OFF-MANIFOLD ROBUSTNESS IN SYNTHESIZER INVERSION WITH JOINT DISTRIBUTION FLOW MATCHING

Anonymous Authors
Anonymous Affiliations
anonymous@ismir.net

ABSTRACT

Recent work on synthesizer inversion shows that generative models outperform deterministic approaches by explicitly modeling the ambiguity in mapping audio to parameters. Training such models, however, requires audio-parameter pairs, which are typically obtained by rendering sampled or preset parameters through the synthesizer itself. This creates a train-test mismatch that can degrade performance on off-manifold real-world recordings, for which ground-truth parameter annotations do not exist. To circumvent this obstacle, we propose to model the *joint* distribution of audio and parameters with a multi-modal continuous normalizing flow using independent noise schedules for each modality. This formulation allows us to train joint and conditional densities with paired synthesizer data, while unpaired real recordings train the audio marginal alone, exposing the model to off-manifold signals without requiring parameter labels. Further, because the model learns to map from audio to parameters at all noise levels, we find that partially noising the audio reference at inference improves real-audio reconstruction, consistent with reducing sensitivity to distribution-specific detail while preserving coarse structure. Evaluating on SURGE XT and DEXED, we find our method substantially improves inversion of real-world audio while remaining competitive in-domain.

1. INTRODUCTION

A considerable body of work has sought to develop systems that automatically program audio synthesizers to recreate or approximate a given audio input [1–15]. This task is intrinsically challenging: it is an ill-posed inverse problem [11] that must be solved under highly unfavourable conditions, as synthesizers are black boxes exhibiting strong nonlinearity, stochasticity, and complex parameter interactions. Recent work [11] has shown some of these difficulties can be mitigated by approaching the problem probabilistically, modelling the conditional distribution of parameters given audio while accounting for structural symmetries inherent to the synthesizer.

A gap remains, however, between these results and likely real-world use cases, in which a user presents the model with audio that was not produced by the synthesizer—i.e., audio that lies off the synthesizer's audio manifold (Fig. 1). Training such models requires audio paired with corresponding synthesizer parameters, and so training data is limited to audio produced by the synthesizer itself. As we show empirically, this distribution mismatch is associated with degraded performance when the trained model is presented with off-manifold, real audio at inference time.

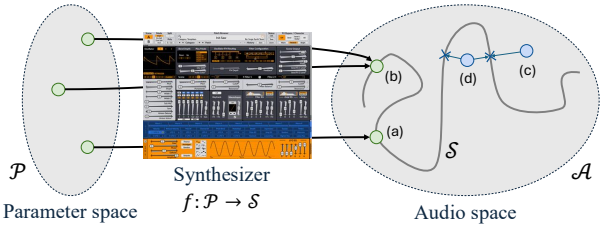

Figure 1. (a) An audio synthesizer f maps from points in a parameter space $\mathcal{P}$ to points on a manifold $\mathcal{S}$ embedded in an ambient audio signal space $\mathcal{A}$. (b) Multiple points in $\mathcal{P}$ sometimes map to the same point on $\mathcal{S}$. (c) Off-manifold points in $\mathcal{A}$ can be optimally projected onto $\mathcal{S}$ under some distance metric. (d) Sometimes one off-manifold point admits multiple optimal projections, and sometimes multiple off-manifold points share an optimal projection on $\mathcal{S}$.

The obvious remedy would be to expose the model to real audio during training, but the natural mechanisms for doing so each carry significant costs. Differentiable digital signal processing [7, 16, 17] permits direct supervision from real audio but sacrifices the benefits of generative training, and binds the method to specific differentiable implementations with their accompanying instability and optimisation challenges [17–19]. Reinforcement learning approaches [12] offer a complementary fine-tuning route, but synth-in-the-loop reward function computation renders them difficult to scale as a primary training objective, and requires substantial per-synthesizer design effort.

To sidestep these issues, we instead build on the generative synthesizer inversion framework [11] and repurpose joint multi-modal flow matching [20, 21] as a mechanism for *off-manifold robustness*. Rather than treating audio as a fixed conditioning variable and learning only the conditional distribution of parameters given audio, we learn a single flow over the joint audio-parameter space with independent noise schedules for each modality. While this framework was originally designed to unify multiple generative tasks under a single objective, we observe that its factorised noise

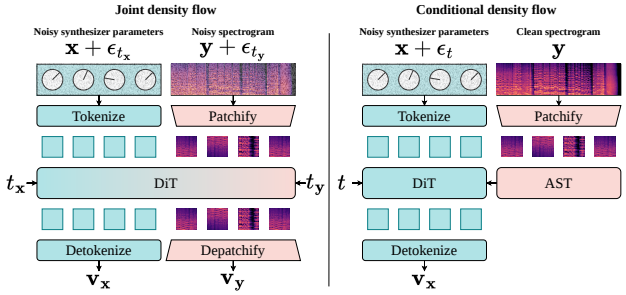

Figure 2. Left: Joint distribution flow matching for synthesizer inversion. Audio spectrogram patches and per-parameter tokens are concatenated and processed by a DiT with per-modality time conditioning. Right: conditional distribution flow matching for synthesizer inversion.

schedule is also well suited to enhance the off-manifold robustness of a conditional generative model, for three reasons. First, training on partially noised audio states constrains the velocity field away from the clean synthesized-audio manifold, reducing the extrapolation required when conditioning on real recordings. Second, its independent noise schedules admit both the conditional parameter distribution and marginal audio distribution as special cases, allowing unlabeled real audio to be used alongside synthesizer audio as first-class training data. Third, at inference, conditioning on a partially noised version of the target audio yields a noise-smoothed posterior that suppresses unreproducible recording-specific detail while preserving higher level structure.

We evaluate our approach on two widely-used software synthesizers, SURGE XT and DEXED [1], comparing against a prior generative inversion baseline [11] on both in-domain and out-of-domain audio. Our experiments demonstrate substantial improvements on real-world inputs while maintaining competitive in-domain performance. Audio examples are available at `https://anon9119.github.io/`.

2. BACKGROUND

2.1 Synthesizer Inversion

Let $\mathcal{P} \subset \mathbb{R}^k$ denote the space of synthesizer parameters and $\mathcal{A} \subset \mathbb{R}^n$ the space of audio signals. A synthesizer is the forward map $f : \mathcal{P} \to \mathcal{A}$. In synthesizer inversion, we seek to recover parameters $\mathbf{x} \in \mathcal{P}$ from a target signal $\mathbf{y} \in \mathcal{A}$.

In general, f is non-injective: multiple parameter configurations can produce the same signal. Deterministic regression methods [10, 22] therefore tend to collapse onto suboptimal solutions corresponding to averages over equivalent parameter sets. Methods based on Differentiable Digital Signal Processing (DDSP) [7, 9, 23] similarly rely on a deterministic mapping and cannot disambiguate equivalent or uncertain solutions. They further require a differentiable synthesizer implementation, with attendant training instability.

[1] SURGE XT: `https://surge-synthesizer.github.io/`; DEXED: `https://asb2m10.github.io/dexed/`

Recently, the task has been approached generatively [11], modelling the conditional distribution $p(\mathbf{x} \mid \mathbf{y})$ of parameters given audio. Generative approaches can place predictive mass on multiple plausible solutions, naturally accommodating the ill-posedness of the inverse problem while benefiting from the training stability of modern generative models. However, such models require training audio to be annotated with corresponding synthesizer parameters, and are thus exclusively trained on synthesized audio, which typically lies on a manifold $\mathcal{S} \subset \mathcal{A}$ (Fig. 1). This leads to degraded performance on real-world, off-manifold recordings.

Complementary work [12] addresses domain gaps by fine-tuning a model with reinforcement learning on out-of-distribution audio for which parameter labels are unavailable. As their method discretises each continuous parameter into a fixed number of classes and formulates inversion as classification, a like-for-like comparison would require non-trivial redesign of their architecture to operate on continuous parameter spaces. This is beyond the scope of the current work, and so we do not adopt it as a baseline. We do, however, consider this a worthwhile avenue for future work, noting that our generative model could, itself, serve as a policy within such an RL framework.

2.2 Flow Matching

Flow matching [24, 25] is a technique for training a continuous normalizing flow without relying on numerical integration during training. Given data $X_1 \sim p_{\text{data}}$ and independent noise $X_0 \sim p_0$, we define the stochastic interpolant:

$$X_t = (1-t)X_0 + tX_1, \qquad t \in [0,1], \tag{1}$$

with distribution q_t at time t. By construction, $q_0 = p_0$ and $q_1 = p_{\text{data}}$. The *marginal velocity field* is the conditional expectation

$$u_t(\mathbf{x}) = \mathbb{E}[X_1 - X_0 \,|\, X_t = \mathbf{x}]\,, \tag{2}$$

which satisfies the continuity equation $\partial_t q_t = -\nabla \cdot (u_t\, q_t)$. Transporting p_0 along u_t from $t = 0$ to $t = 1$ therefore recovers p_{data}.

A model v_θ is trained to regress u_t via the flow-matching objective

$$\mathcal{L} = \mathbb{E}_{t,X_t} \left\| v_\theta(X_t, t) - u_t(X_t) \right\|^2. \tag{3}$$

Direct minimisation is intractable since u_t is unknown, but [24, 25] show that Eq. 3 shares its minimiser with the tractable conditional objective:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,X_0,X_1} \left\| v_\theta(X_t, t) - (X_1 - X_0) \right\|^2, \tag{4}$$

whose targets are samples rather than conditional expectations. The minimiser of Eq. 4 is, pointwise in t, the marginal velocity u_t. At inference, samples are generated by integrating v_θ from $t = 0$ to $t = 1$ with an ODE solver; by the continuity equation, this pushes p_0 forward to p_{data}.

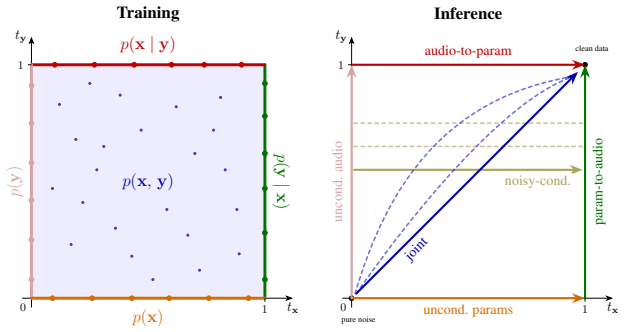

Figure 3. Flow-matching time $\mathbf{t} = (t_\mathbf{x}, t_\mathbf{y})$ forms a unit square. **Training:** interior samples fit the joint density $p(\mathbf{x}, \mathbf{y})$; samples on the top and right edges fit the conditionals $p(\mathbf{x} \mid \mathbf{y})$ and $p(\mathbf{y} \mid \mathbf{x})$; samples on the left and bottom edges fit the marginals $p(\mathbf{y})$ and $p(\mathbf{x})$. **Inference:** different sampling trajectories correspond to different tasks.

2.3 Multi-modal Flow Matching

Bao et al. [21] observed that, given multi-modal data, diffusion models over marginal, conditional, and joint distributions can be unified by training a single model and assigning each modality its own independent noise level. OmniFlow [20] extends this construction to flow matching: given n modalities, each with its own time variable $t_i \in [0, 1]$, different generation tasks correspond to different paths through the n-dimensional time cube $[0, 1]^n$. When a modality is unobserved, its time variable is set to 0 (pure noise), and the model learns only from the remaining modalities. This enables joint training on heterogeneous data in which different subsets of modalities are available for different examples. The method we develop in Section 3 builds directly on this framework, specialised to the two-modality setting of parameters and audio.

3. JOINT DISTRIBUTION FLOW MATCHING FOR SYNTHESIZER INVERSION

Let $X_1 \in \mathcal{P}$ and $Y_1 \in \mathcal{A}$ denote paired synthesizer parameters and audio, $(X_1, Y_1) \sim p_{\text{data}}$, and let $X_0 \sim p_0^\mathbf{x}$ and $Y_0 \sim p_0^\mathbf{y}$ be noise samples, with X_0, Y_0, and (X_1, Y_1) mutually independent. We assign each modality its own time variable, collected into the *time square* $\mathbf{t} = (t_\mathbf{x}, t_\mathbf{y}) \in [0, 1]^2$, and define the per-modality interpolants:

$$\begin{aligned} X_{t_\mathbf{x}} &= (1 - t_\mathbf{x})X_0 + t_\mathbf{x} X_1, \\ Y_{t_\mathbf{y}} &= (1 - t_\mathbf{y})Y_0 + t_\mathbf{y} Y_1. \end{aligned} \tag{5}$$

Their joint distribution $q_\mathbf{t}$ is a family of distributions on $\mathcal{P} \times \mathcal{A}$ indexed by the time square. Its corners recover the data distribution $p_{\text{data}} = q_{(1,1)}$, a pair of independent noise samples $p_0^\mathbf{x} \otimes p_0^\mathbf{y} = q_{(0,0)}$, and the clean-parameter and clean-audio marginals paired with the opposite modality's noise, i.e. at $\mathbf{t} = (1, 0)$ and $\mathbf{t} = (0, 1)$. Its edges recover conditional and marginal interpolants: the top edge $\{t_\mathbf{y} = 1\}$ parameterises $p(X_{t_\mathbf{x}} \mid Y_1)$, the conditional interpolant from noise to clean parameters given clean audio; the left edge $\{t_\mathbf{x} = 0\}$ factors as independent noise over $\mathcal{P}$ times the audio marginal interpolant $q_{t_\mathbf{y}}^\mathbf{y}$; and the right and bottom edges are their symmetric counterparts.

A model $v_\theta = (v_\theta^\mathbf{x}, v_\theta^\mathbf{y})$ predicting per-modality velocities is trained by the multi-modal conditional flow-matching objective of [20],

$$\begin{aligned} \mathcal{L}_{\text{joint}} = \mathbb{E}_{\mathbf{t}\sim\mu}\Big[&\big\| v_\theta^\mathbf{x}(X_{t_\mathbf{x}}, Y_{t_\mathbf{y}}, \mathbf{t}) - (X_1 - X_0) \big\|^2 \\ &+ \big\| v_\theta^\mathbf{y}(X_{t_\mathbf{x}}, Y_{t_\mathbf{y}}, \mathbf{t}) - (Y_1 - Y_0) \big\|^2 \Big], \end{aligned} \tag{6}$$

where the expectation is also over X_0, Y_0, X_1, Y_1, and μ is a sampling distribution over the time square. For unpaired real-audio examples, we restrict to the audio-marginal edge $t_\mathbf{x} = 0$ and evaluate only the audio velocity term. By the pointwise-minimiser property of the conditional FM objective [25] (Section 2.2), v_θ approximates the marginal velocities of $q_\mathbf{t}$ at each $\mathbf{t} \in \text{supp}(\mu)$, and is otherwise unconstrained. Moreover, a sampling path $\gamma : [0, 1] \to \text{supp}(\mu)$ specifies which task is performed at inference: as γ moves through the time square, only the velocity components corresponding to changing time coordinates are integrated [20].

μ determines the density being learned: Since Eq. 6 constrains v_θ only on $\text{supp}(\mu)$, μ determines which distributions the model learns, as illustrated in Fig. 3. Supporting μ on the full interior $[0, 1]^2$ fits the joint distribution and its conditional and marginal paths. Support only on the top edge recovers standard conditional flow-matching inversion, while support only on the left edge learns the audio marginal $p_{\text{data}}^\mathbf{y}$ without parameter labels, allowing unpaired real recordings to be used directly.

γ determines the density being sampled: At inference, the sampling path γ selects the task, as illustrated in Fig. 3. Integrating along the top edge with fixed reference audio $\mathbf{y}^*$ samples $p(\mathbf{x} \mid \mathbf{y}^*)$, the diagonal samples the joint distribution, and the left edge samples the audio marginal. Interior horizontal paths $\{t_\mathbf{y} = \tau\}$ condition on partially noised audio, yielding a smoothed posterior that trades reference specificity against robustness to off-manifold detail.

4. EXPERIMENTS

We evaluate our method on both *in-domain* (synthesized) and *off-manifold* (real-world) audio to assess its ability to generalize beyond the synthesizer manifold. Our setup is designed to isolate the effect of joint density flow matching and the inclusion of unpaired audio, while controlling for architecture and training budget.

4.1 Data

We evaluate on two software synthesizers spanning distinct synthesis paradigms.[2] SURGE XT is a hybrid subtractive/wavetable synthesizer used as the primary testbed in [11]. DEXED is an open-source clone of the Yamaha DX7, a six-operator frequency modulation synthesizer. FM synthesis [26] poses a challenge for parameter estimation algorithms [14, 27]: small parameter perturbations can produce large, non-local timbral changes, and the operator routing ("algorithm") reorganises the entire signal flow. Prior work on DEXED inversion has therefore restricted the

[2] We run full ablations on SURGE XT only, to make optimal use of available computational resources.

problem by fixing or limiting the algorithm [5, 12, 14, 15]. We make no such restriction: our model jointly predicts the algorithm alongside all continuous parameters, across all 32 routings. This is a substantially harder setting, as the semantics of most parameters depend on the algorithm.

Following [11], we generate paired audio–parameter examples by sampling parameters uniformly over their respective ranges and rendering audio through the synthesizer. For SURGE XT we adopt a similar set of active parameters as in [11], but disable audio effects. For both synthesizers, we disable any modulator settings known to introduce non-determinism (e.g. Sample & Hold). Rather than fixing a dataset size, we render online during training, giving effectively unlimited data. Uniform parameter sampling produces a long tail of silent or near-silent outputs, which would otherwise dominate training; we therefore reject any sample with broadband RMS below -60 dBFS. For validation and testing we fix a random seed and freeze a 10k-example set per synthesizer, ensuring determinism across model comparisons.

Continuous parameters are linearly mapped to $[-1, 1]$, and discrete parameters (e.g., DEXED algorithms, Surge oscillator types) are one-hot encoded. Audio is rendered at 44.1 kHz in stereo with a duration of 4.0 seconds. Inputs to the model are log-mel spectrograms with a 25 ms window and 10 ms hop. To avoid hand-tuned input normalisation, we estimate channel-wise mean and variance over the first 8k training spectrograms using Welford's online algorithm and freeze the resulting statistics for the remainder of training.

4.1.1 Off-manifold audio

Our central claim concerns inversion of audio that does not lie on the synthesizer's manifold. To stress-test this regime, we curate a dataset that approximates the intended deployment setting — short, isolated sounds of the kind a user would plausibly hand to an inversion system. We draw from two sources. First, we filter Freesound [28] for files flagged as containing a single acoustic event by the Audio Commons single-event descriptor [29], discarding loops, ambiences, and multi-event recordings. Second, we include a proprietary library of one-shot samples used in music production, covering instruments, percussion, and sound-design material. After deduplication, trimming/padding to 4.0 s, and resampling to 44.1 kHz, we obtain $617{,}016$ training files together with held-out validation and test sets of 10k files each. This data carries no parameter annotations and is consumed exclusively along the audio-marginal edge $\{t_{\mathbf{x}} = 0\}$ defined in Section 3.

4.2 Model variants

We compare four models to isolate the effects of joint training, real-audio marginal supervision, and architectural unification. Across the JDF variants, the only training-time difference is the time-sampling distribution μ. Exact task-mixture weights for μ are given in the supplementary material.

- **JDF**: Our joint formulation, trained on the full time square $\mathrm{supp}(\mu) = [0, 1]^2$ with additional mass on the conditional inversion edge $\{t_{\mathbf{y}} = 1\}$ and forward-synthesis edge $\{t_{\mathbf{x}} = 1\}$.
- **JDF-A**: JDF with additional training on the audio-marginal edge $\{t_{\mathbf{x}} = 0\}$ using both synthetic and unpaired real audio.
- **Unif-**PARAMONLY: Matches the unified JDF architecture, but trained only on the conditional inversion edge $\{t_{\mathbf{y}} = 1\}$.
- **Conditional**: An encoder-based generative inversion baseline following [11], where an AST [30] audio encoder conditions a parameter DiT [31] trained by conditional flow matching over $p(\mathbf{x} \mid \mathbf{y})$.

4.2.1 Architecture and training

All models are trained for 500k steps using a rectified flow parameterisation [24] with linear interpolants. Except where they are fixed to a constant value, time variables are sampled from a logit-normal schedule [32].

Classifier-free guidance (CFG) [33] is supported in all variants. For JDF and JDF-A, the audio-marginal edge $\{t_{\mathbf{x}} = 0\}$ provides a natural unconditional branch. For Unif-PARAMONLY and Conditional, which are not trained on this edge, we use 10% conditioning dropout by replacing audio conditioning with pure noise for the former, and a learnt CFG token for the latter. We sweep guidance weight in Fig. 5. Further training and architecture details are provided in the supplementary material.

4.2.2 Inference

All samples are drawn by integrating v_θ with an Euler solver for 20 steps along the appropriate path in $[0, 1]^2$. For inversion, this is the top edge $\{t_{\mathbf{y}} = 1\}$ with audio fixed at the (possibly partially noised) reference. The conditioning noise level τ (Section 3) and the CFG weight are treated as inference-time hyperparameters and swept in Figs. 4 and 5. CFG is applied on the limited interval $[0.15, 0.95]$ [34].

4.3 Metrics

We report four reconstruction metrics between each input audio signal and its resynthesis obtained by passing the inferred parameters back through the synthesizer, following [11]. These are the multi-scale spectral distance (MSS), warped-MFCC distance (wMFCC), spectral optimal-transport distance (SOT), and the cosine similarity between RMS-energy envelopes (RMS). Each metric is reported on two test conditions of 10k examples each: an in-distribution split drawn from the synthetic data generator, and the off-manifold split of Section 4.1.1.

5. RESULTS

Tables 1 and 2 report the main comparison. JDF-A substantially improves inversion of real audio compared to the Conditional baseline: on SURGE XT, MSS falls from 26.70 to 9.97, and on DEXED from 22.71 to 11.34. The same pattern holds across the remaining metrics, including a large increase in RMS-envelope agreement on DEXED ($0.294 \rightarrow 0.671$). Importantly, the off-manifold improvement is not achieved at the expense of *in-distribution* performance: JDF-A also outperforms the baseline on synthetic audio across all metrics and both synthesizers.

Model	SURGE XT MSS ↓	wMFCC ↓	SOT ↓	RMS ↑
(a) On-manifold (Synth audio)				
JDF	13.62	14.74	0.155	0.870
JDF-A	12.77	14.38	0.144	0.879
Unif-PARAMONLY	**12.30**	**13.93**	**0.131**	**0.883**
Conditional	15.95	17.94	0.188	0.863
(b) Off-manifold (Real audio)				
JDF	10.45	13.68	0.311	**0.749**
JDF-A	**9.97**	**12.94**	**0.236**	0.731
Unif-PARAMONLY	15.32	16.25	0.368	0.659
Conditional	26.70	20.47	0.521	0.500

Table 1. Audio reconstruction results on SURGE XT.

Model	DEXED MSS ↓	wMFCC ↓	SOT ↓	RMS ↑
(a) On-manifold (Synth audio)				
JDF-A	**16.59**	**15.27**	**0.067**	**0.763**
Conditional	20.23	17.63	0.076	0.728
(b) Off-manifold (Real audio)				
JDF-A	**11.34**	**17.25**	0.602	**0.671**
Conditional	22.71	25.61	**0.589**	0.294

Table 2. Audio reconstruction results on DEXED.

While the Conditional baseline's performance degrades off-manifold, the MSS of JDF-A surprisingly *improves*. This is likely explained, in part, by differences in the audio distributions. The strong presence of one-shot audio samples for music production in the off-manifold dataset, for example, means the ends of many windows are silent. In this sense, the framewise energy invariant SOT metric acts as an anchor, and indeed reveals that even for JDF-A, matching fine spectral details is more challenging off manifold.

5.1 Training on off-manifold audio

To isolate the effect of training the audio-marginal edge with unpaired real audio, we compare JDF-A to JDF, which has the same joint formulation but only sees synthesizer audio during training. JDF-A improves over JDF on most metrics, especially off manifold, supporting the hypothesis that training the audio-marginal edge on real recordings helps narrow the train-test gap. Interestingly, JDF-A also gives small gains on several in-domain metrics, which is not directly explained by exposure to real audio. This should not be read as evidence that real-audio inversion is intrinsically easier, since the synthetic and off-manifold test sets differ in content and temporal structure. We speculate that the auxiliary audio-generation objective may regularise the shared representation, in a manner reminiscent of recent representation-alignment effects in diffusion models [35,36]. We thus propose to probe the learnt representations with downstream audio tasks in subsequent work.

5.2 Isolating the effect of joint distribution training

The JDF variants differ from the Conditional baseline both in objective and in conditioning architecture. Conditional uses an AST audio encoder whose representation conditions a parameter DiT decoder, whereas JDF processes audio and parameter tokens jointly in a single shared DiT. This removes the AST bottleneck and AdaLN-style conditioning [31], replacing them with direct self-attention between audio and parameter tokens. Thus, any improvement over Conditional could reflect a more expressive conditioning pathway rather than the joint-density objective itself. Unif-PARAMONLY controls for this possibility by using the same shared-token DiT as JDF while training only on the conditional inversion edge $\{t_{\mathbf{y}} = 1\}$. Its strong in-domain performance, including the best SURGE XT MSS, shows that architectural unification does contribute to the overall improvement. However, its off-manifold performance remains substantially worse than JDF and JDF-A, indicating that architecture alone does not explain the robustness gain.

This supports the central role of the joint distribution formulation. When $\mathrm{supp}(\mu) = [0,1]^2$, most audio states encountered during training are partially noised rather than clean synthesized spectrograms. As a result, the model learns a velocity field in a neighbourhood around the synthesizer manifold, making off-manifold inputs less of an extrapolation problem at inference.

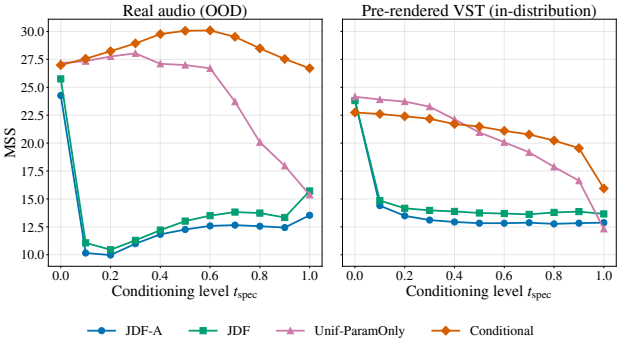

Figure 4. Effect of conditioning noise level on MSS.

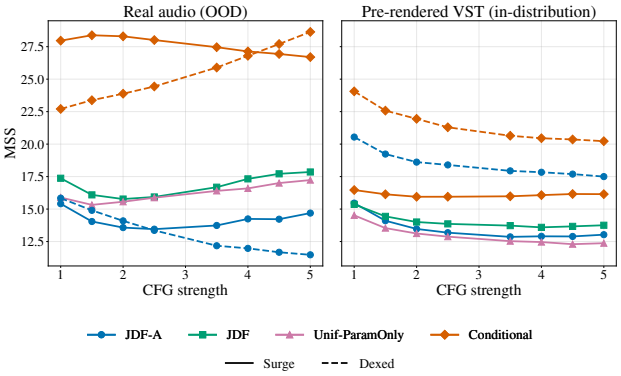

Figure 5. Effect of classifier-free guidance (CFG) weight. CFG is available for JDF and JDF-A models without conditioning dropout, as fully noised audio creates a natural unconditional branch.

5.3 Classifier-free guidance for free

JDF variants obtain an unconditional branch directly from the bottom edge $\{t_{\mathbf{y}} = 0\}$, whereas Unif-PARAMONLY and the Conditional baseline require conditioning dropout to

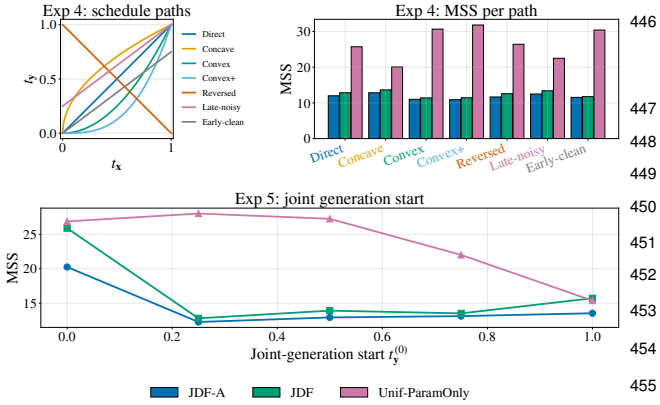

Figure 6. Top left: Schedules for time-varying conditioning noise. Top right: MSS on real audio for each schedule. Bottom: joint sampling from a partially noised reference at $(0, t_{\mathbf{y}}^{(0)})$, integrating to $(1, 1)$.

approximate the same branch. Fig. 5 shows that CFG modestly improves in-domain performance for all models. Off manifold, however, the Conditional baseline behaves less predictably: on DEXED, CFG degrades performance, while on SURGE XT the curve is seemingly inverted, worsening at low weights and improving at high weights.

We interpret this as a sign that CFG amplifies the Conditional model's already biased off-manifold predictions [33]. By contrast, all other models exhibit improved performance under CFG, suggesting that this behaviour relates to the architectural differences identified in Section 5.2.

5.4 Noisy conditioning as posterior smoothing

Because v_θ is trained over the full time square, JDF models learn audio-conditioned parameter posteriors at intermediate audio noise levels, not only at $t_{\mathbf{y}} = 1$. We can therefore condition on a partially noised reference, $Y_\tau = (1 - t_{\mathbf{y}})Y_0 + t_{\mathbf{y}}\mathbf{y}^*$, and sample from a smoothed posterior that interpolates between the parameter marginal at $\tau = 0$ and the clean-conditioned posterior at $\tau = 1$. Intuitively, lowering τ suppresses recording-specific detail that the synthesizer cannot reproduce while retaining coarse temporal and spectral structure.

Fig. 4 supports this view. On-manifold, conditioning noise harms Conditional and Unif-PARAMONLY, which were trained for clean conditioning, while JDF variants remain stable except at the highest noise levels. Off-manifold, the pattern reverses: JDF and JDF-A improve substantially at high noise levels, especially $\tau = 0.1$ and 0.2, whereas the baselines degrade. Thus the benefit is not generic input corruption, but a trained inference mode that trades specificity against robustness without retraining.

We also vary the full path through the time square in Fig. 6. JDF and JDF-A are relatively insensitive to path choice, even under the pathological reversed path. Unif-PARAMONLY, by contrast, performs poorly away from its training edge. This suggests that the learned JDF velocity field remains useful across multiple regions of the time square, allowing inference-time path choices to affect reconstruction performance without retraining.

5.5 Joint sampling from a partially noised reference

In the idealised flow-matching setting, paths within $\mathrm{supp}(\mu)$ correspond to transports between intermediate distributions. We therefore test whether such paths remain useful in the learned model. Fig. 6 (bottom) integrates along a path beginning at $(0, t_{\mathbf{y}}^{(0)})$ and ending at $(1, 1)$, allowing parameters and audio to co-evolve from a partially noised reference. At $t_{\mathbf{y}}^{(0)} = 1$ the path collapses to standard inversion; at $t_{\mathbf{y}}^{(0)} = 0$ we sample the joint distribution unconditionally. JDF and JDF-A find a shallow optimum near $t_{\mathbf{y}}^{(0)} = 0.25$, close to but slightly worse than fixed-noise conditioning. Unif-PARAMONLY fails everywhere except the degenerate $t_{\mathbf{y}}^{(0)} = 1$ endpoint, again as expected: it has no training experience anywhere off the inversion edge. Joint sampling is therefore possible in our trained models, but does not outperform fixed-noise conditioning in this setting. We report it primarily to illustrate that JDF models remain usable away from the conditional edge.

6. CONCLUSION

We have shown that reframing synthesizer inversion as joint modelling of parameters and audio, rather than purely conditional estimation, improves robustness to real-world off-manifold inputs while preserving in-domain performance. By training a single multi-modal rectified flow over the time square $[0, 1]^2$, paired synthesizer data and unpaired real recordings can be incorporated in one objective, and partially noised conditioning provides a simple inference-time mechanism for trading specificity against robustness. Experiments on SURGE XT and DEXED show substantial improvements over a generative baseline on real-audio inversion, without sacrificing synthetic-audio performance.

Several directions remain open. The flexibility of the joint distribution flow matching framework enables us to include further "views" of our training data which may further improve robustness, and unlocked further options for conditioning and control. Moreover, our model's probabilistic framing suits it well to fine-tuning by reinforcement learning, which may serve to further refine the learnt distribution for off-manifold inputs.

We acknowledge some limitations of our presented approach. First, noting that for off-manifold audio the notion of a matching synthesizer patch is loosely defined, we suggest that human evaluation of such a system should be a crucial component of subsequent work. This should, secondly, be paired with an analysis of failure modes, as aggregate reconstruction metrics obscure potential commonalities between inputs which lead to worse outputs. Finally, certain design choices were selected heuristically and still warrant ablation. Noting the significant effect that tuning the noise schedule has had on other diffusion and flow matching models [32], for example, it is likely that more rigorous exploration of the time square sampling distribution will prove a fruitful avenue for future work.

7. AI USAGE STATEMENT

An AI coding assistant was used during the development of the research codebase, for writing experiment management tooling, and for collating and plotting results. All AI code outputs were subject to extensive human verification to ensure the expected functionality was implemented. The manuscript was drafted and edited with the aid of an LLM. However, all scientific claims, experimental results, and final wording were reviewed and approved by the author.

8. ETHICS STATEMENT

As in the work we build on [11], we train exclusively on synthetically generated data, publicly available audio datasets, and legally obtained private datasets. Our choice of synthesizer reflects a bias toward the conventions of western popular music production. While we expect our method to generalise to other synthesis paradigms, we have not empirically verified this. We view synthesizer inversion as a tool to augment, rather than replace, creative workflows.

9. REFERENCES

[1] A. Horner, J. Beauchamp, and L. Haken, "Machine Tongues XVI: Genetic Algorithms and Their Application to FM Matching Synthesis," *Computer Music Journal*, vol. 17, no. 4, pp. 17–29, 1993, publisher: The MIT Press.

[2] O. Barkan, D. Tsiris, O. Katz, and N. Koenigstein, "InverSynth: Deep Estimation of Synthesizer Parameter Configurations from Audio Signals," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 2385–2396, Dec. 2019, arXiv: 1812.06349.

[3] O. Barkan, M. Laufer, S. Shvartzman, N. Uzrad, A. Elharar, and N. Koenigstein, "InverSynth II: Sound matching via self-supervised synthesizer-proxy and inference-time finetuning," in *Proc. of the 24rd Int. Society for Music Information Retrieval Conf.*, Milan, Italy, 2023.

[4] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, "Flow Synthesizer: Universal Audio Synthesizer Control with Normalizing Flows," *Applied Sciences*, vol. 10, no. 1, p. 302, Dec. 2020.

[5] G. L. Vaillant, T. Dutoit, and S. Dekeyser, "Improving Synthesizer Programming From Variational Autoencoders Latent Space," in *2021 24th International Conference on Digital Audio Effects (DAFx)*. Vienna, Austria: IEEE, Sep. 2021, pp. 276–283.

[6] G. L. Vaillant and T. Dutoit, "Synthesizer Preset Interpolation Using Transformer Auto-Encoders," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2023, pp. 1–5, iSSN: 2379-190X.

[7] N. Masuda and D. Saito, "Synthesizer Sound Matching with Differentiable DSP," Online, Nov. 2021.

[8] ——, "Quality Diversity for Synthesizer Sound Matching," in *2021 24th International Conference on Digital Audio Effects (DAFx)*, Sep. 2021, pp. 300–307, iSSN: 2413-6689.

[9] ——, "Improving Semi-Supervised Differentiable Synthesizer Sound Matching for Practical Applications," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 863–875, 2023.

[10] F. Bruford, F. Blang, and S. Nercessian, "Synthesizer Sound Matching Using Audio Spectrogram Transformers," in *Proceedings of the 27th International Conference on Digital Audio Effects*, Guildford, Surrey, Sep. 2024.

[11] B. Hayes, C. Saitis, and G. Fazekas, "Audio synthesizer inversion in symmetric parameter spaces with approximately equivariant flow matching," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, Daejeong, Korea, Jun. 2025.

[12] W. Shin and K. Lee, "SynthRL: Cross-domain Synthesizer Sound Matching via Reinforcement Learning," vol. 11, Sep. 2025, pp. 10 162–10 170, iSSN: 1045-0823.

[13] J. Shier, "The synthesizer programming problem: improving the usability of sound synthesizers," Master's thesis, University of Victoria, 2021.

[14] Z. Chen, Y. Jing, S. Yuan, Y. Xu, J. Wu, and H. Zhao, "Sound2Synth: Interpreting sound via FM synthesizer parameters estimation," in *Proceedings of the thirty-first international joint conference on artificial intelligence, IJCAI-22*, L. D. Raedt, Ed. International Joint Conferences on Artificial Intelligence Organization, Jul. 2022, pp. 4921–4928.

[15] J. Shier, G. Tzanetakis, and K. McNally, "SpiegeLib: An automatic synthesizer programming library," *Journal of the Audio Engineering Society*, no. 10377, May 2020.

[16] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," Jan. 2020, arXiv:2001.04643 [cs, eess, stat].

[17] B. Hayes, J. Shier, G. Fazekas, A. McPherson, and C. Saitis, "A Review of Differentiable Digital Signal Processing for Music & Speech Synthesis," *Frontiers in Signal Processing*, 2023.

[18] B. Hayes, C. Saitis, and G. Fazekas, "Sinusoidal Frequency Estimation by Gradient Descent," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2023, pp. 1–5.

[19] J. Turian and M. Henry, "I'm Sorry for Your Loss: Spectrally-Based Audio Distances Are Bad at Pitch," *arXiv:2012.04572 [cs, eess]*, Dec. 2020, arXiv: 2012.04572.

[20] S. Li, K. Kallidromitis, A. Gokul, Z. Liao, Y. Kato, K. Kozuka, and A. Grover, "OmniFlow: Any-to-Any Generation with Multi-Modal Rectified Flows," in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2025, pp. 13 178–13 188, iSSN: 2575-7075.

[21] F. Bao, S. Nie, K. Xue, C. Li, S. Pu, Y. Wang, G. Yue, Y. Cao, H. Su, and J. Zhu, "One transformer fits all distributions in multi-modal diffusion at scale," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23, vol. 202. Honolulu, Hawaii, USA: JMLR.org, Jul. 2023, pp. 1692–1717.

[22] M. J. Yee-King, L. Fedden, and M. d'Inverno, "Automatic Programming of VST Sound Synthesizers Using Deep Networks and Other Techniques," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 150–159, Apr. 2018.

[23] Y. Yang, Z. Jin, C. Barnes, and A. Finkelstein, "White box search over audio synthesizer parameters," in *Proc. of the 24rd Int. Society for Music Information Retrieval Conf.*, Milan, Italy, 2023.

[24] X. Liu, C. Gong, and Q. Liu, "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow," Feb. 2023.

[25] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow Matching for Generative Modeling," Feb. 2023.

[26] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *J. Audio Eng. Soc*, vol. 21, no. 7, pp. 526–534, 1973.

[27] F. Caspe, A. McPherson, and M. Sandler, "DDX7: Differentiable FM Synthesis of Musical Instrument Sounds," Aug. 2022, arXiv:2208.06169 [cs, eess].

[28] F. Font Corbera, G. Roma Trepat, and X. Serra, "Freesound technical demo," *MM'13. Proceedings of the 21st ACM international conference on Multimedia; 2013 Oct 21-25; Barcelona, Spain. New York: ACM; 2013. p. 411-2.*, 2013, iSBN: 9781450324045 Publisher: ACM Association for Computer Machinery.

[29] A. Pearce, T. Brookes, and R. Mason, "Timbral Attributes for Sound Effect Library Searching." Audio Engineering Society, Jun. 2017.

[30] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," 2021, pp. 571–575.

[31] W. Peebles and S. Xie, "Scalable Diffusion Models with Transformers," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Paris, France: IEEE, Oct. 2023, pp. 4172–4182.

[32] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, D. Podell, T. Dockhorn, Z. English, and R. Rombach, "Scaling rectified flow transformers for high-resolution image synthesis," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24, vol. 235. Vienna, Austria: JMLR.org, Jul. 2024, pp. 12 606–12 633.

[33] J. Ho and T. Salimans, "Classifier-Free Diffusion Guidance," Dec. 2021.

[34] T. Kynkäänniemi, M. Aittala, T. Karras, S. Laine, T. Aila, and J. Lehtinen, "Applying guidance in a limited interval improves sample and distribution quality in diffusion models," in *Proceedings of the 38th International Conference on Neural Information Processing Systems*, ser. NIPS '24, vol. 37. Red Hook, NY, USA: Curran Associates Inc., Dec. 2024, pp. 122 458–122 483.

[35] S. Yu, S. Kwak, H. Jang, J. Jeong, J. Huang, J. Shin, and S. Xie, "Representation Alignment for Generation: Training Diffusion Transformers Is Easier Than You Think," Oct. 2024.

[36] G. Wu, S. Zhang, R. Shi, S. Gao, Z. Chen, L. Wang, Z. Chen, H. Gao, Y. Tang, J. Yang, M.-M. Cheng, and X. Li, "Representation Entanglement for Generation: Training Diffusion Transformers Is Much Easier Than You Think," Oct. 2025.