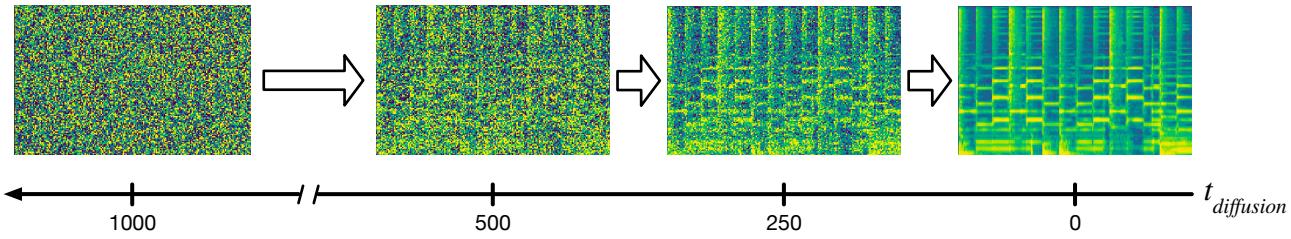

Figure 2. Our decoder stack is trained as a Denoising Diffusion Probabilistic Model (DDPM) [1]. The model starts with Gaussian noise as input and is trained to iteratively refine that noise toward a target, conditioned on a sequence of note events and the spectrogram of the previously rendered segment. This figure illustrates the diffusion process for one example segment.

dio samples. Our model for the first stage is an encoder-decoder Transformer architecture, shown in Figure 1. The encoder takes in a sequence of note events and, optionally, a second encoder uses an earlier part of the spectrogram as context. These embeddings are passed to a decoder, which generates a spectrogram corresponding to the input note sequence. Here, we explore training the decoder autoregressively (Section 3.1) or as a Denoising Diffusion Probabilistic Model (DDPM) [1] (Section 3.2).

Our architecture uses the encoder-decoder Transformer from T5 [51] with T5.1.1 [2] improvements and hyperparameters. We use the same note event vocabulary and note sequence encoding procedure as MT3 [8]. We find that training on a full song is prohibitive in terms of memory and compute due to the quadratic scaling of self-attention with sequence length, so we split note sequences into segments. Specifically, we train models with 2048 input positions for note events and 256 output positions for spectrogram frames. Each spectrogram frame represents 20 ms of audio, so each segment is 5.12 seconds.

Rendering audio segments independently introduces the problem of how to ensure smooth transitions between the segments once they are eventually concatenated together to form a full musical piece. We address this problem by providing the model with the spectrogram from the previously rendered segment, meaning that this model is autoregressive at the segment level. This context segment has its own encoder stack with 256 input positions. As seen in Figure 1, the outputs of both encoder stacks are concatenated together as input for cross-attention in the decoder layers.

We use sinusoidal position encodings, as in the original Transformer paper [52]. However, we found better performance if we decorrelated the position encodings of each network, as they have different meanings. We decorrelate the encodings by applying a unique random channel permutation and phase offset for the sinusoids used in each of the encoder and decoder stacks.

3.1 Autoregressive Decoder

As an initial approach, we took inspiration from Tacotron [34, 35], and trained the decoder as an autoregressive model on the continuous spectrograms. In this setting, standard causal masking was applied throughout the self-attention layers. The inputs and outputs are continuous spectrogram frames, and the model was trained with a mean-squared error (MSE) loss on those frames. Mathematically, this is equivalent to training a continuous autoregressive model with a Gaussian output distribution with isotropic and fixed variance. We also tried training models using mixtures of several Gaussians, but they tended to be unstable during sampling.

For sampling we use a fixed variance (0.2 in units of log magnitude) that is added to the outputs of every inference step. In practice, this "dithering" reduced strong audio artifacts due to overly smooth outputs.

Despite this, these baseline models still produce spectrogram outputs that tend to be blurry and contain jarring artifacts in some frames. We hypothesized an approach enabling incremental, bidirectional refinement, and model dependencies between frequency bins, could help resolve these issues.

3.2 Diffusion Decoder

Taking inspiration from recent successes in image generation such as DALL-E 2 and Imagen [53, 54], we investigated training the decoder as a Denoising Diffusion Probabilistic Model (DDPM) [1].

DDPMs are models that generate data from noise by reversing a Gaussian diffusion process that turns an input signal $\mathbf{x}$ into noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This *forward* process occurs over diffusion timesteps $t \in [0, 1]$ (not to be confused with time for the audio) resulting in noisy data $\mathbf{x_t} = \alpha_t \mathbf{x} + \sigma_t \epsilon$, where $\alpha_t \in [0, 1]$ and $\sigma_t \in [0, 1]$ are noise schedules that blend between the original signal and the noise over the diffusion time. In this work, the decoder, ϵ_θ with parameters θ, is trained to predict the added noise given the noisy data by minimizing an L1 objective,

$$\mathop{\mathbb{E}}_{\mathbf{x},\mathbf{c},\epsilon,t} w_t ||\epsilon_\theta(\mathbf{x_t}, \mathbf{c}, t) - \epsilon||_1^1 \quad (1)$$

where w_t is a loss weighting for different diffusion timesteps and $\mathbf{c}$ is additional conditioning information for the decoder. Schedules for w_t, α_t and ϵ_t are hyperparameters chosen to selectively emphasize certain steps in the reverse diffusion process.

[2] https://github.com/google-research/text-to-text-transfer-transformer/blob/main/released_checkpoints.md#t511

During sampling, we follow the *reverse* diffusion process by starting from pure independent Gaussian noise for each frame and frequency bin and iteratively use noise estimates to generate new spectrograms with gradually decreasing noise content. Full details of this process can be found in Ho et al. [1] and source code is provided for clarity [3]. A visualization of the forward and reverse diffusion process can be seen in Figure 2.

Because the diffusion process expects inputs to be in the range $[-1, 1]$, we scale spectrograms used for training targets and audio context to that range before using them. During inference, we scale model outputs back to the expected spectrogram range before converting them to audio.

During training, we used a uniformly-sampled noise time step in $[0, 1]$ for each training example and run the diffusion process forward to that step with randomly sampled Gaussian noise. The noisy example is then used for input, and the model is trained to predict the Gaussian noise component with Equation (1). During inference, we run the diffusion process in reverse using 1000 linearly spaced steps in $[0, 1]$.

Based on the implementation of Imagen [54], we also incorporated several recent DDPM improvements, including using logSNR [55, 56], a cosine noise schedule [57] of $\cos(t * \pi/2)^2$, and Classifier-Free Guidance [58] during sampling. In particular, we found that using Classifier-Free Guidance during sampling led to samples that were less noisy and more crisp sounding. After a coarse sweep of values, we train with a conditioning dropout probability of 0.1 and sample with a conditioned sample weight of 2.0.

Many diffusion models use a UNet [59] architecture, but in order to stay close to the architecture used in MT3, we used a standard Transformer decoder with no causal masking. We incorporate the noise time information by first converting the continuous time value to a sinusoid embedding using the same method as the position encodings in the original Transformer paper [52] followed by an MLP block. That embedding is then incorporated at each decoder layer using a FiLM [60] layer before the self-attention block and after the cross-attention block. The outputs of note events and spectrogram context encoder stacks are concatenated together and incorporated using a standard encoder-decoder cross-attend layer.

3.3 Spectrograms to Audio

To translate the model's magnitude spectrogram output to audio, we use a convolutional spectrogram inversion network as proposed in MelGAN [2]. In particular, we base our implementation on the more recent SEANet [61] and SoundStream [31]. This model first applies a 1D-convolution with kernel size 3 and stride 1. It then cascades four blocks of layers. Each block is composed of a transposed convolution followed by three residual units, applying a dilated convolution with kernel size 3 and dilation rate 1, 3, and 9 respectively. ELUs [62] are used after each convolution. As in [31], the model is trained with a mix of three loss functions. The first is a multi-scale spectral reconstruction loss, inspired by [63]. The second and third losses are an adversarial loss and a feature matching loss, computed with two discriminators, one STFT-based and one waveform-based. We train this spectrogram inverter on an internal dataset of 16k hours of uncurated music with Adam [64] for 1M steps with a batch size of 128.

The spectrograms used for training both the spectrogram inverter and the synthesis models used audio with a sample rate of 16 KHz, an STFT hop size of 320 samples (20 ms), a frame size of 640, and 128 log magnitude mel bins ranging in frequency from 0 to 8 KHz.

4. DATASETS

Our architecture is flexible enough to train on any dataset containing paired audio and MIDI examples, even when multiple instruments are mixed together in the same track (e.g., individual instrument stems are unavailable). Thus, we are able to use all the same datasets used to train the MT3 transcription model: MAESTROv3 [14] (piano), Slakh2100 [65] (synthetic multi-instrument), Cerberus4 [66] (synthetic multi-instrument), Guitarset [67] (guitar), MusicNet [68] (orchestral multi-instrument), and URMP [69] (orchestral multi-instrument).

We use the same preprocessing of these datasets as MT3, including the data augmentation strategy used for Slakh2100 where 10 random subsets of instruments from each track were selected to increase the number of tracks. We also use the same train/validation/test splits. Our coarse hyperparameter sweeps (e.g., for selecting Classifier-Free Guidance weighting) were done using only validation sets. Results are reported on the test splits, except for Guitarset and URMP which do not have a test split, and so the validation split was used for final results.

The datasets used in these models contain a wide variety of instruments, both synthetic and real. In order to simplify training, we map all MIDI instruments to the 34 Slakh2100 classes plus drums. These mappings cover all General MIDI classes other than "Synth Effects", "Other", "Percussive", and "Sound Effects". As a result, the trained model is capable of synthesizing arbitrary MIDI files while retaining clear instrument identity.

We use the same note event vocabulary as MT3, which is similar to MIDI events. Specifically, there are events for Instrument (128 values), Note (128 values), On/Off (2 values), Time (512 values), Drum (128 values), End Tie Section (1 value), and EOS (1 value). A full description of these events can be found in Section 3.2 of the MT3 paper [8]. Because not all datasets include velocity information, we do not currently include velocity events (as in MT3) and rely on the model's decoder to produce natural sounding velocities given the music context.

Training on such a diverse set of examples gives the model flexibility during synthesis. For example, we can synthesize a track with realistic piano sounds learned from MAESTRO, orchestral instruments from URMP, and synthesizers and drum beats from Slakh.

[3] https://github.com/magenta/music-spectrogram-diffusion

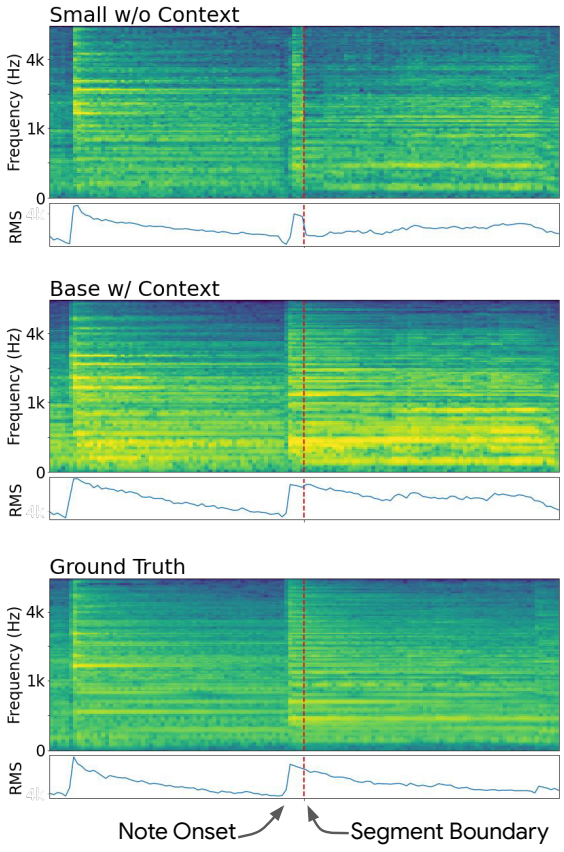

Figure 3. The "Small w/o Context" model (top) does not receive the previously rendered segment as conditioning input and exhibits clear artifacts at segment boundaries (note the dip in RMS after the segment boundary). The "Base w/ Context" model (middle) has access to the previous segment as conditioning information and achieves smooth transition between segments. By repeating the process of rendering a segment and then feeding that segment in as context for the next segment, this model is capable of rendering arbitrary length pieces.

5. EXPERIMENTS

We used the MT3 codebase as a starting point [72], and experiments were implemented using `t5x` for model training and `seqio` for data processing and evaluation [6].

We used the T5.1.1 "small" or "base" Transformer hyperparameters. For "small", there are 8 layers on the encoder and decoder stacks, 6 attention heads with 64 dimensions each, 1024 MLP dimensions, and 512 embedding dimensions. For "base", there are 12 layers on the encoder and decoder stacks, 12 attention heads with 64 dimensions each, 2048 MLP dimensions, and 758 embedding dimensions. We used float32 activations for all models.

Using the datasets described above, we trained four versions of the synthesis model: an 85M parameter "small" autoregressive model with no spectrogram context, an 85M parameter "small" diffusion model with no context, a 104M parameter "small" diffusion model with context, and a 412M parameter "base" diffusion model with context. Training used a batch size of 1024 on 64 TPUv4s with a constant learning rate of $1e^{-3}$ for 500k steps with the Adafactor [73] optimizer. Depending on model size and hardware availability, training took 44–134 hours.

5.1 Metrics

We evaluate model performance on the following metrics:

Reconstruction Embedding Distance For this metric, we pass an individual audio clip and a synthesis model's reconstruction of that audio into a classification network and calculate the distance in the classification network's embedding between the two signals. Here, we report numbers from both VGGish [70] and TRILL [71]. To compute the distance we use the Frobenius norm of the network's embedding layer, averaged over time frames. VGGish outputs 1 embedding per input audio second, and we use the model's output layer as the embedding layer. TRILL outputs $\sim$5.9 embeddings per input audio second, and we use the network's dedicated "embedding" layer.

Fréchet Audio Distance (FAD) FAD measures the perceptual similarity of the distribution of all of the model's output over the entire evaluation set of note events to the distribution of all of the ground truth audio [74].

MT3 Transcription This metric measures how well the synthesis model is reproducing the notes and instruments specified in the input data. We pass the synthesis model output through a pretrained MT3 transcription model and compute an F1 score using the "Full" metric from the MT3 paper as calculated by `mir_eval` [75]. A note is considered correct if has a matching note onset within ± 50 ms, an offset within $0.2 \cdot$ reference_duration or 50 ms (whichever is greater), and has the exact instrument program number as the input data.

Realtime (RT) Factor This measures how fast synthesis is compared to the duration of the audio being synthesized. For example, an RT Factor of 2 means that 2 seconds of audio can be created with 1 second of wall time compute. Inference is performed on a single TPUv4 with a batch size of 1. Here, we include both spectrogram inference and spectrogram-to-audio inversion.

The reconstruction and FAD metrics require embedding both model output and ground truth audio using a model sensitive to perceptual differences. Following the original FAD formulation, we use the VGGish model [70]. To avoid any biases from that particular model, we also compute embeddings using the TRILL model [71]. Pretrained models for computing these embeddings were obtained from the VGGish [4] and TRILL [5] pages on TensorFlow Hub. Due to the large size of the datasets and the high compute and memory requirements for these metrics, we limit metrics calculation to the first 10 minutes of audio for each example.

[4] https://tfhub.dev/google/vggish/1
[5] https://tfhub.dev/google/nonsemantic-speech-benchmark/trill/3

Model	VGGish [70]		TRILL [71]		Transcription F1 ↑	RT Factor ↑
	Recon. ↓	FAD ↓	Recon. ↓	FAD ↓		
Small Autoregressive	3.70	1.03	7.27	0.39	0.31	**10.55**
Small w/o Context	3.49	1.07	6.16	0.46	0.31	2.82
Small w/ Context	3.56	1.10	6.40	0.53	0.25	3.07
Base w/ Context	**3.13**	**1.00**	**2.74**	**0.27**	**0.36**	1.05
Ground Truth Encoded	2.14	0.51	1.54	0.05	0.36	12.25
Ground Truth Raw	0.00	0.00	0.00	0.00	0.63	–

Table 1. Synthesis model experiment results. Metrics are the mean across all evaluation datasets. "Small" and "Base" refer to model capacity (Section 5). "Ground Truth Encoded" refers to the ground truth audio encoded by the spectrogram inverter (Section 3.3) and represents an upper limit on synthesis model performance. "Ground Truth Raw" refers to the unprocessed ground truth audio and represents an upper limit on transcription model performance. Metrics are fully described in Section 5.1 and results are discussed in Section 5.2.

5.2 Results

We first evaluated a "small" autoregressive model (Section 3.1). Initial results were promising, but quantitative and qualitative evaluation of results made it clear that this approach would not be sufficient.

We then investigated using a diffusion approach (Section 3.2). Results from a "small" model were impressive (nearly maxing out the Transcription metric), but we noticed abrupt timbre shifts and audible artifacts at segment boundaries, as illustrated in Figure 3. This makes sense because the synthesis problem from raw MIDI is underspecified: input to the model specifies notes and instruments, but the model has been trained on a wide variety of sources, and the audio corresponding to a given note and instrument combination could just as likely be synthetic or real, played in a dry or reverberant room, played with or without vibrato, etc.; the model has to sample from this large space of possibilities independently for each segment.

To address this issue, we added a context spectrogram input encoder to encourage the model to be consistent over time. We first experimented with adding this context input to a "small" model, but got poor results, possibly due to insufficient decoder capacity for incorporating the additional encoder output. We then tried scaling up to a "base" model that included context. The additional model capacity resulted in generally higher audio quality and also did not have segment boundary problems. This is reflected in the metrics, where this model achieves the best scores by far. Also, even with the larger model size, the synthesis process is still slightly faster than realtime.

Qualitatively, we find that the model does an especially impressive job rendering instruments where the training data came from real audio recordings as opposed to synthetic instruments. For example, when given a note event sequence from the Guitarset validation split, the model not only accurately reproduces the notes but also adds fret and picking noises. Sequences rendered for orchestral instruments such as the ones found in URMP add breath noise and vibrato. The model also does a remarkably good job of rendering natural-sounding note velocities, even though no velocity information is present in our note vocabulary.

Results for these experiments are presented in Table 1. Additional results, including audio examples demonstrating the ability to render out-of-domain MIDI files and modify their instrumentation, are in our online supplement [6] . Results on a per-dataset basis are in Appendix A.

These results are encouraging, but there is still plenty of room for improvement. Particularly, the synthesized audio has occasional issues with inconsistent loudness and audio artifacts, and its overall fidelity does not match the training data. However, in all our experiments, we observed no overfitting on the validation sets, so we suspect that even larger models could be trained for higher fidelity audio.

Another limiting factor of our approach is the spectrogram inversion process, which represents an upper bound on audio quality. This is apparent in the Transcription F1 score, where our models have already achieved the maximum score possible, even though that score is only a little over half of what is achievable with raw audio.

6. CONCLUSION

This work represents a step toward interactive, expressive, and high fidelity neural audio synthesis for multiple instruments. Our flexible architecture allows incorporating any training data with paired audio and MIDI examples. Improved automatic music transcription systems such as MT3 [8] point toward being able to generate high quality MIDI annotations for arbitrary audio, which would greatly expand the available training data and range of instruments and acoustic settings. Using generic Transformer encoder stacks also presents the possibility of utilizing conditioning information beyond note events and spectrogram context. For example, we could add finer grained conditioning such as control over note timbre, or more global controls such as free text descriptions. This model is already slightly faster than realtime, but ongoing research into diffusion models shows promising directions for speeding up inference [56], giving plenty of room for larger and more complicated models to remain interactive. This may be especially important as we explore options for higher quality audio output than our current spectrogram inversion process enables.

[6] Online supplement and examples: `https://bit.ly/3wxSS4l`

7. ACKNOWLEDGEMENTS

We would like to thank William Chan, Mohammad Norouzi, Chitwan Saharia, and Jonathan Ho for help with the diffusion implementation and Sander Dieleman for helpful discussions about diffusion approaches.

8. REFERENCES

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf

[2] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/6804c9bca0a615bdb9374d00a9fcba59-Paper.pdf

[3] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=B1x1ma4tDr

[4] Y. Wu, E. Manilow, Y. Deng, R. Swavely, K. Kastner, T. Cooijmans, A. Courville, C.-Z. A. Huang, and J. Engel, "Midi-ddsp: Detailed control of musical performance via hierarchical modeling," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=UseMOjWENv

[5] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[6] A. Roberts, H. W. Chung, A. Levskaya, G. Mishra, J. Bradbury, D. Andor, S. Narang, B. Lester, C. Gaffney, A. Mohiuddin *et al.*, "Scaling up models and data with `t5x` and `seqio`," *arXiv preprint arXiv:2203.17189*, 2022.

[7] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan, "Flamingo: a visual language model for few-shot learning," 2022. [Online]. Available: https://arxiv.org/abs/2204.14198

[8] J. P. Gardner, I. Simon, E. Manilow, C. Hawthorne, and J. Engel, "MT3: Multi-task multitrack music transcription," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=iMSjopcOn0p

[9] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel, "Sequence-to-sequence piano transcription with transformers," *arXiv preprint arXiv:2107.09142*, 2021.

[10] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *9th ISCA Speech Synthesis Workshop*, 2016, pp. 125–125.

[11] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model," *arXiv preprint arXiv:1612.07837*, 2016.

[12] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1068–1077.

[13] N. Mor, L. Wolf, A. Polyak, and Y. Taigman, "A universal music translation network," Tech. Rep., 2018. [Online]. Available: https://ytaigman.github.io/musicnet

[14] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=r1lYRjC9F7

[15] R. Manzelli, V. Thakkar, A. Siahkamari, and B. Kulis, "Conditioning deep generative raw audio models for structured automatic music," *arXiv preprint arXiv:1806.09905*, 2018.

[16] J. W. Kim, R. Bittner, A. Kumar, and J. P. Bello, "Neural music synthesis for flexible timbre control," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 176–180.

[17] E. Cooper, X. Wang, and J. Yamagishi, "Text-to-speech synthesis techniques for midi-to-audio synthesis," 2021. [Online]. Available: https://arxiv.org/abs/2104.12292

[18] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *arXiv*, 2018, pp. 2415–2424.

[19] L. H. Hantrakul, J. Engel, A. Roberts, and C. Gu, "Fast and flexible neural audio synthesis." in *ISMIR*, 2019.

[20] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ByMVTsR5KQ

[21] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial neural audio synthesis," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=H1xQVn09FX

[22] G. Greshler, T. R. Shaham, and T. Michaeli, "Catch-a-waveform: Learning to generate audio from a single short example," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021. [Online]. Available: https://openreview.net/forum?id=XmHnJsiqw9s

[23] M. Morrison, R. Kumar, K. Kumar, P. Seetharaman, A. Courville, and Y. Bengio, "Chunked autoregressive gan for conditional waveform synthesis," in *International Conference on Learning Representations (ICLR)*, April 2022.

[24] X. Wang and J. Yamagishi, "Neural harmonic-plus-noise waveform model with trainable maximum voice frequency for text-to-speech synthesis," Tech. Rep., 2019.

[25] J. Engel, R. Swavely, L. H. Hantrakul, A. Roberts, and C. Hawthorne, "Self-supervised pitch detection by inverse audio synthesis," in *ICML 2020 Workshop on Self-supervision in Audio and Speech*, 2020. [Online]. Available: https://openreview.net/forum?id=RlVTYWhsky7

[26] R. Castellon, C. Donahue, and P. Liang, "Towards realistic midi instrument synthesizers," in *NeurIPS Workshop on Machine Learning for Creativity and Design (2020)*, 2020.

[27] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter waveform models for statistical parametric speech synthesis," *arXiv preprint arXiv:1904.12088*, 2019.

[28] A. Caillon and P. Esling, "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," *arXiv preprint arXiv:2111.05011*, 2021.

[29] ——, "Streamable neural audio synthesis with non-causal convolutions," 2022.

[30] A. Défossez, N. Zeghidour, N. Usunier, L. Bottou, and F. Bach, "Sing: Symbol-to-instrument neural generator," *Advances in neural information processing systems*, vol. 31, 2018.

[31] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021.

[32] C. Hawthorne, A. Jaegle, C. Cangea, S. Borgeaud, C. Nash, M. Malinowski, S. Dieleman, O. Vinyals, M. Botvinick, I. Simon *et al.*, "General-purpose, long-context autoregressive modeling with perceiver ar," *arXiv preprint arXiv:2202.07765*, 2022.

[33] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. USA: Curran Associates Inc., 2017, pp. 6309–6318. [Online]. Available: http://dl.acm.org/citation.cfm?id=3295222.3295378

[34] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, and Others, "Tacotron: Towards end-to-end speech synthesis," in *Proc. Interspeech*, 2017, pp. 4006–4010.

[35] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgiannakis, and Y. Wu, "Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2018-April, 2018, pp. 4779–4783. [Online]. Available: https://google.github.io/tacotron/publications/tacotron2.

[36] R. J. Weiss, R. Skerry-Ryan, E. Battenberg, S. Mariooryad, and D. P. Kingma, "Wave-tacotron: Spectrogram-free end-to-end text-to-speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5679–5683.

[37] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3617–3621.

[38] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," in *International Conference on Learning Representations*, 2020.

[39] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "Wavegrad: Estimating gradients for waveform generation," in *International Conference on Learning Representations*, 2020.

[40] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov, "Grad-TTS: A diffusion probabilistic model for text-to-speech," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8599–8608.

[41] M. Jeong, H. Kim, S. J. Cheon, B. J. Choi, and N. S. Kim, "Diff-TTS: A denoising diffusion model for text-to-speech," *arXiv preprint arXiv:2104.01409*, 2021.

[42] J. Lee and S. Han, "Nu-wave: A diffusion probabilistic model for neural audio upsampling," *Proc. Interspeech 2021*, pp. 1634–1638, 2021.

[43] H. Kameoka, T. Kaneko, K. Tanaka, N. Hojo, and S. Seki, "Voicegrad: Non-parallel any-to-many voice conversion with annealed langevin dynamics," *arXiv preprint arXiv:2010.02977*, 2020.

[44] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, M. Kudinov, and J. Wei, "Diffusion-based voice conversion with fast maximum likelihood sampling scheme," *arXiv preprint arXiv:2109.13821*, 2021.

[45] J. Liu, C. Li, Y. Ren, F. Chen, P. Liu, and Z. Zhao, "Diffsinger: Diffusion acoustic model for singing voice synthesis," *CoRR*, vol. abs/2105.02446, 2021. [Online]. Available: https://arxiv.org/abs/2105.02446

[46] S. Liu, Y. Cao, D. Su, and H. Meng, "Diffsvc: A diffusion probabilistic model for singing voice conversion," *arXiv preprint arXiv:2105.13871*, 2021.

[47] S. Rouard and G. Hadjeres, "Crash: Raw audio score-based generative modeling for controllable high-resolution drum sound synthesis," *arXiv preprint arXiv:2106.07431*, 2021.

[48] N. Kandpal, O. Nieto, and Z. Jin, "Music enhancement via image translation and vocoding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 3124–3128.

[49] G. Mittal, J. Engel, C. Hawthorne, and I. Simon, "Symbolic music generation with diffusion models," *arXiv preprint arXiv:2103.16091*, 2021.

[50] K. Goel, A. Gu, C. Donahue, and C. Ré, "It's raw! audio generation with state-space models," *arXiv preprint arXiv:2202.09729*, 2022.

[51] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[53] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022. [Online]. Available: https://arxiv.org/abs/2204.06125

[54] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *arXiv preprint arXiv:2205.11487*, 2022.

[55] D. P. Kingma, T. Salimans, B. Poole, and J. Ho, "Variational diffusion models," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021. [Online]. Available: https://openreview.net/forum?id=2LdBqxc1Yv

[56] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=TIdIXIpzhoI

[57] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8162–8171. [Online]. Available: https://proceedings.mlr.press/v139/nichol21a.html

[58] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. [Online]. Available: https://openreview.net/forum?id=qw8AKxfYbI

[59] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[60] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[61] M. Tagliasacchi, Y. Li, K. Misiunas, and D. Roblek, "SEANet: A multi-modal speech enhancement network," in *Interspeech*, 2020, pp. 1126–1130.

[62] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1511.07289

[63] A. A. Gritsenko, T. Salimans, R. van den Berg, J. Snoek, and N. Kalchbrenner, "A spectral energy distance for parallel speech synthesis," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/9873eaad153c6c960616c89e54fe155a-Abstract.html

[64] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[65] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 45–49.

[66] E. Manilow, P. Seetharaman, and B. Pardo, "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 771–775.

[67] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, "GuitarSet: A dataset for guitar transcription." in *ISMIR*, 2018, pp. 453–460.

[68] J. Thickstun, Z. Harchaoui, and S. M. Kakade, "Learning features of music from scratch," in *International Conference on Learning Representations (ICLR)*, 2017.

[69] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications," *Trans. Multi.*, vol. 21, no. 2, p. 522–535, feb 2019. [Online]. Available: https://doi.org/10.1109/TMM.2018.2856090

[70] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, New Orleans, USA, Mar. 2017.

[71] J. Shor, A. Jansen, R. Maor, O. Lang, O. Tuval, F. de Chaumont Quitry, M. Tagliasacchi, I. Shavitt, D. Emanuel, and Y. Haviv, "Towards learning a universal non-semantic representation of speech," in *Interspeech*, 2020, pp. 140–144.

[72] "MT3: Multi-task multitrack music transcription," https://github.com/magenta/mt3, accessed: 2022-05-01.

[73] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4596–4604. [Online]. Available: https://proceedings.mlr.press/v80/shazeer18a.html

[74] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms." in *INTERSPEECH*, 2019, pp. 2350–2354.

[75] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.

DDX7: DIFFERENTIABLE FM SYNTHESIS OF MUSICAL INSTRUMENT SOUNDS

Franco Caspe, Andrew McPherson, Mark Sandler
Queen Mary University of London
{f.s.caspe, a.mcpherson, mark.sandler}@qmul.ac.uk

ABSTRACT

FM Synthesis is a well-known algorithm used to generate complex timbre from a compact set of design primitives. Typically featuring a MIDI interface, it is usually impractical to control it from an audio source. On the other hand, Differentiable Digital Signal Processing (DDSP) has enabled nuanced audio rendering by Deep Neural Networks (DNNs) that learn to control differentiable synthesis layers from arbitrary sound inputs. The training process involves a corpus of audio for supervision, and spectral reconstruction loss functions. Such functions, while being great to match spectral amplitudes, present a lack of pitch direction which can hinder the joint optimization of the parameters of FM synthesizers. In this paper, we take steps towards enabling continuous control of a well-established FM synthesis architecture from an audio input. Firstly, we discuss a set of design constraints that ease spectral optimization of a differentiable FM synthesizer via a standard reconstruction loss. Next, we present Differentiable DX7 (DDX7), a lightweight architecture for neural FM resynthesis of musical instrument sounds in terms of a compact set of parameters. We train the model on instrument samples extracted from the URMP dataset, and quantitatively demonstrate its comparable audio quality against selected benchmarks.

1. INTRODUCTION

Sound generation and transformation tools are ubiquitous in music composition and production. The electronic synthesizer has enabled musicians to access forms of timbre beyond the capabilities of acoustic or amplified instruments. Chowning's FM Synthesis [1] is a widely used technique that is flexible to create complex, harmonic and inharmonic spectra from a reduced set of controls. Its long-standing presence in the audio industry has shaped traditional and contemporary sound design techniques [2–4] across musicians and producers, with a wide number of current commercial synthesizer keyboards, modules and software plugins featuring it.

For most of its history, the digital synthesizer has been inextricably associated with the MIDI keyboard. Several historical technical and social factors contributed to making the keyboard the de-facto control interface for the synthesizer [5]. The legacy of this association continues to steer synth design, favouring triggers and envelope generators over continuous control strategies.

Developing alternative interfaces for controlling digital synthesis remains an active area of research [6]. Many such digital musical instruments (DMIs) are based on mappings between sensor data and synthesis parameters [7, 8]. An alternative approach uses features extracted from the audio signal of an acoustic instrument to control a digital synthesis process. In that context, previous works employed audio signals from musical instrument as oscillators [9], in a configuration that can be seen as a special application of Adaptive Digital Audio Effects [10, 11].

More recently, Neural Audio Synthesis (NAS) algorithms have employed Deep Neural Networks (DNNs) to map audio features to synthesizer parameters, in tasks such as realistic audio synthesis from a compact set of control signals [12–14], timbre transfer from one instrument to another [15, 16] and enhancement of symbolic musical expression [17]. While the results are impressive, we observe that the employed synthesis architectures are overly complex, featuring dozens of time-evolving parameters that hinder intervention, with users defaulting to indirect methods to intervene with the synthesis process, such as network bending [18].

We are interested in an audio-based, continuous control technique for a well-established synthesis architecture, that can potentially offer musicians similar sound design primitives and outcomes available to keyboard players. Furthermore, we want the model to be compatible with live use, therefore it should be able to work in real-time.

In this work, we take steps towards enabling interpretable sound design controls for NAS algorithms, and present Differentiable DX7 (DDX7), a causal and lightweight DNN architecture that maps continuous audio features to the synthesis parameters of a well-known FM synthesizer. We train DDX7 in single instrument datasets and evaluate its resynthesis performance against selected benchmarks. We provide full source code, and an online supplement [1] with audio examples and a preliminary analysis of the model's real-time execution capabilities.

[1] https://fcaspe.github.io/ddx7

2. BACKGROUND

2.1 Linear FM Synthesis

Linear FM modulation for audio signals, originally described by Chowning [1] in the early 1970s, is a well-known sound design technique that powered the first massively available and commercially successful digital synthesizer, the Yamaha DX7, defining an era in music production [2]. Since then, it that has retained a fair amount of attention by the research community [19], being present in a variety of topics from timbre semantic analysis [13], to empirical [4] and computer assisted sound design [20], and adaptive effects [11]. Commercially, manufacturers continue to release as of today a sizable number of new instruments based on this technology [21, 22].

Linear FM synthesis is actually a phase modulation technique aimed at generating different timbres with few parameters and low computational resources. Expressed in term of sine waves, the instantaneous FM modulated signal can be written as shown in Equation 1, where f_c is the carrier frequency, f_m is the modulator frequency and I is the modulation index. In this work, we denote this particular linear phase modulation technique simply as FM Synthesis.

$$y(t) = sin(2\pi f_c t + Isin(2\pi f_m t)) \quad (1)$$

The side-bands of an FM signal are equally spaced around the carrier frequency, and their separation determined by f_m, while the number of harmonics depends on the modulation index and follows a Bessel function of the first order (Equation 2).

$$y(t) = \sum_{n=-\infty}^{n=+\infty} J_n(I) sin(2\pi(f_c + nf_m) \cdot t) \quad (2)$$

By controlling the modulation indexes and ratios, users can generate rich and complex timbre with a small set of oscillators modulated in frequency. For instance, the simple modulator-carrier setup can be extended with a new carrier that is modulated by the output of the previous pair. This *stacked* arrangement spreads the bands of the initial modulator-carrier pair across another carrier, adding a new layer of complexity to the timbre. Furthermore, outputs from many stacks can be mixed in an *additive* approach. Next, by setting the frequency ratio r in such a way that $f_c = r \cdot f_m$, with $r \in \mathbb{Q}$, FM generates harmonic spectra.

2.2 The DX7 Synthesizer

The Yamaha DX7 synthesizer is probably the best-known FM synthesizer, and it features a linear FM synthesis architecture that has been previously used by other synth models as well, making it an excellent candidate for integration to a continuous control strategy.

The sound of the DX7 is generated by six frequency-modulated sinusoidal oscillators, and it is programmed by means of setting up a *patch*. For each oscillator, a patch describes their routing (i.e. how the oscillators are interconnected in a stacked or additive fashion), the Attack-Decay-Sustain-Release (ADSR) envelope generator parameters and the frequency ratios with respect to the note that is being played. Fixed frequencies can also be assigned to the oscillators. The ADSR parameters control the *output levels* of each oscillator; affecting their output volume or modulation index depending on their interconnection. Finally, in the DX7, the oscillators' frequency ratios and routing remain *fixed* during audio rendering. Sound dynamics are generated mainly by the ADSR envelopes controlling the volume or modulation index of the carriers and modulators respectively.

2.3 Sound Matching

Sound matching is the process of estimating a set of synthesis parameters that approximate a given audio signal as best as possible. The task of matching an audio input to synthesizer controls is not new especially for FM, where evolutionary methods have been used to optimize a patch for a particular given sound [20, 23].

More recently, DNNs have been studied for supervised sound matching from an annotated dataset of patches and synthesized audio excerpts. The approaches include classification [24], where the network predicts a patch based on an input spectrogram, variational inference [25], with the DNN learns an invertible mapping between a dataset of audio and patches, and multi-modal analysis [26] that employs multiple aggregated audio features for prediction.

While these methods are effective, they usually process long audio windows on the order of seconds, required to estimate the values of the ADSR envelope generator. Furthermore they can only estimate parameters from audio excerpts at specific pitch values. These approaches are not suitable for continuous control of a synthesizer from an audio signal.

2.4 Neural Audio Synthesis

DNNs can capture complex relations from a dataset. This can be exploited in a generative approach to produce realistic sounding audio. Neural Audio Synthesis (NAS) algorithms employ DNNs as powerful synthesizers that can capture structure and nuances from a corpus of music and produce new audio with similar characteristics during inference. Furthermore, these algorithms can be conditioned at train or inference time, modifying the output on-the-fly.

Since the introduction of the seminal autoregressive architectures Wavenet [27] and SampleRNN [28], the NAS field has included generation approaches such as Generative Adversarial Neural Networks (GANs) [29–31], probabilistic models [32, 33] and style transfer methods [15, 34]. Furthermore, a branch of the field has focused on controllable music generation, in an effort to bring interaction possibilities to users, with NAS algorithms supporting control inputs such as MIDI [17], timbre descriptors [29] and pitch and loudness signals [12–14].

2.5 Differential Digital Signal Processing

Differential Digital Signal Processing (DDSP) [12] is one approach for efficient, high quality audio rendering from