Week 8 Report

Kieran Grant - 2357351G

What I've done this week

- Not had much time to spend on the project this week - should be back to normal from the middle of Week 9.
- Met with Krzysztof after John's introduction.
  - Had a very good chat about our respective projects and their overlaps/differences.
  - It was useful to talk through some of the ideas we had for our projects and the problems we had ran into.
  - Going to keep in touch and meet up again at some point before the end of semester.
- Signed up to both Weights and Biases and the Andrew Ng newsletter as discussed last week.
  - I will try using W&B for the MNIST audio tutorial which I'll start next week.
- Read and summarised *One-Shot Parameteric Audio Production Style Transfer with Application to Frequency Equalization (2020)*
  - Short summary and takeaways below.
- Created architecture diagrams for two of the main ideas I had for project direction.
  - One idea is to allow for arbitrary effects to be used in the style transfer architecture.
  - The other is to allow exploration of parameter values that are similar to those found via style matching in a low dimensional space.
  - Diagrams are at the end of the report.

Questions

- None this week.

Plan for next week

- Organise papers read/to be read into groups for interim report literature review.
- Prepare a mental plan of where I'd like to be by Christmas.
  - Plan of what to do in remaining weeks to achieve it.
- Start Neural Nets for Generating Audio series.
  - Use W&B to track model training.
- Try to get Style Transfer code working locally.

Current state of project

- Have a decent idea of the state of the field.
- Need to start actually implementing some models and create a plan for organising project as well as data generation, how evaluation should be performed.

One-Shot Parameteric Audio Production Style Transfer with Application to Frequency Equalization (2020)

Short Summary

The model proposed in this paper predicts the parameter values required for a DAFX in order to match the style of an input signal to a target signal. In particular, the predicted values are discrete quantized values of a 4-band parametric equaliser.

The main architecture first computes the Mel-spectrogram of the input and reference signals using a 20ms window, 3ms hop size and 128 Mel-bands. A feature extractor (CNN) with shared weights created embeddings from these spectrograms of size $1 \times F$ where F is the feature size. These two embeddings are concatenated into a matrix of size $2 \times F$ and a convolution network reduces these back into a $1 \times F$ feature vector.

The final module in the architecture is the parameter prediction. The module consists of two feed-forward neural networks. The first takes the $1 \times F$ output embedding above as input and outputs a representation vector of length $1 \times (P * D)$ where P is the number of parameters and D is the number of discrete values. The second network reshapes this into a $P \times D$ matrix and performs a soft-max to find the highest probability discrete parameter value for each parameter.

Data generation is performed using a similar method to [1] where for M iterations: two random audio signals are taken from an audio dataset (speech recordings for example) called X and Y. An identical 'scene augmentation' is applied to both (background noise, reverb etc). Each signal is then divided into N overlapping segments, which are shuffled. For each segment a random parameter setting is applied to the i'th Y audio segment (Y_i). The X_i, Y_i segments and parameter settings are then stored as a tuple in the generated dataset.

For the experiment, the above method is applied to speech recordings (DAPS dataset) using a 4-band parametric EQ. Each parameter value is discretized to 21 values. The authors use a 50/10/40 train/validation/test split of the ~5 hours of speech audio. The actual number of data generations steps above are $M = \{100, 80, 20\}$ respectively. The loss is computed between the estimated and actual parameter settings using absolute error. For testing, the MAE and standard deviation are reported.

The authors compare the proposed model using a baseline from [2] which uses both linearly valued Mel-Spectrograms to decibel valued (log) Spectrograms. The results show that using decibel scaling performs better than linear scaling, but the proposed method (with learned comparison) performs best of the three, with the most noticeable improvement to the prediction of high frequency bands.

Takeaways

- The authors don't make it clear whether the sampled audio signal from the database is the same audio signal or two different audio signals. In [1] the same audio signal is used and I think this makes the most sense.
- The authors also mention that they discretized the values as 'they found [it] more convenient for modelling generic DAFX' - without further explanation. I would have assumed that outputting a normalised parameter value would be more flexible and also lower the complexity of the model.
- Overall I think [1] is a direct evolution of this method which is more generalisable and also uses loss in the audio domain rather than the parameter domain.

Architecture Diagrams

Using Arbitrary Audio Effects

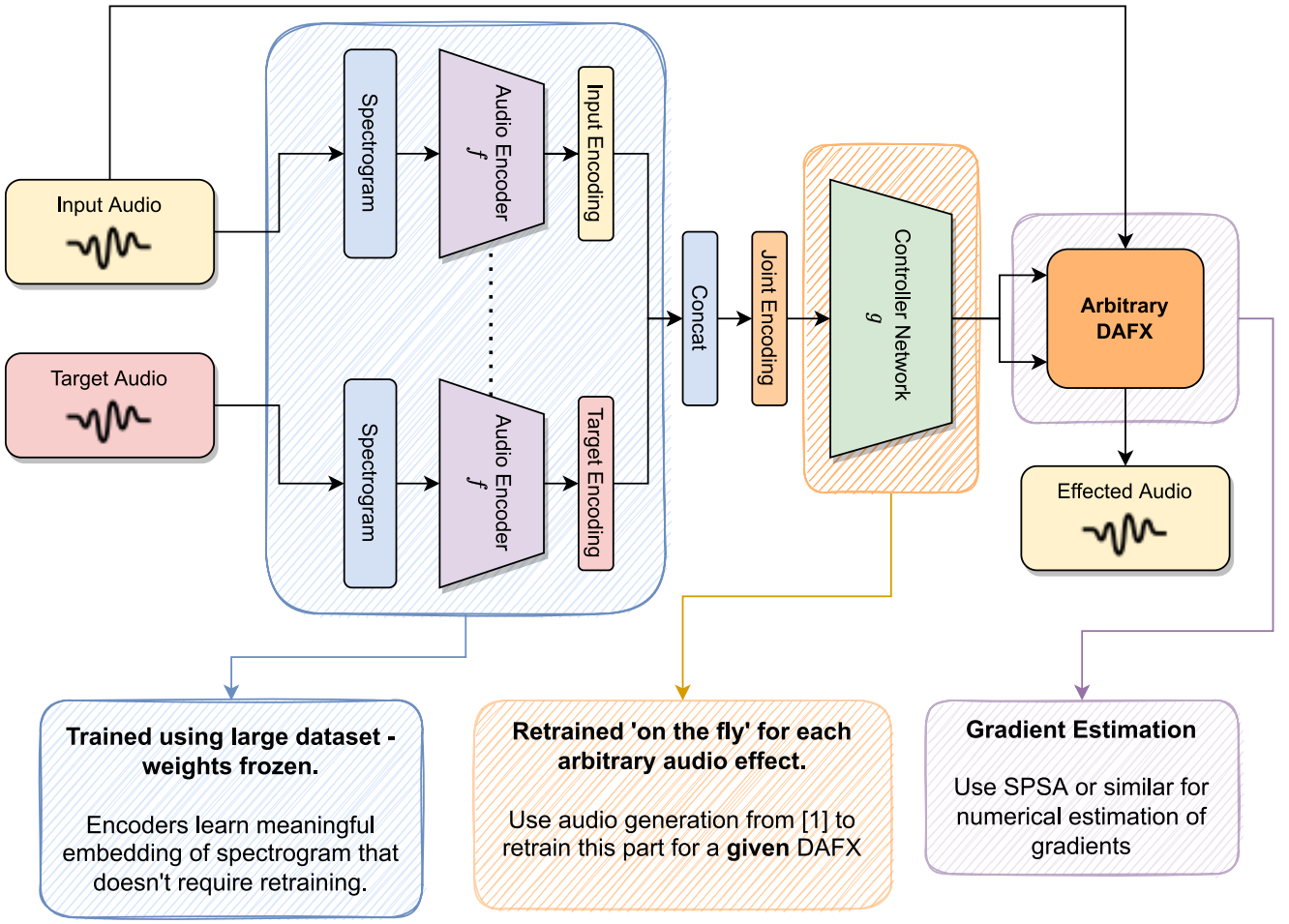

Figure 1: Implementing arbitrary DAFX in the architecture from [1]. The encoder part is 'pre-trained' to create meaningful representations of the audio signals. The Controller network is retrained for each new effect (and model weights stored) which can be used for multiple style transfers.

Exploring Latent Space

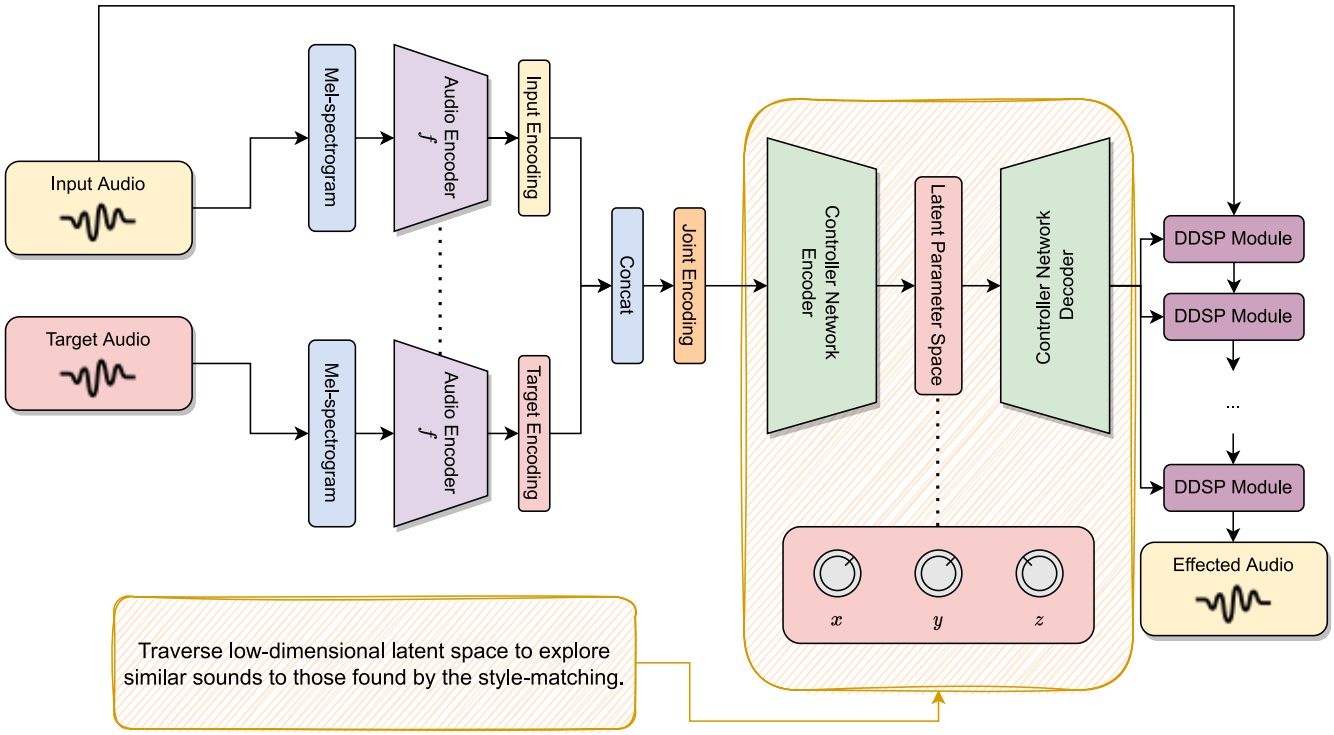

Figure 2: Adding a bottleneck in the controller network to allow for a low-level representation of the parameter space which can be explored to find similar sounds to those found by the style transfer.

References

[1] Steinmetz, Christian J., Nicholas J. Bryan, and Joshua D. Reiss. "Style Transfer of Audio Effects with Differentiable Signal Processing." arXiv preprint arXiv:2207.08759 (2022).

[2] D. Sheng and G. Fazekas, "A feature learning Siamese model for intelligent control of the dynamic range compressor," in 2019 International Joint Conference on Neural Networks (IJCNN), July 2019, pp. 1–8.