Classification and Neural Audio Synthesis on a Raspberry Pi: a Practice-based-Research Enquiry

Mark Hanslip
Department of Music, University of York, UK
mwh512@york.ac.uk

The Raspberry Pi is a popular choice for creative audio applications due to its combination of low cost, computational power, and community support. Its utility as a platform for creative machine learning-based audio applications remains under-explored though. My talk is a progress report on my efforts to port AI-driven interactive systems featuring neural audio classification and generation to the Raspberry Pi. The goals of this project are to answer the basic question of feasibility and, assuming success, to provide a template for other practitioners and to generate tacit knowledge of its usefulness.

The format of the talk is as follows:

1) Introduction to me and my background as a musician and researcher (~1 min)
2) Overview of system design and development (~3 mins)
3) Audition of extracts from some of my recordings made using these systems (~2 mins)
4) Report on progress made towards porting this work to Raspberry Pi (~3 mins)
5) Summary of learnings and future work (~1 min)

System development and design

I first created two recordings capturing differing aspects of my practice as an improvising saxophonist. These can be broadly characterised as melodic improvisation played mostly with a conventional technique, and more timbre-oriented playing using extended techniques such as multiphonics. These recordings were converted to constant-Q spectrograms; I then trained CNN image classifiers on the resulting datasets. I also trained WaveGAN models on each recording. Data augmentations using pitch shift and waveform polarity inversion were applied to achieve appropriate quantities of data for training WaveGANs and avoid overfitting.

The overall system design is such that, upon receipt of an audio input, the classifier infers which dataset the input most closely resembles. A corresponding WaveGAN generator can then be prompted with a noise vector, after which generated samples are concatenated and played back. In other pieces, pre-assigned audio samples are simply triggered. Non-playing inputs are filtered out using loudness analysis. All code is written in Python, with component reusability prioritised by design. Machine learning components are written in PyTorch. Development of the classification and real-time interaction aspects took place on a laptop; WaveGAN models were trained in Google Colab.

Rasperry Pi Implementation

I've since set about porting these systems to a Raspberry Pi 4B (4GB RAM Model). Audio is handled by the PiSound interface which provides excellent i/o functionality and pleasing form factor. Because PiSound is designed for 32-bit OS only, PyTorch had to be compiled from source. Finding compatible versions of Numba, LLVM, Librosa and Scikit-Learn for the Pi was also challenging. All machine learning components work on the Pi though – I can pass in an audio file, classify it correctly and generate from a trained WaveGAN in response. Multi-threaded audio playback also works, allowing for overlapping sample playback.

At time of writing, a loud hum on the PiSound's line-level input is delaying development and needs to be addressed. WaveGAN samples generated from models trained using the author's Tensorflow implementation are higher quality than my PyTorch version, so installing Tensorflow on the Pi is a priority, as is adding a method for toggling between patches.

Relevance of Talk

- It describes a strategy for transferring laptop-designed programs containing HPC-trained elements to embedded hardware.
- It brings neural audio synthesis to embedded devices.
- It presents for practitioners a functional combination of technologies for real-time-optimized audio and machine learning in an embedded context.
- It considers accessibility through the inclusion of an installation guide. I would share system code via GitHub ahead of the talk.