ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10095540

EXPLORING APPROACHES TO MULTI-TASK AUTOMATIC SYNTHESIZER PROGRAMMING

Daniel Faronbi[1] *Iran Roman*[1] *Juan Pablo Bello*[1]

[1] Music and Audio Research Laboratory, New York University, New York, USA

ABSTRACT

Automatic Synthesizer Programming is the task of transforming an audio signal that was generated from a virtual instrument, into the parameters of a sound synthesizer that would generate this signal. In the past, this could only be done for one virtual instrument. In this paper, we expand the current literature by exploring approaches to automatic synthesizer programming for multiple virtual instruments. Two different approaches to multi-task automatic synthesizer programming are presented. We find that the joint-decoder approach performs best. We also evaluate the performance of this model for different timbre instruments and different latent dimension sizes.

Index Terms— Automatic Synthesizer Programming, VAE, Synthesizer Sound Matching

1 Introduction

Since the advent of electronic instruments, sound synthesizers have become increasingly important in many different genres of music. They are most popular in electronic dance music (EDM) genres, but have also seen prominence in film scores, jazz fusion, contemporary classical music, and many other genres. Despite the ubiquity of the synthesizer as an instrument, understanding how to program the synthesizer to achieve a specific sound still presents a challenge to many aspiring music producers. As such, it may be of use to provide a tool to allow for people to more easily be able to associate specific sounds with synthesizer parameters.

Automatic Synthesizer Programming (ASP) aims to fix this problem. The goal is to build algorithms that can extract synthesizer parameter data from a given audio signal. Previous approaches have accomplished this task for a single virtual instrument at a time, that is the extraction of parameters from audio is specific to a given synthesizer. However, this has severe limitations for real world usage. Most music producers use many synthesizers, and the above formulation would mean learning separate ASP models for each of them. Alternatively, we propose novel methods for performing parameter inference for several synthesizers at a time. We achieve this by using a multi-task ASP framework. Code and data for this project is available on Github[1] and Zenodo[2].

2 Literature Review

There have been many different approaches to ASP throughout the years. These include harmonic analysis [1], genetic algorithms [2, 3, 4], machine learning approaches like linear models [5], convolutional neural networks [6, 7], differiential DSP [8, 9], and graphical methods [10]. All of these approaches have shown some success, however the introduction of Variational Autoencoders (VAEs) [11] stands out as especially promising. VAEs have been widely used in deep learning because of their ability to generate new samples based on interpretable latent codes. Generative applications where specific controls are needed are suited for VAEs. In the field of audio and music, VAEs have been used to generate controllable real time audio [12] [8] and music scores [13]. The addition of normalizing flows to VAEs [14] in ASP allowed for very accurate parameter inference [15] since it allowed the model to learn complex multi-modal distributions in its latent space. This along with one-hot encoding [16] for categorical parameters allowed models to converge for synthesizers with high parameter counts.

3 Approach

We propose two multi-task variations of the VAE approach in [15, 16]. Our first method uses a VAE with a decoder for representation learning and separate decoders for each synthesizer parameter vector to be inferred. This approach has the advantage of decoders that could potentially be trained individually for parameter inference of additional synthesizers. Our second method uses the same VAE for representation learning. However, this method uses only a single decoder for every synthesizer. Smaller parameter vectors are padded to equal the dimensions of the output of the decoder. Unlike the original models, we removed normalizing flows to simplify the approach and better isolate the effect of the multi-task variations.

3.1 Model

We construct two different multi-task ASP models as shown in *Figure 1*. Each model has a mel spectrogram

[1] https://github.com/dafaronbi/Multi-Task-Automatic-Synthesizer-Programming

[2] https://zenodo.org/record/7686668#.ZAodmOzMK3J

encoder $p_\phi(z|x_s)$ and decoder $p_\theta(\hat{x}_s|z)$ for representation learning, where x_s is mel-spectrogram audio data. Each approach also includes n parameter decoders $p_{\psi i}(\hat{p}|z, m_i)$ for parameter inference. Where m_i is a mask inputted into the model to give an understanding of what synthesizer parameter vector is being inferred.

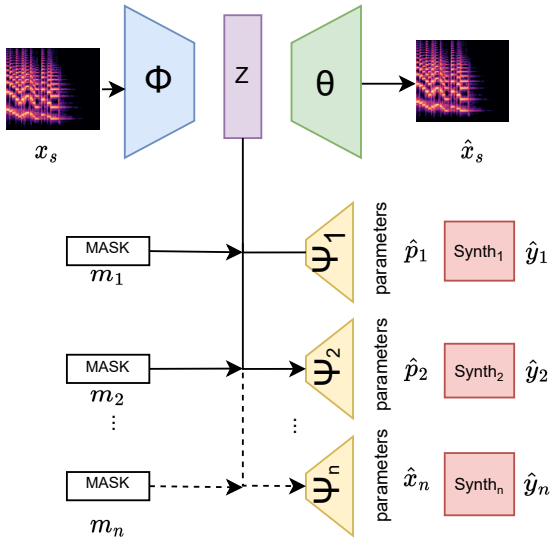

Fig. 1. Diagram for our approach to multi-task ASP.

3.1.1 Separate-Decoder

The separate-decoder approach uses n parameter decoders and n masks for each decoder to infer parameters for each synthesizer. For our experiments, $n = 3$ with k_i parameter dimensions of each synthesizers parameter vector. The output $\hat{p}_i$ varies for each decoder $\hat{p}_1 \in \mathbb{R}^{480}$, $\hat{p}_2 \in \mathbb{R}^{759}$, and $\hat{p}_3 \in \mathbb{R}^{327}$. The masks m_i also vary for each decoder. $m_1 \in \{0,1\}^{480}$, $m_2 \in \{0,1\}^{759}$, and $m_3 \in \{0,1\}^{327}$. The values of the mask are set during training to indicate which synthesizer was used to generate the ground truth audio signal. if $m_i = \{1\}^{k_i}$, the ground truth was generated by the synthesizer i and decoder i's weights will be updated during training. if $m_i = \{0\}^{k_i}$ then that decoder will be ignored during training. For inference, masks can be set to $\{1\}^{k_i}$ or $\{0\}^{k_i}$ depending on the desired decoder's output.

3.1.2 Joint-Decoder

The joint-decoder approach attaches a single decoder to infer parameters. The dimensiality of the output parameters is the same $\hat{p}_i \in \mathbb{R}^{759}$ for all synthesizer parameter vectors, so the parameter vector $\hat{p}$ is padded with zeros so that the size of the vector is always equal to 759. $\hat{p}_1 \in \mathbb{R}^{480} * \{0\}^{279}$, $\hat{p}_2 \in \mathbb{R}^{759} * \{0\}^{0}$, and $\hat{p}_3 \in \mathbb{R}^{327} * \{0\}^{432}$. The dimensionality of all masks in this model are the same $m_i \in \mathbb{R}^{759}$, where $m_1 \in \{1\}^{480} * \{0\}^{279}$, $m_2 \in \{1\}^{759} * \{0\}^{0}$, and $m_3 \in \{1\}^{327} * \{0\}^{432}$ and $*$ is a padding operator to extend the dimensions of a vector space.

3.2 Training

We train each model for 500 epochs using the ADAM optimizer [17]. We use warmup [18] for 100 epochs to ensure that model learns proper spectrogram reconstruction before it begins to regularize. The loss of each model is a combination of the reconstruction loss of the audio representation and parameters, combined with the regularization of the latent space to be close to a unit spherical Gaussian prior $p_\theta(z)$:

$$\begin{aligned}\mathcal{L}_{\phi,\theta,\psi} = & \underbrace{\mathbb{E}_{\phi(z|x_s)}[logp_\theta(\hat{x}_s|z)]}_{\text{spectrogram reconstruction accuracy}} \\ & + \sum_{i=1}^{n} \underbrace{\mathbb{E}_{\phi(z|x_s)}[logp_{\psi_i}(\hat{p}_i|z, m_i)]}_{\text{parameter reconstruction accuracy}} \\ & + \underbrace{\mathbb{E}_{\phi(z|x_s)}[log\mathrm{p}_\theta(z) - logq_\phi(z|x_s)]}_{\text{regularization term}}\end{aligned} \tag{1}$$

4 Experiment Design

We perform experiments to compare the performance of each of our approaches to multi-task ASP. We first compare the accuracy of the parameter inference using three different metrics. Next, we evaluate how accurate the multi-decoder model performs for different timbre presets. Finally, we evaluate the accuracy of the model with different sized latent dimensions.

We construct baseline models with separate encoders and decoders for each synthesizer to provide an understanding of how the multi-task parameter inference models compare to single-task models. Each of these models were trained with only one synthesizer.

To evaluate our model, we use metrics that compare the ground truth and predicted parameter vectors and also compare the audio generated by these parameters. To evaluate the parameter estimation accuracy, we use different metrics for continuous and categorical parameters. For continuous parameters, we simply take the mean squared error between ground truth and predicted parameters. For categorical parameters, we calculate the percentage of parameters that are categorized correctly. For the audio domain, we use log spectral distance (LSD) as in [9]. We generate an STFT $\hat{y}_i$ of the audio signal generated from inferred parameters $\hat{p}_i$ and compare this to the generated STFT y_i of audio generated from ground truth parameters p_i.

$$LSD = \| \log(|\hat{y}|^2) - \log(|y|^2) \| \tag{2}$$

where $\| \cdot \|$ is the frobenius norm of a matrix. This metric captures the distance in spectral characteristics of the audio signal.

4.1 Data

4.1.1 Synthesizers

We collect synthesizer presets from three different virtual instruments: Serum, Diva, and TyrellN6. These synthesizers were chosen because they have a large presence of presets

available online and are diverse in complexity and parameter vector size. Serum has 315 parameters (273 continuous and 42 discrete), Diva has 281 parameters (190 continuous and 91 discrete), and TyrellN6 has 92 paramers (68 continuous and 24 discrete). Previous studies use data sets from synthesizers like the Dexed or custom made synthesizer constructed with Jsyn [19], but we choose these three because of their diversity in synthesis techniques and complexity. Serum is an advanced wave-table synthesizer created by Xfer Records that can create sounds using almost every modern type of synthesis technique (subtractive, additive, FM, etc). Diva is an FM subtractive synthesizer created by U-He that can create analog-sounding sounds. This synthesizer was also used in the first ASP paper with a VAE model [15]. TyrellN6 is a virtual instrument created by U-He. Similarly to Diva, it uses FM and subtractive synthesis to make analog-like sounds. However, its variety of oscillators and filters are much more restrictive when compared to Diva. We collect 2505 presets for Serum, 1694 presets for Diva, and 1870 presets for TyrellN6 and used an 80-10-10 Train-Test-Validate data set split for our experiments.

4.1.2 Parameter Encoding

We chose to encode continuous and categorical parameters differently as in [16]. Continuous parameters keep the same dimensionality but categorical parameters are one hot encoded. This means that categorical parameters take up more dimensions after being one hot encoded. For example, if a synthesizer has parameters $\in \mathbb{R}^2$ where one of the parameters is continuous and one parameter can be one of three categories, our encoding will expand the data to $\mathbb{R}^4$ where the first dimension is the same and the 2-4 dimensions are used to one-hot encode the categorical parameter. When running inference with our models, the model output will select the highest value in the one-hot encoded section of the model and set all others to zero. The parameters will then be collapsed to their original dimension length.

4.1.3 Generating Sound

Presets were gathered for each synthesizer in FXB format. This format was used because it is a preset format that is compatible with multiple digital audio workstations. We use DAW Dreamer[20] to generate 5 second audio samples from all of these presets by holding a note down for three seconds and allowing for attenuation after. We generate an audio sample for midi notes C4-B4 for each preset. We filter out audio samples that have an RMS < -20 dBFS to make sure that silent presets were not included in training. We extract mel-spectrograms from these audio clips using librosa[21] with 128 mel bins and a hop size of 512. Each audio sample is associated with its parameter label. Each parameter sets categorical variables are one hot encoded as in [16]. For each sample we do harmonic percussive source seperation [22] with residual [23] and create timbre groups for when the harmonic component is between 0-20%, 20-40%, 40-60%, 60-80% and 80-100% of the separation.

5 Results & Discussion

Table 1. Metrics for all synthesizers

Model		Metrics					
Encoder	Decoder	LSD		Param MSE		Parm % AC	
		Mean	STD	Mean	STD	Mean	STD
separate	separate	86.8	432.8	5.0	**3.1**	**75.3**	15.7
joint	separate	**72.9**	**297.5**	**4.9**	**3.1**	75.1	15.6
joint	joint	74.2	310.0	5.3	3.2	75.1	**15.5**

Table 2. Metrics for Serum

Model		Metrics					
Encoder	Decoder	LSD		Param MSE		Parm % AC	
		Mean	STD	Mean	STD	Mean	STD
separate	separate	64.9	170.5	**4.7**	**3.5**	**81.7**	**13.3**
joint	separate	**54.9**	**144.6**	4.8	**3.5**	81.1	13.7
joint	joint	55.2	150.6	5.0	3.6	81.2	13.7

Table 3. Metrics for Diva

Model		Metrics					
Encoder	Decoder	LSD		Param MSE		Parm % AC	
		Mean	STD	Mean	STD	Mean	STD
separate	separate	183.6	777.9	6.2	3.1	76.9	10.3
joint	separate	**151.4**	**522.5**	**6.0**	**2.9**	76.5	10.3
joint	joint	156.5	544.3	6.5	3.0	**76.9**	**9.8**

Table 4. Metrics for TyrellN6

Model		Metrics					
Encoder	Decoder	LSD		Param MSE		Parm % AC	
		Mean	STD	Mean	STD	Mean	STD
separate	separate	28.8	42.8	4.2	2.3	66.4	18.0
joint	separate	26.0	**35.9**	**4.1**	**2.2**	**66.6**	17.6
joint	joint	**25.5**	38.1	4.6	2.5	66.1	**17.3**

Model Accuracy: We report the mean and standard deviation of all three metrics for each synthesizer. For mean LSD and MSE, lower is better. For mean parameter accuracy, higher is better. For all standard deviation metrics, lower is better. The best model is marked in bold.

5.1 Model Accuracy

We compare all three of our approaches to multi-task parameter inference approaches by evaluating each model on the same test set data and analyzing how each model performs per synthesizer using the aforementioned metrics. Results for each model are shown in *Tables 1, 2, 3, and 4*. The joint-encoder separate-decoder module outperforms the others overall in the error of generated audio and continuous parameters. A one-way anova test of LSD values revealed statistically significant differences between the groups (p-value of 0.0313). However, the separate-encoder separate-decoder module outperforms others in classification accuracy. This difference is likely due to the separate-encoder separate-decoder module over-fitting to specific synthesizer parameter vectors while the joint-encoder separate-decoder module is forced to learn a general representation for all synthesizers that is more aligned with the generated audio signal. Difference in variance for parameter metrics is negligible between

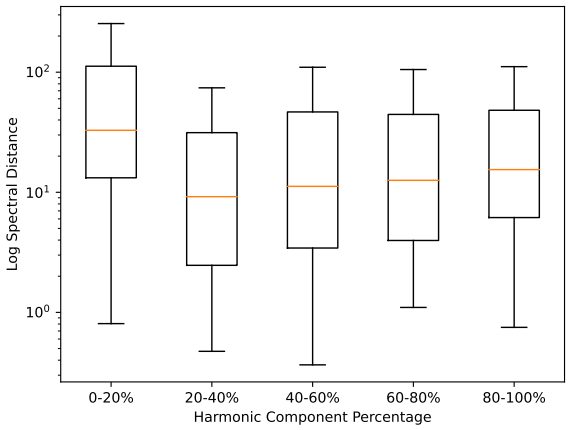

Fig. 2. We plot the change in performance measured by Log Spectral Distance vs the change in the percentage of harmonic component in each test set preset. Lower is better

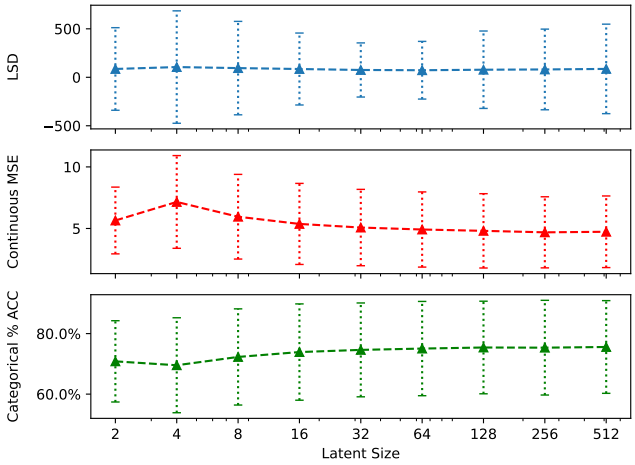

Fig. 3. We plot the change in our metrics as the latent dimensions in our separate-decoder model increase. For MSE and LSD, lower is better. for % accuracy, higher is better

all three modules. However, the variance in audio metrics is always significantly better in the single-encoder separate-decoder module. This is because the separate-decoder module receives more training examples than the separate encoder module and has specialized decoders that the joint-decoder module does not. This allows it to have an advantage when inferring parameters. When comparing performance between synthesizer types, it seems as though simpler synthesizers are easier to estimate than complex synthesizers. When you compare Diva and Tyrell (since they have similar representation in the training set), TyrellN6 outperforms Diva by a significant margin in the audio domain. Serum also outperforms Diva, but this is likely due to more training examples of Serum being used. However, in categorical parameter accuracy, Diva and Serum outperform TyrellN6. This is because many of TyrellN6's categorical parameters have minimal effect on the generated audio signal.

5.2 Timbre Evaluation

Next we evaluate the effect of timbre on the multi-task ASP setup using the joint encoder separate decoder model. To accomplish this, we use the log spectral distance metric. We draw box plots of the LSD value for 5 different timbre groups shown in *Figure 2*. These box plots give us quantitative evidence that sounds with extremely percussive components (HPSS harmonic component less than 20%) perform worse in the model than all other timbre categories. There is a slight decrease in performance as the harmonic percentage increases after 20%. However, this change is visibly negligible. We conclude that the model relies primarily on harmonic information to perform parameter inference.

5.3 Latent Size

We then try to understand the limitation of the latent space size for encoding multiple synthesizer parameters into a shared latent space. We train similar joint encoder separate decoder models with various latent space dimensions $\in \{2, 4, 8, 16, 32, 64, 128, 256, 512\}$. We evaluate the average log spectral distance, continuous parameter mean squared error, and categorical parameter accuracy for each different latent dimension size and plot the changes in *Figure 3*. We normalize the data to range from 0-1 and take the finite difference of each series to find where the series starts to saturate. We heuristically decide that a finite difference less than 0.1 is set as the saturation point. This saturation points happens when the latent dimension size is 64. This is also the latent size where the LSD metrics variance is at a valley.

6 Conclusion

We have shown a variety of approaches to multi-task ASP, and come to the conclusion that the separate-decoder approach is best. However, there are still many other types of experiments that could be done. Each model did not contain flow transformations in the latent space like past VAE parameter inference models. We could expand on our work by exploring multi task approaches with flow transformations from the latent space to parameter vectors instead of multi layer perception decoders. It may also be worth exploring a modified spectrogram decoder that uses DDSP to generate audio. this may allow for more useful representations to be learned in the latent space. We hope that future work is able to explore these topics and build upon the research presented in this paper.

References

[1] James Justice. “Analytic signal processing in music computation”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 27.6 (1979), pp. 670–684.

[2] Andrew Horner, James Beauchamp, and Lippold Haken. “Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis”. In: *Computer Music Journal* 17.4 (1993), pp. 17–29.

[3] Christopher R Houck, Jeff Joines, and Michael G Kay. “A genetic algorithm for function optimization: a Matlab implementation”. In: *Ncsu-ie tr* 95.09 (1995), pp. 1–10.

[4] Naotake Masuda and Daisuke Saito. “Quality diversity for synthesizer sound matching”. In: *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx20in21)*. 2021.

[5] Katsutoshi Itoyama and Hiroshi G Okuno. “Parameter estimation of virtual musical instrument synthesizers”. In: *ICMC*. 2014.

[6] Oren Barkan et al. “Inversynth: Deep estimation of synthesizer parameter configurations from audio signals”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.12 (2019), pp. 2385–2396.

[7] Matthew John Yee-King, Leon Fedden, and Mark d’Inverno. “Automatic programming of VST sound synthesizers using deep networks and other techniques”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 2.2 (2018), pp. 150–159.

[8] Jesse Engel et al. “DDSP: Differentiable Digital Signal Processing”. In: 2020. URL: https://openreview.net/forum?id=B1x1ma4tDr.

[9] Naotake Masuda and Daisuke Saito. “Synthesizer Sound Matching with Differentiable DSP.” In: *ISMIR*. 2021, pp. 428–434.

[10] Mark Cartwright and Bryan Pardo. “Synthassist: an audio synthesizer programmed with vocal imitation”. In: *Proceedings of the 22nd ACM international conference on Multimedia*. 2014, pp. 741–742.

[11] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[12] Antoine Caillon and Philippe Esling. *RAVE: A variational autoencoder for fast and high-quality neural audio synthesis*. 2022. URL: https://openreview.net/forum?id=cdwobSbmsjA.

[13] Ashis Pati and Alexander Lerch. “Is disentanglement enough? On latent representations for controllable music generation”. In: *arXiv preprint arXiv:2108.01450* (2021).

[14] Danilo Rezende and Shakir Mohamed. “Variational inference with normalizing flows”. In: *International conference on machine learning*. PMLR. 2015, pp. 1530–1538.

[15] Philippe Esling et al. “Flow synthesizer: Universal audio synthesizer control with normalizing flows”. In: *Applied Sciences* 10.1 (2020), p. 302.

[16] Gwendal Le Vaillant, Thierry Dutoit, and Sébastien Dekeyser. “Improving synthesizer programming from variational autoencoders latent space”. In: *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx20in21)*. 2021.

[17] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[18] Casper Kaae Sønderby et al. *How to Train Deep Variational Autoencoders and Probabilistic Ladder Networks*. English. WorkingPaper. 2016.

[19] Phil Burk. “Jsyn-a real-time synthesis api for java”. In: *ICMC*. 1998.

[20] David Braun. “DawDreamer: Bridging the Gap Between Digital Audio Workstations and Python Interfaces”. In: *arXiv preprint arXiv:2111.09931* (2021).

[21] Brian McFee et al. “librosa: Audio and music signal analysis in python”. In: *Proceedings of the 14th python in science conference*. Vol. 8. Citeseer. 2015, pp. 18–25.

[22] Derry Fitzgerald. “Harmonic/percussive separation using median filtering”. In: *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Vol. 13. 2010, pp. 1–4.

[23] Jonathan Driedger, Meinard Müller, and Sascha Disch. “Extending Harmonic-Percussive Separation of Audio Signals.” In: *ISMIR*. 2014, pp. 611–616.