Jukebox Empty

May 22, 2023

[ ]:
```
!nvidia-smi
```

[ ]:
```
!python --version
```

[ ]:
```
!pip install --upgrade git+https://github.com/craftmine1000/jukebox-saveopt.git
```

[ ]:
```
import jukebox
import torch as t
import librosa
import os
from IPython.display import Audio
from jukebox.make_models import make_vqvae, make_prior, MODELS, make_model
from jukebox.hparams import Hyperparams, setup_hparams
from jukebox.sample import sample_single_window, _sample, \
                           sample_partial_window, upsample, \
                           load_prompts
from jukebox.utils.dist_utils import setup_dist_from_mpi
from jukebox.utils.torch_utils import empty_cache

try:
    if device is not None:
        pass
except NameError:
    rank, local_rank, device = setup_dist_from_mpi()
```

[ ]:
```
model = "5b_lyrics" # or "1b_lyrics"
hps = Hyperparams()
hps.sr = 44100
hps.n_samples = 3 if model=='5b_lyrics' else 8
hps.name = 'Jukebox Samples Mozart noLyrics'
chunk_size = 16 if model=="5b_lyrics" else 32
max_batch_size = 3 if model=="5b_lyrics" else 16
hps.levels = 3
hps.hop_fraction = [.5,.5,.125]

vqvae, *priors = MODELS[model]
vqvae = make_vqvae(setup_hparams(vqvae, dict(sample_length = 1048576)), device)
```

```
top_prior = make_prior(setup_hparams(priors[-1], dict()), vqvae, device)
```

[ ]:
```
sample_length_in_seconds = 30          # Full length of musical sample to␣
 ↪generate - we find songs in the 1 to 4 minute
                                       # range work well, with generation time␣
 ↪proportional to sample length.
                                       # This total length affects how quickly␣
 ↪the model
                                       # progresses through lyrics (model also␣
 ↪generates differently
                                       # depending on if it thinks it's in the␣
 ↪beginning, middle, or end of sample)

hps.sample_length = (int(sample_length_in_seconds*hps.sr)//top_prior.
 ↪raw_to_tokens)*top_prior.raw_to_tokens
assert hps.sample_length >= top_prior.n_ctx*top_prior.raw_to_tokens, f'Please␣
 ↪choose a larger sampling rate'
```

[ ]:
```
metas = [dict(artist = "wolfgang amadeus mozart",
            genre = "classical",
            total_length = hps.sample_length,
            offset = 0,
            lyrics = """
            """,
            ),
          ] * hps.n_samples
labels = [None, None, top_prior.labeller.get_batch_labels(metas, 'cuda')]
```

[ ]:
```
sampling_temperature = .98

lower_batch_size = 16
max_batch_size = 3 if model == "5b_lyrics" else 16
lower_level_chunk_size = 32
chunk_size = 16 if model == "5b_lyrics" else 32
sampling_kwargs = [dict(temp=.99, fp16=True, max_batch_size=lower_batch_size,
                        chunk_size=lower_level_chunk_size),
                    dict(temp=0.99, fp16=True, max_batch_size=lower_batch_size,
                         chunk_size=lower_level_chunk_size),
                    dict(temp=sampling_temperature, fp16=True,
                         max_batch_size=max_batch_size, chunk_size=chunk_size)]
```

[ ]:
```
zs = [t.zeros(hps.n_samples,0,dtype=t.long, device='cuda') for _ in␣
 ↪range(len(priors))]
zs = _sample(zs, labels, sampling_kwargs, [None, None, top_prior], [2], hps)
```

[ ]:
```
Audio(f'{hps.name}/level_2/item_0.wav')
```

[ ]:
```
Audio(f'{hps.name}/level_2/item_1.wav')
```

[ ]:
```
Audio(f'{hps.name}/level_2/item_2.wav')
```

[ ]:
```
if True:
    del top_prior
    empty_cache()
    top_prior=None
upsamplers = [make_prior(setup_hparams(prior, dict()), vqvae, 'cpu') for prior␣
 ↪in priors[:-1]]
labels[:2] = [prior.labeller.get_batch_labels(metas, 'cuda') for prior in␣
 ↪upsamplers]
```

[ ]:
```
zs = upsample(zs, labels, sampling_kwargs, [*upsamplers, top_prior], hps)
```

[ ]:
```
Audio(f'{hps.name}/level_0/item_0.wav')
```

[ ]:
```
Audio(f'{hps.name}/level_0/item_1.wav')
```

[ ]:
```
Audio(f'{hps.name}/level_0/item_2.wav')
```

[ ]:
```

```