ECS7013: Deep Learning for Audio and Music

Lab 5 – Autoencoder Representations

In this lab, we will experiment with variational autoencoder (VAE).

1. Take the PyTorch VAE (variational autoencoder) code example *https://github.com/pytorch/examples/tree/master/vae* and run it.

You may need to download the MNIST digits dataset manually (if you get a 403 error on doing it automatically): *http://yann.lecun.com/exdb/mnist/*
Note that the code expects the MNIST raw data to be located at *../data/MNIST/raw/*

The code example will generate image output in a subfolder called "results". Inspect the results briefly.

2. Modify the code so that instead of using only fully-connected layers, it uses convolution (in the encoder) and deconvolution (in the decoder).

Replace EVERY fully-connected layer with a convolutional layer, having filters of size 5. Do not use any downsampling or upsampling. For each conv layer in the encoding, double the number of channels at each step (e.g. the first one maps from 1 channel to 2 channels), and in the decoder, halve the number of channels at each step.

For simplicity in this exercise, we keep the time and frequency axes the same size all the way through the network (no downsampling/upsampling). The latent representation "z" will thus be a distribution over a time-frequency tensor too: the mean and stdev will both be time-frequency tensors.

Implementation tips:

- Note that *fc21* and *fc22* are applied in parallel not in series: one calculates the mean, and one the stdev, of the variational latent representation.
- The original code flattens the 2D image data into 1D using "*x.view(-1, 784)*" in the *forward()* function. This collapses the x and y dimensions. You will need to remove this flattening in order to use convolutions.

Run it. Inspect the results.

- In what way are they different from the original results?
- Why?
- Which of the two latent representations (the original, or your modified version) would be better for representing audio scenes such as 10-second wildlife recordings?
- How could we improve the latent representation for audio purposes?

3. Create a PyTorch Dataset class to represent an audio dataset (e.g. the data we've used before, or one of your choice).
https://pytorch.org/tutorials/beginner/data_loading_tutorial.html
Tips:
(a) The raw data are waveforms, but we will be using spectrograms. Choose a spectrogram representation to use in this task (free choice).
(b) Remember that the data are not inherently normalised. Calculate a single global mean and standard deviation for the training data, and then modify your class to normalise data automatically as they are loaded. (For the purpose of today's task, it's OK to hard-code the mean and standard deviation, for simplicity.)

--- (If sufficient time…) ---

4. Add maxpooling (downsampling) in the encoding layers, and upsampling in the decoding layers (use the *Upsample* class).

Get the system training at least once. Inspect the spectrograms produced.

Tip: for inspiration on implementation, see e.g. this u-net code example:
https://github.com/milesial/Pytorch-UNet/tree/master/unet

5. Modify the variational autoencoder code example to use your dataset class.

Get the system training at least once. Inspect the spectrograms produced.

6. Also resynthesise the generated spectrograms back to audio files. You can use the resynthesis code from the lecture notebook in Week 6. Listen to the examples and diagnose how to improve the VAE method for audio.