huggingface.co

facebook/musicgen-small · Hugging Face

10–12 minutes

MusicGen is a text-to-music model capable of genreating high-quality music samples conditioned on text descriptions or audio prompts. It is a single stage auto-regressive Transformer model trained over a 32kHz EnCodec tokenizer with 4 codebooks sampled at 50 Hz. Unlike existing methods, like MusicLM, MusicGen doesn't require a self-supervised semantic representation, and it generates all 4 codebooks in one pass. By introducing a small delay between the codebooks, we show we can predict them in parallel, thus having only 50 auto-regressive steps per second of audio.

MusicGen was published in Simple and Controllable Music Generation by *Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, Alexandre Défossez*.

Four checkpoints are released:

- **small** (this checkpoint)
- medium
- large
- melody

Example

Try out MusicGen yourself!

- Audiocraft Colab:

  Open in Colab

- Hugging Face Colab:

  Open in Colab

- Hugging Face Demo:

  Open in HF Spaces

🤗 Transformers Usage

You can run MusicGen locally with the 🤗 Transformers library from version 4.31.0 onwards.

1. First install the 🤗 Transformers library and scipy:

```
pip install --upgrade pip
pip install --upgrade transformers scipy
```

2. Run inference via the `Text-to-Audio` (TTA) pipeline. You can infer the MusicGen model via the TTA pipeline in just a few lines of code!

```
from transformers import pipeline
import scipy

synthesiser = pipeline("text-to-audio",
"facebook/musicgen-small")

music = synthesiser("lo-fi music with a soothing
melody", forward_params={"do_sample": True})
```

```
scipy.io.wavfile.write("musicgen_out.wav",
rate=music["sampling_rate"],
music=audio["audio"])
```

3. Run inference via the Transformers modelling code. You can use the processor + generate code to convert text into a mono 32 kHz audio waveform for more fine-grained control.

```
from transformers import AutoProcessor,
MusicgenForConditionalGeneration

processor =
AutoProcessor.from_pretrained("facebook/musicgen-
small")
model =
MusicgenForConditionalGeneration.from_pretrained(
small")

inputs = processor(
    text=["80s pop track with bassy drums and
synth", "90s rock song with loud guitars and
heavy drums"],
    padding=True,
    return_tensors="pt",
)

audio_values = model.generate(**inputs,
max_new_tokens=256)
```

3. Listen to the audio samples either in an ipynb notebook:

```
from IPython.display import Audio

sampling_rate =
model.config.audio_encoder.sampling_rate
Audio(audio_values[0].numpy(),
rate=sampling_rate)
```

Or save them as a `.wav` file using a third-party library, e.g. `scipy`:

```
import scipy

sampling_rate =
model.config.audio_encoder.sampling_rate
scipy.io.wavfile.write("musicgen_out.wav",
rate=sampling_rate, data=audio_values[0,
0].numpy())
```

For more details on using the MusicGen model for inference using the 🤗 Transformers library, refer to the MusicGen docs.

Audiocraft Usage

You can also run MusicGen locally through the original Audiocraft library:

1. First install the `audiocraft` library

```
pip install git+https://github.com
/facebookresearch/audiocraft.git
```

2. Make sure to have `ffmpeg` installed:
3. Run the following Python code:

```
from audiocraft.models import MusicGen
from audiocraft.data.audio import audio_write

model = MusicGen.get_pretrained("small")
model.set_generation_params(duration=8)  #
generate 8 seconds.

descriptions = ["happy rock", "energetic EDM"]

wav = model.generate(descriptions)  # generates 2
samples.

for idx, one_wav in enumerate(wav):
    # Will save under {idx}.wav, with loudness
normalization at -14 db LUFS.
    audio_write(f'{idx}', one_wav.cpu(),
model.sample_rate, strategy="loudness")
```

Model details

Organization developing the model: The FAIR team of Meta AI.

Model date: MusicGen was trained between April 2023 and May 2023.

Model version: This is the version 1 of the model.

Model type: MusicGen consists of an EnCodec model for audio tokenization, an auto-regressive language model based on the transformer architecture for music modeling. The model comes in different sizes: 300M, 1.5B and 3.3B parameters ; and two variants: a model trained for text-to-music generation task and a model trained for melody-guided music generation.

Paper or resources for more information: More information can be found in the paper Simple and Controllable Music Generation.

Citation details:

```
@misc{copet2023simple,
      title={Simple and Controllable Music Generation},
      author={Jade Copet and Felix Kreuk and Itai Gat and Tal Remez and David Kant and Gabriel Synnaeve and Yossi Adi and Alexandre Défossez},
      year={2023},
      eprint={2306.05284},
      archivePrefix={arXiv},
      primaryClass={cs.SD}
}
```

License: Code is released under MIT, model weights are released under CC-BY-NC 4.0.

Where to send questions or comments about the model: Questions and comments about MusicGen can be sent via the Github repository of the project, or by opening an issue.

Intended use

Primary intended use: The primary use of MusicGen is research on AI-based music generation, including:

- Research efforts, such as probing and better understanding the limitations of generative models to further improve the state of science
- Generation of music guided by text or melody to understand

current abilities of generative AI models by machine learning amateurs

Primary intended users: The primary intended users of the model are researchers in audio, machine learning and artificial intelligence, as well as amateur seeking to better understand those models.

Out-of-scope use cases: The model should not be used on downstream applications without further risk evaluation and mitigation. The model should not be used to intentionally create or disseminate music pieces that create hostile or alienating environments for people. This includes generating music that people would foreseeably find disturbing, distressing, or offensive; or content that propagates historical or current stereotypes.

Metrics

Models performance measures: We used the following objective measure to evaluate the model on a standard music benchmark:

- Frechet Audio Distance computed on features extracted from a pre-trained audio classifier (VGGish)
- Kullback-Leibler Divergence on label distributions extracted from a pre-trained audio classifier (PaSST)
- CLAP Score between audio embedding and text embedding extracted from a pre-trained CLAP model

Additionally, we run qualitative studies with human participants, evaluating the performance of the model with the following axes:

- Overall quality of the music samples;

- Text relevance to the provided text input;
- Adherence to the melody for melody-guided music generation.

  More details on performance measures and human studies can be found in the paper.

  Decision thresholds: Not applicable.

Evaluation datasets

The model was evaluated on the MusicCaps benchmark and on an in-domain held-out evaluation set, with no artist overlap with the training set.

Training datasets

The model was trained on licensed data using the following sources: the Meta Music Initiative Sound Collection, Shutterstock music collection and the Pond5 music collection. See the paper for more details about the training set and corresponding preprocessing.

Evaluation results

Below are the objective metrics obtained on MusicCaps with the released model. Note that for the publicly released models, we had all the datasets go through a state-of-the-art music source separation method, namely using the open source Hybrid Transformer for Music Source Separation (HT-Demucs), in order to keep only the instrumental part. This explains the difference in objective metrics with the models used in the paper.

Model	Frechet Audio Distance	KLD	Text Consistency	Chroma Cosine Similarit
facebook/musicgen-small	4.88	1.42	0.27	-
facebook/musicgen-medium	5.14	1.38	0.28	-
facebook/musicgen-large	5.48	1.37	0.28	-
facebook/musicgen-melody	4.93	1.41	0.27	0.44

More information can be found in the paper Simple and Controllable Music Generation, in the Results section.

Limitations and biases

Data: The data sources used to train the model are created by music professionals and covered by legal agreements with the right holders. The model is trained on 20K hours of data, we believe that scaling the model on larger datasets can further improve the performance of the model.

Mitigations: Vocals have been removed from the data source using corresponding tags, and then using a state-of-the-art music source separation method, namely using the open source Hybrid Transformer for Music Source Separation (HT-Demucs).

Limitations:

- The model is not able to generate realistic vocals.

- The model has been trained with English descriptions and will not perform as well in other languages.
- The model does not perform equally well for all music styles and cultures.
- The model sometimes generates end of songs, collapsing to silence.
- It is sometimes difficult to assess what types of text descriptions provide the best generations. Prompt engineering may be required to obtain satisfying results.

Biases: The source of data is potentially lacking diversity and all music cultures are not equally represented in the dataset. The model may not perform equally well on the wide variety of music genres that exists. The generated samples from the model will reflect the biases from the training data. Further work on this model should include methods for balanced and just representations of cultures, for example, by scaling the training data to be both diverse and inclusive.

Risks and harms: Biases and limitations of the model may lead to generation of samples that may be considered as biased, inappropriate or offensive. We believe that providing the code to reproduce the research and train new models will allow to broaden the application to new and more representative data.

Use cases: Users must be aware of the biases, limitations and risks of the model. MusicGen is a model developed for artificial intelligence research on controllable music generation. As such, it should not be used for downstream applications without further investigation and mitigation of risks.