Generating Detailed Music Datasets with Neural Audio Synthesis

Yusong Wu [1] **Joshua Gardner** [2 3] **Ethan Manilow** [4 3] **Ian Simon** [3] **Curtis Hawthorne** [3] **Jesse Engel** [3]

Abstract

Generative models are increasingly able to generate realistic high-quality data with systematic control over conditional attributes. These models are ideally suited for creating voluminous and detailed synthetic datasets, which has led to large improvements in low-resource tasks in language and vision. However, such generative data amplification has yet to be demonstrated for the domain of music (symbolic music (i.e. MIDI) or raw audio). In this work, we address this gap by using a generative model of MIDI (Coconet trained on Bach Chorales) with a structured audio synthesis model (MIDI-DDSP trained on URMP). We demonstrate a system capable of producing unlimited amounts of realistic chorale music with rich annotations through controlled synthesis of MIDI through generative models. We call this system the **Chamber Ensemble Generator (CEG)**, and use it to generate a large dataset of chorales (CocoChorales). We demonstrate that data generated using our approach improves state-of-the-art models for music transcription and source separation, and we release both the system and the dataset as an open-source foundation for future work.

1. Introduction

Recent advances in generative modeling have led to systems that can produce realistic, high-quality data in the image (Ramesh et al., 2022), video (Wu et al., 2021), speech (Oord et al., 2016; Shen et al., 2018) and music domains (Dhariwal et al., 2020). Other than generating one-off examples given a user input, generative models can be used to create large-scale datasets with high-quality labels, which in turn can be used to train downstream supervised models. This process is called *dataset amplification*[1], where a generative model is used to create larger datasets from existing data, thus "amplifying" small amounts of data (Jahanian et al., 2022; Zhang et al., 2021; Christiano et al., 2018; Axelrod et al., 2020). It is now well-established that neural networks perform better when they are larger and have access to more data (Kaplan et al., 2020; Brown et al., 2020; Henighan et al., 2020). Thus, applying dataset amplification by generating high-quality data and labels could ease the amount of human annotations needed or improve the resulting model trained on an "amplified" dataset. For example, Zhang et al. (2021) trained an additional semantic map decoder on the latent space of a pre-trained StyleGAN (Karras et al., 2019), making it an infinite dataset generator with paired semantic maps; this achieved similar performance as a model trained on 100x more annotated data.

Amplifying data is particularly promising for the domain of Music Information Retrieval (MIR). In MIR tasks, data is often costly to collect and label, and the scaling of MIR systems is bottlenecked by the lack of data with high-quality labels. In fact, MIR tasks that do have abundant data–such as music tagging (Bogdanov et al., 2019; Won et al., 2019; 2020) or piano transcription (Hawthorne et al., 2019; Kong et al., 2021; Hawthorne et al., 2021)–have seen performance gains with architectures (e.g., Transformers (Vaswani et al., 2017)) that can take advantage of large-scale data. Given a sufficiently good generative model trained on a small dataset, one could use the model to create a large amount of data and paired labels. This data could then be used to train other large models, which would be impossible with the original dataset.

Despite having powerful generative models in both symbolic music (chorales (Huang et al., 2017), piano performance (Huang et al., 2018), or symphonic (Liu et al., 2022)) and audio (Dhariwal et al., 2020; Castellon et al., 2020; Dong et al., 2022; Wu et al., 2022), dataset amplification for music data is relatively rare. Some existing work investigates generating symbolic note data (Peracha, 2021; Liu et al., 2020), but excludes audio synthesis. Other works create synthetic audio datasets using MIDI synthesizers (Engel et al., 2017;

[*]Equal contribution [1]Université de Montréal, Mila [2]University of Washington [3]Google Brain [4]Northwestern University. Correspondence to: Yusong Wu <yusong.wu@umontreal.ca>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

[1]"Amplification" here does not mean using an electronic amplifier (or amp simulation software), but rather the *expansion* of a dataset using generative models.

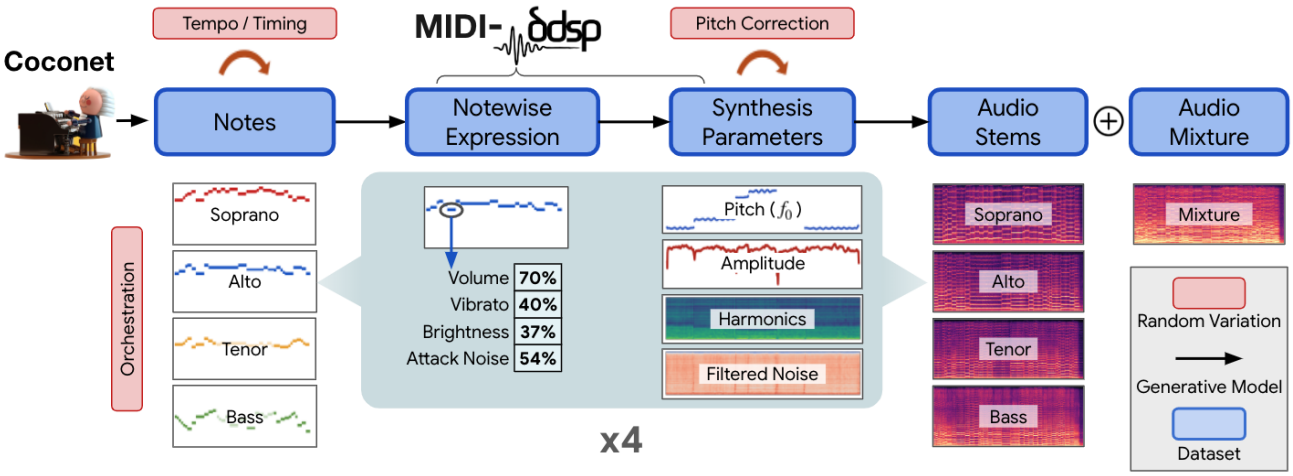

Figure 1. The **Chamber Ensemble Generator (CEG)** generates a dataset of realistic-sounding performances of Bach Chorales by pipelining two generative models, CocoNet (Huang et al., 2017) and MIDI-DDSP (Wu et al., 2022). Because these models form a structured hierarchy, we are able to produce data at many different stages in the generation process, including aligned note data with instrument labels, notewise expression data, synthesis parameter data, audio stem data, and audio mixture data. The structured hierarchy also enables us to manually add manipulation at certain stages to introduce variation. In the dataset accompanying this paper, **CocoChorales**, we vary the orchestration of the pieces and overall tempo, we alter the microtiming of notes, and we randomly apply f_0 augmentation to the performances.

Manilow et al., 2019), but due to being synthesized from MIDI, the audio lacks the realism that a generative model might provide. We are unaware of any attempts in the MIR domain to apply data amplification to create large audio datasets with paired labels.

In this paper, we present the **Chamber Ensemble Generator (CEG)**, a pipeline of structured generative models which we use to create a large synthetic audio dataset. Using structured models produces many types of high-quality labels and allows us to manipulate the generative processes in many different ways. Specifically, we use a music composition model, Coconet (Huang et al., 2017), to generate four-part note data in the style of Bach Chorales, and we use an audio generation model, MIDI-DDSP (Wu et al., 2022), to turn the note data into audio for each instrument in the ensemble. We use the CEG to generate $240,000$ performances containing audio mixture data with high-quality annotations for stems, MIDI, note-level performance attributes, and fine-grained synthesis parameters. We call the resulting dataset **CocoChorales**, and we make both data and code publicly available.[2] We show that state-of-the-art models for music transcription and source separation benefit greatly from the expanded dataset and accompanying labels.

[2] https://chamber-ensemble-generator.github.io/

2. The Chamber Ensemble Generator

In this section, we will first describe the two generative models used in our Chamber Ensemble Generator (CEG), which is used to create the CocoChorales dataset. Then we will introduce each level of data manipulation while providing our choice of manipulation used in CocoChorales. Last, we will show an analysis of the CocoChorales. The dataset creation process and paired data included for each piece are illustrated in Figure 1.

2.1. Coconet

Coconet (Huang et al., 2017) is a music composition model that generates four-part Bach harmony. Coconet is trained on J.S. Bach chorales dataset (Boulanger-Lewandowski et al., 2012; jsb, 2022), which consists of 382 pieces. Coconet trains a convolutional neural networks to complete partial musical scores by infilling a randomly masked input score. During inference, Coconet iteratively applies blocked Gibbs sampling to rewrite its generation and samples a music score.

In this paper, we used an open-source implementation of Coconet (coc, 2022). However, our model slightly differs from the original Coconet in the following ways: (1) We apply pitch augmentation during training by randomly transposing the input by $\{-3, -2, -1, 0, 1, 2, 3\}$ semitones. (2) The loss used during training is simply unweighted cross-entropy, compared to the reweighted loss in the original

paper (Huang et al., 2017). (3) Our model does not generate rest notes.

2.2. MIDI-DDSP

MIDI-DDSP (Wu et al., 2022) is a score-to-audio generation model which uses a three-level structured hierarchy (notes, performance, synthesis) when generating single-note audio. Given input MIDI, the audio synthesis via MIDI-DDSP proceeds as follows. First, MIDI-DDSP generates a set of "note expression" characteristics for each note, each controlling one aspect of a note's performance: volume, volume fluctuation, volume peak position, vibrato, brightness, attack noise. Second, MIDI-DDSP uses the note expressions and the accompanying MIDI to generate frame-wise synthesis parameters. The synthesis parameters of an audio clip consist of a fundamental frequency f_0, an amplitude curve, a distribution of amplitudes for each harmonic frequency above the f_0, and a set of filtered noise magnitudes. Finally, the Differentiable Digital Signal Processing (DDSP) (Engel et al., 2020) modules synthesize a waveform using the generated synthesis parameters. Figure 1 provides an overview of this process.

The intermediate representations (i.e., note expressions and synthesis parameters) generated by MIDI-DDSP can provide rich annotations for many MIR tasks. For example, these representations can be used for tasks like multi-f_0 estimation or performance analysis.

MIDI-DDSP is trained on the URMP dataset (Li et al., 2018), which consists of 3.75 hours of solo recordings. MIDI-DDSP is thus capable of generating the 13 common orchestral instruments present in URMP: violin, viola, cello, double bass, flute, oboe, clarinet, saxophone, bassoon, trumpet, horn, trombone, and tuba.

3. CocoChorales

Using our Chamber Ensemble Generator pipeline, we generated a dataset which we call **CocoChorales**. CocoChorales consists of $240,000$ pieces, totaling 1411 hours of mixture data. The CocoChorales is orders of magnitude larger than existing MIR datasets (Li et al., 2018; Rafii et al., 2017; Foster et al., 2021; Boulanger-Lewandowski et al., 2012; jsb, 2022; Thickstun et al., 2017; Manilow et al., 2019). Every example in the dataset contains MIDI data, note expression data, synthesis parameter data, and audio for each instrument stem, audio of the mixture, and additional metadata about tempi, ensemble type, etc. We make train/valid/test split of CocoChorales in $80\%/10\%/10\%$ portion of the overall data. Further information about CocoChorales–including download links–can be found in the online supplement. The rest of this section is dedicated to describing the creation process for CocoChorales.

3.1. MIDI Generation and Augmentation

Coconet (Huang et al., 2018) is a generative model of Bach Chorales. We use Coconet to compose 8 measures of chorale, each conforming to the standard four-part chorale structure (Soprano, Alto, Tenor, Bass, or SATB) in $\frac{4}{4}$ time, and output the compositions in MIDI format. We generate samples from Coconet by running 1024 sampling steps.

In order to ensure that each part generated by Coconet falls into the range of expected pitches for the given SATB part (e.g., a soprano note should not be too low), we reject a generated sample piece if any of the pitches fall 3 semitones outside of the min/max pitch used in the J.S. Bach Chorales dataset (Boulanger-Lewandowski et al., 2012) for that part.

Once we have a set of valid chorales as MIDI, we augment this data to add more variety to the dataset. Because we are operating on MIDI data rather than audio data, we can apply tempo and timing variations without worrying about a time-stretching algorithm introducing artifacts. To that end, we randomly set the tempo (in BPM) to an integer drawn from $\text{Uniform}([50, 150])$. Furthermore, the output of Coconet is quantized at the granularity of 16^{th} notes, so to add an extra level of expressiveness to the MIDI, we add microtiming offsets to the notes. Following the observation that human timing approximates a normal distribution (Sogorski et al., 2018; Naveda et al., 2011), we add a random timing offset to each note sampled from a truncated normal distribution between $[-50, 50]$ ms with $\mu = 0$ ms, $\sigma = 15$ ms.

The final way that we add variation to the MIDI data is by changing the orchestration of the ensembles. We define four ensembles–String, Brass, Woodwind, and Random–with $60,000$ examples in each. The first three ensembles have a fixed orchestration throughout, and the random ensemble has a varied orchestration, with instruments randomly selected from a pool of instruments according to the pitch range of the SATB part (see Section B).

Once a composition is generated by Coconet, tempi are chosen, microtiming added to the notes, and the orchestration is determined, the MIDI dataset is given to MIDI-DDSP to synthesize into audio performances.

3.2. Audio Synthesis and Mixing

After the MIDI data is generated, MIDI-DDSP is used to synthesize the MIDI into a realistic-sounding performance. Because MIDI-DDSP can only synthesize monophonic audio, each SATB part is rendered separately. As described in Section 2.2, MIDI-DDSP offers multiple ways to manipulate the sonic characteristics of its output.

The first way MIDI-DDSP output can be manipulated is by making edits to the note expression parameters (e.g., note-wise volume, vibrato, etc), and even though it is possible

to manually edit these, here we opt to use the expressions that are automatically generated by MIDI-DDSP without manipulation or augmentation.

The second way MIDI-DDSP output can be manipulated is by altering the synthesis parameters, which directly influences how the output sounds. MIDI-DDSP is trained on the URMP dataset (Li et al., 2018), which contains instrument intonations that deviate from equal temperament tuning and is thus reflected in MIDI-DDSP's output. We apply a random amount of f_0 augmentation, which helps ensure that the synthesized audio corresponds to the MIDI notes, while allowing a realistic amount of deviation from "perfect" equal temperament tuning in the output audio. This avoids the bias of perfect intonation and makes models trained on CocoChorales robust to intonation errors that are likely to occur in real-life music performances.

MIDI-DDSP predicts an f_0 curve in semitone space as a decimal offset from the in-tune integer pitch value. We randomly transpose the pitch value of each note like so

$$\hat{f}_0 = f_0^{\text{Note}} + \hat{f_0^{\Delta}} - \alpha\, \bar{f_0^{\Delta}} \text{ where } \alpha \sim \mathcal{U}[0,1], \quad (1)$$

and $\hat{f}_0$ is the new fundamental frequency in semitones, f_0^{Note} is the known frequency of the note in semitones, $\hat{f_0^{\Delta}}$ is the model's predicted f_0 offset in semitones, $\bar{f_0^{\Delta}}$ is the model's predicted f_0 offset in semitones averaged across the note, and α is a random scaling factor.

We synthesize each of the four instrument parts independently and save the audio at 16kHz and 16-bit PCM. We mix each of the four instrument stems following the mixing strategy used by Slakh (Manilow et al., 2019): First, each stem is normalized to have integrated loudness of -13dB, calculated according to the ITU-R BS.1770-4 specification (BS.1770-4, 2017). Second, all the stems are summed create a single mixture. Finally, if that mixture has a peak loudness larger than -1dB, a uniform gain is applied to the mix (and stems) to ensure that the mix does not clip.

4. Experiments

We conduct experiments in multitrack music transcription and source separation to demonstrate the benefits of additional data. Our goal in these experiments is not to propose novel models or architectures for these tasks; instead, it is to demonstrate how the additional data in CocoChorales can be leveraged to improve existing models for the respective tasks. Experiment details are provided in Section C.

For music transcription, we conduct two experiments to determine whether data amplification can help transcription model in learning on a "low-resource" dataset, and whether it can improve state-of-the-art transcription systems. We use the MT3 model of (Gardner et al., 2022), the current state-of-the-art in music transcription, in these experiments. Our results show that CocoChorales increases transcription performance on URMP by approximately 100% as measured by both multi-instrument onset-offset F1 and (instrument-agnostic) onset-offset F1 score (Table 2). When adding the CocoChorales training dataset to the dataset combination used to train the MT3 model, CocoChorales improves transcription performance on the URMP datasest without cost to performance on any other datasets, but does not improve performance on other datasets (Table 3).

For source separation, we show that dataset amplification enables source separation models to train on instruments that were previously scarce in data (Table 4). We further show an example of source separation on Woodwind ensembles in the URMP dataset. We note that the original URMP dataset only contains less than 4 *minutes* of Woodwind ensemble audio, whereas in CocoChorales, we amplify to have 360 *hours*, which significantly improves the learned model when using CocoChorales.

5. Conclusion and Future Applications

In this paper we introduce the Chamber Ensemble Generator (CEG), which we use to produce the CocoChorales dataset. The CEG is a combination of a generative model for notes (Coconet) and a generative model for audio (MIDI-DDSP) that we set up as a structured hierarchy. In doing this, we can produce an unlimited amount of chamber ensemble mixture data with a rich set of aligned annotation data, with note data, notewise expression data, synthesis parameter data and stem audio data. Using CocoChorales, we trained a state-of-the-art transcription system and showed how the data can boost performance on low-resource transcription datasets. We also showed separation results for sources on which prior separation work has underperformed.

The experiments we show in Section C are only two of many possible applications enabled by the Chamber Ensemble Generator and CocoChorales. Because of the rich annotations in the dataset, we are excited by the many other potential applications that this work will enable. Future applications could include performance analysis (Lerch et al., 2019; Pati et al., 2018) (e.g., using the note expressions in the dataset, or deriving new ones from the synthesis parameters), multi-f_0 estimation (Bittner et al.; Cuesta et al., 2020) (e.g., using the f_0's in this dataset used to synthesize each instrument), or new advances in source separation (e.g., separating multiple instances of similar sounding sources like the string ensemble (Petermann et al., 2020; Schulze-Forster et al., 2022; Kawamura et al., 2022), or separating random ensembles (Lee et al., 2019; Manilow et al., 2020b; Chen et al., 2022)). We look forward to the new directions the MIR community will explore using this data and what new variations on dataset amplification will be explored.

References

Coconet-pytorch. https://github.com/lukewys/coconet-pytorch, 2022. [Online; accessed 01-May-2022].

JSB-Chorales-dataset. https://github.com/czhuang/JSB-Chorales-dataset, 2022. [Online; accessed 01-May-2022].

Axelrod, B., Garg, S., Sharan, V., and Valiant, G. Sample amplification: Increasing dataset size even when learning is impossible. In *International Conference on Machine Learning*, pp. 442–451. PMLR, 2020.

Bittner, R. M., McFee, B., Salamon, J., Li, P., and Bello, J. P. Deep salience representations for f0 estimation in polyphonic music.

Bogdanov, D., Won, M., Tovstogan, P., Porter, A., and Serra, X. The mtg-jamendo dataset for automatic music tagging. 2019.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *International Conference on Machine Learning*, 2012.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

BS.1770-4, R. I.-R. Algorithms to measure audio programme loudness and true-peak audio level. 2017.

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

Chen, K., Du, X., Zhu, B., Ma, Z., Berg-Kirkpatrick, T., and Dubnov, S. Zero-shot audio source separation through query-based learning from weakly-labeled data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

Chen, Y. and Lerch, A. Melody-conditioned lyrics generation with seqgans. In *2020 IEEE International Symposium on Multimedia (ISM)*, pp. 189–196. IEEE, 2020.

Christiano, P., Shlegeris, B., and Amodei, D. Supervising strong learners by amplifying weak experts. *arXiv preprint arXiv:1810.08575*, 2018.

Cuesta, H., McFee, B., and Gómez, E. Multiple f0 estimation in vocal ensembles using convolutional neural networks. *arXiv preprint arXiv:2009.04172*, 2020.

Défossez, A., Usunier, N., Bottou, L., and Bach, F. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254*, 2019.

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

Dong, H.-W., Zhou, C., Berg-Kirkpatrick, T., and McAuley, J. Deep performer: Score-to-audio music performance synthesis. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 951–955. IEEE, 2022.

Emiya, V., Bertin, N., David, B., and Badeau, R. Maps-a piano database for multipitch estimation and automatic transcription of music. 2010.

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

Engel, J., Hantrakul, L., Gu, C., and Roberts, A. DDSP: Differentiable digital signal processing. In *International Conference on Learning Representations*, 2020.

Ens, J. and Pasquier, P. Building the metamidi dataset: Linking symbolic and audio musical data. In *Proceedings of 22st International Conference on Music Information Retrieval, ISMIR*, 2021.

Foster, D., Dixon, S., et al. Filosax: A dataset of annotated jazz saxophone recordings. In *Proceedings of 22st International Conference on Music Information Retrieval, ISMIR*, 2021.

Gardner, J. P., Simon, I., Manilow, E., Hawthorne, C., and Engel, J. MT3: Multi-task multitrack music transcription. In *International Conference on Learning Representations*, 2022.

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

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

Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Huang, C.-Z. A., Cooijmans, T., Roberts, A., Courville, A., and Eck, D. Counterpoint by convolution. In *Proceedings of 18st International Conference on Music Information Retrieval, ISMIR*, 2017.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

Jahanian, A., Puig, X., Tian, Y., and Isola, P. Generative models as a data source for multiview representation learning. In *International Conference on Learning Representations*, 2022.

Jonason, N., Sturm, B., and Thomé, C. The control-synthesis approach for making expressive and controllable neural music synthesizers. In *2020 AI Music Creativity Conference*, 2020.

Ju, Z., Lu, P., Tan, X., Wang, R., Zhang, C., Wu, S., Zhang, K., Li, X., Qin, T., and Liu, T.-Y. Telemelody: Lyric-to-melody generation with a template-based two-stage method. *arXiv preprint arXiv:2109.09617*, 2021.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Kawamura, M., Nakamura, T., Kitamura, D., Saruwatari, H., Takahashi, Y., and Kondo, K. Differentiable digital signal processing mixture model for synthesis parameter extraction from mixture of harmonic sounds. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 941–945. IEEE, 2022.

Kong, Q., Li, B., Song, X., Wan, Y., and Wang, Y. High-resolution piano transcription with pedals by regressing onset and offset times. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3707–3717, 2021.

Le Roux, J., Wisdom, S., Erdogan, H., and Hershey, J. R. Sdr–half-baked or well done? In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 626–630. IEEE, 2019.

Lee, J. H., Choi, H.-S., and Lee, K. Audio query-based music source separation. *arXiv preprint arXiv:1908.06593*, 2019.

Lerch, A., Arthur, C., Pati, A., and Gururani, S. Music performance analysis: A survey. *arXiv preprint arXiv:1907.00178*, 2019.

Li, B., Liu, X., Dinesh, K., Duan, Z., and Sharma, G. Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications. *IEEE Transactions on Multimedia*, 21(2): 522–535, 2018.

Liu, A., Fang, A., Hadjeres, G., Seetharaman, P., and Pardo, B. Incorporating music knowledge in continual dataset augmentation for music generation. *arXiv preprint arXiv:2006.13331*, 2020.

Liu, J., Dong, Y., Cheng, Z., Zhang, X., Li, X., Yu, F., and Sun, M. Symphony generation with permutation invariant language model. *arXiv preprint arXiv:2205.05448*, 2022.

Manilow, E., Wichern, G., Seetharaman, P., and Le Roux, J. Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019.

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

Manilow, E., Wichern, G., and Le Roux, J. Hierarchical musical instrument separation. In *International Society for Music Information Retrieval (ISMIR) Conference*, pp. 376–383, 2020b.

Naveda, L., Gouyon, F., Guedes, C., and Leman, M. Microtiming patterns and interactions with musical properties in samba music. *Journal of New Music Research*, 40(3): 225–238, 2011.

Nikolenko, S. I. et al. *Synthetic data for deep learning*. Springer, 2021.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Pati, K. A., Gururani, S., and Lerch, A. Assessment of student music performances using deep neural networks. *Applied Sciences*, 8(4):507, 2018.

Peracha, O. Js fake chorales: a synthetic dataset of polyphonic music with human annotation. *arXiv preprint arXiv:2107.10388*, 2021.

Petermann, D., Chandna, P., Cuesta, H., Bonada, J., and Gómez, E. Deep learning based source separation applied to choir ensembles. *arXiv preprint arXiv:2008.07645*, 2020.

Raffel, C. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

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

Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., and Bittner, R. The MUSDB18 corpus for music separation, December 2017. URL https://doi.org/10.5281/zenodo.1117372.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Schulze-Forster, K., Doire, C. S., Richard, G., and Badeau, R. Unsupervised audio source separation using differentiable parametric source models. *arXiv preprint arXiv:2201.09592*, 2022.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4779–4783. IEEE, 2018.

Sogorski, M., Geisel, T., and Priesemann, V. Correlated microtiming deviations in jazz and rock music. *PloS one*, 13(1):e0186361, 2018.

Thickstun, J., Harchaoui, Z., and Kakade, S. Learning features of music from scratch. 2017.

Tremblay, J., To, T., and Birchfield, S. Falling things: A synthetic dataset for 3d object detection and pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2038–2041, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, B. and Yang, Y.-H. Performancenet: Score-to-audio music generation with multi-band convolutional residual network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1174–1181, 2019.

Won, M., Chun, S., and Serra, X. Toward interpretable music tagging with self-attention. *CoRR*, abs/1906.04972, 2019. URL http://arxiv.org/abs/1906.04972.

Won, M., Ferraro, A., Bogdanov, D., and Serra, X. Evaluation of cnn-based automatic music tagging models. *arXiv preprint arXiv:2006.00751*, 2020.

Wood, E., Baltrušaitis, T., Hewitt, C., Dziadzio, S., Cashman, T. J., and Shotton, J. Fake it till you make it: Face analysis in the wild using synthetic data alone. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3681–3691, 2021.

Wu, C., Liang, J., Ji, L., Yang, F., Fang, Y., Jiang, D., and Duan, N. N\" uwa: Visual synthesis pre-training for neural visual world creation. *arXiv preprint arXiv:2111.12417*, 2021.

Wu, Y., Manilow, E., Deng, Y., Swavely, R., Kastner, K., Cooijmans, T., Courville, A., Huang, C.-Z. A., and Engel, J. MIDI-DDSP: Detailed control of musical performance via hierarchical modeling. In *International Conference on Learning Representations*, 2022.

Xi, Q., Bittner, R. M., Pauwels, J., Ye, X., and Bello, J. P. Guitarset: A dataset for guitar transcription. In *ISMIR*, pp. 453–460, 2018.

Zhang, Y., Ling, H., Gao, J., Yin, K., Lafleche, J.-F., Barriuso, A., Torralba, A., and Fidler, S. Datasetgan: Efficient labeled data factory with minimal human effort. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10145–10155, 2021.

Zhao, W., Queralta, J. P., and Westerlund, T. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 737–744. IEEE, 2020.

A. Further Related Work

Training on synthetic data to help real-world downstream applications has long been used in Machine Learning, and Reinforcement Learning (Engel et al., 2017; Nikolenko et al., 2021; Zhao et al., 2020; Tremblay et al., 2018; Axelrod et al., 2020). Recently, the use of generative models for dataset amplification, i.e., enlarging the dataset by generating large synthetic datasets, has become more common in vision and other domains (Jahanian et al., 2022; Zhang et al., 2021; Christiano et al., 2018; Wood et al., 2021). In the image domain, Zhang et al. (Zhang et al., 2021) trained an additional semantic map decoder on the latent space of a pre-trained StyleGAN (Karras et al., 2019). By training the decoder on a few annotated examples, the model becomes an infinite dataset generator with paired semantic maps, achieving the similar performance as a model trained on 100x more annotated data. Here, we implement a similar approach in the music domain, with the hope of enlarging the effective size of our training data. Specifically, we train a set of generative models from scratch instead of using a pre-trained model.

Synthetic datasets have been proposed a handful of times in the music domain. The JS Fake Chorales dataset (Peracha, 2021) consists of 500 synthetic Bach Chorales generated by an RNN-based generative model. However, the JS Fake Chorales dataset is small compared to this work and consists of symbolic note data, lacking any audio performance data. Similarly, Liu et. al. (Liu et al., 2020) propose a system to grade model's generated data and feed high-quality data from the model back into its training set. This work, also, only used symbolic data and no audio performance data. The Synthesized Lakh (Slakh) Dataset (Manilow et al., 2019) is constructed by synthesizing 2100 MIDI files using professional-grade sample-based synthesizers, resulting in 145 hours of mixture audio with paired stem audio and MIDI files. However, Slakh lacks finer-grained annotations like f_0's for each note, and due to being synthesized from MIDI, it does not sound like it was performed by live musicians.

Many possible generative music models could be used to amplify or generate a dataset. One could use a composition model (piano performance (Huang et al., 2018), singing (Ju et al., 2021; Chen & Lerch, 2020), or symphonic (Liu et al., 2022)) paired with a score-to-audio synthesis model (Wang & Yang, 2019; Jonason et al., 2020; Castellon et al., 2020; Dong et al., 2022), or even use a more general-purpose audio generation model (Dhariwal et al., 2020). However, in this work we chose Coconet (Huang et al., 2017) and MIDI-DDSP (Wu et al., 2022) because of the high amount of structure these models contain (i.e., Coconet generates Bach chorales, and MIDI-DDSP has a 3-level hierarchy), and thus they enable more variation of the generation process and support more tasks.

B. Additional Info about the Dataset

In Table 1, we compare many existing datasets to our proposed dataset, CocoChorales. Because we leverage generative models in our Chamber Ensemble Generator (CEG), we are able to create an order of magnitude more data than previous datasets with paired audio & MIDI data.

In CocoChorales, we define four ensembles with different sets of instrumentation. Each ensemble has four instruments, each one serving one of the Soprano, Alto, Tenor, or Bass parts (SATB). Details about each ensembles, in SATB order, are shown below:

- **String**: Violin 1, Violin 2, Viola, Cello.
- **Brass**: Trumpet, French Horn, Trombone, Tuba.
- **Woodwind**: Flute, Oboe, Clarinet, Bassoon.
- **Random**: Each part is randomly assigned an instrument according to:
  - Soprano: Violin, Flute, Trumpet, Clarinet, Oboe.
  - Alto: Violin, Viola, Flute, Clarinet, Oboe, Saxophone, Trumpet, French Horn.
  - Tenor: Viola, Cello, Clarinet, Saxophone, Trombone, French Horn.
  - Bass: Cello, Double Bass, Bassoon, Tuba.

Table 1. This table compares our work, CocoChorales (last 5 rows), to previously proposed datasets: Bach Chorales (Boulanger-Lewandowski et al., 2012), JS Fake Chorales (Peracha, 2021), Lakh MIDI (Raffel, 2016), Meta MIDI (Ens & Pasquier, 2021), MAPS (Emiya et al., 2010), MAESTRO (Hawthorne et al., 2019), GuitarSet (Xi et al., 2018), FiloSax (Foster et al., 2021), MUSDB18 (Rafii et al., 2017), MusicNet (Thickstun et al., 2017), URMP (Li et al., 2018), Slakh (Manilow et al., 2019). Durations are counted in hours of mixture data (where applicable) and is rounded to the nearest hour. Duration of Bach Chorales is calculated as if every piece is in 60 BPM, durations for the other MIDI dataset use the tempos provided.

Name	Instrumentation	# Examples	Duration (hrs)	Content
MIDI-only Datasets				
Bach Chorales	4-part chorale	382	26	MIDI
JS Fake Chorales	4-part chorale	500	1	MIDI
Lakh MIDI	128 MIDI instruments	176,581	10,521	MIDI
Meta MIDI	128 MIDI instruments	436,631	19,224	MIDI
Audio/MIDI Datasets – Single Instrument				
MAPS	Piano	270	18	Audio, MIDI
MAESTRO	Piano	1,276	199	Audio, MIDI
GuitarSet	Guitar	360	3	Audio, MIDI, Multi-f_0, Tempo, Chords
FiloSax	Saxophone	240	24	Audio, MIDI
Audio/MIDI Datasets – Ensembles				
MUSDB18	4 instruments	150	10	Mix and stem audio
MusicNet	11 instruments	330	34	Mix audio, MIDI
URMP	14 instruments	23	1	Mix and stem audio, video, MIDI
Slakh	34 instruments	2,100	145	Mix and stem audio, MIDI
CocoChorales (ours)				
String	String ensemble	60,000	350	Mix and stem audio, MIDI, note expression, synthesis parameters
Brass	Brass ensemble	60,000	350	
Woodwind	Woodwind ensemble	60,000	350	
Random	Random instruments	60,000	350	
Total	13 instruments	240,000	1,400	

Table 2. Transcription results when training an MT3 (Gardner et al., 2022) model on URMP (Li et al., 2018) alone versus training one on URMP and CocoChorales, higher is better. URMP is a small dataset (1 hour), so using dataset amplification, as done in here to create CocoChorales, has a large effect on transcription performance.

Training Dataset(s)	**On/Off F1**	**Multi-Inst. F1**
URMP	0.28	0.22
URMP + CocoChorales	**0.55**	**0.48**

Table 3. Transcription results on the combination of MT3 datasets from (Gardner et al., 2022), higher is better. (Note that we use the MIDI-DDSP (Wu et al., 2022) train-test split for URMP, not the MT3 train-test split for URMP.)

Model	**MAESTRO**	**Cerberus4**	**GuitarSet**	**MusicNet**	**Slakh2100**	**URMP**
Onset-Offset F1						
MT3 Datasets	0.83	0.80	0.78	0.33	0.57	0.61
+ CocoChorales	0.83	0.80	0.79	0.34	0.57	**0.66**
Multi-Instrument F1						
MT3 Datasets	0.83	0.79	0.78	0.30	0.58	0.50
+ CocoChorales	0.83	0.75	0.79	0.30	0.57	**0.56**

Table 4. Mean separation results (dB) for the Woodwind ensemble over the CocoChorales test set using SI-SDR (Le Roux et al., 2019), higher is better.

Network	Flute	Oboe	Clarinet	Bassoon
Demucs v2 (Défossez et al., 2019)	18.7	17.3	21.0	20.7
MI+TR (Manilow et al., 2020a)	12.5	6.9	10.7	6.9

C. Experiments

C.1. Music Transcription

We conduct two sets of music transcription experiments as a demonstration of the effectiveness of our Chamber Ensemble Generator (CEG) and the resulting CocoChorales dataset.

Music transcription, the task of producing a symbolic representation from a raw waveform, is often challenging due to the "low-resource" nature of many transcription datasets – that is, high-quality paired data (audio and aligned note annotations) is scarce and expensive to collect. Dataset amplification, as proposed in this paper, has the potential to provide unlimited data, alleviating this resource limitation. However, whether data from generative models can improve music transcription models is yet unproven.

To explore this, we perform a set of experiments designed to investigate how dataset amplification can improve existing state-of-the-art transcription systems, both in a very low-resource setting (on URMP (Li et al., 2018), the "amplified" dataset in our study) and in a mixed setting where we combine many existing transcription datasets of various sizes. For all experiments, we use the MT3 transcription model (Gardner et al., 2022), with no modifications. For all experiments, we use MIDI-DDSP's train/test split of URMP in order to ensure that no inputs used to train the MIDI-DDSP generative model occur in the test set for the transcription model.

Our first study compares the transcription performance of an MT3 model trained only on URMP to an identical model trained on a combined CocoChorales and URMP dataset. This allows us to observe the effectiveness of dataset amplification on the original dataset, URMP. The results of this experiment are shown in Table 2. Our results clearly demonstrate the usefulness of our proposed approach, with the transcription performance on URMP increasing approximately 100% as measured by both multi-instrument onset-offset F1 and (instrument-agnostic) onset-offset F1 score. For details on the computation of this score, we refer the reader to (Raffel et al., 2014).

Our second study evaluates whether the addition of the CocoChorales can help improve MT3's transcription performance even in the presence of a more diverse collection of datasets. We do this by adding the CocoChorales training dataset to the original dataset combination used to train the MT3 model, and compare an MT3 model trained on the original combination of datasets to an MT3 model trained on the union of those datasets with CocoChorales. This mixture, described in detail in (Gardner et al., 2022), contains six standard transcription datasets with a mix of styles, instrumentations, and audio characteristics (i.e. synthesized vs. real).

The results of our second study are shown in Table 3. The results show that our dataset amplification improves transcription performance on the URMP dataset even over the performance of the state-of-the-art MT3 model trained on the largest available combination of datasets. We note that this improvement from the addition of CocoChorales does not come at a cost to performance on any other datasets, but it also does not improve performance on the datasets besides URMP. To our knowledge, this model also achieves the best published transcription results on the URMP dataset.

C.2. Source Separation

To further demonstrate the usefulness of the Chamber Ensemble Generator (CEG) and resulting CocoChorales dataset, we conduct a set of brief source separation experiments trained on the CocoChorales dataset. The majority of music separation systems have focused on separating sources where data is ample, which has historically excluded many instruments (e.g., flutes, oboes, clarinets, or bassoons). Because the CEG enables us to generate a limitless quantity of high-quality ensemble data, we are able to train separation systems on instruments that have been neglected by existing separation systems.

To that end, we train two separation networks to separate instruments from the Woodwind ensemble in CocoChorales.

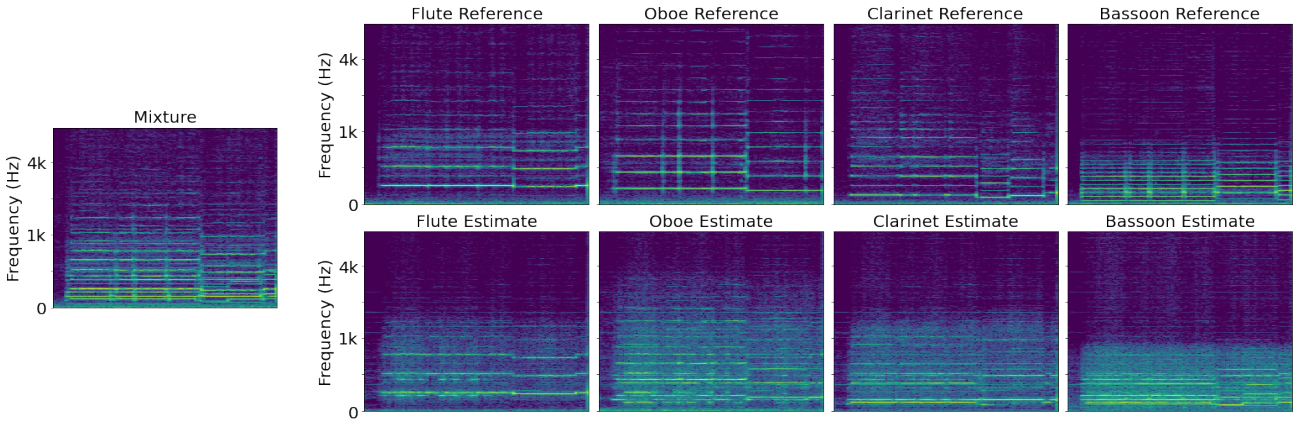

Figure 2. A Woodwind ensemble from URMP (Li et al., 2018), where the mixture (left) is separated by a Demucs v2 (Défossez et al., 2019) (bottom row) trained on CocoChorales. The ground truth, reference signals are shown in the top row.

Specifically, we train a Demucs v2 (Défossez et al., 2019), which is a state-of-the-art waveform-to-waveform U-Net, and a Cerberus-style (Manilow et al., 2020a) separation network, which is a set of 4 LSTM layers that create a mask which is applied to the mixture spectrogram. The Cerberus-style network is the same as Cerberus, except we omit the clustering head and only use the Mask Inference and TRanscription heads. We refer to this as MI+TR in Table 4, and only report its separation performance. For more network details, refer the reader to (Défossez et al., 2019; Manilow et al., 2020a). We train these models for 55k steps and report results over all 5 second segments of the CocoChorales test set in Table 4, where we report mean SI-SDR (Le Roux et al., 2019).

Furthermore, we also showcase a separation example from URMP (Li et al., 2018) in Figure 2. As noted, URMP alone is an extremely small dataset to train a separator on. To train a woodwind ensemble separator, URMP only has two recordings that are woodwind ensembles, totalling less than 4 minutes. Even if one wanted to just make an oboe separator, URMP has less than 12 *minutes* of oboe data, which would quickly lead to overfitting using modern networks! For this reason, we do not have a URMP-trained baseline for Table 4. With CocoChorales, the woodwind ensemble has 360 *hours* of oboe data, which enables us to train separation models that work well on URMP.