NEURAL MUSIC INSTRUMENT CLONING FROM FEW SAMPLES

Nicolas Jonason and Bob L. T. Sturm

Division of Speech, Music and Hearing
KTH Royal Institute of Technology
Stockholm, Sweden
njona@kth.se | bobs@kth.se

ABSTRACT

Neural music instrument cloning is an application of deep neural networks for imitating the timbre of a particular music instrument recording with a trained neural network. One can create such clones using an approach such as DDSP [1], which has been shown to achieve good synthesis quality for several instrument types [2]. However, this approach needs about ten minutes of audio data from the instrument of interest (target recording audio). In this work, we modify the DDSP architecture and apply transfer learning techniques used in speech voice cloning [3] to significantly reduce the amount of target recording audio required. We compare various cloning approaches and architectures across durations of target recording audio, ranging from four to 256 seconds. We demonstrate editing of loudness and pitch as well as timbre transfer from only 16 seconds of target recording audio. Our code is available online[1] as well as many audio examples.[2]

1. INTRODUCTION

Neural music instrument cloning reproduces the timbre of a given music instrument recording (target recording audio). For instance, one can clone the recording of a specific saxophone playing in an acoustic environment and synthesise audio having similar timbre. A clone can be used to modify attributes of a recording, like adjusting pitch or loudness; or to transfer timbres between recordings, whether the same type of instrument or not. One recent approach to creating neural music instrument clones of harmonic instruments is provided by differentiable digital signal processing (DDSP) [1], although Engel et al. do not use the term "clone." Neural music instrument cloning is closely related to neural voice cloning [3], where the identifiable characteristics of a given speaker are extracted from example recordings and can then be used to synthesise new speech from that speaker.

The synthesis quality of a clone, whether for voice or music instrument synthesis, depends on the amount and variety of training data available. The DDSP approach can imitate the timbre of a recording using about ten minutes of target recording data [1, 2]. Our work significantly reduces that amount, e.g., to the order of tens of seconds, by modifying the DDSP architecture and using transfer learning. Our main contributions are:

- We apply transfer learning techniques from neural voice cloning [3] to DDSP;
- We introduce a novel trainable reverb design whose constrained nature makes it better for fitting on small amounts of data;
- We show how including the confidence of the pitch estimation as conditioning to a DDSP model helps decrease synthesis artefacts when training neural music instrument clones on little data.

In the next section, we briefly review neural music audio synthesis and speech voice cloning. Section 3 presents our model architecture. In Sec. 4 we present experiments comparing various cloning approaches and model architectures. Section 5 demonstrates some applications. Section 6 discusses our results, presenting several future avenues of research.

2. RELATED WORK

Neural music instrument synthesis Neural audio synthesis applies neural networks to the synthesis of audio, including music instrument sounds [1, 4–11] Neural synthesis models often represent recording-specific data with latent encodings that are produced by inputting audio data into a trained encoder [1, 4, 8–10]. The DDSP algorithm [1] combines spectral modelling synthesis [12] with convolution reverberation to synthesise particularly realistic instrument timbres that can be controlled in pitch and loudness. We use the term *loudness* to refer to a measurable characteristic computed from an A-Weighting of the power spectrum [11]. DDSP requires about ten minutes of audio training data of a solo pitched instrument to achieve good synthesis quality [1, 2].

Neural voice cloning Neural voice cloning applies neural networks to synthesise the voices of specific speakers from small amounts of audio [3, 13, 14]. Arik et al. [3] compare multiple approaches for cloning voices from minutes to seconds of target speaker audio. These approaches start with a pre-training phase where they train a multi-speaker model on many speakers. For each speaker, a speaker-specific embedding is trained alongside the multi-speaker model. Once the multi-speaker model is trained they proceed with a cloning phase in which voice cloning is performed using speaker adaptation or speaker encoding. Speaker adaptation fine-tunes a multi-speaker model on a few samples of the target speaker while speaker encoding trains an encoder model to compute the necessary adaptation in a single inference step. Arik et al. [3] compare speaker adaptation by tuning only the embedding or by tuning the whole model and find that whole-model adaptation gives the best results in terms of naturalness and target-speaker similarity. The authors also find that speaker encoding

[1] https://github.com/erl-j/neural-instrument-cloning

[2] https://erl-j.github.io/neural-music-instrument-cloning-web-supplement

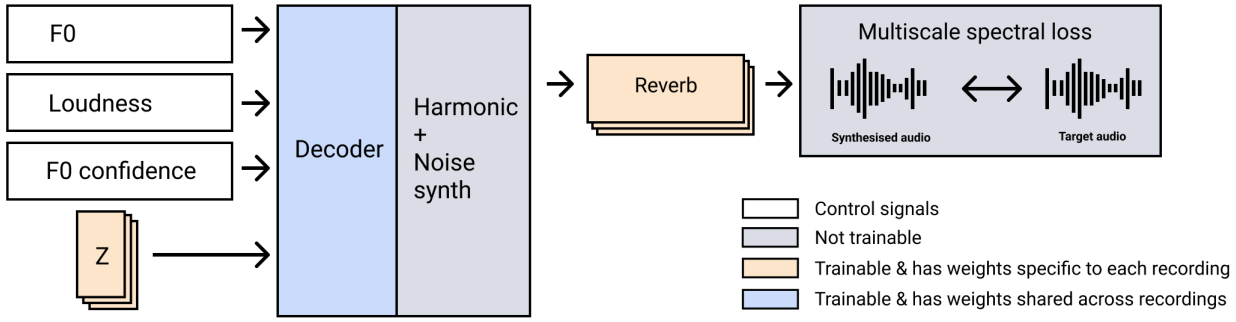

Figure 1: *Our modified DDSP architecture. Three control signals are fed to the decoder along with a trainable latent embedding Z. The neural decoder in turn controls a harmonic + noise synthesiser. A trainable reverb is applied. The decoder is trained across multiple recordings. Separate instances Z and reverb are trained for each recording.*

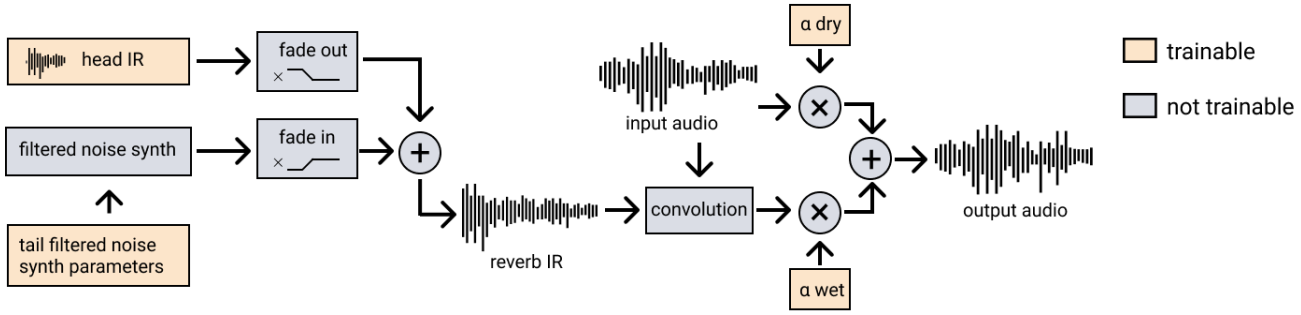

Figure 2: *Diagram of the proposed two-part reverb design. A short trainable IR is used to model the early reflections and a filtered white noise with trainable filter magnitude contours is used to model the late reverberation. These two components are combined with a cross-fade. Two trainable scalars – α_{dry}, α_{wet} – are used to scale the amplitude of the dry signal and the wet signal, respectively.*

is many times faster than speaker adaptation but generates speech that is less natural and less similar to the target speaker. These are both instances of *transfer learning* [15], the technique of leveraging a model trained on a particular domain for a different domain.

3. MODEL ARCHITECTURE

Overview Figure 1 gives an overview of our modified DDSP [1] architecture. We make the following modifications:

- We provide the decoder with a trainable recording-specific embedding notated Z. Note that we do not use an encoder to produce Z, instead, we train a separate instance of Z for each recording jointly with the decoder as done by Arik et al. [3]. This implicitly enforces a constraint on Z to be constant throughout each recording regardless of what is being played on the instrument.
- We introduce a new reverb design, detailed in section 3, motivated by acoustic models of natural impulse responses (IR). This new reverb design is used during cloning (see 4.3.2). Like with Z, the reverb is specific to each recording.
- We condition the decoder on the confidence of the F0 estimate given by CREPE [16], as well as F0 and loudness contours. This is intended to help the model determine whether the sound to be produced is pitched or not (exhales, inhales, key presses, etc.).

Details Our model synthesises audio at a sampling rate of 48 kHz. We adjust the loss, harmonic + noise synthesiser and decoder used by DDSP, mainly to accommodate for this sampling rate [1]. The multi-scale spectral loss uses the following FFT-sizes: 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384. The harmonic synthesiser has 192 harmonics, and the filtered noise synthesiser has 192 magnitudes. The recording-specific embedding Z has 512 dimensions. During the forward pass, Z is repeated over the time dimension so that it has the same frame rate as F0, loudness, and F0-confidence contours (250 frames per second). We apply a tanh non-linearity to Z before the decoder. As in DDSP [1], the decoder uses a bidirectional 512-channel GRU. Each input including Z has its own 3-layer fully connected stack with 512 neurons per hidden layer applied before the GRU. A final 3-layer fully connected stack is applied after the GRU. Finally, the output of the decoder is stacked with the F0 contour to produce the synthesiser parameters. In total, our decoder has 12,548,993 parameters.

Reverb design Figure 2 illustrates our new reverb design. Our early attempts at training models from little target training data using an unconstrained reverb, as done in [1], suffered from excessive reverberation (see 4.3.2). We also experimented with using filtered white noise as the IR [2, 17] but found that the results sounded unnatural (see 4.3.2). This led us to implement a new two-part reverb design motivated by acoustics.

Prior work in acoustics shows how IR of natural environments

Table 1: *Summary of the Solos Wind Clean (SWC) dataset*

instrument type	pre-training set #excerpts	#videos	total duration	mean excerpt duration	cloning set #excerpts	#videos	total duration	mean excerpt duration
bassoon	10	8	0:42:46	0:04:16	5	4	0:18:09	0:03:37
clarinet	34	23	0:59:40	0:01:45	4	4	0:12:55	0:03:13
flute	30	29	1:55:03	0:03:50	4	4	0:09:33	0:02:23
horn	21	16	1:19:54	0:03:48	2	2	0:08:52	0:04:26
oboe	28	21	1:17:09	0:02:45	7	4	0:24:13	0:03:27
trumpet	16	16	0:16:44	0:01:02	2	2	0:05:13	0:02:36
trombone	29	20	1:17:27	0:02:40	1	1	0:01:53	0:01:53
tuba	11	11	0:27:59	0:02:32	5	3	0:09:58	0:01:59
saxophone	27	20	0:51:00	0:01:53	2	2	0:04:35	0:02:17
total	206	164	9:07:42	0:02:43	32	26	1:35:21	0:02:52

consist of early reflections and dense reverberation [18]. Our design mimics this structure by modelling the IR as a superposition of two parts: a free portion to model early reflections, and a portion of filtered white noise to model dense reverberation. We linearly cross-fade between the two parts between 100 to 200 ms, to create an IR with a total duration of one second. Furthermore, our reverb module has two trainable scalars corresponding to the amplitudes of the dry and wet signals: α_{dry} and α_{wet}. There are 9600 parameters for the early reflections part (200 ms), and 25000 parameters for modelling the dense reverberation, representing 250 frames of 100 filter-band amplitudes.

4. EXPERIMENTS

We now perform experiments to compare various approaches for training neural music instrument clones, as well as to evaluate the impacts of our modifications to the DDSP architecture. We evaluate these different configurations by looking at the multi-scale spectral reconstruction loss of test excerpts. We also perform informal listening evaluation, paying particular attention to the naturalness of the synthesis as well as its timbral similarity to the target [3, 19]. We invite the reader to follow along with the accompanying audio examples in the experiments section of the web supplement. We describe the datasets we use before presenting our experiments.

4.1. Datasets

4.1.1. Solos Wind Clean (SWC)

Our experiments use a subset of *Solos* [20], a dataset containing 755 videos of 13 different instrument types collected from YouTube using queries like "audition" and "solo". Since DDSP only works with monophonic recordings, we keep only the *Solos* videos that are of strictly monophonic instruments. We manually clean up this data by listening and removing video sections having silences, loud ambient noise relative to the instrument sound, exotic playing styles (e.g. flute beatboxing), mislabelling of instrument type, stomping, metronome sounds, heavy compression artefacts, clipping, and noise-reduction artefacts. Removing these sections provides partition points from which we extract excepts while keeping track of their originating video. The reason for this is that some recordings from *Solos* come from separate takes spliced together. This can cause problems if the takes are recorded in different conditions.

We partition the cleaned data into *pre-training* and *cloning* sets using the following procedure. For each instrument type, we randomly sample four videos from those contributing at least one excerpt of duration exceeding 64 seconds. These videos are assigned to an initial cloning set, and the remaining videos are assigned to the pre-training set. To avoid test data leakage, we retrieve the YouTube channel ID of each video using the YouTube API and remove videos from the initial cloning set with a channel whose videos also appear in the pre-training set. The reason for this is that we do not want recordings of the same specific instrument and recording condition to appear across sets. We designate the resulting dataset as *Solos Wind Clean (SWC)*.

4.1.2. Saxophone-extra

To study the effectiveness of various cloning approaches as a function of training data size we find an audition video on YouTube with a suitable duration, i.e., providing at least 256 seconds of suitable training material, and at least 32 seconds for testing. We query YouTube with "saxophone audition" with a duration filter of 4-20 minutes, and then select the first video where the position of the player is consistent throughout, and most of it is instrument performance. We manually trim speech and long silences, leaving audio data of duration 08:29. We call this single recording dataset *saxophone-extra*. We use the first 06:40 as training data (*saxophone-extra-train*), and the final 01:48 as test data (*saxophone-extra-test*).

4.2. Comparing training approaches

We now compare approaches for training clones, first when the instrument type (saxophone) appears in pre-training, and then when it does not appear.

4.2.1. Cloning an instrument of a type seen during pre-training

Pre-training phase We first pre-train a model without constraints on the IR using the saxophone excerpts from *SWC-pre-train- saxophone*. This corresponds to 27 excerpts from 20 different videos, totalling 51 minutes. The mean excerpt duration is 1 minute and 53 seconds. The model is pre-trained using the ADAM optimizer with a starting learning rate of 1×10^{-4} and a learning rate decay occurring every 10k steps with a rate of 0.98 [1]. All training data is windowed into four-second windows with a hop-size of one second. The batch size for pre-training is six. We train for 380k

Table 2: *Hyper-parameters used during cloning. The number of cloning epochs is shown for each cloning approach across different training data sizes along with approximate training time using a single NVIDIA GeForce RTX 2080 Ti in parentheses. Learning rate of* 3×10^{-5} *is used for init-whole and sax/nosax-whole and* 3×10^{-3} *is used for sax/nosax-parts.*

training data size (seconds)	4	8	16	32	64	128	256
init-whole, epochs	2900 (5h)	1000 (1h40)	1000 (3h)	1000 (6h)	1000 (12h)	300 (6h30)	300 (13h)
sax/nosax-whole, epochs	100 (5 min)	100 (10 min)	100 (20 min)	100 (47 min)	100 (1h45)	100 (3h)	100 (4h)
sax/nosax-parts, epochs	300 (15 min)	300 (25 min)	300 (50 min)	100 (50 min)	100 (1h35)	100 (3h)	100 (5h)

steps (around 36 hours on three NVIDIA GeForce RTX 2080 Ti) and find an average training loss of 7.553 over the last 1k steps.

Cloning phase We perform cloning with three approaches across seven different training data sizes from *saxophone-extra-train*: 4, 8, 16, 32, 64, 128, and 256 seconds. We extract all training data starting from the beginning of *saxophone-extra-train*. As in the pre-training phase, all training data is windowed into four-second windows with a hop-size of one second. During the cloning phase, we change the reverb design to the two-part design. The three cloning approaches we test are:

- *sax-parts*: Load decoder pre-trained on *SWC-pre-training-saxophone*. Initialize and tune only Z and the reverb.
- *sax-whole*: Load decoder pre-trained on *SWC-pre-training-saxophone*. Tune decoder, Z and the reverb.
- *init-whole*: Initialize decoder, Z and reverb. Tune decoder, Z and the reverb.

All cloning experiments use the ADAM optimizer with fixed learning rates and a batch size of one. We use manual search to select the number of epochs and the learning rate for each configuration. Table 2 summarises the learning rates and the number of epochs we use for each configuration. Loss plots from cloning are available in the web supplement.

After cloning, we compute the multi-scale spectral reconstruction loss on the entirety of *saxophone-extra-test*. We also synthesise the first 32 seconds of *saxophone-extra-test* for listening. Since our implementation only supports rendering 4 seconds at a time, we render excerpts longer than 4 seconds by windowing control signals with four-second windows with a hop size of 3 seconds, and then linearly cross-fade the results.

Comparison to nearest instrument recording from pre-training In addition to the aforementioned cloning approaches, we synthesise *saxophone-extra-test* with each pair of Z and reverb from the pre-training stage. We record the lowest reconstruction loss and the resulting synthesis. We refer to this approach as *sax-nearest*. We use this as an additional point of comparison with the aforementioned approaches. Additionally, it gives us a general idea of how different our cloning target is from the recordings seen during pre-training.

Evaluation Figure 3) shows that starting from a pre-trained model (*sax-parts*, *sax-whole*) results in the lowest test losses for small training data sizes (4, 8, and 16 seconds). We observe with listening that the approach involving pre-training and then fine-tuning parts of the model (*sax-parts*) sounds the most natural for these training data sizes. However, we hear two noticeable artefacts.

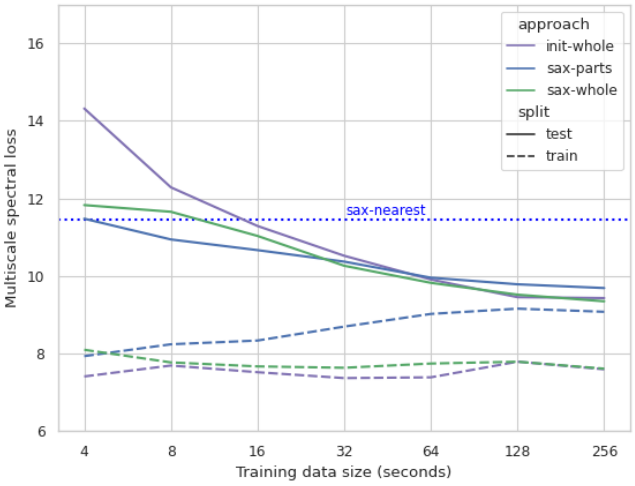

Figure 3: *Comparing different cloning approaches. We observe that when less than 32 seconds of training data are used, starting from a pre-trained model (sax-parts,* sax-parts-whole*) results in the lowest test losses.*

First, it sounds as if the instrument was recorded at a greater distance from the microphone than in the target. This artefact is present for all training data sizes but becomes less noticeable as the training data size increases. The second artefact is an echo not present in the target occurring for training data sizes under 32 seconds. This artefact is most noticeable for the 4- and 8-seconds train data sizes but can also be heard faintly in the 16-seconds examples. We do not hear much improvement in the naturalness or target similarity past 32 seconds of training data. Up to 32 seconds, tuning the entire model during cloning with the approach leveraging pre-training (*sax-whole*) sounds slightly more natural than the approach that does not leverage pre-training (*init-whole*). Additionally, the approach that leverages pre-training (*sax-whole*) requires less training time during cloning (see Table 2). Overall, we find that for less than 32 seconds of training data, pre-training and then tuning parts of the model (*sax-parts*) sounds more natural and similar to the target. With more than 32 seconds of training data, tuning the whole model during cloning (*init-whole*, *sax-whole*) produces better naturalness and similarity to the target.

*4.2.2. Cloning an instrument of a type **not** seen during pre-training*

We perform the following experiment to determine how well the aforementioned cloning approaches work if the target recording is of an instrument type not seen during pre-training. We pre-train a model on all excerpts in *SWC-pre-training* except for the saxophone excerpts. This corresponds to 179 excerpts from 144 different videos, totalling 8 hours 16 minutes, and 46 seconds. The mean excerpt duration is 2 minutes and 47 seconds. We pre-train the model using the same setup as in 4.2. We train for 558k steps (around 60 hours) and find an average training loss of 7.746 over the last 1k steps.

We define the following cloning approaches:

- *nosax-parts*: Load decoder pre-trained on all of *SWC-pre-training* except for the saxophone recordings. Initialize and tune only Z and the reverb.
- *nosax-whole*: Load decoder pre-trained on all of *SWC-pre-training* except for the saxophone recordings. Tune decoder, Z and the reverb.

Similar to our previous experiment, we render *saxophone-extra-test* with each pair of Z and reverb trained during the pre-training stage and record the lowest reconstruction loss, and the resulting synthesis. We refer to this approach as *nosax-nearest*.

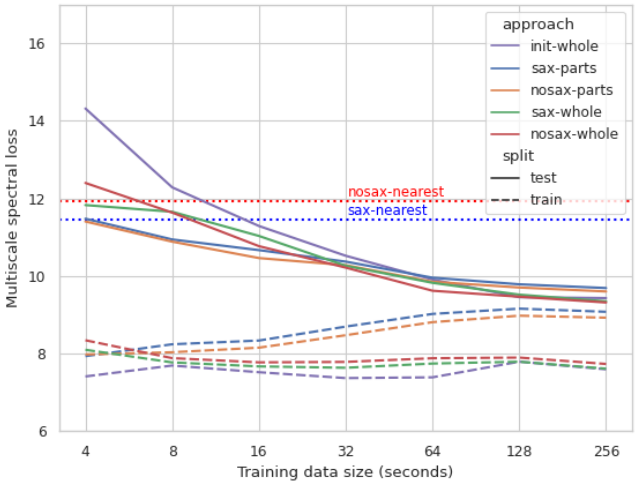

Figure 4: *Cloning an instrument type seen during pre-training (sax-parts, sax-whole) vs cloning an instrument type not seen during pre-training (nosax-parts, nosax-whole). We observe similar test losses regardless if the model was pre-trained on recordings of other saxophone or on recordings of wind instruments other than saxophone.*

Figure 4 shows pre-training decreases the reconstruction losses to a similar extent regardless of if the model was pre-trained on saxophones or if it was pre-trained on other wind instruments. Our listening evaluation finds that the type of pre-training data used makes a significant difference when tuning only parts of the model in the cloning phase. We find that while both approaches sound natural, the approach that leverages pre-training on saxophone recordings (*sax-parts*) sounds more similar to the target. When the pre-training data does not include the saxophone, the clone (*nosax-parts*) sounds between a saxophone and a horn. Regardless of whether a model was pre-trained on saxophones (*sax-whole*) or on other wind instruments (*nosax-whole*), we find tuning the whole model during cloning results in sounds of similar naturalness and similarity to the target across all training data sizes.

4.3. How much do the architecture changes alone improve model performance?

One surprising result from the experiments in Sec. 4.2 is that even the models without pre-training achieve good results when training on as little as 32 seconds of data. Prior work [1, 2] recommends around 10 minutes of target audio for training such a model. This discrepancy could be due to several factors:

- The training data covers a particularly wide range of notes despite its short duration;
- The additional information provided by F0-confidence makes the modelling task easier;
- Our two-part reverb design reduces the need for data.

We thus perform two ablation studies to validate that the changes we make to the DDSP architecture improve model performance. Both ablation studies introduce variations to the *-init* approach where we clone a recording without any pre-training. All ablation experiments reuse the same hyper-parameters as *init-whole*. We see the choice of hyperparameters is appropriate overall according to the loss convergence (provided in the web supplement).

4.3.1. Is F0-confidence conditioning beneficial?

We build and train a model that uses only F0 and loudness contours (*init-whole-no_f0conf*), and compare its performance to our full model (*init-whole*). Figure 5 shows that the model with F0-confidence conditioning achieves lower test losses except at the lowest amount of training data used (4 seconds). The unusually high training losses for the model trained without F0-confidence conditioning (*init-whole-no_f0conf*) for the two lowest training data sizes could be due to stopping training too early. In listening, we find the F0-confidence conditioning reduces artefacts, even for larger training data sizes. These particular artefacts occur during the onsets and offsets of notes.

4.3.2. Comparing reverb designs

To evaluate the impact of our two-part reverb design, we compare the *init-whole* approach with the following approaches:

- *init-whole-free_reverb* which is identical to *init-whole* except that it does not impose constraints on the reverb IR other than its duration
- *init-whole-fn_reverb* which is identical to *init-whole* except that it uses a filtered noise to produce the entire IR similar to [2].

Figure 6 shows that the model using the two-part reverb design (*init-whole*) obtains the lowest losses across all training data sizes. We also find that these models sound the most natural and similar to the target. We observe that the model with the unconstrained reverb (*init-whole-free_reverb*) suffers from a clear artefact in the form of excessive reverberation – an artefact particularly noticeable when the training data is small. We also observe that the

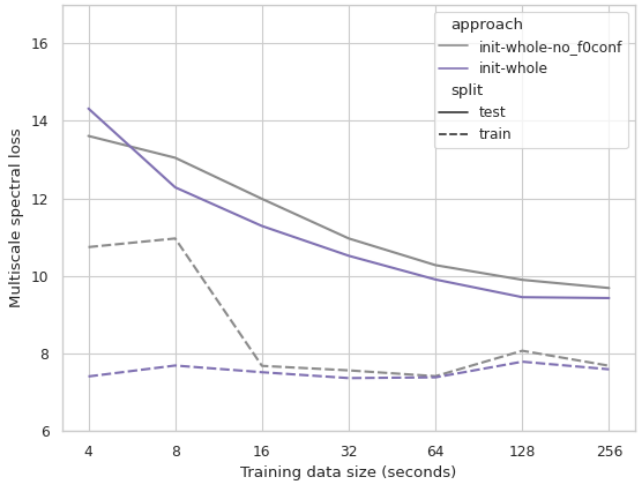

Figure 5: *Is F0-confidence conditioning beneficial? We observe that the model using F0-confidence conditioning (init-whole) obtains lower losses across all training data sizes larger than 4 seconds.*

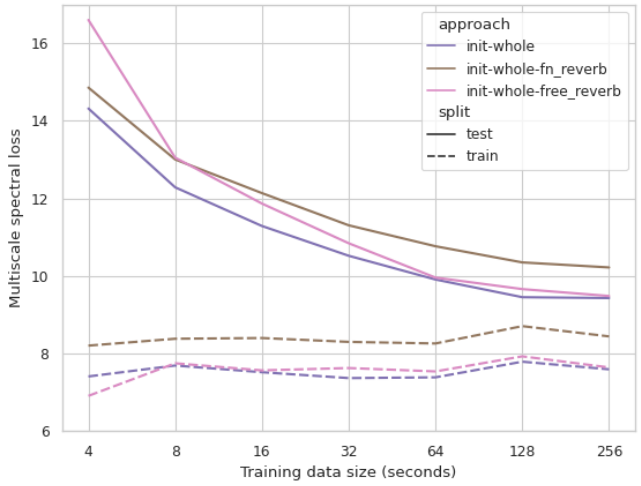

Figure 6: *Comparing reverb designs. The model using the two-part reverb (init-whole) design obtains the lowest test losses for all training data sizes.*

model using the filtered noise reverb (*init-whole-fn_reverb*) has poor naturalness and poor similarity to the target for all training data sizes. In general, the synthesised examples sound too dry compared to the target. The only exception is a subtle unnatural reverberation occurring with the smallest training data sizes (4 and 8 seconds).

5. APPLICATIONS

Engel et al. [1] demonstrates multiple applications enabled by the DDSP architecture. We demonstrate these applications are viable with our approach using 16 seconds of target recording data or less. We train clones from both excerpts in *SWC-cloning-saxophone* using our pre-trained saxophone model (Sec. 4.2.1). We also do the same for flute and trombone, i.e., first pre-training models on *SWC-pre-training-flute* or *SWC-pre-training-trombone* and then training clones on each excerpt from *SWC-cloning-flute* (4 excerpts) or *SWC-cloning-trombone* (1 excerpt). All clones in this section are trained on 16 seconds of training data using the *-parts* approach from Sec. 4.2.1, meaning that only the recording-specific embedding Z and the reverb are tuned.

We invite the reader to listen to the accompanying audio examples in the applications section of the web supplement. We demonstrate the following applications:

Loudness editing We produce versions of the excerpt that sound as if the instrument was played louder or quieter. We do this by shifting the loudness contour up or down. We notice that some re-synthesised examples sound unnatural, especially when the loudness contour is shifted by a large amount (-12 dB, +12 dB) (see flute #2). We observe by listening that shifting the loudness contour has an effect on the timbre similar to what we would expect from playing a real instrument quieter or louder. For example, the saxophones and trombones sound more bright when played louder and less bright when played quieter.

Pitch editing We produce versions of the excerpt that sound as if they were played in a different key. We do this by scaling the F0 contour., We use the pitch shifting effect from *librosa* [21] as a naive baseline. We observe with listening that some artefacts appear when using the clone to perform transposition. We also see that the transposed examples retain more timbral similarity to the cloning target than the naive baseline. The latter is particularly noticeable in the transpositions of trombone and saxophone #1.

Timbre transfer We synthesise a recording (control source) with the timbre of a different recording (timbre source) We use the first 32 seconds of *saxophone-extra-test* as our control source. In the presented examples, we do not use the trick by Engel et. al [1] where you first apply the trained reverb from the timbre source to the control source before extracting the control signals.

We observe that we obtain better results by shifting the loudness input so that the maximum value of the control signal matches the maximum loudness present in the training data of the clone. We also noticed some artefacts in the results: A subtle trailing echo can be heard in some examples (see audio examples for saxophone #1), some of the onsets and note transitions sound strange (see trombone) and some examples have excessive noise which sounds unnatural (see flute #4).

6. DISCUSSION

6.1. Results

Our first experiment compares cloning approaches when the target instrument is of a type seen during model pre-training. Here we see that when only small amounts of target training data are available ($<$ 16 seconds), tuning only Z and the reverb produce better

results than tuning the entire model. Our second experiment looks at when the target instrument is of a type not seen during model pre-training. We see the same result as before, but with slightly worse similarity to the target recording. Beyond a certain amount of training data (32 seconds in our case), tuning the whole model results in sound more similar to the target than tuning only part of the model. Interestingly, starting from a pre-trained model does not lead to significant improvements in naturalness or similarity to the timbre of the target when the whole model is tuned during cloning. However, starting from a pre-trained model allows us to clone instruments faster (see Table 2), even if the target recording was of an instrument of a different type than those seen in pre-training.

Our third and fourth experiments are ablation studies, looking at the impact of our modifications to the DDSP architecture – namely, F0-confidence conditioning of the decoder and the two-part reverb design. We observe that F0-confidence conditioning reduces artefacts around note onsets and offsets, especially if only a small amount of training data is available. Although we do not observe this, there might be a drawback with using this additional control signal in a timbre-transfer setting since the F0-confidence contour can be entangled with the timbre of an instrument. This could lead to unnatural output from a model trained on a different instrument type.

In our fourth experiment, we observe that our two-part reverb design achieves better results than either unconstrained reverb or a filtered noise reverb. However, as the amount of target data increases, the difference in performance between the two-part reverb design and unconstrained reverb diminishes. Further investigation could test if the two-part reverb design performs as well when applied to the synthesis of a broader range of instrument types and acoustic environments.

There is difficulty in setting training hyperparameters to provide a fair comparison of different architectures across multiple training data sizes. We aimed for a fair comparison between configurations by setting hyper-parameters such that each reached convergence of the test loss. Of the 56 configurations that we test, only four appear to not have fully converged after the given number of training epochs (see loss plots in web supplement): *init-whole-no_f0conf* at 4 seconds and 8 seconds, *init-whole-free_reverb* at 4 seconds, and *init-whole-fn_reverb* at 16 seconds. Further investigation into over-fitting during cloning is warranted. Interestingly, we see that the approaches tuning only Z and the reverb show no evidence of over-fitting, regardless of how long they are trained. Additionally, in light of the variable training data sizes, we believe the number of model updates might be a more useful measure than the number of epochs for reasoning about model training.

Overall, our experiments show how combining transfer learning with our modifications to the DDSP architecture has clear advantages. It allows one to create natural-sounding synthesis models of specific instrument recordings from smaller amounts of target instrument audio than before.

6.2. Future work

In this paper, we evaluate models by inspecting the reconstruction loss as well as generating a variety of examples for audition. Future work will evaluate such models using more formal listening tests with subjects drawn from the intended audience, as well as their use in creative settings. We have applied our approaches to a small number of instruments. Further work will extend to a larger set of instrument types (non-harmonic, percussive, etc.) and other synthesis architectures. Previous work extends DDSP to synthesise audio from MIDI [22–24]. Sharing parts of a model across recordings could help improve synthesis quality from MIDI as well as help reduce the amount the training data required.

Arik et al. [3] use a trained encoder to produce latent embeddings for neural voice cloning. This approach greatly accelerates the cloning process at a slight cost in synthesis quality. Earlier work in neural instrument synthesis also employs a trainable encoder [1,4,8–10]. Future work will explore integrating an encoder into our proposed approach. This could pose some challenges. Arik et al. [3] report that training the encoder jointly with the rest of the model is challenging and so opt for a multi-stage training process. Additionally, our approach uses a recording-dependent reverb, which could require some modification.

Differentiable rendering techniques have been used to transcribe polyphonic passages [25]. Other work has successfully learned to synthesise harmonic instruments from a mixture of instruments given a transcription [26]. We believe that training a multi-instrument model instils broad knowledge about musical instrument timbre into the decoder. Thus, future work will explore if using a multi-instrument model as a differentiable renderer could accomplish the simultaneous transcription and music instrument cloning with polyphonic music recordings, e.g., piano and ensembles.

7. ACKNOWLEDGMENTS

We would like to thank Laura Cros Vila, Joris Grouwels and Carl Thomé for their invaluable feedback. This paper is an outcome of MUSAiC, a project that has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 864189).

8. REFERENCES

[1] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts, "DDSP: Differentiable Digital Signal Processing," *arXiv:2001.04643 [cs, eess, stat]*, Jan. 2020, arXiv: 2001.04643.

[2] Michelle Carney, Chong Li, Edwin Toh, Nida Zada, Ping Yu, and Jesse Engel, "Tone Transfer: In-Browser Interactive Neural Audio Synthesis," in *Joint Proceedings of the ACM IUI 2021 Workshops*, 2021, p. 6.

[3] Sercan O. Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou, "Neural Voice Cloning with a Few Samples," *arXiv:1802.06006 [cs, eess]*, Oct. 2018, arXiv: 1802.06006.

[4] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi, "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders," *arXiv:1704.01279 [cs]*, Apr. 2017, arXiv: 1704.01279.

[5] Andy Sarroff and Michael A. Casey, "Musical Audio Synthesis Using Autoencoding Neural Nets," *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR2014)*, Oct. 2014, Publisher: International Society for Music Information Retrieval.

[6] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and

Yoshua Bengio, "SampleRNN: An Unconditional End-to-End Neural Audio Generation Model," *arXiv:1612.07837 [cs]*, Feb. 2017, arXiv: 1612.07837.

[7] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *arXiv:1609.03499 [cs]*, Sept. 2016, arXiv: 1609.03499.

[8] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts, "GANSynth: Adversarial Neural Audio Synthesis," *arXiv:1902.08710 [cs, eess, stat]*, Apr. 2019, arXiv: 1902.08710.

[9] Sicong Huang, Qiyang Li, Cem Anil, Xuchan Bao, Sageev Oore, and Roger B. Grosse, "TimbreTron: A WaveNet(CycleGAN(CQT(Audio))) Pipeline for Musical Timbre Transfer," *arXiv:1811.09620 [cs, eess, stat]*, May 2019, arXiv: 1811.09620.

[10] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman, "A Universal Music Translation Network," *arXiv:1805.07848 [cs, stat]*, May 2018, arXiv: 1805.07848.

[11] Lamtharn Hantrakul, Jesse H Engel, Adam Roberts, and Chenjie Gu, "Fast and Flexible Neural Audio Synthesis.," in *ISMIR*, 2019.

[12] Xavier Serra, "Spectral Modeling Synthesis: A Sound Analysis/Synthesis Based on a Deterministic plus Stochastic Decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[13] Ye Jia, Yu Zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, and Yonghui Wu, "Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis," *arXiv:1806.04558 [cs, eess]*, Jan. 2019, arXiv: 1806.04558.

[14] Hieu-Thi Luong and Junichi Yamagishi, "Nautilus: a versatile voice cloning system," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2967–2981, 2020, Publisher: IEEE.

[15] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, pp. 9, May 2016.

[16] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello, "CREPE: A Convolutional Representation for Pitch Estimation," *arXiv:1802.06182 [cs, eess, stat]*, Feb. 2018, arXiv: 1802.06182.

[17] Christian J. Steinmetz, Vamsi Krishna Ithapu, and Paul Calamia, "Filtered Noise Shaping for Time Domain Room Impulse Response Estimation From Reverberant Speech," *arXiv:2107.07503 [cs, eess]*, July 2021, arXiv: 2107.07503.

[18] Heinrich Kuttruff, *Room Acoustics*, CRC Press, Boca Raton, 6 edition, Sept. 2016.

[19] Mirjam Wester, Zhizheng Wu, and Junichi Yamagishi, "Analysis of the Voice Conversion Challenge 2016 Evaluation Results," in *Interspeech 2016*. Sept. 2016, pp. 1637–1641, ISCA.

[20] Juan F Montesinos, Olga Slizovskaia, and Gloria Haro, "Solos: A dataset for audio-visual music analysis," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. 2020, pp. 1–6, IEEE.

[21] Brian McFee, Colin Raffel, Dawen Liang, D. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto, "librosa: Audio and Music Signal Analysis in Python," 2015.

[22] Nicolas Jonason, Bob Sturm, and Carl Thomé, "The control-synthesis approach for making expressive and controllable neural music synthesizers," in *2020 AI Music Creativity Conference*, 2020.

[23] Yusong Wu, Ethan Manilow, Yi Deng, Rigel Swavely, Kyle Kastner, Tim Cooijmans, Aaron Courville, Cheng-Zhi Anna Huang, and Jesse Engel, "MIDI-DDSP: Detailed Control of Musical Performance via Hierarchical Modeling," *arXiv:2112.09312 [cs, eess]*, Mar. 2022, arXiv: 2112.09312.

[24] Rodrigo Castellon, Chris Donahue, and Percy Liang, "Towards realistic midi instrument synthesizers," in *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2020.

[25] Hayato Sumino, Adrien Bitton, Lisa Kawai, Philippe Esling, and Tatsuya Harada, "Automatic Music Transcription and Instrument Transposition with Differentiable Rendering," Oct. 2020, Conference Name: Joint Conference on AI Music Creativity (AIMC 2019) Pages: 10 Publication Title: Proceedings of the 1st Joint Conference on AI Music Creativity Publisher: AIMC.

[26] Masaya Kawamura, Tomohiko Nakamura, Daichi Kitamura, Hiroshi Saruwatari, Yu Takahashi, and Kazunobu Kondo, "Differentiable Digital Signal Processing Mixture Model for Synthesis Parameter Extraction from Mixture of Harmonic Sounds," *arXiv:2202.00200 [cs, eess]*, Jan. 2022, arXiv: 2202.00200.