openai/whisper-large-v3

like 5.13k Follow OpenAI 26.9k

Automatic Speech Recognition | Transformers | PyTorch | JAX | Safetensors

99 languages | whisper | audio | hf-asr-leaderboard | arxiv:2212.04356 | License: apache-2.0

Deploy Use this model

Model card Files xet Community 216

Edit model card

Whisper

Whisper is a state-of-the-art model for automatic speech recognition (ASR) and speech translation, proposed in the paper Robust Speech Recognition via Large-Scale Weak Supervision by Alec Radford et al. from OpenAI. Trained on >5M hours of labeled data, Whisper demonstrates a strong ability to generalise to many datasets and domains in a zero-shot setting.

Whisper large-v3 has the same architecture as the previous large and large-v2 models, except for the following minor differences:

1. The spectrogram input uses 128 Mel frequency bins instead of 80
2. A new language token for Cantonese

The Whisper large-v3 model was trained on 1 million hours of weakly labeled audio and 4 million hours of

pseudo-labeled audio collected using Whisper large-v2 . The model was trained for 2.0 epochs over this

Downloads last month
4,508,552

Safetensors

Model size 2B params

Model tree for openai/whisper-la...

Adapters ..... 140 models

Finetunes ..... 655 models

mixture dataset.

The large-v3 model shows improved performance over a wide variety of languages, showing 10% to 20% reduction of errors compared to Whisper large-v2 . For more details on the different checkpoints available, refer to the section Model details

Disclaimer: Content for this model card has partly been written by the 🤗 Hugging Face team, and partly copied and pasted from the original model card.

Usage

Whisper large-v3 is supported in Hugging Face 🤗 Transformers. To run the model, first install the Transformers library. For this example, we'll also install 🤗 Datasets to load toy audio dataset from the Hugging Face Hub, and 🤗 Accelerate to reduce the model loading time:

```
pip install --upgrade pip
pip install --upgrade transformers dataset
```

The model can be used with the `pipeline` class to transcribe audios of arbitrary length:

```
import torch
from transformers import AutoModelForSpeecl
from datasets import load_dataset


device = "cuda:0" if torch.cuda.is_availab
torch_dtype = torch.float16 if torch.cuda.

model_id = "openai/whisper-large-v3"
```

Merges 1 model

Quantizations 16 models

Spaces using openai/whisper… 100

- openai/whisper ⭐
- nari-labs/Dia2-2B
- alexnasa/HuMo_local
- hf-audio/whisper-large-v3
- elmresearchcenter/open_universal…
- OpenSound/CapSpeech-TTS
- Illia56/Ask-AI-Youtube
- gobeldan/insanely-fast-whisper-we…
- cstr/transcribe_audio
- wcy1122/MGM-Omni
- aiqcamp/ENGLISH-Speaking-Scoring
- ardha27/Youtube-AI-Summarizer

+ 88 Spaces

```
model = AutoModelForSpeechSeq2Seq.from_pre
    model_id, torch_dtype=torch_dtype, low
)
model.to(device)

processor = AutoProcessor.from_pretrained(

pipe = pipeline(
    "automatic-speech-recognition",
    model=model,
    tokenizer=processor.tokenizer,
    feature_extractor=processor.feature_ex
    torch_dtype=torch_dtype,
    device=device,
)

dataset = load_dataset("distil-whisper/lib
sample = dataset[0]["audio"]

result = pipe(sample)
print(result["text"])
```

To transcribe a local audio file, simply pass the path to your audio file when you call the pipeline:

```
result = pipe("audio.mp3")
```

Multiple audio files can be transcribed in parallel by specifying them as a list and setting the `batch_size` parameter:

```
result = pipe(["audio_1.mp3", "audio_2.mp3
```

Transformers is compatible with all Whisper decoding strategies, such as temperature fallback and condition on previous tokens. The following example demonstrates how to enable these

heuristics:

```
generate_kwargs = {
    "max_new_tokens": 448,
    "num_beams": 1,
    "condition_on_prev_tokens": False,
    "compression_ratio_threshold": 1.35,
    "temperature": (0.0, 0.2, 0.4, 0.6, 0.
    "logprob_threshold": -1.0,
    "no_speech_threshold": 0.6,
    "return_timestamps": True,
}

result = pipe(sample, generate_kwargs=gene
```

Whisper predicts the language of the source audio automatically. If the source audio language is known *a-priori*, it can be passed as an argument to the pipeline:

```
result = pipe(sample, generate_kwargs={"la
```

By default, Whisper performs the task of *speech transcription*, where the source audio language is the same as the target text language. To perform *speech translation*, where the target text is in English, set the task to `"translate"`:

```
result = pipe(sample, generate_kwargs={"ta
```

Finally, the model can be made to predict timestamps. For sentence-level timestamps, pass the `return_timestamps` argument:

```
result = pipe(sample, return_timestamps=Tr
print(result["chunks"])
```

```
print(result["chunks"])
```

And for word-level timestamps:

```
result = pipe(sample, return_timestamps="w
print(result["chunks"])
```

The above arguments can be used in isolation or in combination. For example, to perform the task of speech transcription where the source audio is in French, and we want to return sentence-level timestamps, the following can be used:

```
result = pipe(sample, return_timestamps=Tr
print(result["chunks"])
```

▸ For more control over the generation parameters, use the model + processor API directly:

Additional Speed & Memory Improvements

You can apply additional speed and memory improvements to Whisper to further reduce the inference speed and VRAM requirements.

Chunked Long-Form

Whisper has a receptive field of 30-seconds. To transcribe audios longer than this, one of two long-form algorithms are required:

1. **Sequential:** uses a "sliding window" for buffered inference, transcribing 30-second slices one after the other

2. **Chunked:** splits long audio files into shorter ones (with a small overlap between segments),

transcribes each segment independently, and stitches the resulting transcriptions at the boundaries

The sequential long-form algorithm should be used in either of the following scenarios:

1. Transcription accuracy is the most important factor, and speed is less of a consideration
2. You are transcribing **batches** of long audio files, in which case the latency of sequential is comparable to chunked, while being up to 0.5% WER more accurate

Conversely, the chunked algorithm should be used when:

1. Transcription speed is the most important factor
2. You are transcribing a **single** long audio file

By default, Transformers uses the sequential algorithm. To enable the chunked algorithm, pass the `chunk_length_s` parameter to the `pipeline`. For large-v3, a chunk length of 30-seconds is optimal. To activate batching over long audio files, pass the argument `batch_size`:

```
import torch
from transformers import AutoModelForSpeecl
from datasets import load_dataset


device = "cuda:0" if torch.cuda.is_availab
torch_dtype = torch.float16 if torch.cuda.

model_id = "openai/whisper-large-v3"


model = AutoModelForSpeechSeq2Seq.from_pre
```

```
    model_id, torch_dtype=torch_dtype, low
)
model.to(device)

processor = AutoProcessor.from_pretrained(

pipe = pipeline(
    "automatic-speech-recognition",
    model=model,
    tokenizer=processor.tokenizer,
    feature_extractor=processor.feature_ex
    chunk_length_s=30,
    batch_size=16,  # batch size for infer
    torch_dtype=torch_dtype,
    device=device,
)

dataset = load_dataset("distil-whisper/lib
sample = dataset[0]["audio"]

result = pipe(sample)
print(result["text"])
```

Torch compile

The Whisper forward pass is compatible with `torch.compile` for 4.5x speed-ups.

Note: `torch.compile` is currently not compatible with the Chunked long-form algorithm or Flash Attention 2 ⚠️

```
import torch
from torch.nn.attention import SDPBackend,
from transformers import AutoModelForSpeecl
from datasets import load_dataset
from tqdm import tqdm

torch.set_float32_matmul_precision("high")

device = "cuda:0" if torch.cuda.is_availab
```

```
torch_dtype = torch.float16 if torch.cuda.

model_id = "openai/whisper-large-v3"

model = AutoModelForSpeechSeq2Seq.from_pre
    model_id, torch_dtype=torch_dtype, low
).to(device)

# Enable static cache and compile the forw
model.generation_config.cache_implementati
model.generation_config.max_new_tokens = 2
model.forward = torch.compile(model.forwar

processor = AutoProcessor.from_pretrained(

pipe = pipeline(
    "automatic-speech-recognition",
    model=model,
    tokenizer=processor.tokenizer,
    feature_extractor=processor.feature_ex
    torch_dtype=torch_dtype,
    device=device,
)

dataset = load_dataset("distil-whisper/lib
sample = dataset[0]["audio"]

# 2 warmup steps
for _ in tqdm(range(2), desc="Warm-up step
    with sdpa_kernel(SDPBackend.MATH):
        result = pipe(sample.copy(), gener

# fast run
with sdpa_kernel(SDPBackend.MATH):
    result = pipe(sample.copy())

print(result["text"])
```

Flash Attention 2

We recommend using Flash-Attention 2 if your GPU supports it and you are not using torch.compile. To

do so, first install Flash Attention:

```
pip install flash-attn --no-build-isolatio
```

Then pass `attn_implementation="flash_attention_2"` to `from_pretrained`:

```
model = AutoModelForSpeechSeq2Seq.from_pre
```

Torch Scale-Product-Attention (SDPA)

If your GPU does not support Flash Attention, we recommend making use of PyTorch scaled dot-product attention (SDPA). This attention implementation is activated **by default** for PyTorch versions 2.1.1 or greater. To check whether you have a compatible PyTorch version, run the following Python code snippet:

```
from transformers.utils import is_torch_sd

print(is_torch_sdpa_available())
```

If the above returns `True`, you have a valid version of PyTorch installed and SDPA is activated by default. If it returns `False`, you need to upgrade your PyTorch version according to the official instructions

Once a valid PyTorch version is installed, SDPA is activated by default. It can also be set explicitly by specifying `attn_implementation="sdpa"` as follows:

```
model = AutoModelForSpeechSeq2Seq.from_pre
```

For more information about how to use the SDPA refer to the Transformers SDPA documentation.

Model details

Whisper is a Transformer based encoder-decoder model, also referred to as a *sequence-to-sequence* model. There are two flavours of Whisper model: English-only and multilingual. The English-only models were trained on the task of English speech recognition. The multilingual models were trained simultaneously on multilingual speech recognition and speech translation. For speech recognition, the model predicts transcriptions in the *same* language as the audio. For speech translation, the model predicts transcriptions to a *different* language to the audio.

Whisper checkpoints come in five configurations of varying model sizes. The smallest four are available as English-only and multilingual. The largest checkpoints are multilingual only. All ten of the pre-trained checkpoints are available on the Hugging Face Hub. The checkpoints are summarised in the following table with links to the models on the Hub:

Size	Parameters	English-only	Multilingual
tiny	39 M	✓	✓
base	74 M	✓	✓
small	244 M	✓	✓
medium	769 M	✓	✓
large	1550 M	x	✓

large-v2	1550 M	x	✓
large-v3	1550 M	x	✓

Fine-Tuning

The pre-trained Whisper model demonstrates a strong ability to generalise to different datasets and domains. However, its predictive capabilities can be improved further for certain languages and tasks through *fine-tuning*. The blog post Fine-Tune Whisper with 🤗 Transformers provides a step-by-step guide to fine-tuning the Whisper model with as little as 5 hours of labelled data.

Evaluated Use

The primary intended users of these models are AI researchers studying robustness, generalization, capabilities, biases, and constraints of the current model. However, Whisper is also potentially quite useful as an ASR solution for developers, especially for English speech recognition. We recognize that once models are released, it is impossible to restrict access to only “intended” uses or to draw reasonable guidelines around what is or is not research.

The models are primarily trained and evaluated on ASR and speech translation to English tasks. They show strong ASR results in ~10 languages. They may exhibit additional capabilities, particularly if fine-tuned on certain tasks like voice activity detection, speaker classification, or speaker diarization but have not been robustly evaluated in these areas. We strongly recommend that users perform robust evaluations of the models in a particular context and domain before deploying them.

In particular, we caution against using Whisper models to transcribe recordings of individuals taken without their consent or purporting to use these models for any kind of subjective classification. We recommend against use in high-risk domains like decision-making contexts, where flaws in accuracy can lead to pronounced flaws in outcomes. The models are intended to transcribe and translate speech, use of the model for classification is not only not evaluated but also not appropriate, particularly to infer human attributes.

Training Data

The large-v3 checkpoint is trained on 1 million hours of weakly labeled audio and 4 million hours of pseudo-labeled audio collected using Whisper large-v2.

As discussed in the accompanying paper, we see that performance on transcription in a given language is directly correlated with the amount of training data we employ in that language.

Performance and Limitations

Our studies show that, over many existing ASR systems, the models exhibit improved robustness to accents, background noise, technical language, as well as zero shot translation from multiple languages into English; and that accuracy on speech recognition and translation is near the state-of-the-art level.

However, because the models are trained in a weakly supervised manner using large-scale noisy data, the

predictions may include texts that are not actually spoken in the audio input (i.e. hallucination). We hypothesize that this happens because, given their general knowledge of language, the models combine trying to predict the next word in audio with trying to transcribe the audio itself.

Our models perform unevenly across languages, and we observe lower accuracy on low-resource and/or low-discoverability languages or languages where we have less training data. The models also exhibit disparate performance on different accents and dialects of particular languages, which may include higher word error rate across speakers of different genders, races, ages, or other demographic criteria. Our full evaluation results are presented in the paper accompanying this release.

In addition, the sequence-to-sequence architecture of the model makes it prone to generating repetitive texts, which can be mitigated to some degree by beam search and temperature scheduling but not perfectly. Further analysis on these limitations are provided in the paper. It is likely that this behavior and hallucinations may be worse on lower-resource and/or lower-discoverability languages.

Broader Implications

We anticipate that Whisper models' transcription capabilities may be used for improving accessibility tools. While Whisper models cannot be used for real-time transcription out of the box – their speed and size suggest that others may be able to build applications on top of them that allow for near-real-time speech recognition and translation. The real value of beneficial applications built on top of

Whisper models suggests that the disparate performance of these models may have real economic implications.

There are also potential dual use concerns that come with releasing Whisper. While we hope the technology will be used primarily for beneficial purposes, making ASR technology more accessible could enable more actors to build capable surveillance technologies or scale up existing surveillance efforts, as the speed and accuracy allow for affordable automatic transcription and translation of large volumes of audio communication. Moreover, these models may have some capabilities to recognize specific individuals out of the box, which in turn presents safety concerns related both to dual use and disparate performance. In practice, we expect that the cost of transcription is not the limiting factor of scaling up surveillance projects.

BibTeX entry and citation info

```
@misc{radford2022whisper,
  doi = {10.48550/ARXIV.2212.04356},
  url = {https://arxiv.org/abs/2212.04356}
  author = {Radford, Alec and Kim, Jong Wo
  title = {Robust Speech Recognition via La
  publisher = {arXiv},
  year = {2022},
  copyright = {arXiv.org perpetual, non-ex
}
```