with flying cars zooming between towering skyscrapers made of glass and steel, and vibrant neon signs in multiple languages glowing against the skyline.

e generated image of a futuristic city at sunset, featuring flying cars, towering glass and scrapers, and vibrant neon signs. L e know if you'd like any adjustments or further ions!

ChatGPT can make mistakes. Check important info.

AI apps

SUNO

Custom

Upload Audio

v3.5

Song description

Instrumental

Home

Create

Library

Explore

Search

A futuristic techno song featuring AI app dev tools like Transformers

69 / 200

Create

Need ideas?

Invite Friends

40 Credits

Subscribe

What's New? 9

Notifications

More from Suno

Digital Symphony ExasperatingCymbals...

00:13 /

Digital

futuristic

Public

Digital

futuristic

Public

Tasks | Libraries | Datasets | Languages | Licenses | Other

Filter Tasks by name

Multimodal

- Audio-Text-to-Text
- Image-Text-to-Text
- Visual Question Answering
- Document Question Answering
- Video-Text-to-Text
- Any-to-Any

Computer Vision

- Depth Estimation
- Image Classification
- Object Detection
- Image Segmentation
- Text-to-Image
- Image-to-Text
- Image-to-Image
- Image-to-Video
- Unconditional Image Generation
- Video Classification
- Text-to-Video
- Zero-Shot Image Classification
- Mask Generation
- Zero-Shot Object Detection
- Text-to-3D
- Image-to-3D
- Image Feature Extraction
- Keypoint Detection

Natural Language Processing

- Text Classification
- Token Classification
- Table Question Answering
- Question Answering

Models 1,173,133 | Filter by name | Full-text search | ↑↓ Sort: Trending

Qwen/QwQ-32B-Preview
Text Generation • Updated 5 days ago • 33.6k • ♡ 1k

tencent/HunyuanVideo
Updated about 7 hours ago • ♡ 339

Lightricks/LTX-Video
Image-to-Video • Updated 11 days ago • 32.3k • ♡ 574

black-forest-labs/FLUX.1-dev
Text-to-Image • Updated Aug 16 • 1.38M • ♡ 6.9k

Djran[illegible]

AIDC-AI/Marco-o1
Text Generation • Updated 11 days ago • 9.71k • ♡ 605

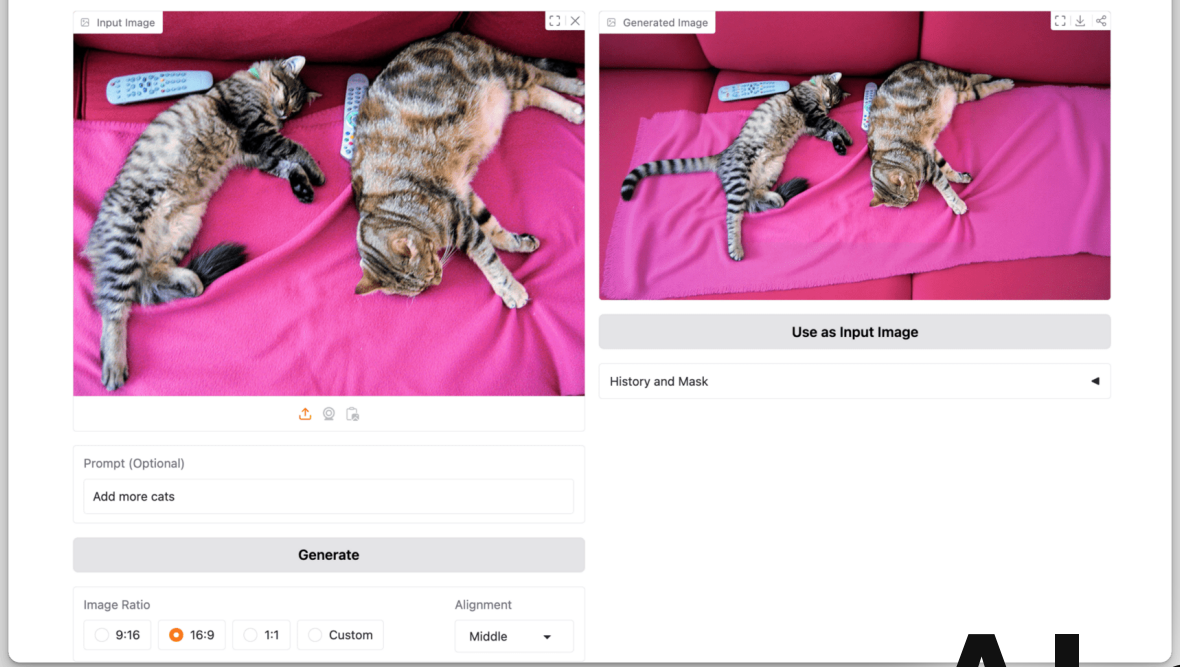
ginipick/flux-lora-eric-cat
Text-to-Image • Updated 2 days ago • 4.18k • ♡ 196

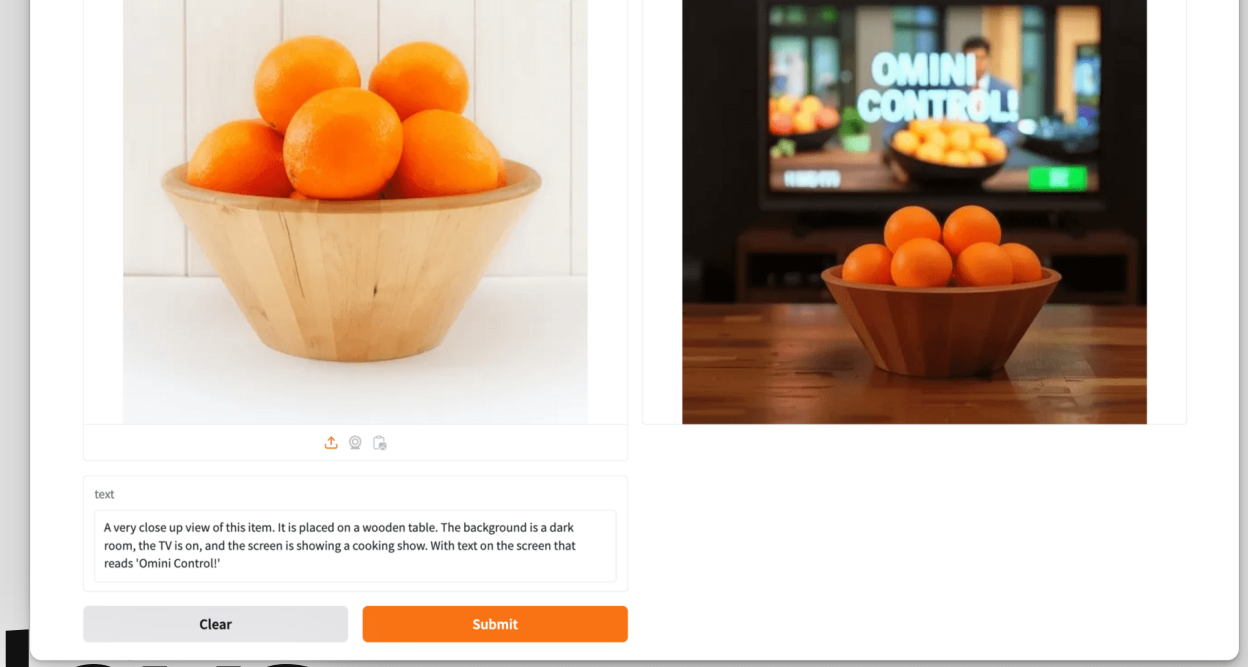
HuggingFaceTB/SmolVLM-Instruct
Image-Text-to-Text • Updated 1 day ago • 21.7k • ♡ 226

seawolf2357/flux-lora-car-rolls-royce
Text-to-Image • Updated Sep 17 • 339 • ♡ 184

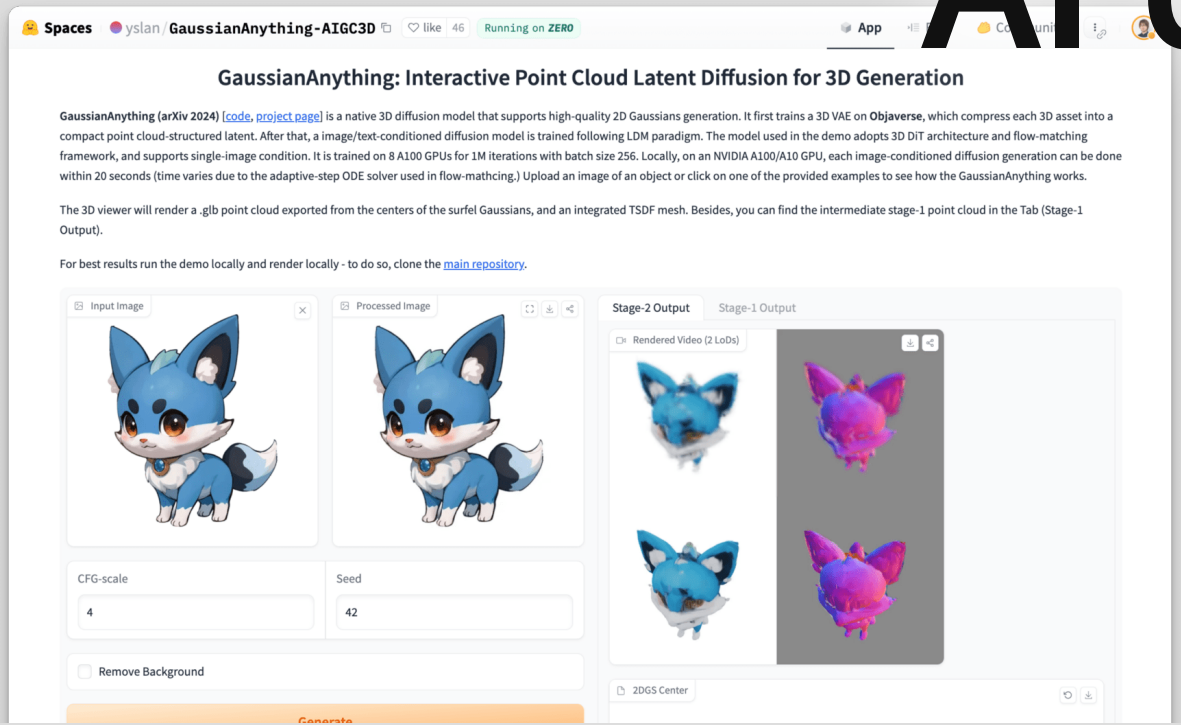
Qwen/Qwen2.5-Coder-32B-Instruct
Text Generation • Updated 16 days ago • 170k • ♡ 1.16k

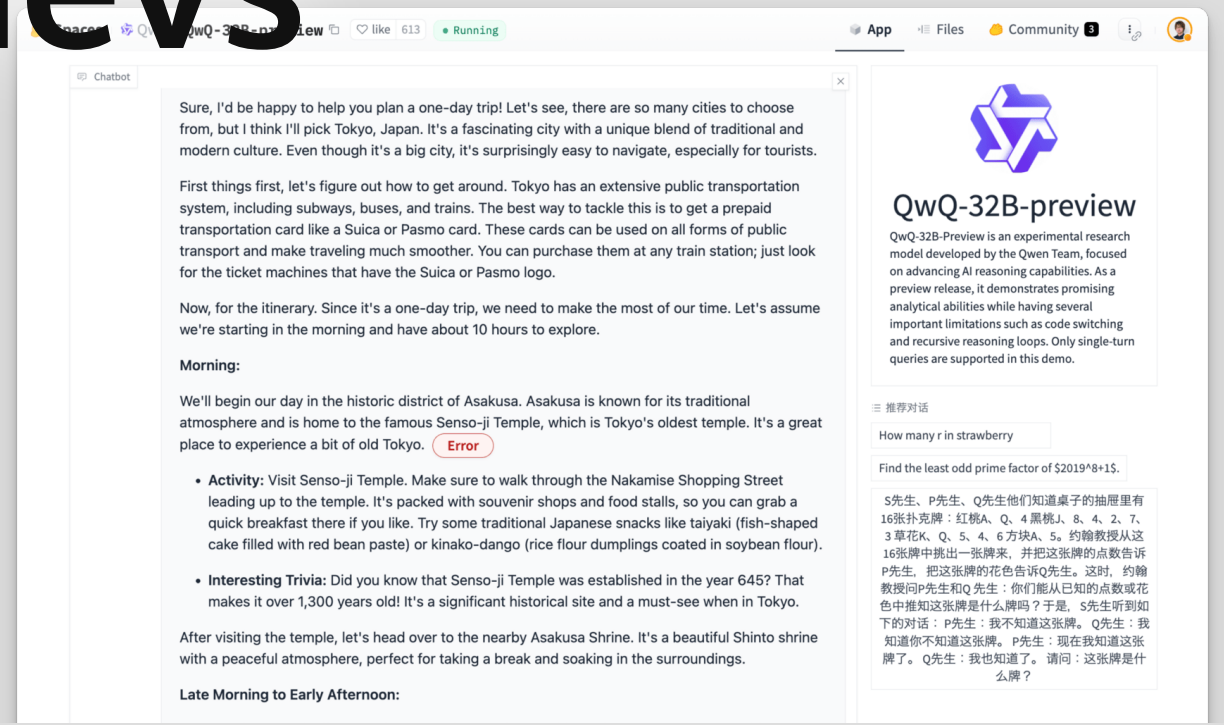
AI devs


Input Image
Generated Image
Use as Input Image
History and Mask
Prompt (Optional)
Add more cats
Generate
Image Ratio
9:16
16:9
1:1
Custom
Alignment
Middle


text
A very close up view of this item. It is placed on a wooden table. The background is a dark room, the TV is on, and the screen is showing a cooking show. With text on the screen that reads 'Omini Control!'
Clear
Submit
OMINI CONTROL!

AI devs


Spaces
yslan / GaussianAnything-AIGC3D
like 46
Running on ZERO
App
GaussianAnything: Interactive Point Cloud Latent Diffusion for 3D Generation
GaussianAnything (arXiv 2024) [code, project page] is a native 3D diffusion model that supports high-quality 2D Gaussians generation. It first trains a 3D VAE on Objaverse, which compress each 3D asset into a compact point cloud-structured latent. After that, a image/text-conditioned diffusion model is trained following LDM paradigm. The model used in the demo adopts 3D DiT architecture and flow-matching framework, and supports single-image condition. It is trained on 8 A100 GPUs for 1M iterations with batch size 256. Locally, on an NVIDIA A100/A10 GPU, each image-conditioned diffusion generation can be done within 20 seconds (time varies due to the adaptive-step ODE solver used in flow-mathcing.) Upload an image of an object or click on one of the provided examples to see how the GaussianAnything works.
The 3D viewer will render a .glb point cloud exported from the centers of the surfel Gaussians, and an integrated TSDF mesh. Besides, you can find the intermediate stage-1 point cloud in the Tab (Stage-1 Output).
For best results run the demo locally and render locally - to do so, clone the main repository.
Input Image
Processed Image
Stage-2 Output
Stage-1 Output
Rendered Video (2 LoDs)
CFG-scale
4
Seed
42
Remove Background
2DGS Center


like 613
Running
App
Files
Community 3
Chatbot
Sure, I'd be happy to help you plan a one-day trip! Let's see, there are so many cities to choose from, but I think I'll pick Tokyo, Japan. It's a fascinating city with a unique blend of traditional and modern culture. Even though it's a big city, it's surprisingly easy to navigate, especially for tourists.
First things first, let's figure out how to get around. Tokyo has an extensive public transportation system, including subways, buses, and trains. The best way to tackle this is to get a prepaid transportation card like a Suica or Pasmo card. These cards can be used on all forms of public transport and make traveling much smoother. You can purchase them at any train station; just look for the ticket machines that have the Suica or Pasmo logo.
Now, for the itinerary. Since it's a one-day trip, we need to make the most of our time. Let's assume we're starting in the morning and have about 10 hours to explore.
Morning:
We'll begin our day in the historic district of Asakusa. Asakusa is known for its traditional atmosphere and is home to the famous Senso-ji Temple, which is Tokyo's oldest temple. It's a great place to experience a bit of old Tokyo.
Error
Activity: Visit Senso-ji Temple. Make sure to walk through the Nakamise Shopping Street leading up to the temple. It's packed with souvenir shops and food stalls, so you can grab a quick breakfast there if you like. Try some traditional Japanese snacks like taiyaki (fish-shaped cake filled with red bean paste) or kinako-dango (rice flour dumplings coated in soybean flour).
Interesting Trivia: Did you know that Senso-ji Temple was established in the year 645? That makes it over 1,300 years old! It's a significant historical site and a must-see when in Tokyo.
After visiting the temple, let's head over to the nearby Asakusa Shrine. It's a beautiful Shinto shrine with a peaceful atmosphere, perfect for taking a break and soaking in the surroundings.
Late Morning to Early Afternoon:
QwQ-32B-preview
QwQ-32B-preview is an experimental research model developed by the Qwen Team, focused on advancing AI reasoning capabilities. As a preview release, it demonstrates promising analytical abilities while having several important limitations such as code switching and recursive reasoning loops. Only single-turn queries are supported in this demo.
推荐对话
How many r in strawberry
Find the least odd prime factor of 2019^8+1.
S先生、P先生、Q先生他们知道桌子的抽屉里有16张扑克牌：红桃A、Q、4 黑桃J、8、4、2、7、3 草花K、Q、5、4、6 方块A、5。约翰教授从这16张牌中挑出一张牌来，并把这张牌的点数告诉P先生，把这张牌的花色告诉Q先生。这时，约翰教授问P先生和Q 先生：你们能从已知的点数或花色中推知这张牌是什么牌吗？于是，S先生听到如下的对话： P先生：我不知道这张牌。 Q先生：我知道你不知道这张牌。 P先生：现在我知道这张牌了。 Q先生：我也知道了。 请问：这张牌是什么牌？

Our Open Source

We are building the foundation of ML tooling with the community.

Transformers
109,705

State-of-the-art ML for Pytorch, TensorFlow, and JAX.

Diffusers
17,025

State-of-the-art diffusion models for image and audio generation in PyTorch.

Safetensors
1,323

Simple, safe way to store and distribute neural networks weights safely and quickly.

Hub Python Library
1,069

Client library for the HF Hub: manage repositories from your Python runtime.

Tokenizers
7,347

Fast tokenizers, optimized for both research and production.

PEFT
8,628

Parameter efficient finetuning methods for large models

Transformers.js
3,270

Community library to run pretrained models from Transformers in your browser.

timm
26,274

State-of-the-art computer vision mode

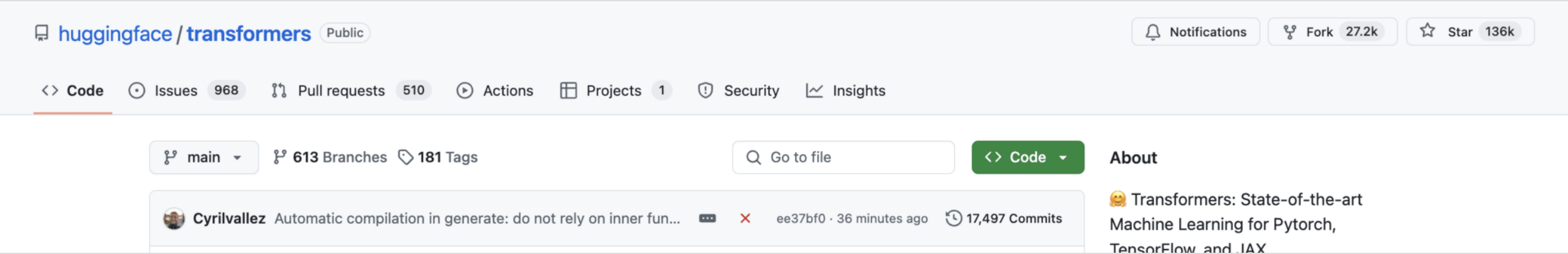
In-browser AI apps with Gradio and Transformers

Transformers

Transformers provides APIs and tools

to easily download and train state-of-the-art pretrained models.

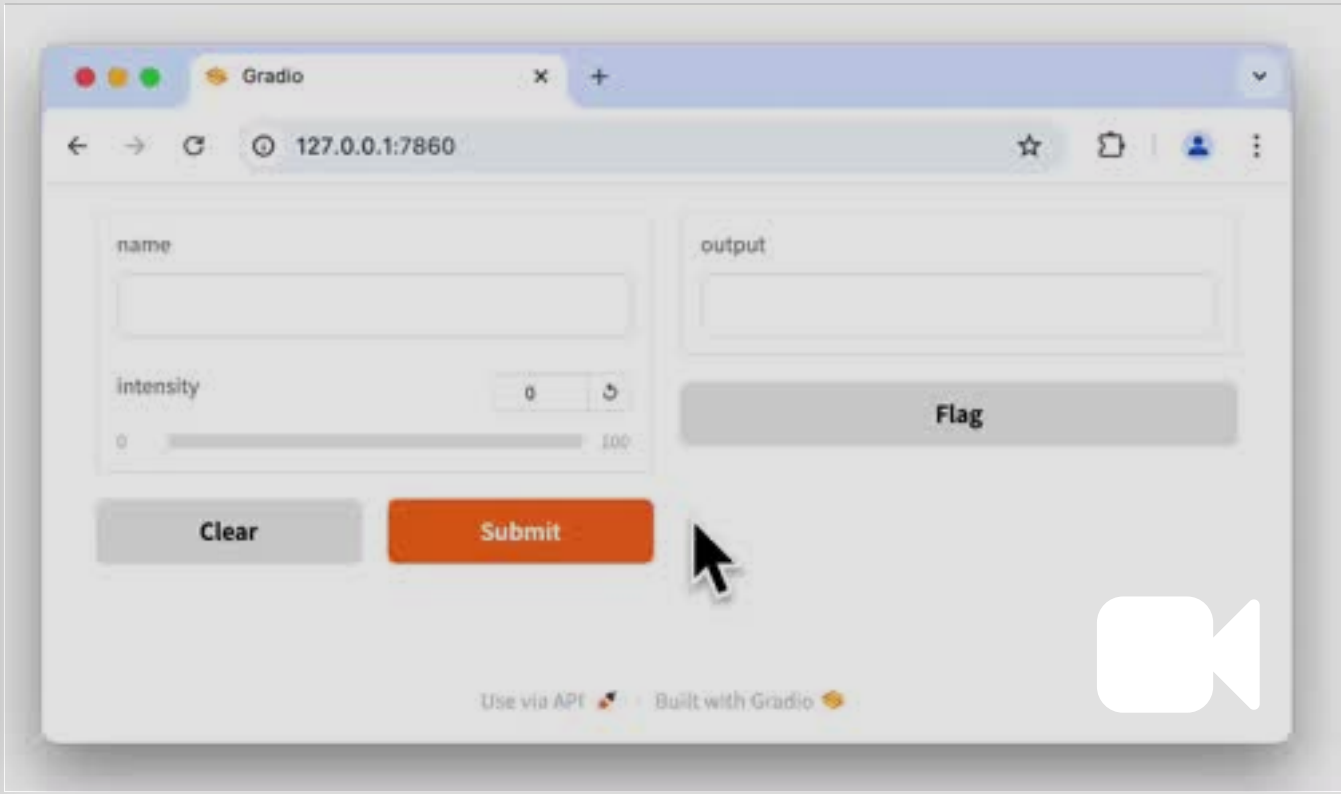
```
>>> from transformers import pipeline
>>> classifier = pipeline("sentiment-analysis")
>>> classifier("We are very happy to show you the 🤗 Transformers library.")
[{'label': 'POSITIVE', 'score': 0.9998}]
```



gradio

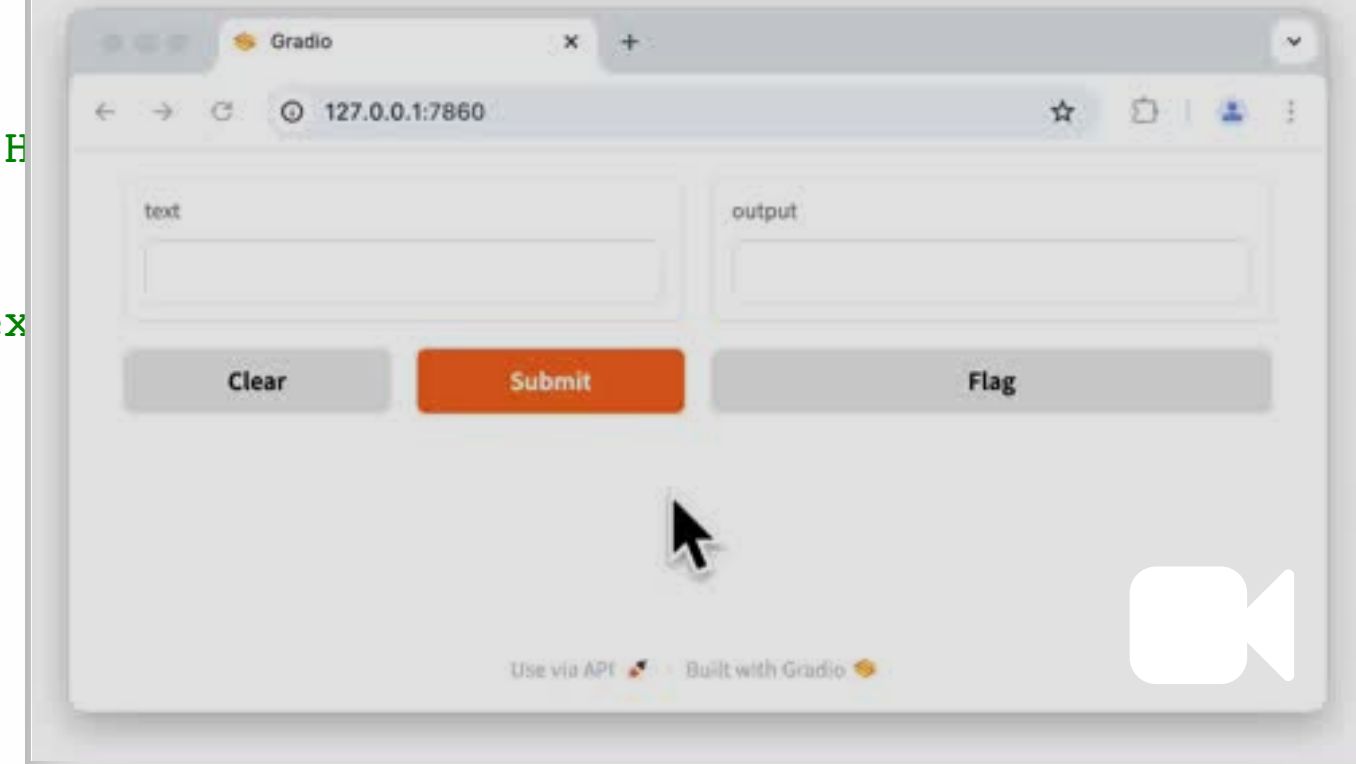
Gradio is an open-source Python package that allows you to quickly build a demo or web application for your machine learning model, API, or any arbitrary Python function.

```python
import gradio as gr

def greet(name, intensity):
    return "Hello, " + name + "!" * int(i

demo = gr.Interface(
    fn=greet,
    inputs=["text", "slider"],
    outputs=["text"],
)

demo.launch()
```



gradio 🤝 Transformers

```python
import gradio as gr
from transformers import pipeline

pipe = pipeline("translation", model="H

def predict(text):
  return pipe(text)[0]["translation_tex

demo = gr.Interface(
  fn=predict,
  inputs='text',
  outputs='text',
)

demo.launch()
```



gradio 🤝 Transformers

```python
import gradio as gr
from transformers import pipeline

pipe = pipeline("translati", model=

def predict(text):
  return pipe(text)[0]["translation_te

demo = gr.Interface(
  fn=predict,
  inputs='text',
  outputs='text',
)

demo.launch()
```

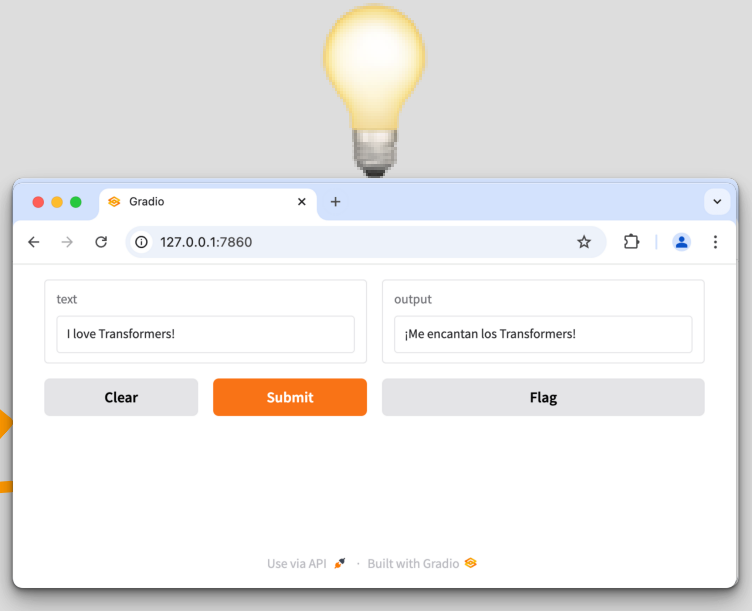
AI inference

Transformers

Web server

gradio

Python runtime



Server


gradio

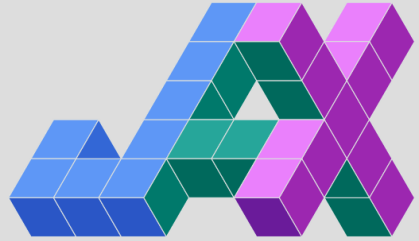

Transformers

Quick & Easy Web-based AI apps

Python

at FEDAY?

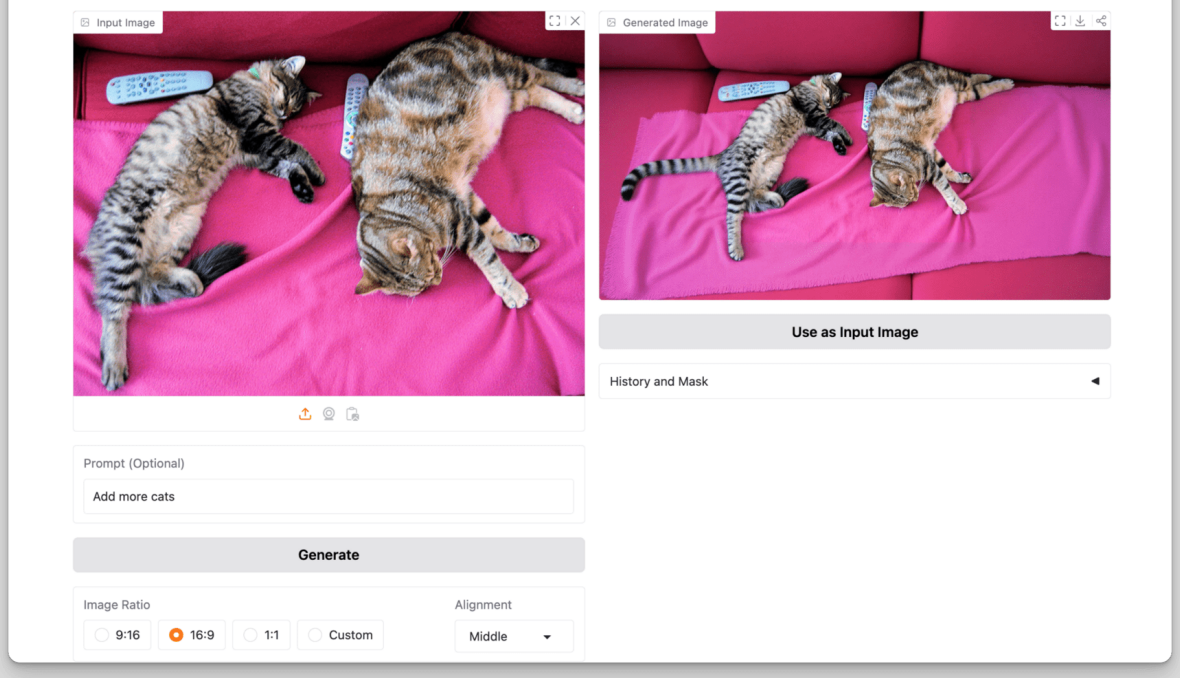


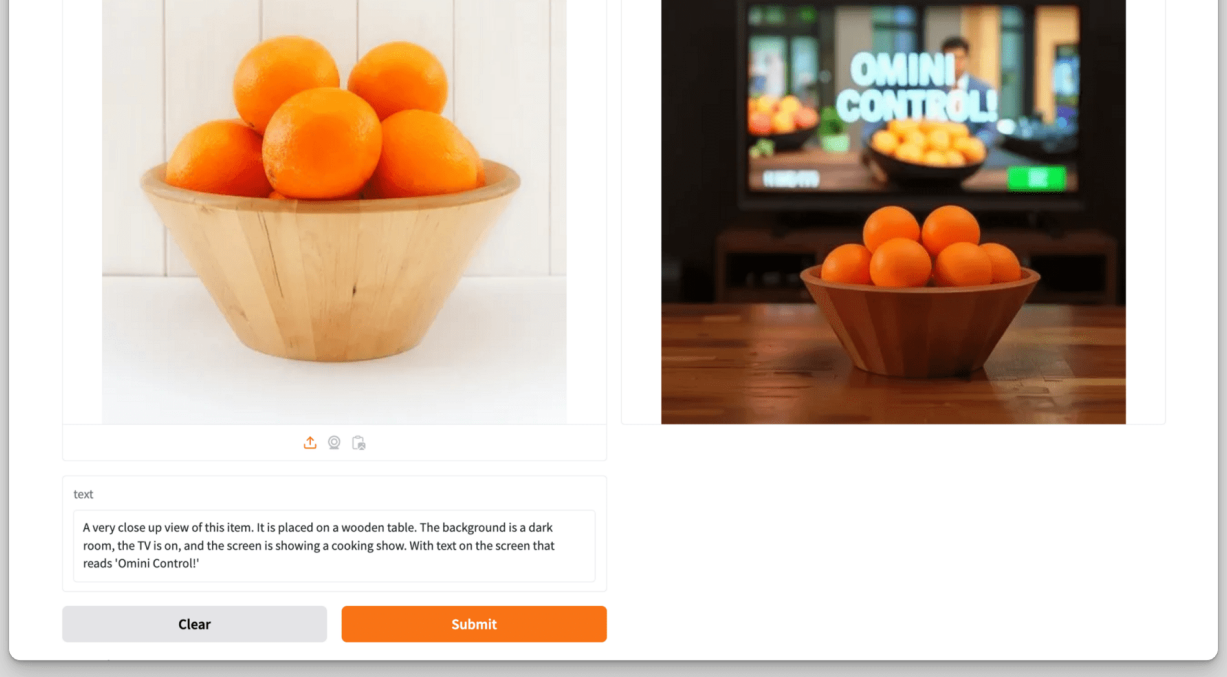



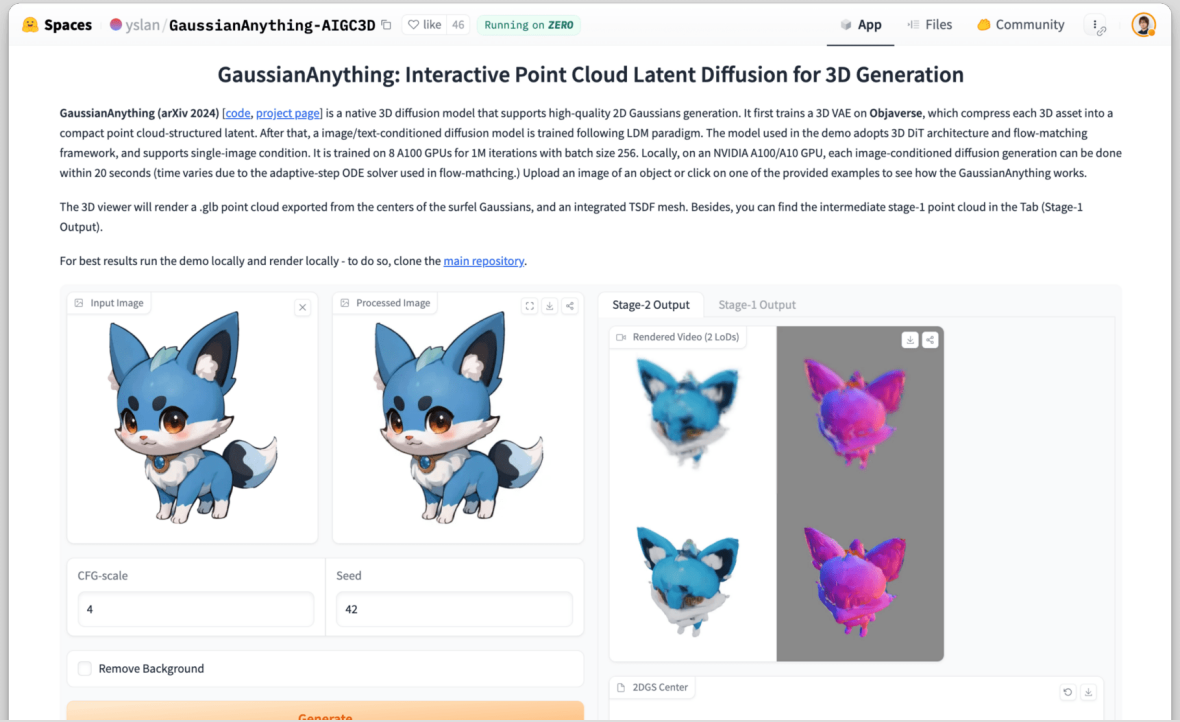

A central part of the AI/ML ecosystem.

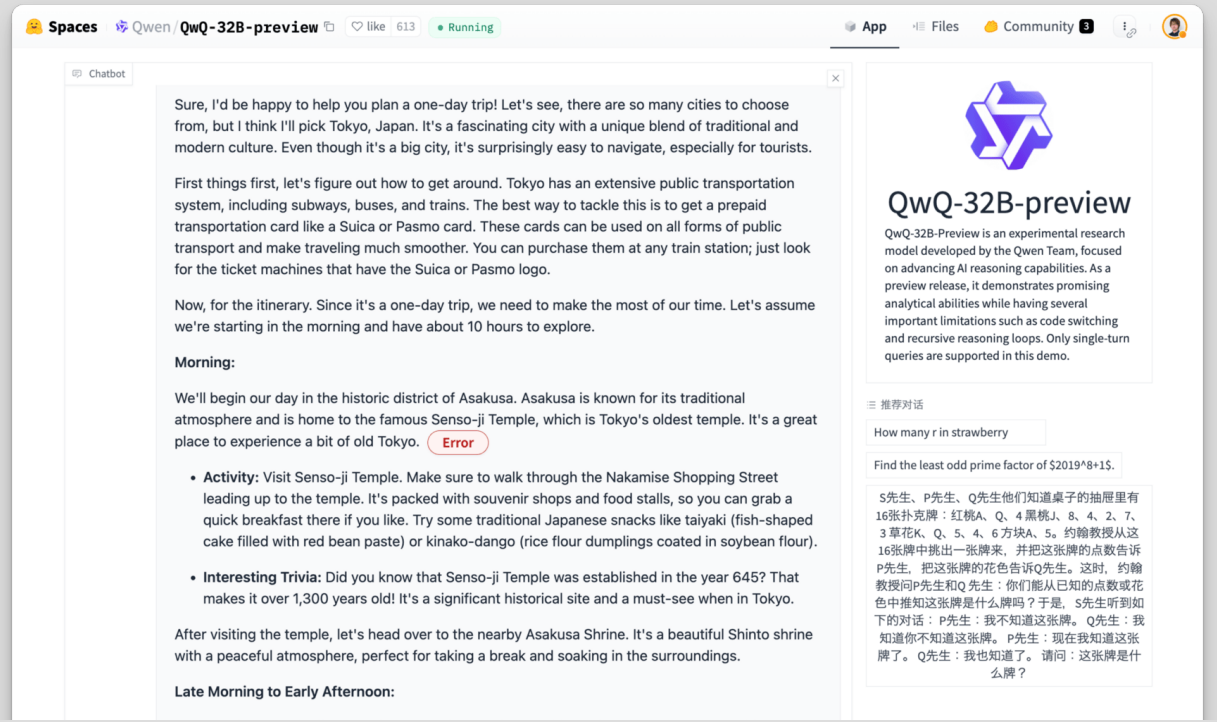






Input Image
Generated Image
Use as Input Image
History and Mask
Prompt (Optional)
Add more cats
Generate
Image Ratio
9:16
16:9
1:1
Custom
Alignment
Middle


text
A very close up view of this item. It is placed on a wooden table. The background is a dark room, the TV is on, and the screen is showing a cooking show. With text on the screen that reads 'Omini Control!'
Clear
Submit
OMINI CONTROL!


Spaces
yslan / GaussianAnything-AIGC3D
like 46
Running on ZERO
App
Files
Community
GaussianAnything: Interactive Point Cloud Latent Diffusion for 3D Generation
GaussianAnything (arXiv 2024) [code, project page] is a native 3D diffusion model that supports high-quality 2D Gaussians generation. It first trains a 3D VAE on Objaverse, which compress each 3D asset into a compact point cloud-structured latent. After that, a image/text-conditioned diffusion model is trained following LDM paradigm. The model used in the demo adopts 3D DiT architecture and flow-matching framework, and supports single-image condition. It is trained on 8 A100 GPUs for 1M iterations with batch size 256. Locally, on an NVIDIA A100/A10 GPU, each image-conditioned diffusion generation can be done within 20 seconds (time varies due to the adaptive-step ODE solver used in flow-mathcing.) Upload an image of an object or click on one of the provided examples to see how the GaussianAnything works.
The 3D viewer will render a .glb point cloud exported from the centers of the surfel Gaussians, and an integrated TSDF mesh. Besides, you can find the intermediate stage-1 point cloud in the Tab (Stage-1 Output).
For best results run the demo locally and render locally - to do so, clone the main repository.
Input Image
Processed Image
Stage-2 Output
Stage-1 Output
Rendered Video (2 LoDs)
CFG-scale
4
Seed
42
Remove Background
2DGS Center


Spaces
Qwen / QwQ-32B-preview
like 613
Running
App
Files
Community 3
Chatbot
Sure, I'd be happy to help you plan a one-day trip! Let's see, there are so many cities to choose from, but I think I'll pick Tokyo, Japan. It's a fascinating city with a unique blend of traditional and modern culture. Even though it's a big city, it's surprisingly easy to navigate, especially for tourists.
First things first, let's figure out how to get around. Tokyo has an extensive public transportation system, including subways, buses, and trains. The best way to tackle this is to get a prepaid transportation card like a Suica or Pasmo card. These cards can be used on all forms of public transport and make traveling much smoother. You can purchase them at any train station; just look for the ticket machines that have the Suica or Pasmo logo.
Now, for the itinerary. Since it's a one-day trip, we need to make the most of our time. Let's assume we're starting in the morning and have about 10 hours to explore.
Morning:
We'll begin our day in the historic district of Asakusa. Asakusa is known for its traditional atmosphere and is home to the famous Senso-ji Temple, which is Tokyo's oldest temple. It's a great place to experience a bit of old Tokyo.
Error
Activity: Visit Senso-ji Temple. Make sure to walk through the Nakamise Shopping Street leading up to the temple. It's packed with souvenir shops and food stalls, so you can grab a quick breakfast there if you like. Try some traditional Japanese snacks like taiyaki (fish-shaped cake filled with red bean paste) or kinako-dango (rice flour dumplings coated in soybean flour).
Interesting Trivia: Did you know that Senso-ji Temple was established in the year 645? That makes it over 1,300 years old! It's a significant historical site and a must-see when in Tokyo.
After visiting the temple, let's head over to the nearby Asakusa Shrine. It's a beautiful Shinto shrine with a peaceful atmosphere, perfect for taking a break and soaking in the surroundings.
Late Morning to Early Afternoon:
QwQ-32B-preview
QwQ-32B-Preview is an experimental research model developed by the Qwen Team, focused on advancing AI reasoning capabilities. As a preview release, it demonstrates promising analytical abilities while having several important limitations such as code switching and recursive reasoning loops. Only single-turn queries are supported in this demo.
推荐对话
How many r in strawberry
Find the least odd prime factor of 2019^8+1.
S先生、P先生、Q先生他们知道桌子的抽屉里有16张扑克牌：红桃A、Q、4 黑桃J、8、4、2、7、3 草花K、Q、5、4、6 方块A、5。约翰教授从这16张牌中挑出一张牌来，并把这张牌的点数告诉P先生，把这张牌的花色告诉Q先生。这时，约翰教授问P先生和Q 先生：你们能从已知的点数或花色中推知这张牌是什么牌吗？于是，S先生听到如下的对话： P先生：我不知道这张牌。 Q先生：我知道你不知道这张牌。 P先生：现在我知道这张牌了。 Q先生：我也知道了。 请问：这张牌是什么牌？

In-browser AI apps with Gradio and Transformers

In-browser AI apps

a.k.a. frontend-only AI apps

Web-based applications that use machine learning models
to perform tasks like text analysis, image processing, or speech recognition
entirely within the user's browser.

Why in-browser?

- Privacy
- Low latency
- Offline capability
- Scalability without servers
- Low cost

In-browser AI apps with Gradio and Transformers

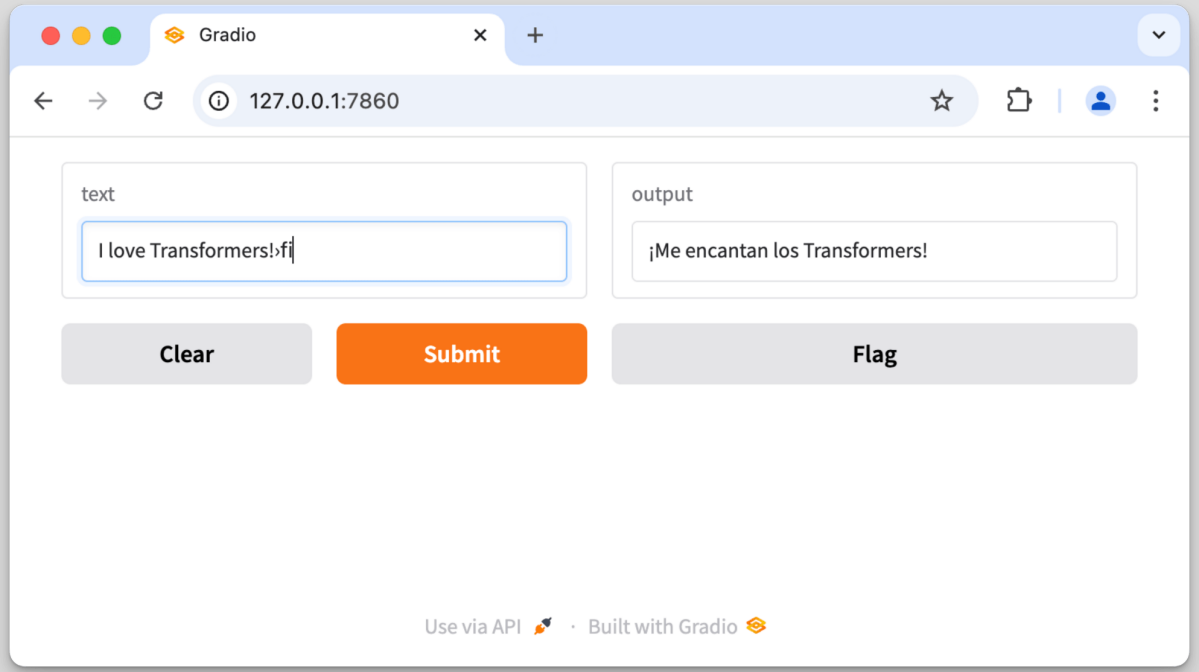

gradio


Transformers

Quick & Easy Web-based AI apps

gradio 🤝 Transformers

```python
import gradio as gr
from transformers import pipeline

pipe = pipeli e(      slation", model="H

def predic
  return      ranslation_tex

demo = gr.Inte      ce(
  fn=predict,
  inputs='text',
  outputs='text',
)

demo.launch()
```



… Python on Frontend?

Python interpreter on a browser

Pyodide is a Python distribution for the browser and Node.js based on WebAssembly.

```
1 import sys
2
3 print(sys.platform)
```

"emscripten"

PYODIDE

WA

Pyodide = Wasm-compiled Pyt

Wasm Runtime

JS engine

Web Browser

pyodide.org/en/stable/consol

pyodide.org/en/stable/console.html

```
Welcome to the Pyodide 0.26.4 terminal emulator 🐍
Python 3.12.1 (main, Nov 15 2024 14:17:00) on WebAssembly/Emscripten
Type "help", "copyright", "credits" or "license" for more information.
>>> message = "Hello Pyodide"
>>> print(message)
Hello Pyodide
>>> import sys
>>> print(sys.platform)
emscripten
>>>
```

jQuery Terminal

stable

gradio 🤝 Transformers

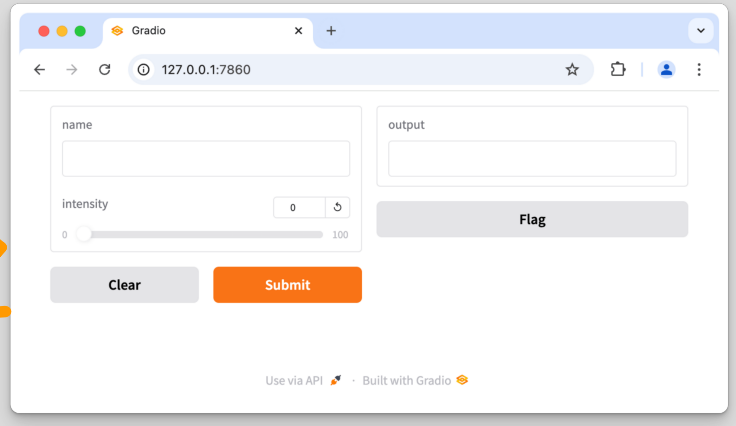
```python
import gradio as gr
from transformers import pipeline

pipe = pipeline("translation", model=

def predict(text):
  return pipe(text)[0]["translation_t

demo = gr.Interface(
  fn=predict,
  inputs='text',
  outputs='text',
)

demo.launch()
```

AI inference

Transformers

Web server

gradio



Python runtime

Server

gradio-lite 🤝 Transformers.JS

```python
import gradio as gr
from transformers_js_py import pipeli

pipe = await pipeline("translation",

async def predict(text):
  res = await pipe(text, {
    "src_lang": 'en',
    "tgt_lang": 'zh',
  })
  return res[0]["translation_text"]

demo = gr.Interface(
  fn=predict,
  inputs='text',
  outputs='text',
)

demo.launch()
```

PYODIDE WA runtime

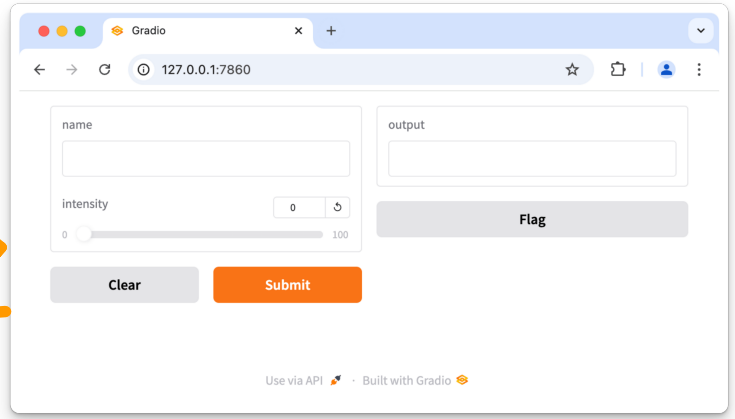
AI inference

Transformers.JS

JS-Py bridge

Pseudo-Web server

gradio-lite



Worker process

Renderer process

Web Browser

Gradio-Lite

Pyodide/Wasm-ported Gradio.

You write Python code, then get a web UI, 100% in the browser.

Transformers.js

JS version of Transformers.

You can use pretrained AI/ML models in the browser.

Check out the presentation 👉 https://www.bilibili.com/video/BV19c411B7QU/

Transformers.js

👉 https://www.bilibili.com/video/BV19c411B7QU/

gradio-lite 🤝 Transformers.JS

```python
import gradio as gr
from transformers_js_py import pipeli

pipe = await pipeline("translation",

async def predict(text):
  res = await pipe(text, {
    "src_lang": 'en',
    "tgt_lang": 'zh',
  })
  return res[0]["translation_text"]

demo = gr.Interface(
  fn=predict,
  inputs='text',
  outputs='text',
)

demo.launch()
```

PYODIDE WA runtime

AI inference

Transformers.JS

JS-Py bridge

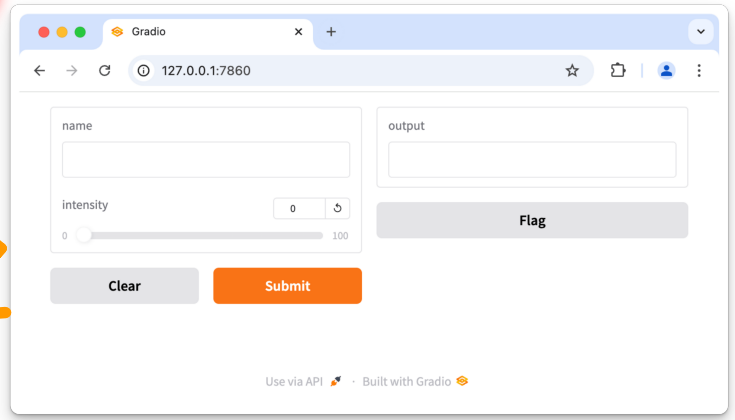
Pseudo-Web server

gradio-lite

Worker process

`Transformers.js.py`

Python wrapper of Transformers.js 🤯



Renderer process

Web Browser

Gradio-Lite

Pyodide/Wasm-ported Gradio.

You write Python code, then get a web UI, 100% in the browser.

Transformers.js

JS version of Transformers.

You can use pretrained AI/ML models in the browser.

Check out the presentation 👉 https://www.bilibili.com/video/BV19c411B7QU/

Transformers.js.py

Use Transformers.js from Pyodide.

gradio-lite 🤝 Transformers.JS

```python
import gradio as gr
from transformers_js_py import pipeli

pipe = await pipeline("translation",

async def predict(text):
  res = await pipe(text, {
    "src_lang": 'en',
    "tgt_lang": 'zh',
  })
  return res[0]["translation_text"]

demo = gr.Interface(
  fn=predict,
  inputs='text',
  outputs='text',
)

demo.launch()
```

PYODIDE WA runtime

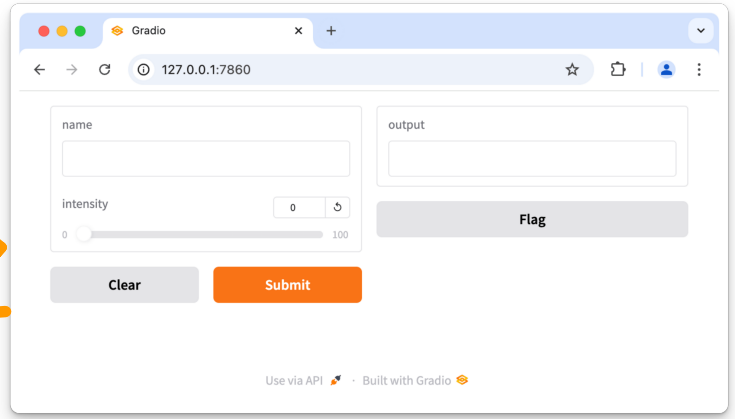
AI inference

Transformers.JS

JS-Py bridge

Pseudo-Web server

gradio-lite



Worker process

Renderer process

Web Browser

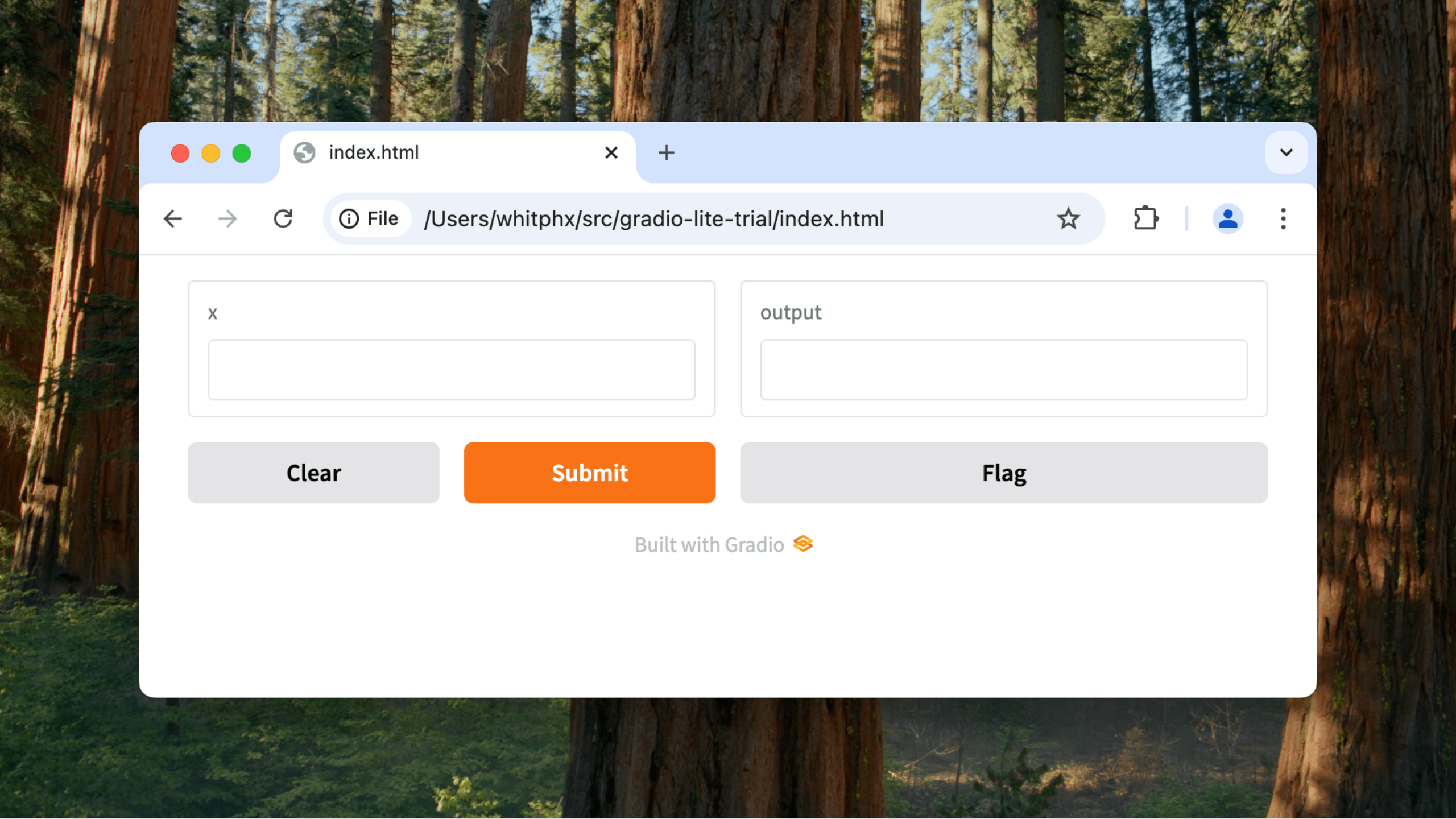
How to use them

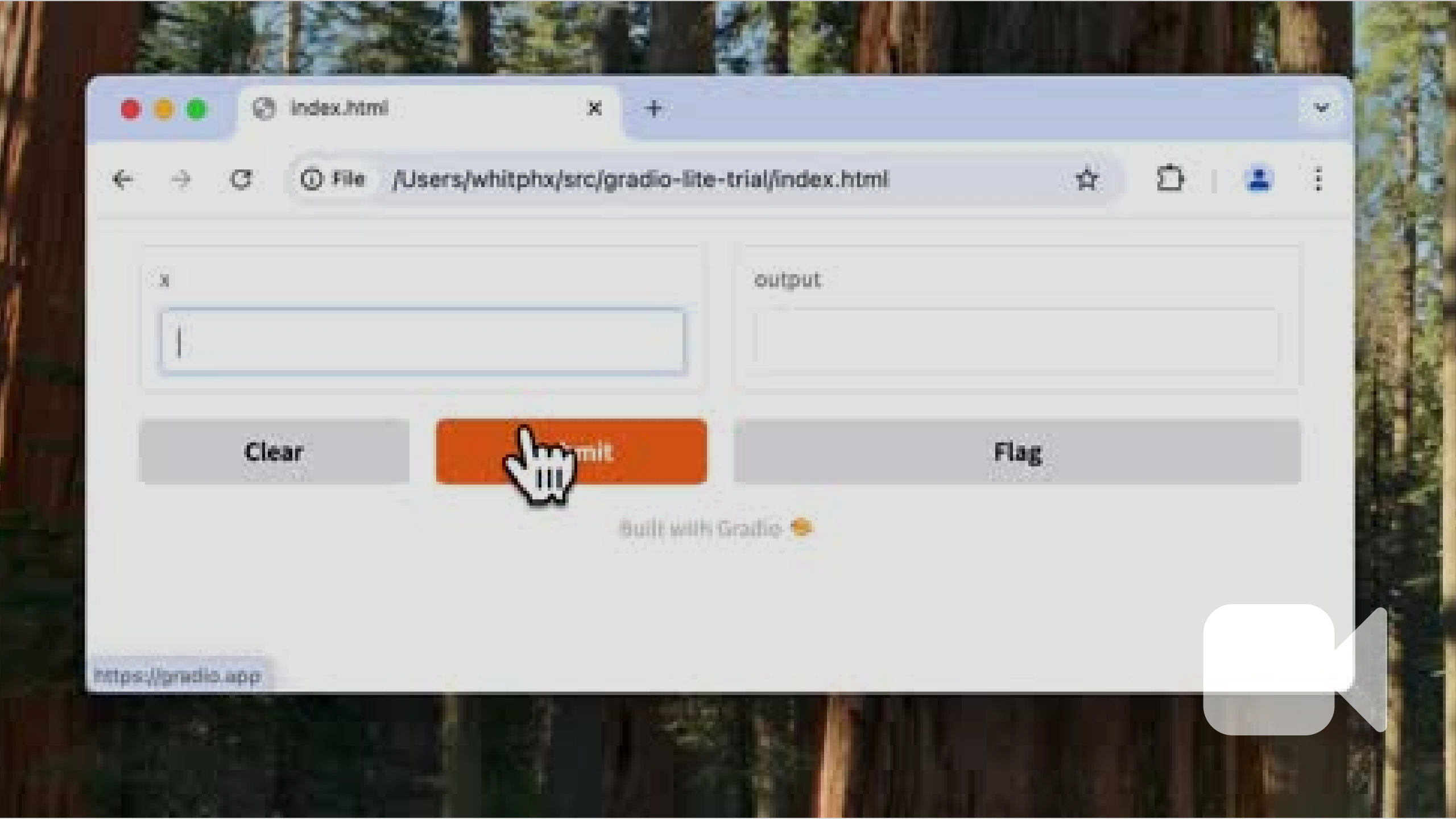
```
1 $ code index.html
```

```
1  <html>
2      <head>
3          <script type="module" crossorigin src="https://cdn.j
4          <link rel="stylesheet" href="https://cdn.jsdelivr.ne
5      </head>
6      <body>
7
8
9      </body>
10 </html>
```

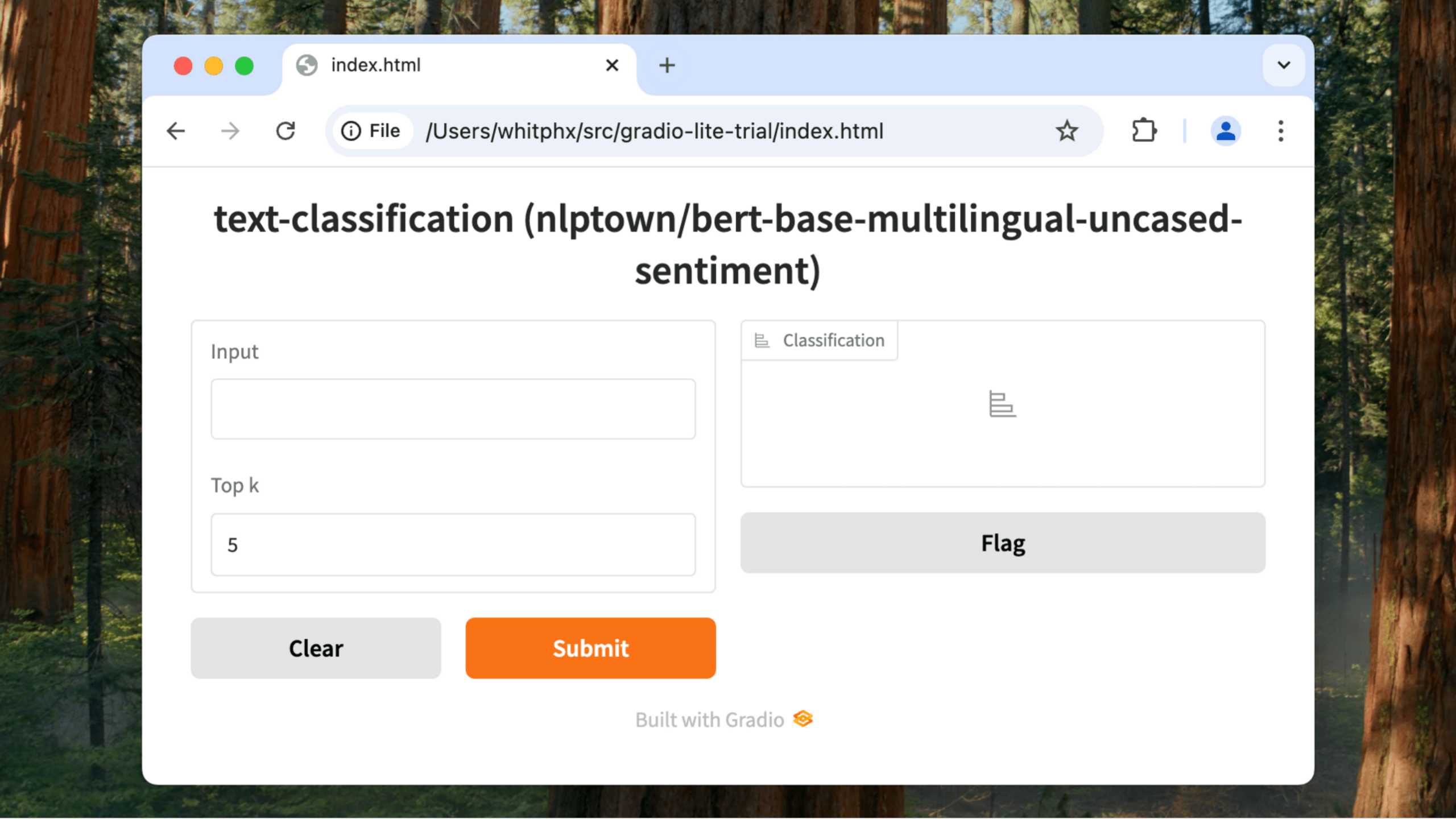
```
<html>
    <head>
        <script type="module" crossorigin src="https://cdn.j
        <link rel="stylesheet" href="https://cdn.jsdelivr.ne
    </head>
    <body>
        <gradio-lite>

        </gradio-lite>
    </body>
</html>
```

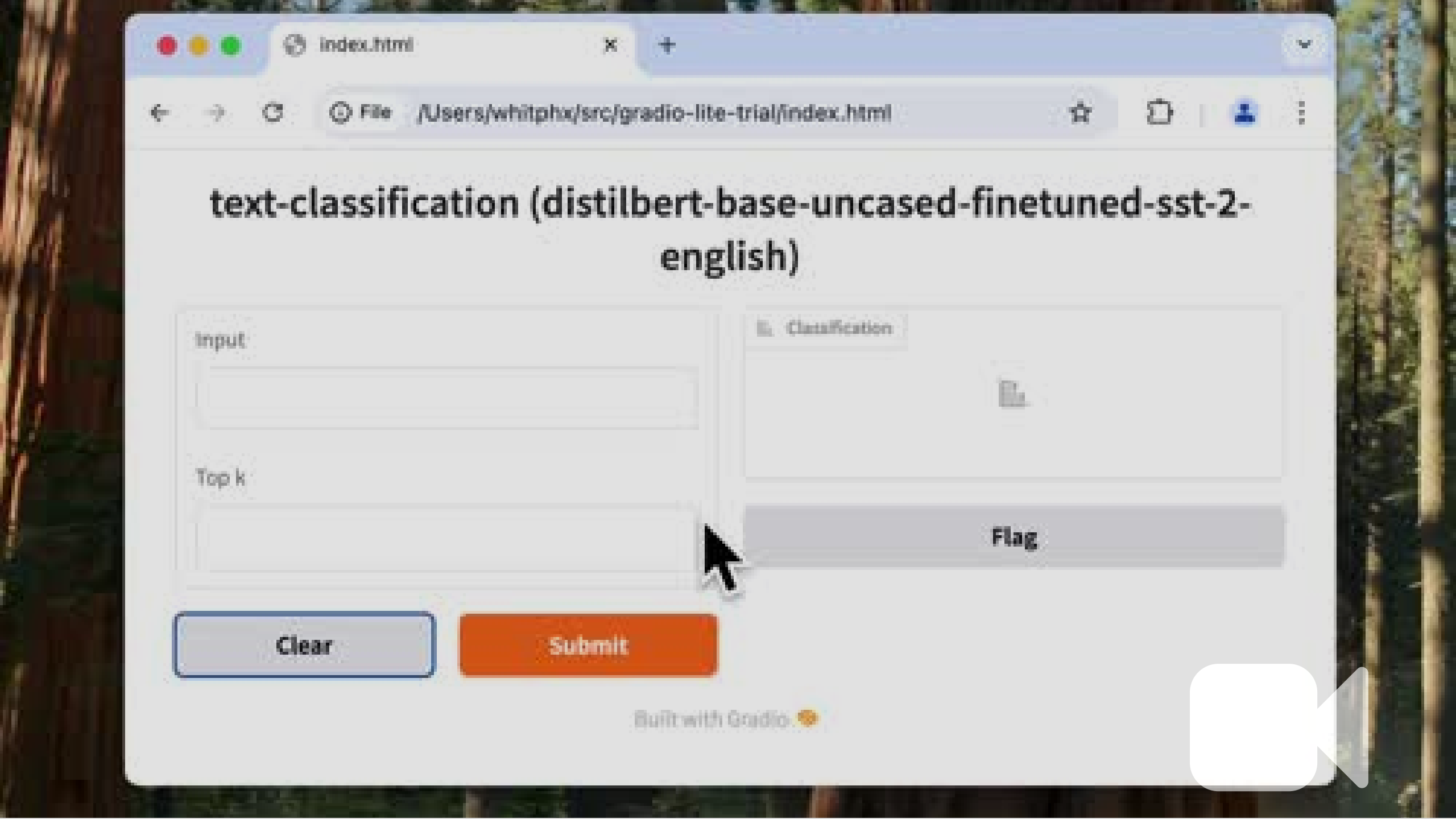
```
<html>
    <head>
        <script type="module" crossorigin src="https://cdn.j
        <link rel="stylesheet" href="https://cdn.jsdelivr.ne
    </head>
    <body>
        <gradio-lite>
import gradio as gr

def greet(name):
    return "Hello, " + name + "!"

gr.Interface(greet, "textbox", "textbox").launch()
        </gradio-lite>
    </body>
</html>
```


index.html
File
/Users/whitphx/src/gradio-lite-trial/index.html
x
output
Clear
Submit
Flag
Built with Gradio


index.html
File
/Users/whitphx/src/gradio-lite-trial/index.html
x
output
Clear
Flag
Built with Gradio
https://gradio.app

```
    <head>
        <script type="module" crossorigin src="https://cdn.j
        <link rel="stylesheet" href="https://cdn.jsdelivr.ne
    </head>
    <body>
        <gradio-lite>
import gradio as gr
from transformers_js_py import pipeline

pipe = await pipeline('sentiment-analysis')

demo = gr.Interface.from_pipeline(pipe)

demo.launch()

            <gradio-requirements>
transformers-js-py
            </gradio-requirements>
```


index.html
File
/Users/whitphx/src/gradio-lite-trial/index.html
text-classification (nlptown/bert-base-multilingual-uncased-sentiment)
Input
Top k
5
Classification
Flag
Clear
Submit
Built with Gradio


index.html
File /Users/whitphx/src/gradio-lite-trial/index.html
text-classification (distilbert-base-uncased-finetuned-sst-2-english)
Input
Top k
Classification
Flag
Clear
Submit
Built with Gradio

```
    <head>
        <script type="module" crossorigin src="https://cdn.j
        <link rel="stylesheet" href="https://cdn.jsdelivr.ne
    </head>
    <body>
        <gradio-lite>
import gradio as gr
from transformers_js_py import pipeline

pipe = await pipeline('sentiment-analysis')

demo = gr.Interface.from_pipeline(pipe)

demo.launch()

            <gradio-requirements>
transformers-js-py
            </gradio-requirements>
```

```
    <head>
        <script type="module" crossorigin src="https://cdn.j
        <link rel="stylesheet" href="https://cdn.jsdelivr.ne
    </head>
    <body>
        <gradio-lite>
import gradio as gr
from transformers_js_py import pipeline

pipe = await pipeline(
    'sentiment-analysis',
    'Xenova/bert-base-multilingual-uncased-sentiment'
)

demo = gr.Interface.from_pipeline(pipe)


demo.launch()
```

huggingface.co/Xenova/bert-base-multilingual-uncased-sentiment

Hugging Face | Searc | Models | Datasets | Spaces | Docs | Enterprise | Pricing | Log In | Sign Up

Xenova / bert-base-multilingual-uncased-sentiment

like 2

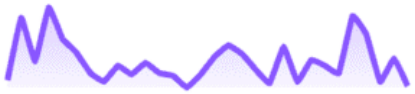
Text Classification | Transformers.js | ONNX | bert

Model card | Files | Community | Use this model

https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment with ONNX weights to be compatible with Transformers.js.

Note: Having a separate repo for ONNX weights is intended to be a temporary solution until WebML gains more traction. If you would like to make your models web-ready, we recommend converting to ONNX using 🤗 Optimum and structuring your repo like this one (with ONNX weights located in a subfolder named `onnx`).

Downloads last month
581



Inference Examples

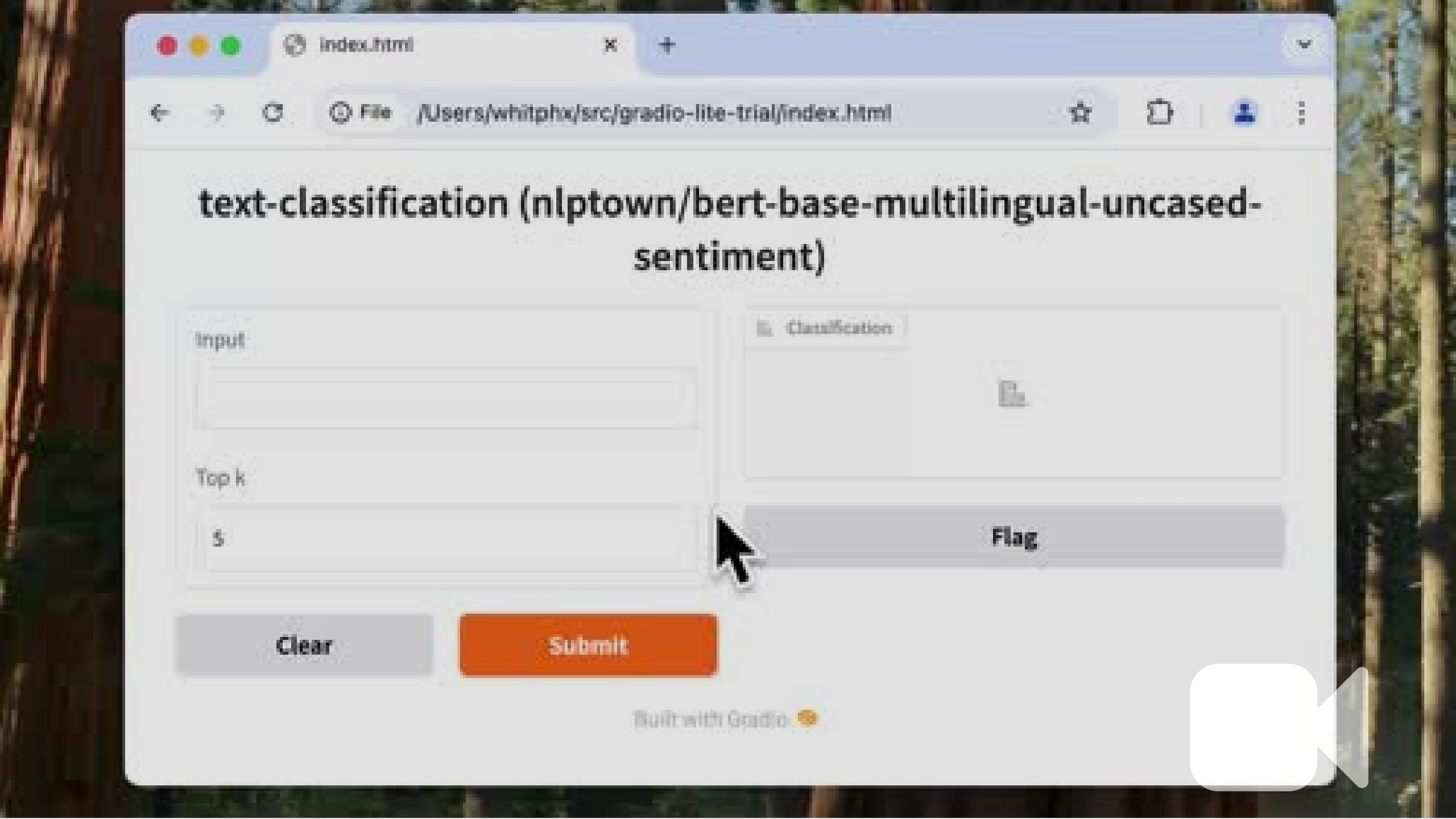
Text Classification

Inference API (serverless) does not yet support transformers.js models for this pipeline type.

Model tree for `Xenova/bert-base-multilingual-...`

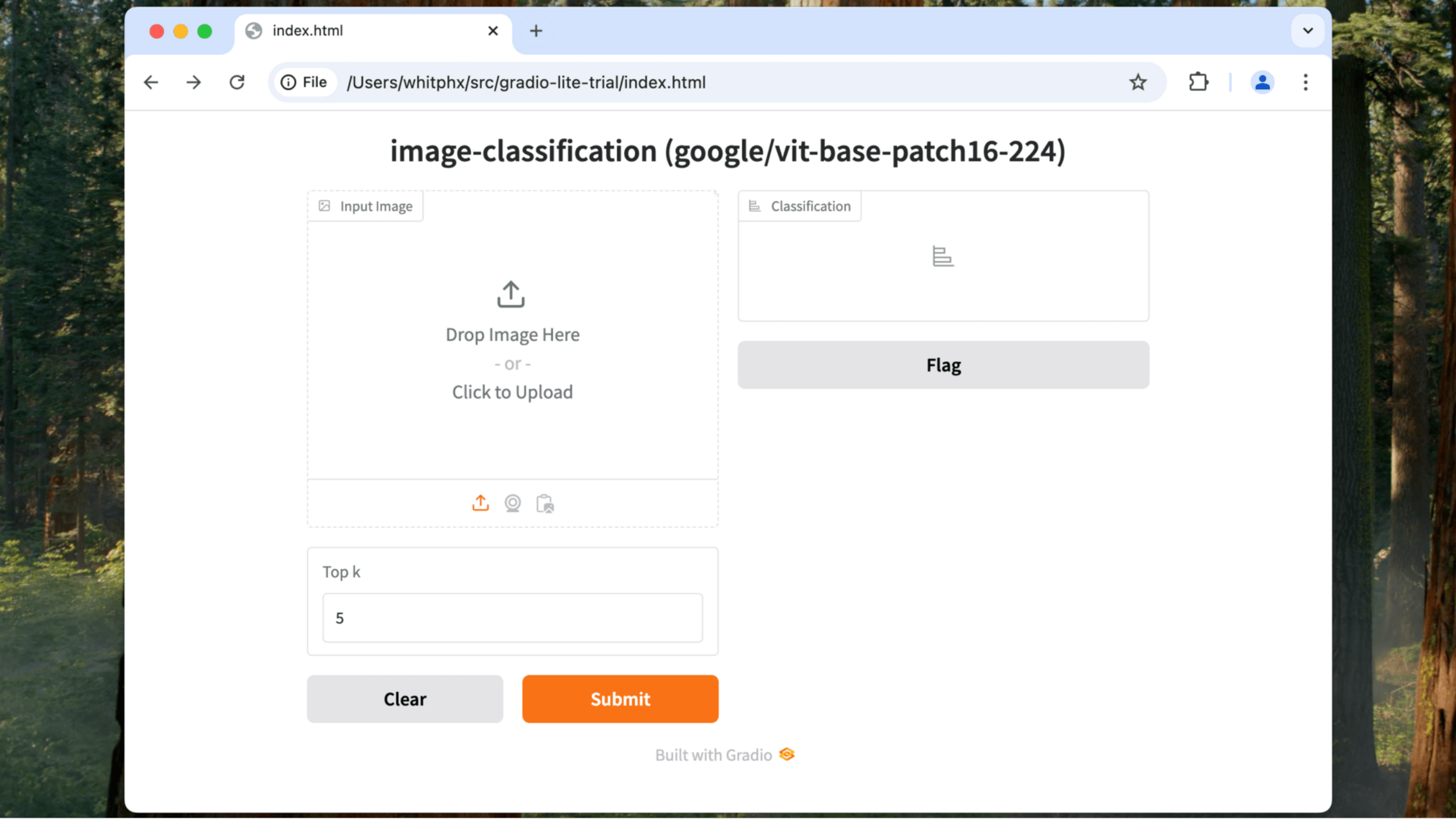
Base model `nlptown/bert-base-multilingual-uncas...`

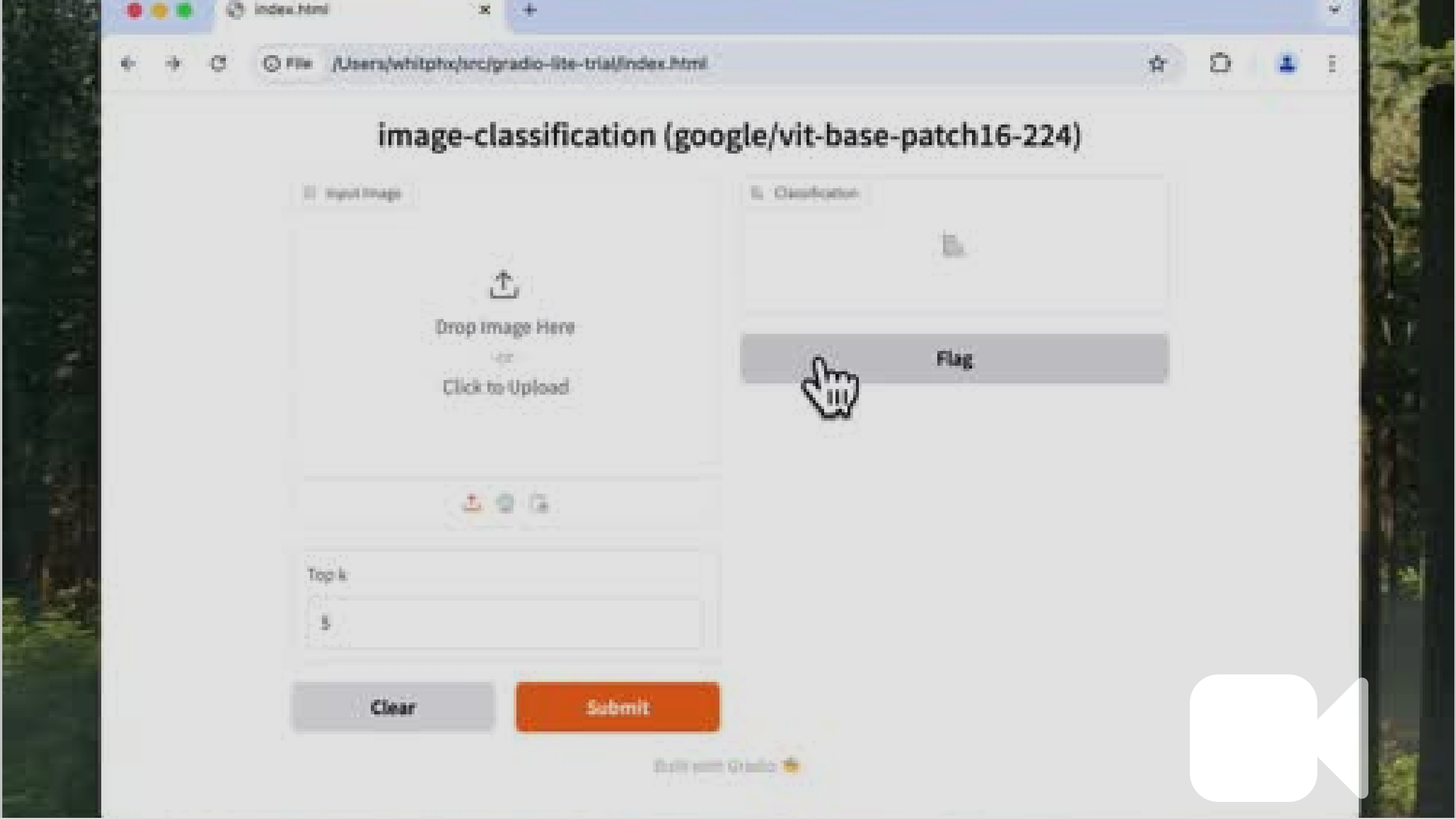
Quantized (1) this model


index.html
File /Users/whitphx/src/gradio-lite-trial/index.html
text-classification (nlptown/bert-base-multilingual-uncased-sentiment)
Input
Classification
Top k
5
Flag
Clear
Submit
Built with Gradio

```
    <head>
        <script type="module" crossorigin src="https://cdn.j
        <link rel="stylesheet" href="https://cdn.jsdelivr.ne
    </head>
    <body>
        <gradio-lite>
import gradio as gr
from transformers_js_py import pipeline

pipe = await pipeline(
    'sentiment-analysis',
    'Xenova/bert-base-multilingual-uncased-sentiment'
)

demo = gr.Interface.from_pipeline(pipe)


demo.launch()
```

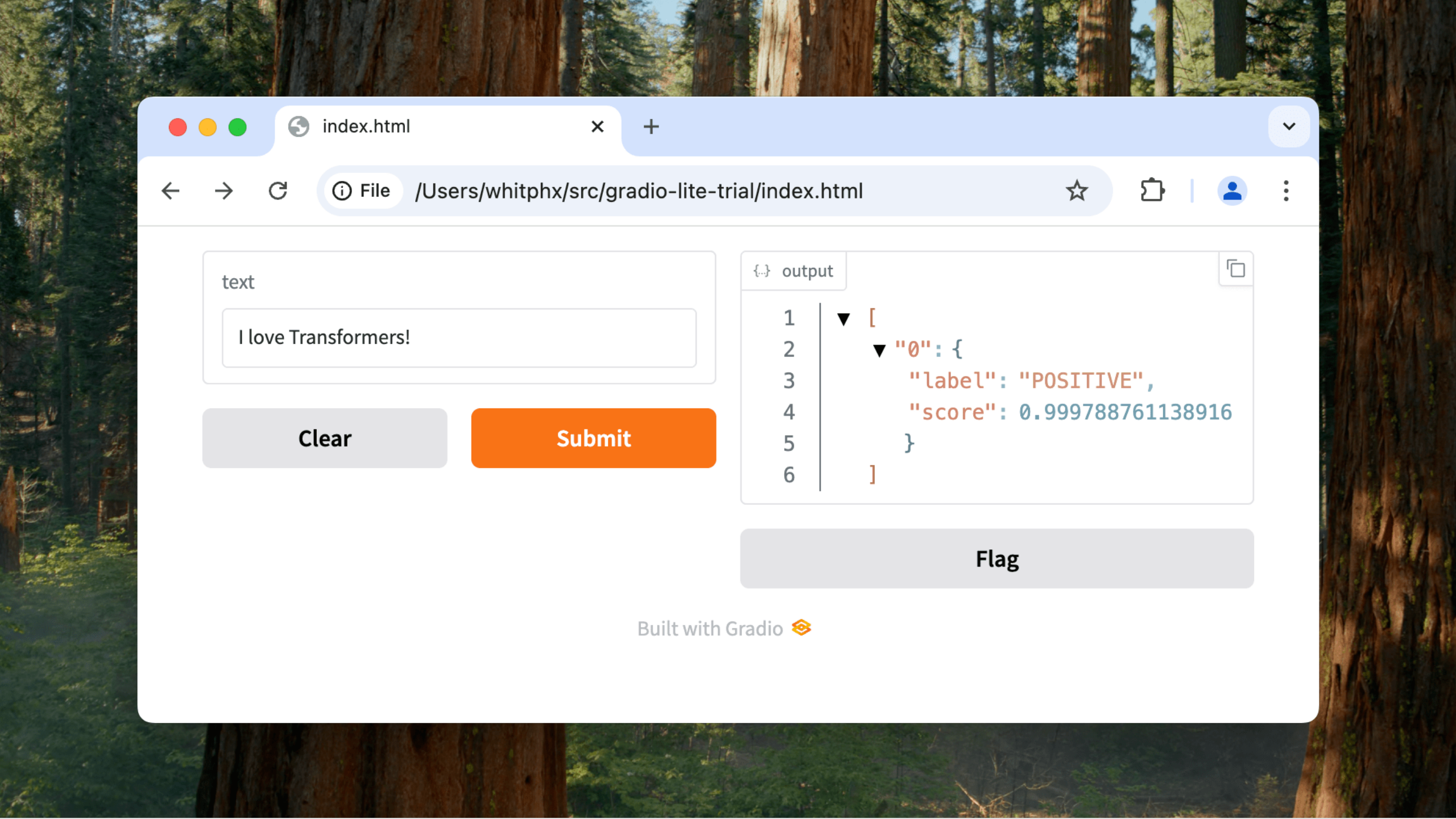
```
    <head>
        <script type="module" crossorigin src="https://cdn.j
        <link rel="stylesheet" href="https://cdn.jsdelivr.ne
    </head>
    <body>
        <gradio-lite>
import gradio as gr
from transformers_js_py import pipeline

pipe = await pipeline('image-classification')

demo = gr.Interface.from_pipeline(pipe)

demo.launch()

            <gradio-requirements>
transformers-js-py
            </gradio-requirements>
```

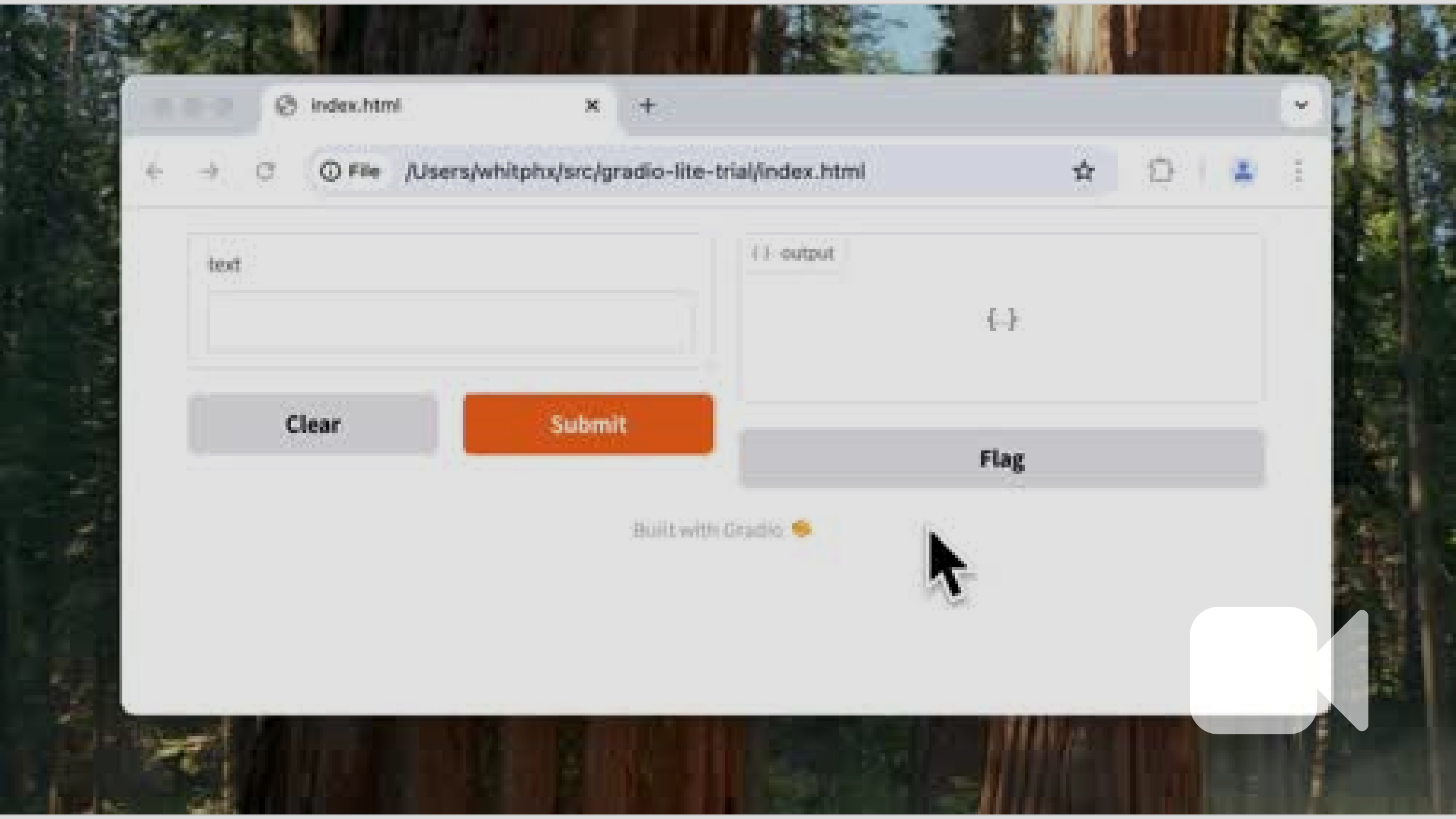

index.html
File
/Users/whitphx/src/gradio-lite-trial/index.html
image-classification (google/vit-base-patch16-224)
Input Image
Drop Image Here
- or -
Click to Upload
Classification
Flag
Top k
5
Clear
Submit
Built with Gradio


index.html
File /Users/whitphx/src/gradio-lite-trial/index.html
image-classification (google/vit-base-patch16-224)
Input Image
Drop Image Here
-or-
Click to Upload
Classification
Flag
Top k
5
Clear
Submit

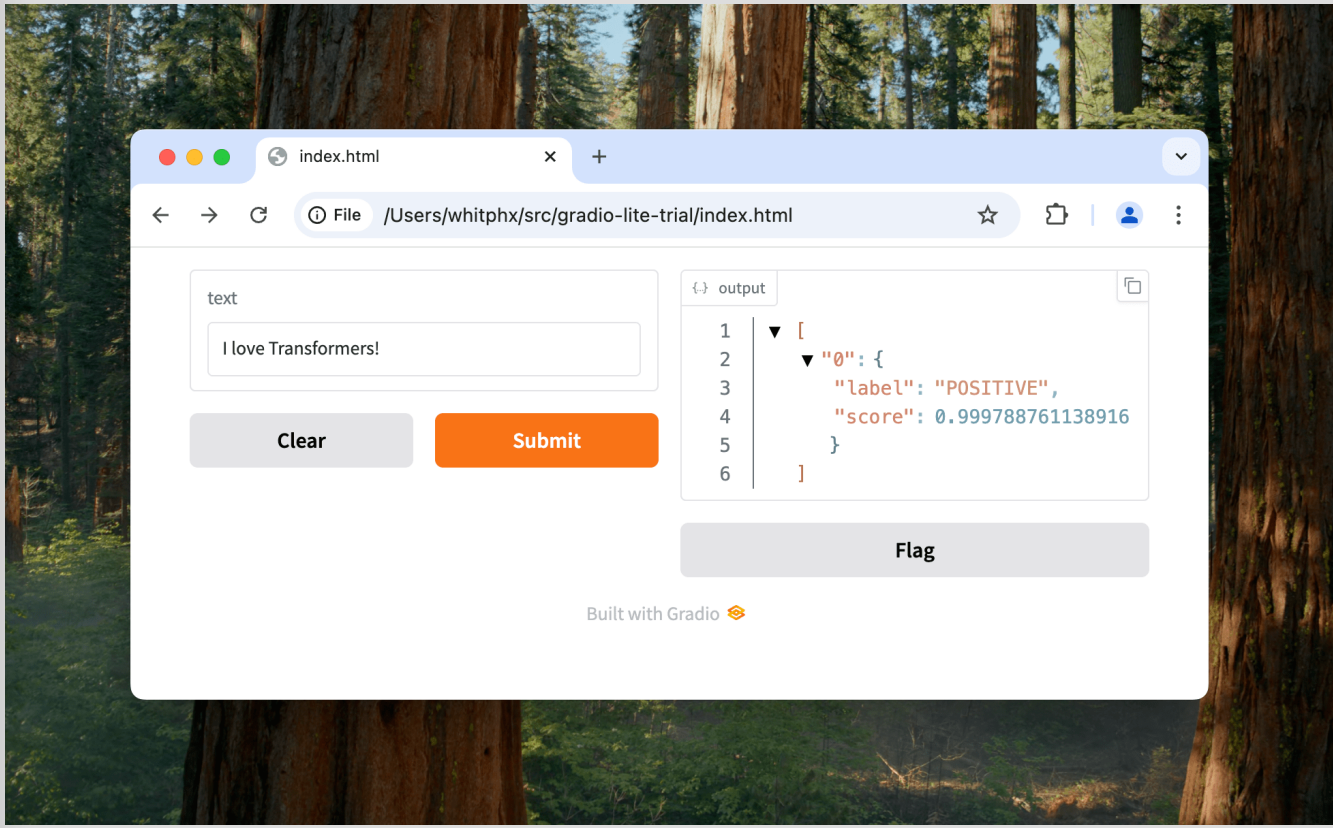
```
    <head>
        <script type="module" crossorigin src="https://cdn.j
        <link rel="stylesheet" href="https://cdn.jsdelivr.ne
    </head>
    <body>
        <gradio-lite>
import gradio as gr
from transformers_js_py import pipeline

pipe = await pipeline('sentiment-analysis')

demo = gr.Interface.from_pipeline(pipe)

demo.launch()

            <gradio-requirements>
transformers-js-py
            </gradio-requirements>
```

```
    <body>
        <gradio-lite>
import gradio as gr
from transformers_js_py import pipeline

pipe = await pipeline('sentiment-analysis')

async def fn(text):
    result = await pipe(text)
    return result

demo = gr.Interface(
    fn=fn,
    inputs=gr.Textbox(),
    outputs=gr.JSON(),
)

demo.launch()
```


index.html
File
/Users/whitphx/src/gradio-lite-trial/index.html
text
I love Transformers!
Clear
Submit
output
1 ▼ [
2 ▼ "0": {
3 "label": "POSITIVE",
4 "score": 0.999788761138916
5 }
6]
Flag
Built with Gradio


index.html
File
/Users/whitphx/src/gradio-lite-trial/index.html
text
output
Clear
Submit
Flag
Built with Gradio

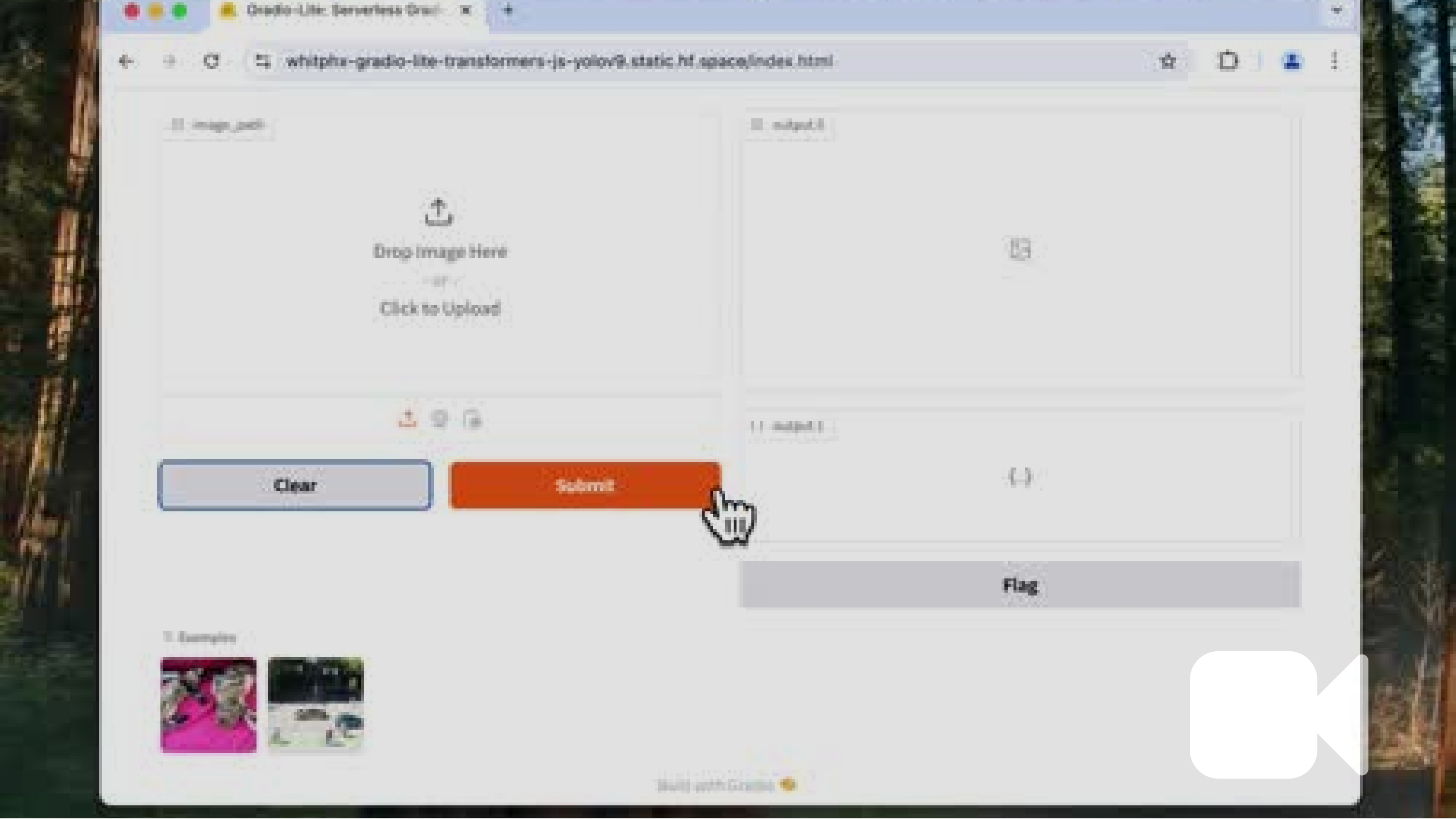
```
<html>
    <head>
        <script type="module" crossorigin src=
        <link rel="stylesheet" href="https://
    </head>
    <body>
        <gradio-lite>
import gradio as gr
from transformers_js_py import pipeline

pipe = await pipeline('sentiment-analysis')

async def fn(text):
    result = await pipe(text)
    return result

demo = gr.Interface(
    fn=fn,
    inputs=gr.Textbox(),
    outputs=gr.JSON(),
)

demo.launch()

            <gradio-requirements>
transformers-js-py
            </gradio-requirements>
        </gradio-lite>
    </body>
```



More examples...

Image: Object Detection

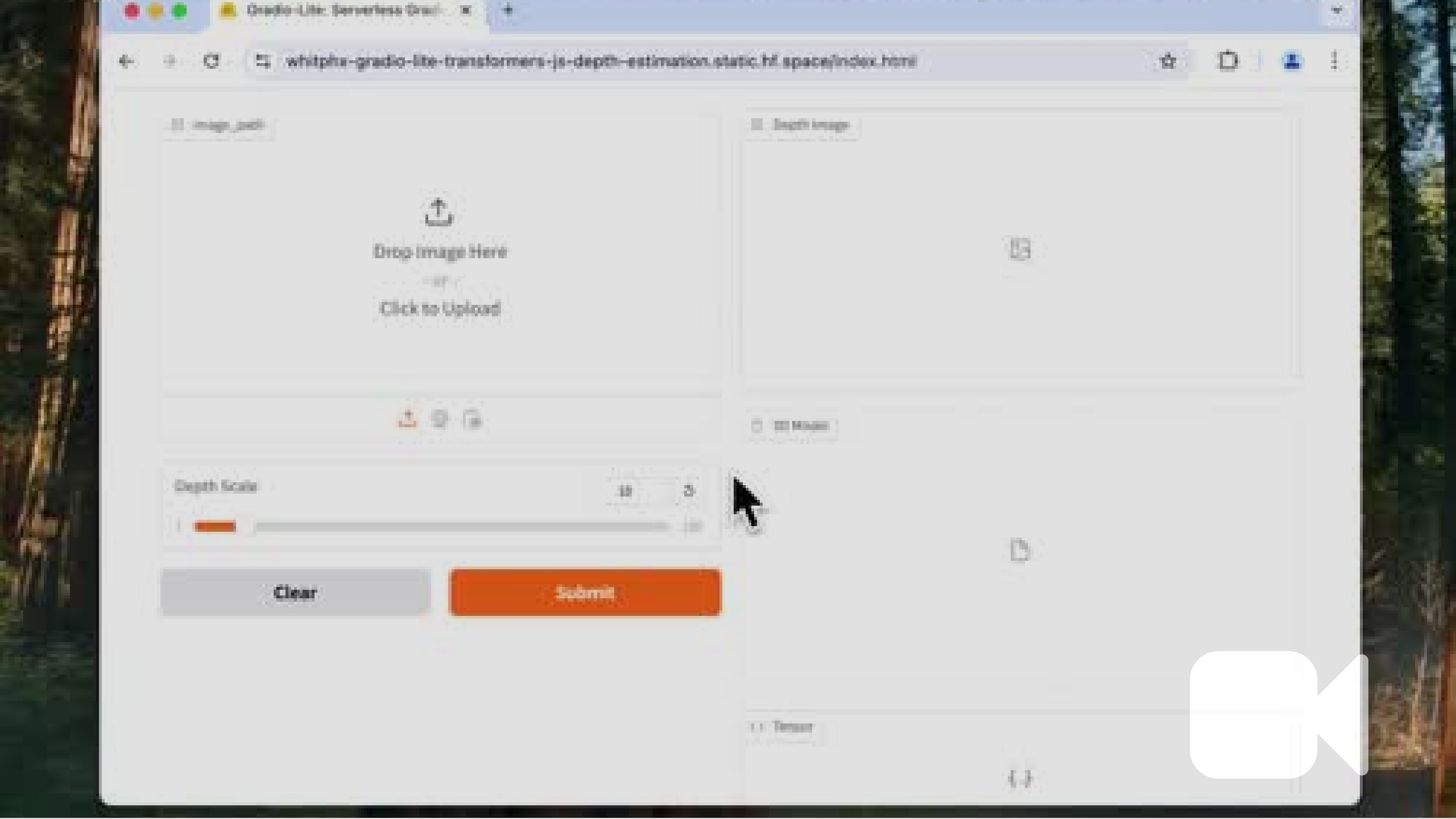
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Gradio-Lite: Serverless Gradio Running Entirely in Your Browser</title>
        <meta name="description" content="Gradio-Lite: Serverless Gradio Running Entirely in Your Browser">

        <script type="module" crossorigin src="https://cdn.jsdelivr.net/npm/@gradio/lite/dist/lite.js"></script>
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@gradio/lite/dist/lite.css" />

        <style>
            html, body {
                margin: 0;
                padding: 0;
                height: 100%;
            }
        </style>
    </head>
    <body>
        <gradio-lite>
            <gradio-file name="app.py" entrypoint>
from transformers_js import import_transformers_js, as_url
import gradio as gr


# Reference: https://huggingface.co/spaces/Xenova/yolov9-web/blob/main/index.js

IMAGE_SIZE = 256;

```


whitphx-gradio-lite-transformers-js-yolov9.static.hf.space/index.html
image_path
Drop Image Here
or
Click to Upload
output
output
Clear
Submit
Flag

https://huggingface.co/spaces/whitphx/gradio-lite-transformers-js-yolov9

Image & 3D: Depth Estimation

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Gradio-Lite: Serverless Gradio Running Entirely in Your Browser</title>
        <meta name="description" content="Gradio-Lite: Serverless Gradio Running Entirely in Your Browser">

        <script type="module" crossorigin src="https://cdn.jsdelivr.net/npm/@gradio/lite/dist/lite.js"></script>
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@gradio/lite/dist/lite.css" />

        <style>
            html, body {
                margin: 0;
                padding: 0;
                height: 100%;
            }
        </style>
    </head>
    <body>
        <gradio-lite>
            <gradio-file name="app.py" entrypoint>
import gradio as gr
import numpy as np
import PIL
import trimesh
from transformers_js import import_transformers_js, as_url


transformers = await import_transformers_js()
```


whitphx-gradio-lite-transformers-js-depth-estimation.static.hf.space/index.html
Drop Image Here
or
Click to Upload
Depth Image
Depth Scale
Clear
Submit

https://huggingface.co/spaces/whitphx/gradio-lite-transformers-js-depth-estimation

Zero-shot Image Classification

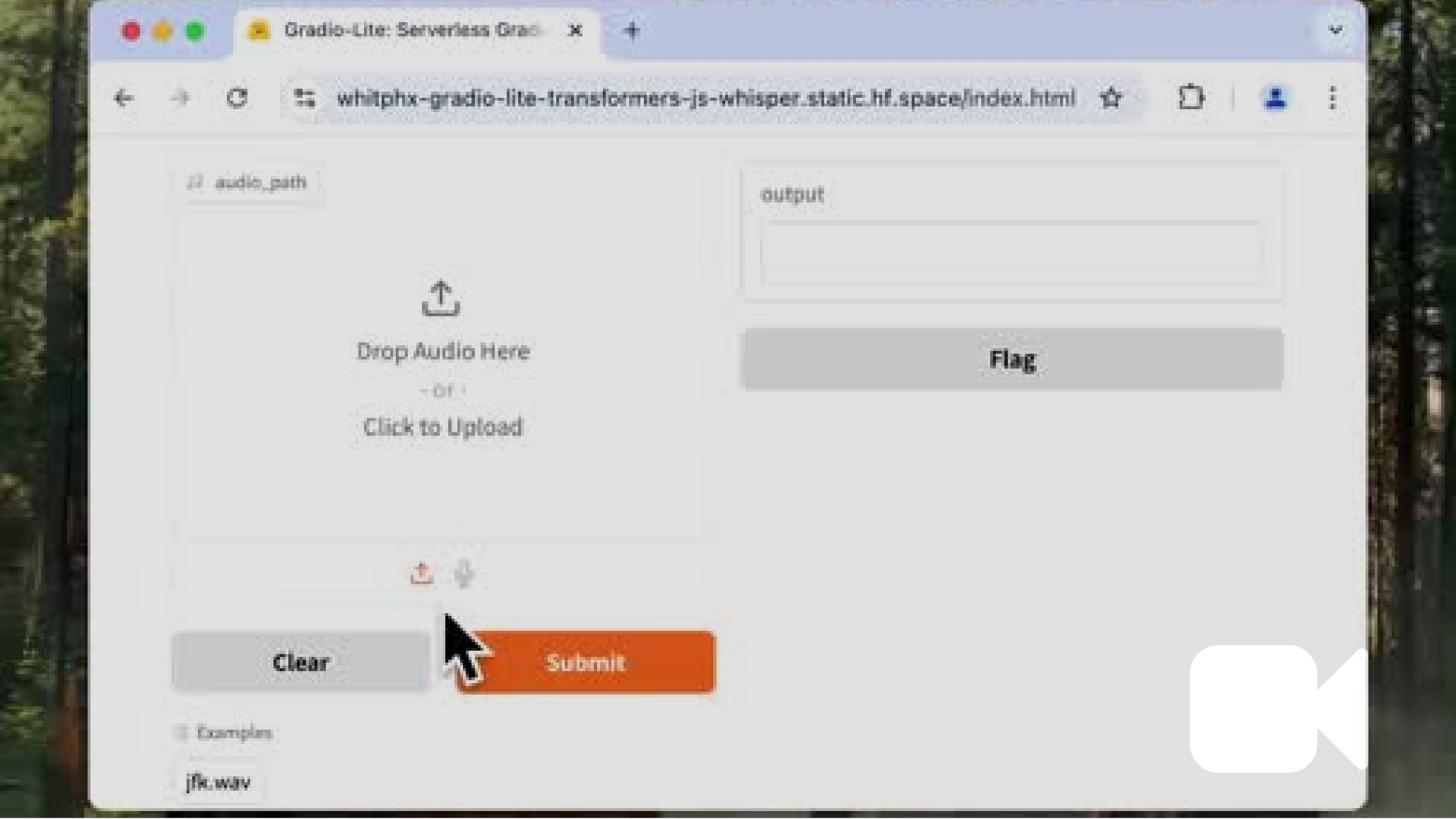
```
<!DOCTYPE html>
<html>
    <head>
        <script type="module" crossorigin src="https://cdn.jsdelivr.net/npm/@gradio/lite/dist/lite.js"></script>
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@gradio/lite/dist/lite.css" />
    </head>
    <body>
<gradio-lite>
<gradio-requirements>
transformers_js_py
</gradio-requirements>

<gradio-file name="app.py" entrypoint>
from transformers_js import import_transformers_js, as_url
import gradio as gr

transformers = await import_transformers_js()
pipeline = transformers.pipeline
pipe = await pipeline('zero-shot-image-classification')

async def classify(image, classes):
    if not image:
        return {}
    classes = [x for c in classes.split(",") if (x := c.strip())]
    if not classes:
        return {}
    data = await pipe(as_url(image), classes)
    result = {item['label']: round(item['score'], 2) for item in data}
    return result

```

- Privacy
- Low latency
- Offline capability
- Scalability without servers
- Low cost

https://huggingface.co/spaces/whitphx/gradio-lite-live-zeroshot-image-classification

Speech-to-Text

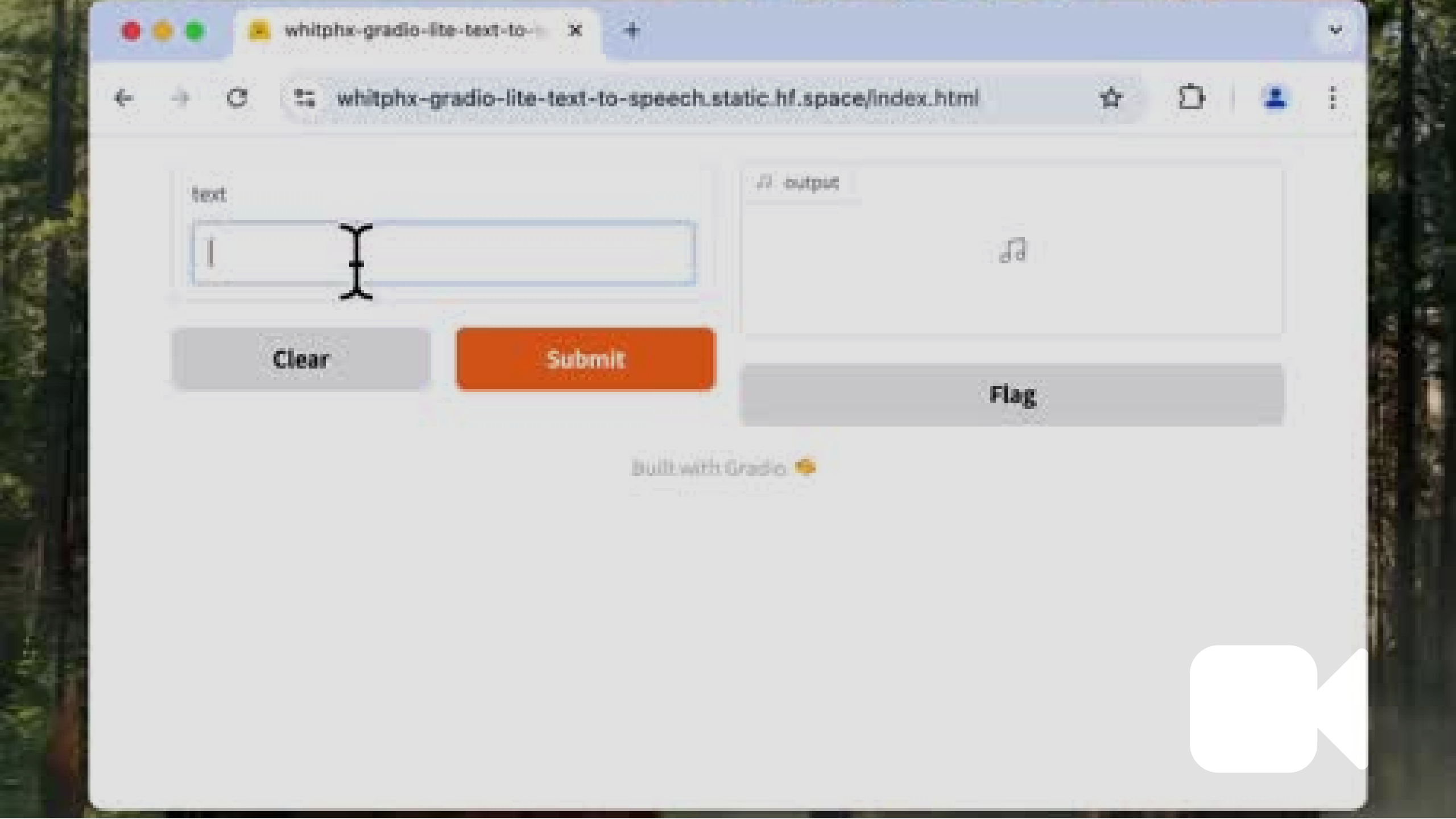
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Gradio-Lite: Serverless Gradio Running Entirely in Your Browser</title>
        <meta name="description" content="Gradio-Lite: Serverless Gradio Running Entirely in Your Browser">

        <script type="module" crossorigin src="https://cdn.jsdelivr.net/npm/@gradio/lite/dist/lite.js"></script>
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@gradio/lite/dist/lite.css" />

        <style>
            html, body {
                margin: 0;
                padding: 0;
                height: 100%;
            }
        </style>
    </head>
    <body>
        <gradio-lite>
            <gradio-file name="app.py" entrypoint>
from transformers_js_py import import_transformers_js, read_audio
import gradio as gr


transformers = await import_transformers_js()
pipeline = transformers.pipeline
pipe = await pipeline('automatic-speech-recognition', 'Xenova/whisper-tiny.en')

```


Gradio-Lite: Serverless Grad
whitphx-gradio-lite-transformers-js-whisper.static.hf.space/index.html
audio_path
Drop Audio Here
- or -
Click to Upload
output
Flag
Clear
Submit
Examples
jfk.wav

https://huggingface.co/spaces/whitphx/gradio-lite-transformers-js-whisper

Text-to-Speech

```
<html>
    <head>
        <script type="module" crossorigin src="https://cdn.jsdelivr.net/npm/@gradio/lite/dist/lite.js"></script>
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@gradio/lite/dist/lite.css" />
    </head>
    <body>
<gradio-lite>

<gradio-requirements>
transformers_js_py
</gradio-requirements>

<gradio-file name="app.py" entrypoint>
from transformers_js_py import import_transformers_js
import gradio as gr
import numpy as np

transformers_js = await import_transformers_js("3.0.2")
pipeline = transformers_js.pipeline

synthesizer = await pipeline(
    'text-to-speech',
    'Xenova/speecht5_tts',
    { "quantized": False }
)
speaker_embeddings = 'https://huggingface.co/datasets/Xenova/transformers.js-docs/resolve/main/speaker_embeddings.bin';


async def synthesize(text):
    out = await synthesizer(text, { "speaker_embeddings": speaker_embeddings });
```


whitphx-gradio-lite-text-to-
whitphx-gradio-lite-text-to-speech.static.hf.space/index.html
text
output
Clear
Submit
Flag
Built with Gradio

https://huggingface.co/spaces/whitphx/gradio-lite-text-to-speech

Supported tasks/models

Text

Question Answering, Summarization, Text Classification, Text Generation, Text-to-text Generation, Token Classification, Translation, Zero-Shot Classification, Feature Extraction, etc...

Image/Video

Depth Estimation, Image Classification, Image SegmentationImage-to-Image, Mask Generation, Object Detection, Video Classification, Unconditional Image Generation, Image Feature Extraction

Audio

Audio Classification, Audio-to-Audio, Automatic Speech Recognition, Text-to-Speech

Multimodal

Document Question Answering, Image-to-Text, Text-to-Image, Visual Question Answering, Zero-Shot Audio Classification, Zero-Shot Image Classification, Zero-Shot Object Detection

Deployment/Delivery

Hosting a static page

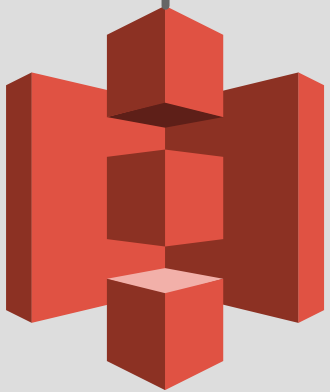
```
<html>
    <head>
        <script type="module" crossorigin src="https://cdn.jsdelivr.net/npm/@gradio/lite/dist/lite.j
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@gradio/lite/dist/lite.css" />
    </head>
    <body>
        <gradio-lite>
import gradio as gr
from transformers_js_py import pipeline

pipe = await pipeline('sentiment-analysis')

async def fn(text):
    result = await pipe(text)
    return result

demo = gr.Interface(
    fn=fn,
```

Static HTML file

GitHub Pages

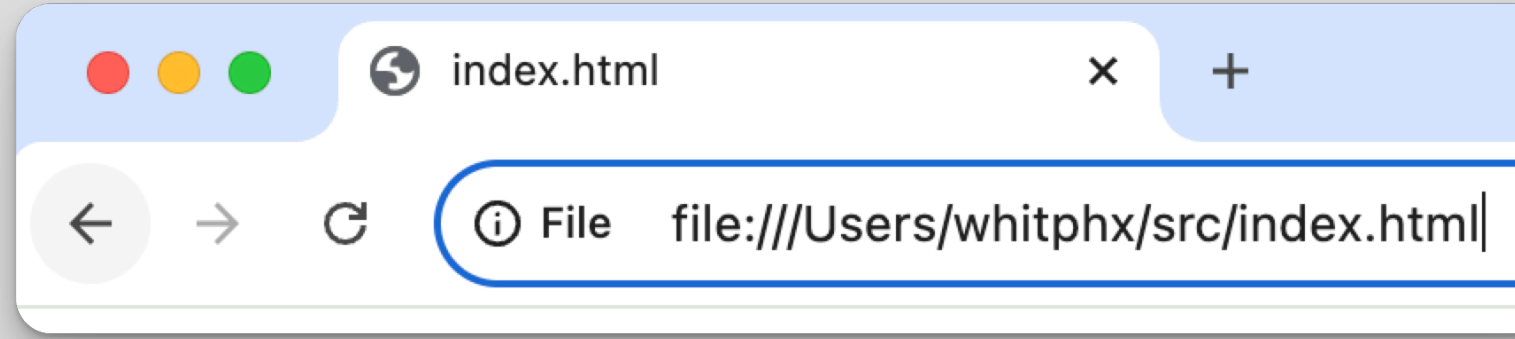
Spaces

hf.co/spaces

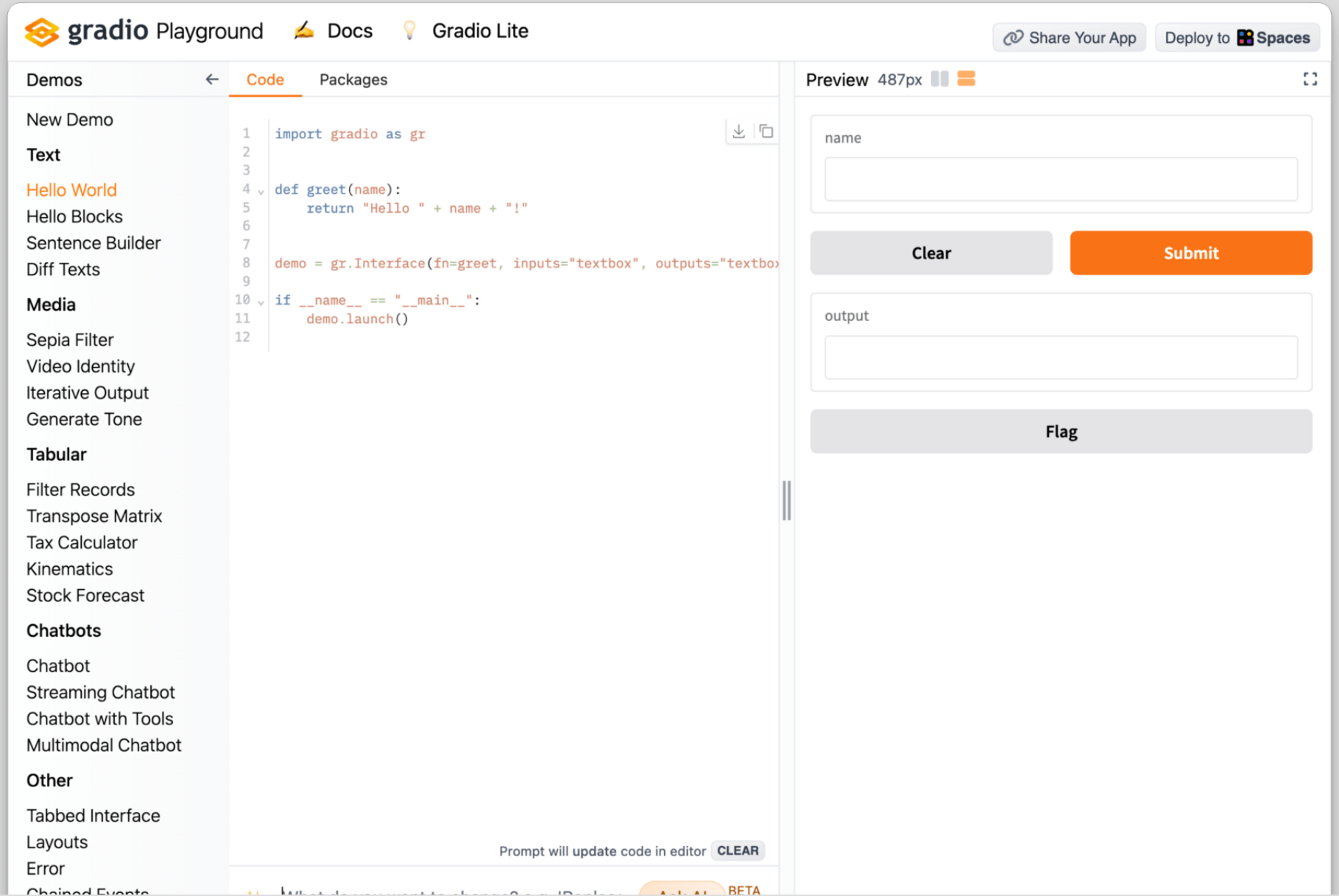


And any your static web server...

...or even distributing the HTML file to your colleagues 😅



Gradio Playground



Use cases

Does it replace JS?

No.

In-browser benefits

- Privacy
- Low latency
- Offline capability
- Scalability without servers
- Low cost

Python/Gradio benefits

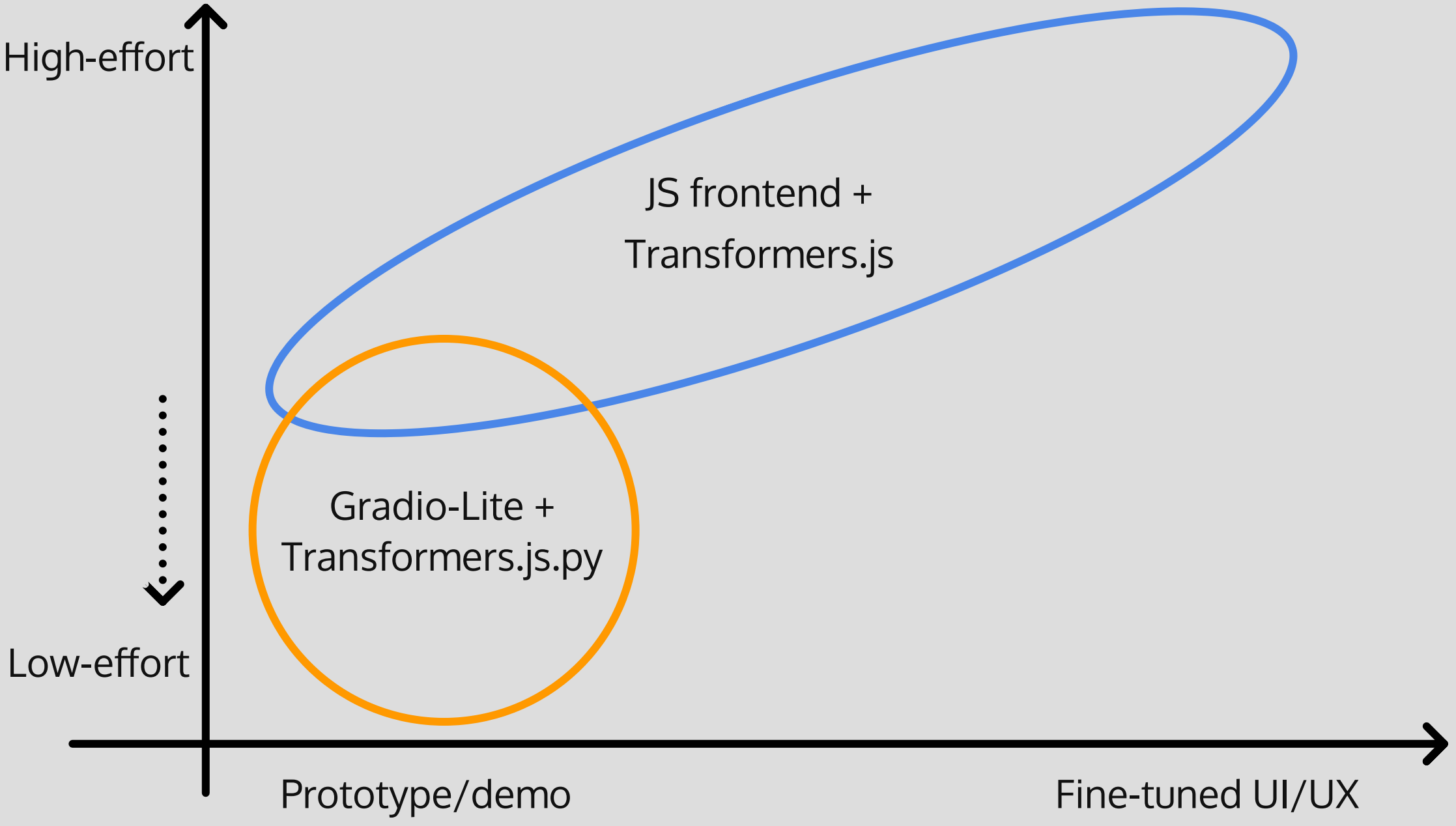
- AI ecosystem
- Built-in components for AI apps
- Easy and quick development

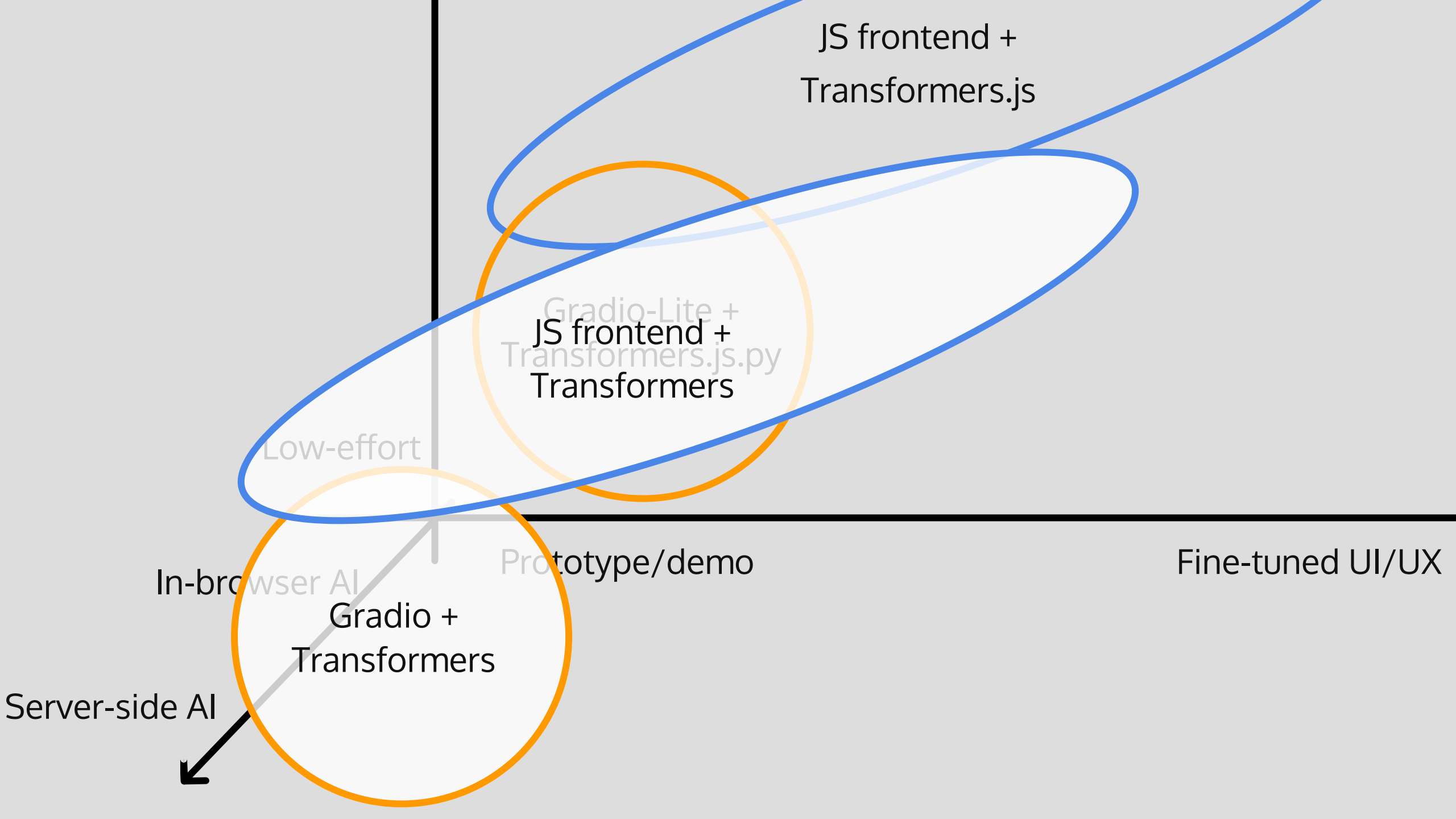
Pros

- Privacy
- Low latency
- Offline capability
- Scalability without servers
- Low cost
- AI ecosystem
- Built-in components for AI apps
- Easy and quick development

Cons

- **Performance**: Initial payload size, loading time, memory usage, etc.
- **Not flexible**: pre-defined components and styles.
- Slight **difference from the normal Python runtime**, e.g. event-loop nature.


High-effort
JS frontend +
Transformers.js
Gradio-Lite +
Transformers.js.py
Low-effort
Prototype/demo
Fine-tuned UI/UX


JS frontend +
Transformers.js
Gradio-Lite +
Transformers.js.py
JS frontend +
Transformers
Low-effort
Prototype/demo
Fine-tuned UI/UX
In-browser AI
Gradio +
Transformers
Server-side AI

Gradio-Lite + Transformers.js are good for...

- Early-stage demo/prototypes
- For communications between researchers, developers, managements, clients, etc
- In-office/private tools
- and...

Please try them out!

Gradio-Lite:
Serverless Gradio Running Entirely in Your Browser

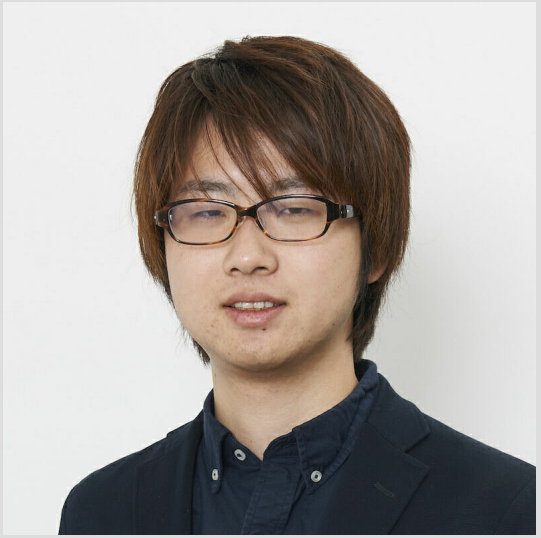
https://huggingface.co/blog/gradio-lite

Building Serverless Machine Learning Apps with Gradio-Lite and Transformers.js

https://www.gradio.app/guides/gradio-lite-and-transformers-js

In-browser LLM app in pure Python:

Gemini Nano + Gradio-Lite



Yuichiro Tachibana

@whitphx

- Pythonista
- OSS enthusiast
- ML Developer Advocate at Hugging Face
- Streamlit Creator