Muzip: Music Compression Using Neural Networks
Michael Arruza-Cruz, David Morales, Divya Saini

Background
Music files are traditionally relatively large and difficult to process, so compression methods are often used to reduce the file size. MP3 files are one such compressed representation that results in lost data. Our project seeks to use convolutional neural networks (CNNs) to convert audio files into a compressed representation that reduces file size, while still allowing reconstruction of the original file. Unlike prevailing literature that tends to center around spectrogram representations of an audio file, we attempt to investigate whether it is possible to garner higher quality and better compressed output on raw audio signals.

The aim of this music compression is to reduce the redundancy in a song in order to be able to store, transmit, and search for the song at low bit rates.

Dataset
We use the FMA music analysis dataset, which provides 917 GB of audio from 106,574 tracks from 16,341 artists and 14,854 albums of 161 genres. Along with this audio, the dataset provides pre-computed features with track- and user-level metadata, tags, and free-form text.

For rapid iteration, we use the smaller version of the dataset containing 8,000 30-second snippets taken from a multitude of songs across 8 balanced genres, in MP3 format.

Approach

![Diagram of compression and decompression process]

Compress a 30-second chunk.

Output sample chunk.

1D Convolution
50 filters, size 1
Stride 1
Tanh

1D Convolution
25 filters, size 1
Stride 1
Tanh

1D Convolution
50 filters, size 1
Stride 1
Tanh

1D Convolution
100 filters, size 1
Stride 1
Tanh

1D Convolution
50 filters, size 1
Stride 1
Tanh

1D Convolution
25 filters, size 1
Stride 1
Tanh

1D Convolution
100 filters, size 1
Stride 1
Tanh

1D Convolution
50 filters, size 1
Stride 1
Tanh

Figure 2. Model Design. We use a convolutional neural network using multiple one-dimensional convolutions over the inputs. Grey boxes show compression layers and pink show decompression layers.

Experiments & Results

![Graphs showing MSE and MAE over batches during training]

Figure 3. MSE over batches during training.

Figure 4. MAE over batches during training.

![Graph showing example 6-second output with actual sample and decompressed sample]

Figure 5. Example 6-second output. Top: Actual sample. Bottom: Decompressed sample.

Using a standard laptop computer and the Keras framework, compression and decompression of a 30-second sample takes 64 milliseconds.

Table 1. Model performance. Performance metrics of our two best music compression and decompression models on 1,231 examples. Both provide 4x compression.

<table>
<thead>
<tr>
<th>Model</th>
<th>MSE</th>
<th>MAE</th>
<th>R²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder:</td>
<td>0.0313</td>
<td>0.0612</td>
<td>0.940</td>
</tr>
<tr>
<td>Decoder:</td>
<td>0.0313</td>
<td>0.0612</td>
<td>0.940</td>
</tr>
</tbody>
</table>

Conclusion & Future Work

- Convolutional Neural Networks are able to compress music samples to a fourth of original size.
- Decompressed samples are noisy, but preserve most of the underlying music sample.
- Taking genre into account may result in better decompression (network will learn to account for similarities in genre).
- Evaluating semantic information captured in compressed representations may reveal potential use for compressed samples.

References
[1] https://github.com/whale-conservation