Audio

`gradio.Audio(...)`

Description

Creates an audio component that can be used to upload/record audio (as an input) or display audio (as an output).

Behavior

As input: depending on `type`, passes the uploaded audio as `str` filepath or a `Tuple(int, numpy.array)` corresponding to (sample rate in Hz, audio data). If the latter, the audio data is a 16-bit int array whose values range from -32768 to 32767 and shape of the audio data array is (samples,) for mono audio or (samples, channels) for multi-channel audio.
As output: expects a `Tuple(int, numpy.array)` corresponding to (sample rate in Hz, audio data as a float or int numpy array) or as a `str` or `pathlib.Path` filepath or URL to an audio file, or bytes for binary content (recommended for streaming). Note: When converting audio data from float format to WAV, the audio is normalized by its peak value to avoid distortion or clipping in the resulting audio.

Initialization

Parameter	Description
`value` *str	Path	tuple[int, np.ndarray]	Callable	None* **default: None**	A path, URL, or [sample_rate, numpy array] tuple (sample rate in Hz, audio data as a float or int numpy array) for the default value that Audio component is going to take. If callable, the function will be called whenever the app loads to set the initial value of the component.

ımeter	Description
`sources` *list[Literal[('upload', 'microphone')]] | None* **default: None**	A list of sources permitted for audio. "upload" creates a box where user can drop an audio file, "microphone" creates a microphone input. The first element in the list will be used as the default source. If None, defaults to ["upload", "microphone"], or ["microphone"] if `streaming` is True.
`type` *Literal[('numpy', 'filepath')]* **default: "numpy"**	The format the audio file is converted to before being passed into the prediction function. "numpy" converts the audio to a tuple consisting of: (int sample rate, numpy.array for the data), "filepath" passes a str path to a temporary file containing the audio.
`label` *str | None* **default: None**	The label for this component. Appears above the component and is also used as the header if there are a table of examples for this component. If None and used in a `gr.Interface`, the label will be the name of the parameter this component is assigned to.
`every` *float | None* **default: None**	If `value` is a callable, run the function 'every' number of seconds while the client connection is open. Has no effect otherwise. Queue must be enabled. The event can be accessed (e.g. to cancel it) via this component's .load_event attribute.
`show_label` *bool | None* **default: None**	if True, will display label.
`container` *bool* **default: True**	If True, will place the component in a container - providing some extra padding around the border.
`scale` *int | None* **default: None**	relative width compared to adjacent Components in a Row. For example, if Component A has scale=2, and Component B has scale=1, A will be twice as wide as B. Should be an integer.

ımeter	Description
`min_width` *int* **default: 160**	minimum pixel width, will wrap if not sufficient screen space to satisfy this value. If a certain scale value results in this Component being narrower than min_width, the min_width parameter will be respected first.
`interactive` *bool* | *None* **default: None**	If True, will allow users to upload and edit an audio file. If False, can only be used to play audio. If not provided, this is inferred based on whether the component is used as an input or output.
`visible` *bool* **default: True**	If False, component will be hidden.
`streaming` *bool* **default: False**	If set to True when used in a `live` interface as an input, will automatically stream webcam feed. When used set as an output, takes audio chunks yield from the backend and combines them into one streaming audio output.
`elem_id` *str* | *None* **default: None**	An optional string that is assigned as the id of this component in the HTML DOM. Can be used for targeting CSS styles.
`elem_classes` *list[str]* | *str* | *None* **default: None**	An optional list of strings that are assigned as the classes of this component in the HTML DOM. Can be used for targeting CSS styles.
`render` *bool* **default: True**	If False, component will not render be rendered in the Blocks context. Should be used if the intention is to assign event listeners now but render the component later.

ımeter	Description
`format` *Literal[('wav', 'mp3')]* **default: "wav"**	The file format to save audio files. Either 'wav' or 'mp3'. wav files are lossless but will tend to be larger files. mp3 files tend to be smaller. Default is wav. Applies both when this component is used as an input (when `type` is "format") and when this component is used as an output.
`autoplay` *bool* **default: False**	Whether to automatically play the audio when the component is used as an output. Note: browsers will not autoplay audio files if the user has not interacted with the page yet.
`show_download_button` *bool | None* **default: None**	If True, will show a download button in the corner of the component for saving audio. If False, icon does not appear. By default, it will be True for output components and False for input components.
`show_share_button` *bool | None* **default: None**	If True, will show a share icon in the corner of the component that allows user to share outputs to Hugging Face Spaces Discussions. If False, icon does not appear. If set to None (default behavior), then the icon appears if this Gradio app is launched on Spaces, but not otherwise.
`editable` *bool* **default: True**	If True, allows users to manipulate the audio file if the component is interactive. Defaults to True.
`min_length` *int | None* **default: None**	The minimum length of audio (in seconds) that the user can pass into the prediction function. If None, there is no minimum length.
`max_length` *int | None* **default: None**	The maximum length of audio (in seconds) that the user can pass into the prediction function. If None, there is no maximum length.

ımeter	Description
`waveform_options` *WaveformOptions* | *dict* | *None* **default: None**	A dictionary of options for the waveform display. Options include: waveform_color (str), waveform_progress_color (str), show_controls (bool), skip_length (int). Default is None, which uses the default values for these options.

Shortcuts

Class	Interface String Shortcut	Initialization
`gradio.Audio`	"audio"	Uses default values
`gradio.Microphone`	"microphone"	Uses sources=["microphone"]

Demos

main_note | generate_tone | reverse_audio

```
from math import log2, pow
import os

import numpy as np
from scipy.fftpack import fft

import gradio as gr

A4 = 440
C0 = A4 * pow(2, -4.75)
name = ["C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", "B"]
```

Event Listeners

Description

Event listeners allow you to capture and respond to user interactions with the UI components you've defined in a Gradio Blocks app. When a user interacts with an element, such as changing a slider value or uploading an image, a function is called.

Supported Event Listeners

The `Audio` component supports the following event listeners. Each event listener takes the same parameters, which are listed in the Event Arguments table below.

Listener	Description
`gradio.Audio.stream(fn, ···)`	This listener is triggered when the user streams the Audio.
`gradio.Audio.change(fn, ···)`	Triggered when the value of the Audio changes either because of user input (e.g. a user types in a textbox) OR because of a function update (e.g. an image receives a value from the output of an event trigger). See `.input()` for a listener that is only triggered by user input.
`gradio.Audio.clear(fn, ···)`	This listener is triggered when the user clears the Audio using the X button for the component.
`gradio.Audio.play(fn, ···)`	This listener is triggered when the user plays the media in the Audio.
`gradio.Audio.pause(fn, ···)`	This listener is triggered when the media in the Audio stops for any reason.
`gradio.Audio.stop(fn, ···)`	This listener is triggered when the user reaches the end of the media playing in the Audio.
`gradio.Audio.pause(fn, ···)`	This listener is triggered when the media in the Audio stops for any reason.
`gradio.Audio.start_recording(fn, ···)`	This listener is triggered when the user starts recording with the Audio.
`gradio.Audio.pause_recording(fn, ···)`	This listener is triggered when the user pauses recording with the Audio.
`gradio.Audio.stop_recording(fn, ···)`	This listener is triggered when the user stops recording with the Audio.
`gradio.Audio.upload(fn, ···)`	This listener is triggered when the user uploads a file into the Audio.

Event Arguments

Parameter	Description

Parameter	Description
`fn` *Callable	None	Literal['decorator']* **default: "decorator"**	the function to call when this event is triggered. Often a machine learning model's prediction function. Each parameter of the function corresponds to one input component, and the function should return a single value or a tuple of values, with each element in the tuple corresponding to one output component.
`inputs` *Component	list[Component]	set[Component]	None* **default: None**	List of gradio.components to use as inputs. If the function takes no inputs, this should be an empty list.
`outputs` *Component	list[Component]	None* **default: None**	List of gradio.components to use as outputs. If the function returns no outputs, this should be an empty list.
`api_name` *str	None	Literal[False]* **default: None**	defines how the endpoint appears in the API docs. Can be a string, None, or False. If set to a string, the endpoint will be exposed in the API docs with the given name. If None (default), the name of the function will be used as the API endpoint. If False, the endpoint will not be exposed in the API docs and downstream apps (including those that `gr.load` this app) will not be able to use this event.
`scroll_to_output` *bool* **default: False**	If True, will scroll to output component on completion
`show_progress` *Literal[('full', 'minimal', 'hidden')]* **default: "hidden"**	If True, will show progress animation while pending

Parameter	Description
`queue` *bool* | *None* **default: None**	If True, will place the request on the queue, if the queue has been enabled. If False, will not put this event on the queue, even if the queue has been enabled. If None, will use the queue setting of the gradio app.
`batch` *bool* **default: False**	If True, then the function should process a batch of inputs, meaning that it should accept a list of input values for each parameter. The lists should be of equal length (and be up to length `max_batch_size`). The function is then ***required*** to return a tuple of lists (even if there is only 1 output component), with each list in the tuple corresponding to one output component.
`max_batch_size` *int* **default: 4**	Maximum number of inputs to batch together if this is called from the queue (only relevant if batch=True)
`preprocess` *bool* **default: True**	If False, will not run preprocessing of component data before running 'fn' (e.g. leaving it as a base64 string if this method is called with the `Image` component).
`postprocess` *bool* **default: True**	If False, will not run postprocessing of component data before returning 'fn' output to the browser.
`cancels` *dict[str, Any]* | *list[dict[str, Any]]* | *None* **default: None**	A list of other events to cancel when this listener is triggered. For example, setting cancels=[click_event] will cancel the click_event, where click_event is the return value of another components .click method. Functions that have not yet run (or generators that are iterating) will be cancelled, but functions that are currently running will be allowed to finish.

Parameter	Description
`every` *float* | *None* **default: None**	Run this event 'every' number of seconds while the client connection is open. Interpreted in seconds. Queue must be enabled.
`trigger_mode` *Literal[('once', 'multiple', 'always_last')]* | *None* **default: None**	If "once" (default for all events except `.change()`) would not allow any submissions while an event is pending. If set to "multiple", unlimited submissions are allowed while pending, and "always_last" (default for `.change()` event) would allow a second submission after the pending event is complete.
`js` *str* | *None* **default: None**	Optional frontend js method to run before running 'fn'. Input arguments for js method are values of 'inputs' and 'outputs', return should be a list of values for output components.
`concurrency_limit` *int* | *None* | *Literal['default']* **default: "default"**	If set, this is the maximum number of this event that can be running simultaneously. Can be set to None to mean no concurrency_limit (any number of this event can be running simultaneously). Set to "default" to use the default concurrency limit (defined by the `default_concurrency_limit` parameter in `Blocks.queue()`, which itself is 1 by default).
`concurrency_id` *str* | *None* **default: None**	If set, this is the id of the concurrency group. Events with the same concurrency_id will be limited by the lowest set concurrency_limit.
`show_api` *bool* **default: True**	whether to show this event in the "view API" page of the Gradio app, or in the ".view_api()" method of the Gradio clients. Unlike setting api_name to False, setting show_api to False will still allow downstream apps to use this event. If fn is None, show_api will automatically be set to False.