PAT 498/598: Special Topics: Generative AI for Music and Audio Creation (Fall 2024)

Assignment 3: Unconditional Music/Audio Generation

Due at 11:59pm ET on November 18

1 Unconditional Music/Audio Generation

In this **open-ended** assignment, you can choose to work on either **unconditional symbolic music generation** or **unconditional music/audio synthesis**. You may use **any dataset of your choice** and **any machine learning model of your choice**. You may also use your own collection of data, and it's fine to use existing codebase, but you need to provide proper citation/reference. The only requirement is that the model needs to be *unconditional*, i.e., generating music or audio from scratch.

1.1 Option 1: Unconditional Symbolic Music Generation

Here are some good datasets that you might want to use:

- Nottingham: 1000+ folk songs in ABC format
- JSB Chorales: 382 Bach chorales in NPY format
- POP 909: 909 C-pop songs in MIDI format, with the melody and accompaniment tracks
- MAESTRO: 200 hours of expressive piano performances in MIDI format
- Groove MIDI: 13.6 hours of expressive drum performances in MIDI format

You will likely want to set a small temporal resolution to start with, e.g., a temporal resolution of 4 time steps per quarter note (i.e., allowing anything greater than a 16th note) is a good starting point for nonexpressive symbolic music datasets. Also, you might find mido, pretty_midi and MusPy useful for processing MIDI files, and MidiTok for tokenizing MIDI files.

Example system A simple option is to implement an n-gram-like model using multilayer perceptrons (i.e., fully-connected feedforward network) or a convolutional neural network (CNN). You will implement a model that learns to predict the next word given the previous $n-1$ words, i.e., learning the mapping $(x_{t-n+1}, x_{t-n+2}, \ldots, x_{t-1}) \rightarrow x_t$. You can then compare the performance of the model for different n, e.g., $n = 1, 10, 100, 1000$, depending on your computing budget.

1.2 Option 2: Unconditional Music/Audio Synthesis

Here are some good datasets that you might want to use:

- NSynth: 305,979 4-sec recordings of single-shot music notes
- MAESTRO: 200 hours of expressive piano performances
- Bach Violin: 6.5 hours of recordings of Bach's six sonatas and partitas for violin
- DCASE Foley: 4,850 audio samples in 7 classes
- ESC-50: 2000 environmental audio recordings in 50 classes, with 40 samples each

You will likely want to downsample/resample the audio files to 8kHz or 16kHz for simplicity if the data come in a higher sampling rate, which can be done using ffmpeg or librosa. If you want to implement a frequency-domain synthesis model, you might want to use the pretrained Hifi-GAN models (note that you need to use their code to compute the mel spectrograms as your training targets if you're using Hifi-GAN as the vocoder).

Example system A not-too-challenging (though not easy either) option is to implement a diffusion model that generates audio spectrograms. You can use and adapt the nicely-written codebase for improved-diffusion. Specifically, you will need to modify the data loader so that it can read mel spectrograms rather than images. You will want to rewrite `load_data()` into, assuming you're storing the mel spectrograms as NPY files:

```
def load_data(
    *, data_dir, batch_size, image_size, class_cond=False,
    deterministic=False
):
    data_dir = pathlib.Path(data_dir)

    # Load filenames
    all_files = list(data_dir.rglob("*.npy"))

    ...
```

You will also want to rewrite `ImageDataset.__getitem__()` into:

```
MELSPEC_MIN = -12.0
MELSPEC_MAX = 3.0

def __getitem__(self, idx):
    # Load the mel spectrogram
    melspec = np.load(self.local_images[idx])

    # Get a random slice
    start = np.random.randint(melspec.shape[1] - self.resolution)
    data = melspec[np.newaxis, :, start : start + self.resolution]
    data = data.astype(np.float32)
    data = 2 * (data - MELSPEC_MIN) / (MELSPEC_MAX - MELSPEC_MIN) - 1
```

Also, you will need to change this line to `input_channels=1` as mel spectrograms have only one channel.

1.3 Task

Please complete the followings:

- Report the training, validation and test losses
- Show some example generated music or audio
- Conduct at least one experiment on any aspect, e.g., datasets, network architecture, training configuration, latent space interpolation, zero-shot generalization

You work will ***not* be graded by the performance of your final model,** but rather **the amount of work you put in exploring different techniques and analyzing the experimental results**. Thus, please also report any negative results that you find not working and discuss why it is not working.

Please submit your code and a report that summarizes the experimental design and your findings. The report should be no more than 2 pages, excluding references, and you may use any template you like. You will receive zero credit if the code is missing.

1.4 Rubrics

- Model implementation (10pt)
- Experimental results (5pt)
- Analysis and discussions (5pt)

2 Submission

- All assignments must be completed on your own. You are welcome to exchange ideas with your peers, but this should be in the form of concepts and discussion, not in the form of writing and code.
- Please provide proper citations/references for any external resources you use in your writing and code.
- Please submit your work to Gradescope.
- Late submissions will be deducted by **3 points per day**.