AUDIO SYNTHESIZER INVERSION IN SYMMETRIC PARAMETER SPACES WITH APPROXIMATELY EQUIVARIANT FLOW MATCHING

Ben Hayes **Charalampos Saitis** **György Fazekas**
Centre for Digital Music, Queen Mary University of London, United Kingdom
ben@benhayes.net, {c.saitis, george.fazekas}@qmul.ac.uk

ABSTRACT

Many audio synthesizers can produce the same signal given different parameter configurations, meaning the inversion from sound to parameters is an inherently ill-posed problem. We show that this is largely due to intrinsic symmetries of the synthesizer, and focus in particular on permutation invariance. First, we demonstrate on a synthetic task that regressing point estimates under permutation symmetry degrades performance, even when using a permutation-invariant loss function or symmetry-breaking heuristics. Then, viewing equivalent solutions as modes of a probability distribution, we show that a conditional generative model substantially improves performance. Further, acknowledging the invariance of the implicit parameter distribution, we find that performance is further improved by using a permutation equivariant continuous normalizing flow. To accommodate intricate symmetries in real synthesizers, we also propose a relaxed equivariance strategy that adaptively discovers relevant symmetries from data. Applying our method to Surge XT, a full-featured open source synthesizer used in real world audio production, we find our method outperforms regression and generative baselines across audio reconstruction metrics.

1. INTRODUCTION

Modern audio synthesizers are intricate systems, combining numerous methods for sound production and manipulation with rich user-facing control schemes. Where, once, many digital synthesis algorithms were accompanied by a corresponding analysis procedure [1–3], selecting parameters of a modern synthesizer to approximate a given audio signal is a challenging open problem [4, 5] which is increasingly approached using powerful optimization and machine learning algorithms. In particular, recent works approach the task with deep neural networks trained on datasets sampled directly from the synthesizer [6–8].

Many synthesizers can produce the same signal given multiple different parameter configurations. This means that inverting the synthesizer is necessarily *ill-posed* — it lacks a unique solution. We argue that this harms the performance of models trained to produce point estimates of the parameters. Maximum likelihood regression objectives that do not account for this ill-posedness are minimized by a suboptimal averaging across equivalent solutions, while invariant loss functions can lead to pathologies such as the *responsibility problem* [9, 10]. This, we suggest, explains the superior performance of generative methods when sound matching complex synthesizers [11, 12] — for a given input, they can assign predictive weight to many possible parameter configurations, as opposed to selecting just one, as illustrated in Fig. 1.

The relationship between equivalent parameters is commonly governed by an underlying symmetry, which arises naturally from the design of the synthesizer. For example, in an additive synthesizer consisting of k independent oscillators, simple permutations yield $k!$ distinct yet equivalent parameter configurations. In this work, we focus on the effects of such permutation symmetries, which frequently

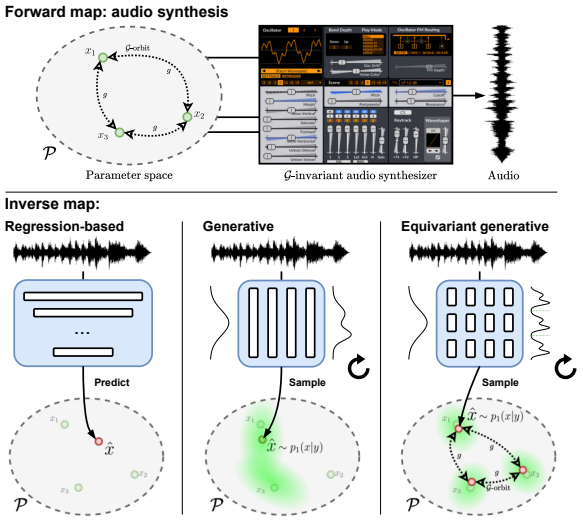

Figure 1: **Top:** Audio synthesis is the forward map we wish to invert. **Bottom-left:** Synthesizer inversion by parameter regression. The neural network produces a point estimate, and does not account for symmetries in the synthesizer. **Bottom-middle:** A generative model approximates the conditional distribution of parameters $\mathbf{x}$ given audio $\mathbf{y}$, but can only learn the appropriate invariances if present in the data. **Bottom-right:** Using an equivariant flow, the learned distribution is inherently invariant to the symmetries of the synthesizer.

occurs in synthesizers due to the use of repeated functional units — filters, oscillators, modulation sources, etc. — in their design. In such cases, we show that by constructing a permutation invariant generative model from equivariant continuous normalizing flows [13], we can improve over the performance of symmetry-naïve generative models. Further, using a toy task in which we can selectively break the invariance of the synthesizer, we show that permutation symmetry degrades the performance of regression-based models.

In real synthesizers, multiple symmetries may act concurrently on different parameters, while some parameters remain unaffected. Hand-designing a model to achieve the appropriate invariance thus scales poorly with synthesizer complexity and requires *a priori* knowledge of the underlying symmetries. Further, some synthesizers exhibit *conditional* and *approximate* symmetries, for which full invariance would be overly restrictive. To address this, we introduce a learnable mapping from synthesizer parameters to model tokens, which is capable of discovering symmetries present in the data, but can break the symmetry where necessary. Applying this technique to a dataset sampled from the Surge XT synthesizer with multiple symmetries, continuous and discrete parameters, and audio effects, we find consistently improved audio reconstruction performance. We provide full source code and audio examples at the following URL: `https://benhayes.net/synth-perm/`

2. BACKGROUND

2.1 Synthesizer inversion & sound matching

Given an audio signal, the sound matching task aims to find a synthesizer parameter configuration that best approximates it [4, 5]. We focus in this paper on *synthesizer inversion*, a sub-task of sound matching in which the audio signal we seek to approximate is known *a priori* to have come from the synthesizer. We do so to eliminate confounding factors due to the non-trivial implicit projection from general audio signals to the set of signals producible by the synthesizer.

For an overview of historical sound matching approaches, we refer the reader to Shier's [5] comprehensive review. The state-of-the-art in regression-based approaches was recently proposed by Bruford et al [8], who proposed to adopt the audio spectrogram transformer [14] architecture. Given its superior performance over MLP and CNN models, we adopt this model as our regression baseline.

Esling et al [11] presented the first generative approach, which was subsequently extended by Le Vaillant et al [12]. These methods train variational autoencoders on audio spectrograms, enriching the posterior distribution with normalizing flows. A second flow is jointly trained with a regression loss to predict synthesizer parameters from this learned audio representation.

Differentiable digital signal processing (DDSP) [15, 16] has also been applied to sound matching [7, 17–20]. Such approaches are effectively regression-based, as the composition of a differentiable synthesizer and an audio-domain loss function is a parameter-domain loss function. If the synthesizer exhibits an invariance, so will this composed loss function, meaning DDSP-based methods are also subject to the responsibility problem. Thus, while we do not conduct specific DDSP experimentation, we expect our findings for permutation invariant loss functions to be applicable also to DDSP with permutation invariant synthesizers.

2.2 Permutation symmetry & set generation

Predicting set-structured data (such as the parameters of a permutation invariant synthesizer) with vector-valued neural networks leads to a pathology known as the *responsibility problem* [9, 10], in which the continuous model must learn a highly discontinuous function. Intuitively, it is always possible to find two similar inputs that induce a change in "responsibility", and hence an approximation to a discontinuity in the network's outputs. Despite these issues, in synthesizer and audio processor inference tasks it is common to ignore the symmetry at the architectural level and simply adopt permutation invariant loss functions [21] or symmetry breaking heuristics [7, 22]. However, such approaches are still subject to the responsibility problem, and thus do not solve the underlying issue.

A variety of methods have been proposed for set prediction, of which the most successful view the task generatively [23–26] by transforming an exchangeable sample to the target set. Effectively, the task is framed as conditional density estimation over the space of sets [23]. Based on this insight, more general generative models such as continuous normalizing flows [27] and diffusion models [28] have been adapted to permutation invariant densities.

2.3 Continuous normalizing flows

Continuous normalizing flows (CNFs) [29, 30] are a family of powerful generative models which define invertible, continuous transformations between probability distributions. The conditional flow matching [31] framework allows us to train CNFs without expensive numerical integration by sampling a *conditional* probability path and regressing a closed form vector field which, in expectation, recovers the exact same gradients as regressing over the marginal field [31, 32]. In this work, we adopt the rectified flow [33] probability path which we pair with a minibatch approximation to the optimal transport coupling [34]. We build on prior work on equivariant flows [35–37] which are known to produce samples from invariant distributions [13].

3. METHOD

Let $\mathcal{P} \subset \mathbb{R}^k$ be the space of synthesizer parameters [1] and $\mathcal{S} \subset \mathbb{R}^n$ be the space of audio signals. A synthesizer is a map between these spaces, $f : \mathcal{P} \to \mathcal{S}$. It is common that f is not injective. That is, there exist multiple sets of parameters, e.g. $\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in \mathcal{P}$, that produce the same signal, i.e. $f(\mathbf{x}^{(1)}) = f(\mathbf{x}^{(2)})$. A trivial example is given when the synthesizer has a global gain control — all sets of parameters with zero global gain will produce an equivalent, silent signal. Clearly, in such cases, f lacks a well-defined inverse.

[1] We include MIDI pitch and note on/off times in our definition of synthesizer parameters. Effectively, we are dealing in single notes.

Therefore, we model our uncertainty over $\mathbf{x}$ as $p(\mathbf{x} \mid \mathbf{y})$, the distribution of parameters $\mathbf{x} \in \mathcal{P}$ given a signal $\mathbf{y} \in \mathcal{S}$.

When there is some transformation — let us denote this g — that acts on parameters such that $f(g \cdot \mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{P}$, we say that g is a symmetry of f. For example, g might permute the oscillators. If a set $\mathcal{G}$ of such transformations contains an identity transformation and an inverse for each $g \in \mathcal{G}$, then under composition it is a group acting on $\mathcal{P}$. For any $\mathbf{x} \in \mathcal{P}$, its $\mathcal{G}$-orbit — the set of points reachable via actions $g \in \mathcal{G}$ — is defined as $O_{\mathbf{x}} = \{g \cdot \mathbf{x} : g \in \mathcal{G}\}$. The set of all $\mathcal{G}$-orbits is a disjoint partition of $\mathcal{P}$:

$$\mathcal{P} = \bigsqcup_{O \in \mathcal{P}/\mathcal{G}} O, \tag{1}$$

This allows any parameter in $\mathcal{P}$ to be expressed as $\mathbf{x} = g \cdot r_O$ for some orbit representative $r_O \in O$ and group element $g \in \mathcal{G}$.[2] Hence, we can factor our conditional parameter density as:[3]

$$p(\mathbf{x} \mid \mathbf{y}) = \underbrace{p(O \mid \mathbf{y})}_{\text{Orbit}} \cdot \underbrace{p(g \mid O, \mathbf{y})}_{\text{Symmetry}} \cdot \underbrace{\eta(O)}_{\text{Stabilizer}}, \tag{2}$$

where $p(O \mid \mathbf{y})$ is invariant under transformations in $\mathcal{G}$, $p(g \mid O, \mathbf{y})$ describes the remaining uncertainty due to symmetry, and $\eta(O)$ is a scaling factor determined by the stabilizer of r_O in $\mathcal{G}$. Depending on the nature of $\mathcal{G}$, the posterior over orbits $p(O \mid \mathbf{y})$ may be considerably simpler than that over parameters $p(\mathbf{x} \mid \mathbf{y})$. For an additive synthesizer of 16 permutable oscillators, $|\mathcal{G}| = 16! \approx 20.9 \times 10^{12}$, meaning that any mode of $p(\mathbf{x} \mid \mathbf{y})$ is accompanied by over 20 trillion symmetries, while it is represented only once in $p(O \mid \mathbf{y})$.

We therefore want to factor out the effect of symmetry. Under two reasonable assumptions, it can be shown that $p(g \mid O, \mathbf{y})$ is necessarily uniform over $\mathcal{G}$. First, we assume $\mathcal{G}$-invariance of the likelihood $p(\mathbf{y} \mid \mathbf{x})$ which arises naturally from the symmetry of our synthesizer. Secondly, we assume $\mathcal{G}$-invariance of the prior $p(\mathbf{x})$. This is a stronger assumption, which we satisfy by randomly sampling our training data from $\mathcal{G}$-invariant parameter distributions, in contrast to some previous work which produces training data from handmade synthesizer presets.

Under a uniform symmetry distribution, we can say that $p(\mathbf{x} \mid \mathbf{y}) \propto p(O \mid \mathbf{y})\eta(O)$ and reduce our task to density estimation over orbits. Of course, the orbit of a point in $\mathcal{P}$ is an abstract equivalence class and can not directly be represented. However, by enforcing $\mathcal{G}$-invariance we ensure that our model is unable to discriminate between points on the same orbit, and thus implicitly learn the orbital posterior.

3.1 Permutation equivariant continuous normalizing flows

Our task, then, is to learn a $\mathcal{G}$-invariant distribution, focusing on the case where $\mathcal{G}$ is the product of permutation (symmetric) subgroups S_k, i.e. $\mathcal{G} = \bigtimes_i S_{k_i}$ of orders k_i. Köhler et al [13] showed that the pushforward of an isotropic Gaussian under an equivariant continuous normalizing flow is a density with the corresponding invariance. We thus seek a permutation *equivariant* architecture, making a Transformer [38] encoder without positional encoding [25] a natural choice. We adopt the Diffusion Transformer (DiT) [39] architecture with Adaptive Layer Norm (Ada-LN) conditioning.

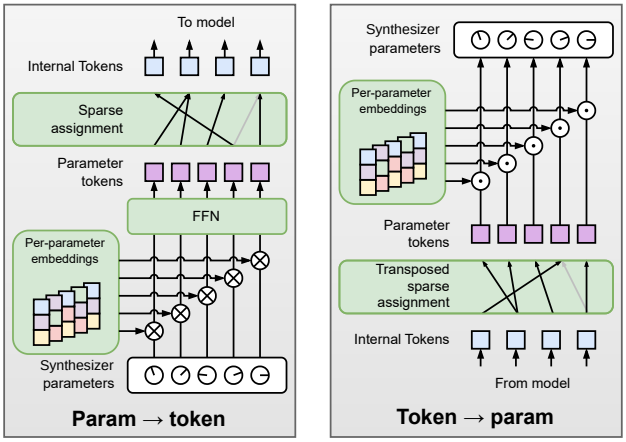

Figure 2: The PARAM2TOK projection for learning to assign parameters to tokens with relaxed equivariance.

Next, we must select an appropriate map from vectors in $\mathcal{P}$ to permutable Transformer tokens. For a simple synthesizer consisting of k permutable oscillators we could simply define k tokens, assigning the parameters of each oscillator to a distinct token. However, for a synthesizer with multiple permutation symmetries, each acting on a distinct subset of parameters, and some further non-permutable parameters, this is more challenging. As well as assigning parameters to tokens, we must indicate which tokens may be permuted with which other tokens and which may not be permuted at all.

Further, we may encounter quasi-symmetric structure in real synthesizers.[4] Specifically, we define *conditional symmetry* to mean a symmetry that acts only on some subset $\mathcal{P}' \subset \mathcal{P}$. For example, in Surge XT (see section 4.2), only certain values of the "routing" parameter allow permutation symmetry between filters. Further, if the actions of a group leave the signal *almost* unchanged, up to some error bound, we call this an *approximate symmetry*. For example, swapping two filters placed before and after a soft waveshaper may yield perceptually similar, but not identical signals. Whilst it may be possible to hand design tokenization strategies to accommodate these behaviours on a case-by-case basis, ideally we would learn such a mapping directly from data.

3.2 Equivariance discovery with PARAM2TOK

To address this, we propose PARAM2TOK, illustrated in Fig. 2, a mapping from parameters to tokens than can

[2] This factorization is unique if and only if the stabilizer of $\mathbf{x}$ (i.e. the subgroup of $\mathcal{G}$ that leaves $\mathbf{x}$ unchanged) is trivial.

[3] A full derivation is given in the supplementary material.

[4] While formal treatment is beyond the scope of this paper, we briefly illustrate them here as they nonetheless represent a source of uncertainty in inverting real world synthesizers.

exploit the Transformer's permutation equivariance when it is beneficial, but is not constrained to doing so.

Given a synthesizer with k parameters, we learn a matrix $\mathbf{Z} \in \mathbb{R}^{k \times d}$, and represent each parameter by scalar multiplication with the corresponding row of $\mathbf{Z}$, followed by a feed-forward neural network h_θ applied row-wise. To map parameter tokens to Transformer tokens, we learn an assignment matrix $\mathbf{A} \in \mathbb{R}^{n \times k}$, from k parameters to n tokens, giving the input to the first layer of our Transformer:

$$\mathbf{X}_0 = \mathbf{A} h_\theta \left(\text{diag}(\mathbf{x}) \mathbf{Z} \right). \tag{3}$$

To return from tokens to parameters, we use weight-tying [40] of the assignment matrix and another set of learned vectors $\mathbf{Z}' \in \mathbb{R}^{k \times d}$ as follows:

$$\tilde{\mathbf{x}} = \left(\mathbf{Z}' \odot \left(\mathbf{A}^T \mathbf{X}_l \right) \right) \mathbf{1}_d, \tag{4}$$

where $\mathbf{1}_d$ is simply a d-dimensional vector of ones and $\mathbf{X}_l$ is the output of the l^{th} Transformer layer.

We initialize $\mathbf{Z}, \mathbf{Z}'$, and $\mathbf{A}$ such that PARAM2TOK is approximately invariant to any permutation of the parameter vector, following the scheme described in the supplementary material. Intuitively, the model starts with the maximum possible symmetry which is then gradually broken through training. Finally, we encourage a sparse assignment with an $\mathcal{L}_1$ penalty on the entries of $\mathbf{A}$. Full initialization and regularization details can be found in the supplementary material, as well as proofs that PARAM2TOK can represent quasi-symmetries.

4. EXPERIMENTS

We train CNF models with the rectified flow probability path [33]. A minibatch approximation to the optimal transport coupling [34] is implemented using the Hungarian algorithm. Conditioning dropout is applied with a probability of 10% and inference is performed with classifier-free guidance [41, 42] with a scale of 2.0. Sampling is performed for 100 steps with the 4^{th} order Runge-Kutta method. For brevity, we omit some details about our models, datasets, and training configurations, but these can be found in the supplementary material as well as plots of learned PARAM2TOK parameters.

4.1 Isolating permutation symmetry with k-osc

To isolate the effect of permutation symmetry, we propose a simple synthetic task called k-osc, in which data is generated with a simple synthesizer which sums the output of k identical oscillators. Each oscillator is parameterized by: angular frequencies $\boldsymbol{\omega}$, amplitudes $\boldsymbol{\alpha}$, and waveform shapes $\boldsymbol{\gamma}$. The waveform shape parameter linearly interpolates between sine, square, and sawtooth waves, of which the latter two are antialiased with PolyBLEP [43].

This synthesizer is permutation invariant, but we can break the symmetry by constraining each oscillator to a disjoint frequency range. This allows us to compare the performance of models under both symmetric and asymmetric conditions. Similarly, we can control the degree of symmetry by varying k, which we test at $k \in \{4, 8, 16, 32\}$.

Data points are created by sampling a parameter vector $\mathbf{x} = \begin{bmatrix} \boldsymbol{\omega} & \boldsymbol{\alpha} & \boldsymbol{\gamma} \end{bmatrix} \in [-1, 1]^{3k}$. Frequency parameters are rescaled internally to $[0, \pi]$ for the symmetric variant or $[(i-1)\frac{\pi}{k}, i\frac{\pi}{k}]$ for $i = 1, \ldots, k$ for the asymmetric variant.

4.1.1 Models

We compare several variants of both generative and regression-based models. To help better interpret results, we also compute metrics over randomly sampled parameter vectors. All models extract a representation of the signal using a 1D frequency domain CNN.

CNF models: We train three CNF models. The first, CNF (Equivariant), parameterizes the flow's vector field with a DiT and Ada-LN conditioning, with the parameter-to-token mapping described in Section 3.1. Specifically, the i^{th} input token is $\begin{bmatrix} \omega_i & \alpha_i & \gamma_i \end{bmatrix}$, and $n = k$. Conversely, CNF (PARAM2TOK) learns the mapping via the relaxed equivariance of PARAM2TOK, again with k model tokens. Finally, to isolate the effect of model equivariance, we train CNF (MLP) using a residual MLP vector field with no intrinsic equivariance.

Regression models: All regression models use a residual MLP prediction head acting on the CNN representation. The first, FFN (MSE) simply regresses parameters with MSE loss. The remaining models mirror approaches to permutation symmetry found in the literature [7, 21, 22]. FFN (Sort) sorts oscillators by their frequency before computing the MSE loss, while FFN (Chamfer) computes a permutation invariant loss using a differentiable Chamfer distance [44]. While both approaches have improved performance in prior work, neither is able to resolve the responsibility problem [9].

4.1.2 Metrics

For every inferred parameter vector, we reconstruct the signal by synthesizing with these parameters. We measure dissimilarity from the ground truth signal using the $\mathcal{L}_2$ *log spectral distance* (LSD). To compare inferred parameters with the ground truth, we simply use the mean squared error (MSE) for the asymmetric condition. For the symmetric condition, we compute the optimal *linear assignment cost* (LAC) — that is, the minimum MSE across permutations.

4.1.3 Results

Results are shown in Fig. 3. Comparing CNF (Equivariant) and FFN (MSE) clearly illustrates the effect of permutation symmetry. Under symmetric conditions, CNF (Equivariant) excels, while FFN (MSE) performs poorly across metrics. Without symmetry, the roles are reversed — FFN (MSE) achieves excellent results, while CNF (Equivariant) imposes an overly restrictive equivariance and fails to discriminate between oscillators.

FFN (Chamfer), FFN (Sort), and CNF (MLP) offer "partial" solutions, being less affected when symmetry is present, but still underperforming CNF (Equivariant). Under asymmetric conditions, FFN (Chamfer) also imposes too stringent a restriction and performs similarly to CNF (Equivariant). CNF (MLP), however, lacks explicit equivariance

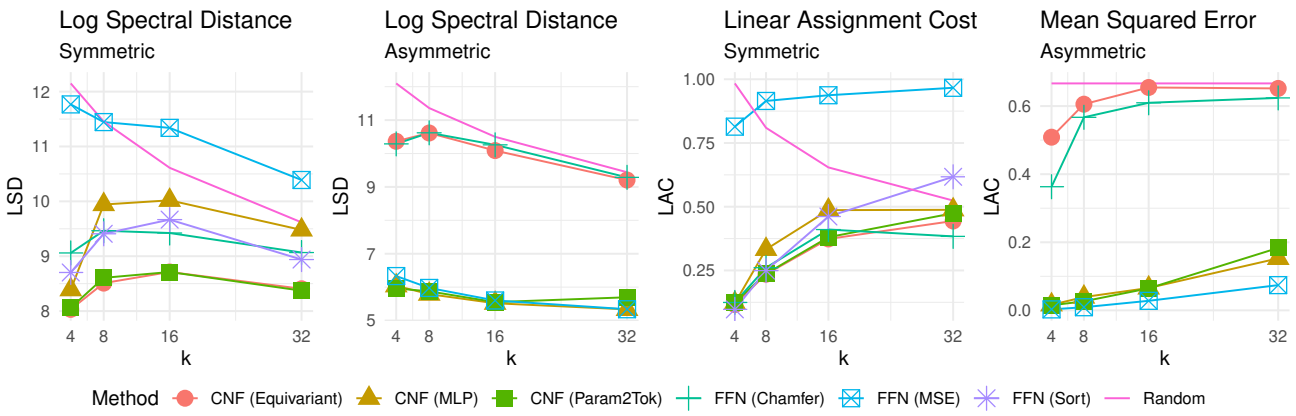

Figure 3: Evaluation results for the symmetric and asymmetric variants of the k-osc task.

and performs on par with FFN (MSE). Across conditions, CNF (PARAM2TOK) is competitive with the best models, demonstrating that PARAM2TOK can take advantage of model equivariance when helpful, but still learn distinct oscillator representations where necessary. Broadly, both parameter- and audio-domain metrics suggest that failing to account for permutation symmetry will harm performance and that CNFs with relaxed equivariance offer good performance across both symmetric and asymmetric scenarios.

4.2 Real-world synthesizer inversion

Our goal, of course, is to invert real-world software synthesizers. To this end, we test our method on Surge XT, an open source, feature rich "hybrid" synthesizer which incorporates multiple methods of producing and shaping sound. It exhibits multiple permutation symmetries, e.g. between oscillators, LFOs, and filters, as well as further conditional and approximate symmetries.

We construct two datasets of 2 million samples each by randomly sampling from the synthesizer's parameter space and rendering the corresponding audio using the `pedalboard` library [45]. [5]

The first dataset, referred to as *Surge Simple*, has 92 parameters including controls for three oscillators, two filters, three envelope generators, 5 low frequency oscillators (LFOs), and some global parameters. Omissions include discrete parameters, and those which affect global signal routing. The second, *Surge Full*, has 165 parameters, of which many are discrete, some alter the internal routing, and some control audio effects. *Surge Full* thus covers a broader sonic range, and introduces uncertainty beyond permutation symmetry.

In both cases, audio was rendered in stereo at 44.1kHz for a duration of 4 seconds and converted to a Mel-spectrogram with 128 Mel bands, using an analysis window of 25ms and a hop length of 10ms. Spectrograms were standardized using train dataset statistics. Like Le Vaillant et al [12], we adopt a one-hot representation of discrete and categorical parameters, concatenating all scalar and one-hot parameter representations to a single vector. All synthesis parameters are scaled to the interval $[-1, 1]$.

4.2.1 Models

CNF models: All CNF models receive audio conditioning from an Audio Spectrogram Transformer (AST) [14] with pre-normalization [46]. Instead of a single `[CLS]` token, we increase conditioning expressivity by learning an individual query token for each layer of the CNF's vector field. All models are trained end-to-end. Following prior findings on mixed continuous-discrete flow-based models [47], we do not constrain flows over discrete parameters to the simplex and simply adopt the same Gaussian source distribution for all dimensions. Our CNF (PARAM2TOK) model again uses the PARAM2TOK module with a DiT vector field. Again, we train a CNF (MLP) model with a residual MLP vector field to help isolate the effect of model equivariance from that of the generative approach.

Baselines: We adopt the AST [14] approach proposed by Bruford et al [8] as our regression baseline, and Le Vaillant et al's [12] VAE + RealNVP method as a further generative baseline.

4.2.2 Metrics

While previous sound matching work has typically included parameter domain distances as an evaluation metric, here we argue that if the synthesizer exhibits symmetry, such metrics are uninformative and possibly misleading. [6] While under simple symmetries we may select an invariant metric, as in Section 4.1.2, this approach does not scale to more complex synthesizers. We thus rely on audio reconstruction metrics which both share the exact invariances of the synthesizer and quantify performance on the task we are actually concerned with, i.e. best reconstructing the signal.

We measure spectrotemporal dissimilarity with a multi-scale spectral (MSS) distance computed over log-scaled Mel spectrograms at multiple resolutions. However, a slight error in one parameter may lead an otherwise good match

[5] Surge XT actually provides comprehensive Python bindings, allowing for direct programmatic interaction with the synthesizer. We opt to use the `pedalboard` in order to ensure our system is applicable to any software synthesizer, which limits us only to parameters exposed to the plugin host.

[6] For more detail on this point, see the supplementary material.

Method	Params	Surge XT (Simple) MSS ↓	wMFCC ↓	SOT ↓	RMS ↑	Surge XT (Full) MSS ↓	wMFCC ↓	SOT ↓	RMS ↑
CNF (Param2Tok)	40.2M	**3.18** ± 0.06	**5.57** ± 0.06	**0.042** ± 0.001	**0.948** ± 0.002	**6.13** ± 0.09	**9.63** ± 0.08	**0.086** ± 0.001	**0.939** ± 0.002
CNF (MLP)	41.6M	3.53 ± 0.06	6.02 ± 0.07	0.045 ± 0.001	0.932 ± 0.002	7.35 ± 0.11	11.01 ± 0.09	0.095 ± 0.001	0.924 ± 0.002
AST	45.3M	6.51 ± 0.10	9.11 ± 0.09	0.076 ± 0.001	0.738 ± 0.005	14.73 ± 0.14	14.42 ± 0.11	0.136 ± 0.002	0.804 ± 0.004
VAE + RealNVP	39.5M / 44.3M*	26.23 ± 0.14	15.80 ± 0.10	0.184 ± 0.002	0.595 ± 0.006	28.86 ± 0.19	18.61 ± 0.12	0.178 ± 0.002	0.703 ± 0.004
		Surge Full → NSynth MSS ↓	wMFCC ↓	SOT ↓	RMS ↑	Surge Full → FSD50K MSS ↓	wMFCC ↓	SOT ↓	RMS ↑
CNF (Param2Tok)	40.2M	**11.04** ± 0.28	**19.71** ± 0.24	**0.158** ± 0.004	0.834 ± 0.006	**15.40** ± 0.17	**17.25** ± 0.12	**0.168** ± 0.003	0.680 ± 0.005
CNF (MLP)	41.6M	13.51 ± 0.32	21.09 ± 0.23	0.175 ± 0.004	**0.840** ± 0.006	16.96 ± 0.18	17.83 ± 0.12	0.183 ± 0.003	**0.710** ± 0.004
AST	45.3M	24.04 ± 0.30	28.62 ± 0.17	0.224 ± 0.003	0.755 ± 0.007	19.09 ± 0.16	21.35 ± 0.14	0.187 ± 0.003	0.682 ± 0.004
VAE + RealNVP	44.3M	35.29 ± 0.26	23.50 ± 0.14	0.266 ± 0.004	0.689 ± 0.007	25.06 ± 0.20	21.24 ± 0.12	0.247 ± 0.003	0.681 ± 0.004

* VAE + RealNVP parameter counts for *Simple* and *Full* datasets, respectively. The difference arises because the latent space dimension is set to the length of the parameter vector.

Table 1: **Top:** Audio reconstruction results on *Simple* and *Full* variants of the Surge XT synthesizer inversion task. **Bottom:** Out-of-domain audio reconstruction results for models trained on the *Surge Full* dataset. **All:** Results reported with 95% confidence interval computed across test dataset.

to be unfairly penalized due to shifts in time or frequency. We thus include a *warped Mel-frequency cepstral coefficient* (wMFCC) metric as a more "malleable" alternative, given by the optimal $\mathcal{L}_1$ dynamic time warping (DTW) cost between two MFCC series. This allows robustness to timing and pitch deviations.

Pointwise spectral distances fail to capture distances "along" the frequency axis, such as pitch dissimilarity [9, 48]. We thus adopt the *spectral optimal transport* (SOT) distance [49] as a pitch-sensitive measure of spectral similarity. Finally, we include a simple cosine similarity between frame-wise RMS energy to capture amplitude envelope similarity.

4.2.3 Results

Metrics are computed on a held-out test dataset of $10,000$ sounds synthesized in the same manner as the training dataset. Results are presented in Table 1, and audio examples can be heard on the companion website. [7]

The CNF models perform consistently with our theoretical expectations and the results of our toy task. In particular, CNF (Param2Tok) outperforms all models across all metrics on both datasets. The learned Param2Tok parameters suggest that the model has discovered a token mapping that corresponds to the intrinsic symmetries of the synthesizer, and thus benefits from the Transformer's permutation equivariance.

The non-equivariant CNF (MLP) also achieves reasonable performance, suggesting that simply adopting a probabilistic framework is already very helpful in dealing with the sources of ill-posedness in real synthesizers. The difference between the MLP and Param2Tok variants is more pronounced for *Surge Full*, suggesting that the effect of introducing greater complexity is magnified in the presence of unresolved symmetry. This aligns with our decomposition of the conditional parameter density in Section 3 — any change in $p(O \mid \mathbf{y})$ is "repeated" in $p(\mathbf{x} \mid \mathbf{y})$ for each element of $\mathcal{G}$.

Despite extensive tuning, VAE + RealNVP consistently collapsed to predicting average values for many parameters. Of course, this conflicts with the results of the original papers [11, 12]. We suggest this may be due to the use of smaller datasets of hand-designed synthesizer presets, which could be sufficiently biased to break the invariance of $p(\mathbf{x})$.

[7] Link to website: `https://benhayes.net/synth-perm/`

4.2.4 Out-of-distribution results

While this is not our focus, we report out-of-distribution results on audio from the NSynth [50] and FSD50K [51] test datasets in Table 1. Across all models, performance suffers compared to in-domain data, but the relationships between models are preserved. Thus, even under such challenging conditions, unhandled symmetry likely detrimentally influences performance. To adapt our method to general sound matching, we intend to explore shared representations of synthesized and non-synthesized audio, domain adaptation techniques, and information bottlenecks, such as quantization, for conditioning.

5. CONCLUSION

The implications of our findings are clear: if the synthesizer has a symmetry, it is better to (i) approach the problem generatively and (ii) learn the corresponding invariant density. This extends beyond synthesizers, as audio effects also commonly exhibit permutation symmetries, as noted by Nercessian [22]. Beyond audio, these results are of relevance anywhere neural networks parameterize an external system with structural symmetries.

A key limitation of our work is the lack of specific experimentation on the role of quasi-symmetries. A k-osc style task which isolates their effect would be very illuminating about the extent to which Param2Tok improves results in their presence, and we intend to include this in a subsequent publication. We also note that we currently lack theoretical guarantees that Param2Tok will discover symmetries. In future work, we thus intend to both study this module theoretically and gather further empirical data on the role of initialization and regularization in its behaviour. We also plan more comprehensive evaluation of the extent to which Param2Tok-based models learn the appropriate invariance.

Our proposed method should generalize effectively to arbitrary software synthesizers. In future work, we will therefore explore multi-task training by simultaneously modelling over many synthesizers. We will also seek to extend our work to the more general sound matching task by exploring the previously discussed strategies for robustifying our system to out-of-distribution inputs.

6. ETHICS STATEMENT

Like any AI model, our work inherently encodes the biases and values of the authors. In particular, our choice of VST synthesizer reflects a bias towards western popular music. However, the more abstract nature of our synthetic experimentation does suggest that our results may reasonably be expected to generalize to tools that better represent the needs of other musical cultures.

The source of training data is of particular importance in assessing the ethical impact of an AI model, and in this instance we have worked entirely with synthetically generated data. Even in our experimentation on a real synthesizer, we opted to generate our dataset by randomly sampling parameters, rather than by scraping the internet for presets or exploiting factory preset libraries. We further note that Surge XT is an open source synthesizer released under the GNU GPL.

Amid growing concern over AI tools displacing workers, particularly in the creative industries, we wish to highlight that our goal in this work is to develop technology to integrate with and enhance existing workflows. We believe the choice of task reflects this by seeking to enhance interaction with an existing family of creative tools, as opposed to outright replacing them.

7. ACKNOWLEDGEMENTS

B.H. would like to thank Christopher Mitcheltree, Marco Pasini, Chin-Yun Yu, Jack Loth, and Julien Guinot for their invaluable feedback on this manuscript in varying stages of completion, and Jordie Shier for the many inspiring and illuminating conversations on this topic.

8. REFERENCES

[1] J. Justice. "Analytic Signal Processing in Music Computation". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 27.6 (Dec. 1979), pp. 670–684. DOI: 10.1109/TASSP.1979.1163321.

[2] R. McAulay and T. Quatieri. "Speech Analysis/Synthesis Based on a Sinusoidal Representation". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34.4 (Aug. 1986), pp. 744–754. DOI: 10.1109/TASSP.1986.1164910.

[3] Xavier Serra and Julius Smith. "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition". In: *Computer Music Journal* 14.4 (1990), pp. 12–24. DOI: 10.2307/3680788.

[4] Martin Roth and Matthew Yee-King. "A Comparison of Parametric Optimization Techniques for Musical Instrument Tone Matching". In: Audio Engineering Society Convention 130. Audio Engineering Society, May 13, 2011.

[5] Jordie Shier. "The Synthesizer Programming Problem: Improving the Usability of Sound Synthesizers". MA thesis. University of Victoria, 2021.

[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 (Dec. 2019), pp. 2385–2396. DOI: 10.1109/TASLP.2019.2944568. arXiv: 1812.06349.

[7] Naotake Masuda and Daisuke Saito. "Improving Semi-Supervised Differentiable Synthesizer Sound Matching for Practical Applications". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 31 (2023), pp. 863–875. DOI: 10.1109/TASLP.2023.3237161.

[8] Fred Bruford, Frederik Blang, and Shahan Nercessian. "Synthesizer Sound Matching Using Audio Spectrogram Transformers". In: *Proceedings of the 27th International Conference on Digital Audio Effects*. DAFx24. Guildford, Surrey, Sept. 3–7, 2024.

[9] Ben Hayes, Charalampos Saitis, and György Fazekas. "The Responsibility Problem in Neural Networks with Unordered Targets". In: *The First Tiny Papers Track at ICLR 2023*. ICLR. Kigali, Rwanda, May 5, 2023.

[10] Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. "FSPool: Learning Set Representations with Featurewise Sort Pooling". In: International Conference on Learning Representations. Sept. 23, 2019.

[11] Philippe Esling et al. "Flow Synthesizer: Universal Audio Synthesizer Control with Normalizing Flows". In: *Applied Sciences* 10.1 (Dec. 2020), p. 302. DOI: 10.3390/app10010302.

[12] Gwendal Le Vaillant, Thierry Dutoit, and Sebastien Dekeyser. "Improving Synthesizer Programming From Variational Autoencoders Latent Space". In: *2021 24th International Conference on Digital Audio Effects (DAFx)*. 2021 24th International Conference on Digital Audio Effects (DAFx). Vienna, Austria: IEEE, Sept. 8, 2021, pp. 276–283. DOI: 10.23919/DAFx51585.2021.9768218.

[13] Jonas Köhler, Leon Klein, and Frank Noe. "Equivariant Flows: Exact Likelihood Generative Learning for Symmetric Densities". In: *Proceedings of the 37th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, Nov. 21, 2020, pp. 5361–5370.

[14] Yuan Gong, Yu-An Chung, and James Glass. "AST: Audio Spectrogram Transformer". In: Proc. Interspeech 2021. 2021, pp. 571–575. DOI: 10.21437/Interspeech.2021-698.

[15] Jesse Engel et al. "DDSP: Differentiable Digital Signal Processing". In: *8th International Conference on Learning Representations*. ICLR 2020. Addis Ababa, Ethiopia, Apr. 2020.

[16] Ben Hayes et al. "A Review of Differentiable Digital Signal Processing for Music & Speech Synthesis". In: *Frontiers in Signal Processing* (2023).

[17] Naotake Masuda and Daisuke Saito. "Synthesizer Sound Matching with Differentiable DSP" (Online). Nov. 7, 2021. DOI: 10.5281/zenodo.5624609.

[18] Noy Uzrad et al. *DiffMoog: A Differentiable Modular Synthesizer for Sound Matching*. Jan. 23, 2024. DOI: 10.48550/arXiv.2401.12570. arXiv: 2401.12570 [eess]. Pre-published.

[19] Yuting Yang et al. "White Box Search over Audio Synthesizer Parameters". In: *Proc. of the 24rd Int. Society for Music Information Retrieval Conf.* ISMIR. Milan, Italy, 2023.

[20] Han Han, Vincent Lostanlen, and Mathieu Lagrange. "Learning to Solve Inverse Problems for Perceptual Sound Matching". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 32 (2024), pp. 2605–2615. DOI: 10.1109/TASLP.2024.3393738.

[21] Jesse Engel, Rigel Swavely, and Adam Roberts. "Self-Supervised Pitch Detection by Inverse Audio Synthesis". In: *Proceedings of the International Conference on Machine Learning*. ICML 2020. 2020, p. 9.

[22] Shahan Nercessian. "Neural Parametric Equalizer Matching Using Differentiable Biquads". In: *Proceedings of the 23rd International Conference on Digital Audio Effects*. DAFx2020. Vienna, Austria, 2020, p. 8.

[23] David W. Zhang, Gertjan J. Burghouts, and Cees G. M. Snoek. "Set Prediction without Imposing Structure as Conditional Density Estimation". In: International Conference on Learning Representations. Jan. 12, 2021.

[24] Yan Zhang, Jonathon Hare, and Adam Prugel-Bennett. "Deep Set Prediction Networks". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

[25] Adam R. Kosiorek, Hyunjik Kim, and Danilo J. Rezende. "Conditional Set Generation with Transformers". In: *Workshop on Object-Oriented Learning*. International Conference on Machine Learning. arXiv, July 1, 2020. arXiv: 2006.16841 [cs].

[26] Jinwoo Kim et al. *SetVAE: Learning Hierarchical Composition for Generative Modeling of Set-Structured Data*. Mar. 29, 2021. DOI: 10.48550/arXiv.2103.15619. arXiv: 2103.15619 [cs]. Pre-published.

[27] Marin Biloš and Stephan Günnemann. "Scalable Normalizing Flows for Permutation Invariant Densities". In: *Proceedings of the 38th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, July 1, 2021, pp. 957–967.

[28] Lemeng Wu et al. "Fast Point Cloud Generation with Straight Flows". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Vancouver, BC, Canada: IEEE, June 2023, pp. 9445–9454. DOI: 10.1109/CVPR52729.2023.00911.

[29] Ricky T. Q. Chen et al. "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.

[30] Will Grathwohl et al. "FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models". In: International Conference on Learning Representations. Sept. 27, 2018.

[31] Alexander Tong et al. "Improving and Generalizing Flow-Based Generative Models with Minibatch Optimal Transport". In: *Transactions on Machine Learning Research* (2024).

[32] Yaron Lipman et al. "Flow Matching for Generative Modeling". In: The Eleventh International Conference on Learning Representations. Feb. 1, 2023.

[33] Xingchao Liu, Chengyue Gong, and Qiang Liu. "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow". In: The Eleventh International Conference on Learning Representations. Feb. 1, 2023.

[34] Aram-Alexandre Pooladian et al. "Multisample Flow Matching: Straightening Flows with Minibatch Couplings". In: *Proceedings of the 40th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, July 3, 2023, pp. 28100–28127.

[35] Yuxuan Song et al. *Equivariant Flow Matching with Hybrid Probability Transport*. Dec. 12, 2023. DOI: 10.48550/arXiv.2312.07168. arXiv: 2312.07168 [cs]. Pre-published.

[36] Majdi Hassan et al. "Equivariant Flow Matching for Molecular Conformer Generation". In: ICML'24 Workshop ML for Life and Material Science: From Theory to Industry Applications. July 17, 2024.

[37] Leon Klein, Andreas Krämer, and Frank Noe. "Equivariant Flow Matching". In: *Advances in Neural Information Processing Systems* 36 (Dec. 15, 2023), pp. 59886–59910.

[38] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[39] William Peebles and Saining Xie. "Scalable Diffusion Models with Transformers". In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023 IEEE/CVF International Conference on Computer Vision (ICCV). Paris, France: IEEE, Oct. 1, 2023, pp. 4172–4182. DOI: 10.1109/ICCV51070.2023.00387.

[40] Ofir Press and Lior Wolf. "Using the Output Embedding to Improve Language Models". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. Valencia, Spain: Association for Computational Linguistics, 2017, pp. 157–163. DOI: 10.18653/v1/E17-2025.

[41] Jonathan Ho and Tim Salimans. "Classifier-Free Diffusion Guidance". In: NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications. Dec. 8, 2021.

[42] Qinqing Zheng et al. *Guided Flows for Generative Modeling and Decision Making*. Dec. 7, 2023. DOI: 10.48550/arXiv.2311.13443. arXiv: 2311.13443 [cs]. Pre-published.

[43] Vesa Välimäki, Jussi Pekonen, and Juhan Nam. "Perceptually Informed Synthesis of Bandlimited Classical Waveforms Using Integrated Polynomial Interpolation". In: *The Journal of the Acoustical Society of America* 131.1 (Jan. 1, 2012), pp. 974–986. DOI: 10.1121/1.3651227.

[44] Harry G Barrow et al. "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching". In: *Proceedings: Image Understanding Workshop*. Science Applications, Inc, 1977, pp. 21–27.

[45] Peter Sobot. *Pedalboard*. Version 0.7.3. Zenodo, Apr. 10, 2023. DOI: 10.5281/ZENODO.7817838.

[46] Ruibin Xiong et al. "On Layer Normalization in the Transformer Architecture". In: *Proceedings of the 37th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, Nov. 21, 2020, pp. 10524–10533.

[47] Ian Dunn and David Ryan Koes. *Mixed Continuous and Categorical Flow Matching for 3D De Novo Molecule Generation*. Apr. 30, 2024. DOI: 10.48550/arXiv.2404.19739. arXiv: 2404.19739 [q-bio]. Pre-published.

[48] Joseph Turian and Max Henry. "I'm Sorry for Your Loss: Spectrally-Based Audio Distances Are Bad at Pitch". Dec. 9, 2020. arXiv: 2012.04572 [cs, eess].

[49] Bernardo Torres, Geoffroy Peeters, and Gaël Richard. "Unsupervised Harmonic Parameter Estimation Using Differentiable DSP and Spectral Optimal Transport". In: *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Seoul, Korea, Republic of: IEEE, Apr. 14, 2024, pp. 1176–1180. DOI: 10.1109/ICASSP48485.2024.10447011.

[50] Jesse Engel et al. "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, Australia, Aug. 6, 2017, pp. 1068–1077.

[51] Eduardo Fonseca et al. "FSD50K: An Open Dataset of Human-Labeled Sound Events". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30 (2022), pp. 829–852. DOI: 10.1109/TASLP.2021.3133208.