INVERSYNTH II: SOUND MATCHING VIA SELF-SUPERVISED SYNTHESIZER-PROXY AND INFERENCE-TIME FINETUNING

Oren Barkan[1] **Shlomi Shvartzman**[2] **Noy Uzrad**[2]
Moshe Laufer[2] **Almog Elharar**[2] **Noam Koenigstein**[2]

[1]The Open Univesity of Israel
[2]Tel Aviv University, Israel

ABSTRACT

Synthesizers are widely used electronic musical instruments. Given an input sound, inferring the underlying synthesizer's parameters to reproduce it is a difficult task known as *sound-matching*. In this work, we tackle the problem of automatic sound matching, which is otherwise performed manually by professional audio experts. The novelty of our work stems from the introduction of a novel differentiable *synthesizer-proxy* that enables gradient-based optimization by comparing the input and reproduced audio signals. Additionally, we introduce a novel self-supervised finetuning mechanism that further refines the prediction at inference time. Both contributions lead to state-of-the-art results, outperforming previous methods across various metrics. Our code is available at: `https://github.com/inversynth/InverSynth2`.

1. INTRODUCTION AND RELATED WORK

Sound synthesis has been an active research field since the end of the previous century [1]. Given a query audio input, the task of crafting a specific sound is known as *sound matching*. Synthesizer sound matching, also known as *inverse synthesis*, involves carefully tuning parameters from an exponentially large number of possible configurations- a task mostly reserved for professional audio experts. This paper presents a novel algorithmic approach for *automated sound matching*.

Algorithmic approaches for inverse synthesis can be loosely categorized into *search-based* methods and *modeling-based* methods [2]. Search-based methods often utilize genetic algorithms (GA) which are based on principles of Darwinian evolution to determine the optimal synthesizer configurations. For instance [3] initiated a set of randomly sampled configurations and used GA optimization to reconstruct the original audio signal. Other search-based methods include Particle Swarm Optimization (PSO) [4] and Hill-Climbing [5]. Search-based methods can employ different objectives such as mel-frequency cepstral coefficients (MFCCs) or a combination of multiple objectives [6, 7]. However, optimizing configurations through search-based methods can be both resource-intensive and time-consuming for every sound sample. Consequently, the rise of deep learning techniques has led to a shift from search-based methods to model-based ones, which avoid the previously mentioned drawbacks. However, search-based methods still possess a unique advantage: they can establish a loss term that directly contrasts the reconstructed audio signal with the input signal.

The aforementioned advantage is absent in most modeling-based methods, as they usually cannot propagate gradients through an external, commercial synthesizer. As a result, they depend on setting an optimization goal focused on reconstructing the parameters rather than the reproduced signal. In general, modeling-based methods employ deep learning in order to predict a synthesizer's configuration based on the input audio signal. For example, [8] employed long short-term memory (LSTM) networks for predicting the parameters in FM synthesizers. InverSynth (IS) [9] employed strided convolution neural networks (CNNs) to estimate a synthesizer's parameters as a multi-objective classification problem. When compared to the LSTMs approach of [8], IS provides improved accuracy with the ability to scale for longer audio sequences. Another direction involves employing variational inference [10] and normalizing flows [11, 12] to automatically tune an open-source replica of the Yamaha DX7 synthesizer [13]. Finally, [14–16] introduce a different versions of audio synthesizer models for sound matching.

A completely different direction for sound matching and synthesis is through neural synthesizers [17–22]. For example, in [19] the authors train Generative Adversarial Networks to synthesize sounds that simulate natural audio samples. However, these directions are inherently different from the current line of work, as they do not deal with the problem of tuning existing musical synthesizers. Instead, these works suggest alternatives to familiar synthesizers, which may be useful for future applications but are less relevant to mainstream musicians that use existing commercial synthesizers.

In this paper, we present InverSynth II (IS2) - an innovative inverse-synthesis model that introduces a differentiable synthesizer-proxy capable of learning to "imitate" the behavior of any given synthesizer. This allows for a differentiable relationship between the synthesizer's parameters and the produced audio signal. As a result, IS2 learns to focus on the synthesizer parameters that have more impact on the reproduced signal. Our evaluations indicate that this approach leads to a better reconstruction of the original audio signal in terms of spectral loss and human perception.

Our contributions are as follows: (1) We introduce IS2 that effectively incorporates the synthesizer's functionality into the computational graph. By learning a differentiable *synthesizer-proxy*, IS2 facilitates self-supervision based on the difference between the input and reproduced audio signals. This is in contrast to previous model-based works that optimized on the predicted synthesizer parameters alone [8, 9, 23]. (2) We introduce a novel self-supervised finetuning technique that utilizes the learned synthesizer-proxy to further refine predictions at inference time. (3) We compare IS2 against the state-of-the-art methods from [10] and [9] on the three datasets, including the datasets from both of these works. Our findings show that IS2 outperforms both methods on all datasets, across all metrics.

2. INVERSYNTH II

2.1 Problem Setup

Let $x \in \mathcal{X}$ be the audio signal i.e., raw waveform, Short-time Fourier transform (STFT) spectrogram, etc. Let $f : \mathcal{Y} \rightarrow \mathcal{X}$ be a synthesizer function that generates a signal $f(y) \in \mathcal{X}$ according to the parameters configuration $y \in \mathcal{Y}$, where y encodes the exact value for each of the configurable synthesizer parameters. For example, these parameters determine the oscillators' waveform types, the amplitudes' values, modulation indexes, ADSR envelopes, filter cutoff frequency, etc. The inverse-synthesis task is to learn an *encoder* function $e_\theta : \mathcal{X} \rightarrow \mathcal{Y}$, parameterized by θ, that receives an audio $x \in \mathcal{X}$ and predicts the parameters configuration $e_\theta(x) \in \mathcal{Y}$ s.t.

$$f(e_\theta(x)) = x' \approx x. \tag{1}$$

2.2 The IS Model

The IS model from [9] receives an input signal x and aims at inferring a parameters configuration $\hat{y}$ which best matches the true yet unknown parameters configuration y that produced $x = f(y)$. To this end, a dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ is generated, where y_i is the synthesizer's configuration used by f to generate the sound x_i, hence $f(y_i) = x_i$. IS trains an encoder network e_θ to predict y_i from x_i by minimizing the objective

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}_p(e_\theta(x_i), y_i), \tag{2}$$

where $\mathcal{L}_p : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ is the *parameters loss* that quantifies the difference between the predicted configuration $e_\theta(x_i)$ and the ground truth configuration y_i. In [9], each synthesizer parameter was treated as a categorical variable (continuous parameters were quantized), hence solving multiple classification problems simultaneously (one for each parameter). Accordingly, the loss $\mathcal{L}_p$ was the sum of P cross-entropy (CE) losses, where P is the number of the synthesizer parameters.

2.3 The IS2 Model

IS does not optimize on the actual reproduced audio signal. Instead, it only optimizes on the parameters configuration according to Eq. 2. However, minimizing $\mathcal{L}_p$ is just a proxy to the original task from Eq. 1 that aims at minimizing the difference between the original signal x and the reproduced signal $f(e_\theta(x))$. This observation motivates an additional self-supervised loss term $\mathcal{L}_a : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, namely the *audio loss*, that measures the discrepancy between the input signal x and the reproduced signal $f(e_\theta(x))$:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}_p(e_\theta(x_i), y_i) + \lambda \mathcal{L}_a(f(e_\theta(x_i)), x_i), \tag{3}$$

where λ is a hyperparameter. The audio loss term $\mathcal{L}_a$ provides feedback on the quality of the reproduced signal $f(e_\theta(x_i))$ itself, hence better aligns with the ultimate task of Eq. 1.

Yet, a key challenge arises - how to backpropagate the error induced by $\mathcal{L}_a$ via f? A naive approach may propose implementing the synthesizer f as part of the computational graph. However, this approach suffers from several limitations: First, it requires a specific implementation per synthesizer and hence does not scale. Second, most commercial synthesizers are not open-source, and even if the source code was provided, it would still require rewriting of the entire codebase to support an auto-differentiation platform (e.g., PyTorch). Furthermore, some synthesizer functionalities are not differentiable and require workarounds that may incur discrepancies and hinder gradient-based optimization.

To this end, IS2 introduces a *synthesizer-proxy* decoder network $d_\phi : \mathcal{Y} \rightarrow \mathcal{X}$, parameterized by ϕ, that serves as a differential replacement to the true synthesizer function f. d_ϕ is trained to minimize $\mathcal{L}_a$ w.r.t. ϕ over the dataset D, which leads to the IS2 training objective

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}_p(e_\theta(x_i), y_i) + \lambda \mathcal{L}_a(d_\phi(e_\theta(x_i)), x_i), \tag{4}$$

with $\Theta = \{\theta, \phi\}$.

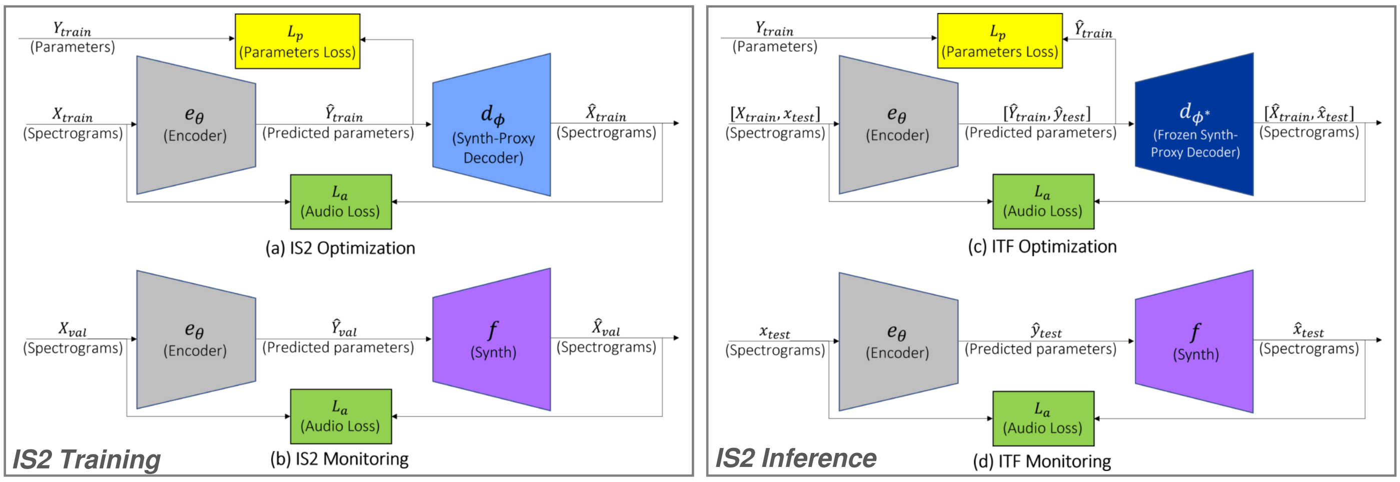
2.4 IS2 Training

IS2 employs stochastic gradient descent optimization [24] on the objective from Eq. 4 as depicted in Fig. 1(a) (the exact implementation and optimization details will follow in Secs. 2.6 and 3.2). We apply a K-fold cross-validation procedure over the dataset D, where each fold defines different training, validation, and test sets. For each fold, we train the IS2 model on the training set and monitor the following measure on the validation set $V \subset \{1..N\}$:

$$\mathcal{L}_V^f := \sum_{i \in V} \mathcal{L}_a(f(e_\theta(x_i)), x_i). \tag{5}$$

Finally, the best-performing model (in terms of $\mathcal{L}_V^f$ across all epochs) is used for reporting results on the test set.

Note that the predicted parameters $e_\theta(x)$ in Eq. 5 are propagated to the *true* synthesizer f and not to the synthesizer-proxy d_ϕ (see Fig. 1(b)). This enables selecting the model that minimizes the discrepancy between x and $f(e_\theta(x))$, which aligns with the ultimate task of Eq. 1. Yet, f does not participate in the optimization objective (Eq. 4) since it is not necessarily differentiable. Instead,



Y_{train} (Parameters)
L_p (Parameters Loss)
X_{train} (Spectrograms)
e_θ (Encoder)
$\hat{Y}_{train}$ (Predicted parameters)
d_ϕ (Synth-Proxy Decoder)
$\hat{X}_{train}$ (Spectrograms)
L_a (Audio Loss)
(a) IS2 Optimization
X_{val} (Spectrograms)
$\hat{Y}_{val}$ (Predicted parameters)
f (Synth)
$\hat{X}_{val}$ (Spectrograms)
(b) IS2 Monitoring
IS2 Training
$[X_{train}, x_{test}]$ (Spectrograms)
$[\hat{Y}_{train}, \hat{y}_{test}]$ (Predicted parameters)
d_{ϕ^*} (Frozen Synth-Proxy Decoder)
$[\hat{X}_{train}, \hat{x}_{test}]$ (Spectrograms)
(c) ITF Optimization
x_{test} (Spectrograms)
$\hat{y}_{test}$ (Predicted parameters)
$\hat{x}_{test}$ (Spectrograms)
(d) ITF Monitoring
IS2 Inference

Figure 1: (a)-(b) depict the IS2 training phase (Sec. 2.4) that utilizes the differentiable synthesizer-proxy d_ϕ, while monitoring for the best model via the true synthesizer f. (c)-(d) depict the IS2 inference phase (Sec. 2.5) that employs ITF, utilizing the optimized d_{ϕ^*} for improved parameters prediction on the specific test example. Again, f is used for monitoring.

the audio loss term $\mathcal{L}_a$ in Eq. 4 utilizes d_ϕ as a differentiable proxy to f in order to propagate gradients as part of the optimization process.

2.5 IS2 Inference

A unique feature of IS2 is the ability to improve the predictions at inference time, by employing Inference-Time Finetuning (ITF). Given a test input x_t, we utilize the audio loss $\mathcal{L}_a$ for leveraging self-supervision from x_t, and refine the prediction specifically for x_t. To this end, we freeze the trained decoder parameters ϕ^* (Eq. 4) and finetune the trained encoder parameters θ^* to obtain finetuned parameters θ^t:

$$\theta^t = \underset{\theta}{\text{argmin}}\ \mathcal{L}_t + \lambda_B \mathcal{L}_B, \tag{6}$$

where

$$\mathcal{L}_t = \mathcal{L}_a(d_{\phi^*}(e_\theta(x_t)), x_t),$$
$$\mathcal{L}_B = \frac{1}{|B|}\sum_{i\in B}\mathcal{L}_p(e_\theta(x_i), y_i) + \lambda\mathcal{L}_a(d_{\phi^*}(e_\theta(x_i)), x_i),$$

and $B \subset \{1..N\}$ is a subset of indexes from the training set (a training batch). While one could optimize only $\mathcal{L}_t$ w.r.t. θ, we found that the inclusion of $\mathcal{L}_B$ serves as a regularization (controlled by the hyperparameter λ_B) that leads to more accurate predictions. This can be explained by the fact that $\mathcal{L}_B$ enforces the encoder to predict accurate configurations for the examples in B, effectively safeguarding the encoder from forgetting what it has learned during the training phase (Sec. 2.4) and avoid overfitting the test example x_t. In practice, the ITF procedure alternates between sampling a batch of examples from the training set $B \subset \{1..N\}$, and performing gradient descent update to θ according to the objective in Eq. 6, until either convergence w.r.t. $\mathcal{L}_t^f := \mathcal{L}_a(f(e_\theta(x_t)), x_t)$ is met or the number of alternations exceeds a prescribed threshold. ITF optimization and monitoring are depicted in Fig. 1(c)-(d).

It is important to clarify that ITF is applied per test example, i.e., for each test example x_t, we first initialize $\theta \leftarrow \theta^*$, where θ^* are the optimal encoder parameters obtained from the IS2 training procedure (Eq. 4). Then, ITF alternations are employed according to Eq. 6 to obtain finetuned encoder parameters θ^t that might improve $\mathcal{L}_t^f$. However, improvement is not guaranteed due to an inherent discrepancy that may exist between the synthesizer-proxy decoder d_{ϕ^*} (used in $\mathcal{L}_t$) and the synthesizer f (used in $\mathcal{L}_t^f$). Therefore, if none of the ITF alternations yield improvement to $\mathcal{L}_t^f$, we fallback to the prediction obtained by the originally trained encoder $e_{\theta^*}(x_t)$ (that serves as a starting point for the ITF procedure).

2.6 IS2 Implementation

In [9], various encoder implementations were investigated and the spectrogram-based strided CNN encoder stood out as the best performer. Following this finding, we implement the encoder e_θ and decoder d_ϕ as strided CNNs. Accordingly, $x \in \mathcal{X}$ stands for the *processed* log-magnitude spectrogram or mel-spectrogram domain (where the spectrogram is obtained by the application of the STFT to the waveform), and $\mathcal{L}_a$ is set to the Mean Absolute Error, hence measuring the spectral difference between the input and reproduced signals.

The synthesizer parameters configuration is encoded by a super-vector $y \in \mathcal{Y}$ that concatenates one-hot vectors and normalized scalars representing the categorical and continuous parameters, respectively. Accordingly, the parameters loss $\mathcal{L}_p$ is set to the average of the cross-entropy and L2 losses for categorical and continuous parameters, respectively. The exact details of the data processing, data representation, and hyperparameters settings appear in Sec. 3.

3. EXPERIMENTAL SETUP AND RESULTS

3.1 Datasets, preprocessing, and data representation

In this study, we present findings from analyses conducted on three distinct datasets. As a consequence of space constraints, it is not feasible to detail all the numerous configurable parameters of every synthesizer utilized in our experiments. Nevertheless, a comprehensive account of

Metric	**Flow**	**IS**	**IS2xITF**	**IS2**
FM Dataset				
Spec (x100)	4.89	1.61	1.54	**1.51**
Melspec (x100)	193.93	56.77	54.65	**53.84**
MFCC (x100)	73.49	28.83	27.74	**27.29**
SC	0.0941	0.0383	0.0367	**0.0361**
ACC (%)	93.01	93.89	93.97	**94.04**
DX7 Dataset				
Spec (x100)	65.31	58.83	58.59	**58.18**
Melspec (x100)	24.04	19.29	19.37	**19.26**
MFCC (x100)	1502.2	1309.5	1300.4	**1280**
SC	1.0472	0.8578	0.8594	**0.8532**
ACC (%)	85.36	86.07	86.34	**86.74**
MAEparam (x100)	10.77	9.79	9.68	**9.56**
TAL Dataset				
Spec (x100)	0.44	0.1809	0.177	**0.173**
Melspec (x100)	106.5	68.06	67.07	**64.64**
MFCC (x100)	8.95	5.85	**5.72**	5.8
SC	0.51	0.512	0.467	**0.424**
ACC (%)	80.94	80.62	80.73	**81.17**

Table 1: Aggregated results on all datasets and metrics.

each synthesizer parameter can be found in the supplementary material accompanying this manuscript. The datasets which were used in this research are as follows: (1) **FM**: is based on the FM synthesizer implementation that is available in IS2 GitHub repository. The synthesizer is composed of a FM oscillator, AM modulation, and low-pass filter. It includes 9 configurable parameters, each represented by a categorical variable. Continuous parameters were discretized and binned to create a finite set of values. A dataset of 180K audio samples (1 second, 16KHz) was generated based on a random sampling of parameter configurations. Samples were transformed into 257x129 spectrograms using log-magnitude STFT (with window size 512 and hop size 128) followed by normalization to [-1,1]. (2) **DX7**: is the dataset from [10] which is based on the Dexed synthesizer [1] which is a virtual replica of the Yamaha DX7 synthesizer with 144 configurable parameters (represented by 54 categorical and 90 continuous variables). It contains 30K audio samples (3 seconds, 22.05KHz). Each sample was transformed into a 257x347 mel-spectrogram (257-bins) of the log-magnitude STFT (with window size 1024 and a hop size 256), followed by normalization to [-1,1]. (3) **TAL** is based on the commercial synthesizer: TAL-NoiseMaker [2]. It consists of 180k audio samples generated using 9 configurable parameters controlling the oscillator, LFO1, LFO2, and cutoff parameters. Each sound has a duration of 1 second sampled at 16kHz and is converted into a 257x129 spectrogram. The spectrograms are normalized to the range [-1, 1] using the same method as the FM synthesizer dataset. The code for the generation of the datasets, including the parameter discretization process is available in our GitHub repository.

The above datasets encompass a broad spectrum of sounds that vary from basic sine waves to intricate waveforms with a wealth of harmonics. The TAL Noisemaker and FM synthesizer datasets comprise a range of sounds including bass, leads, pads, plucks, and percussion, while the DX7 dataset comprises percussive, bell-like, and metallic sounds, in addition to rich pads and complex bass sounds.

[1] `https://github.com/asb2m10/dexed`

[2] `https://tal-software.com/products/tal-noisemaker`

Our GitHub repository includes scripts that can reproduce the these datasets.

3.2 Evaluated methods and hyperparameters setting

The following models were evaluated: (1) **IS2**: our model from Sec. 2. e_θ and d_ϕ are implemented by strided and transposed CNNs with 9 hidden LeakyReLU activated layers and Batch Normalization [25] (the exact hyperparameters which were chosen for each layer can be seen in our GitHub code). The IS2 objective (Eq. 4) was optimized with $\lambda = 1$ using the Adam optimizer [26] with $\beta_1 = 0.9$, $\beta_2 = 0.99$, batch size 64 and learning rate scheduling from 10^{-4} to 10^{-6} for 100 epochs. While training, we monitored $\mathcal{L}_V^f$ (Eq. 5) on the validation set, and the best-performing model was selected eventually. For each test sample, we employed 30 ITF alternations according to the objective from Eq. 6, with $\lambda_B = 1$, and B is a stochastic sample of 64 examples drawn randomly from the training set at each alternation. Finally, L_t^f was monitored for selecting the best result as explained in Sec. 2.5. (2) **IS2xITF**: an ablated version of IS2, in which ITF is not employed and the predictions are performed by the trained encoder e_{θ^*}. (3) **IS**: the IS method from [9]. (4) **Flow**: the method from [10] which is based on variational inference with normalizing flows. We tuned hyperparameters for all models using the validation set.

3.3 Evaluation metrics

We report the average results obtained by a 5-fold cross-validation procedure with 80%-10%-10% (training, validation, test) splits, on the following metrics: (1) **Spec**: the Mean Absolute Error (MAE) between the log-magnitude STFTs of a - the signal reproduced by the application of f to the predicted parameters configuration, and b - the ground truth configuration signal. (2) **Melspec**: the MAE between the mel-spectrograms of a and b. (3) **MFCC**: the MAE between the 40-band MFCCs of a and b. (4) **SC**: the Spectral Convergence [27] between a and b. Note that SC was found less correlated with human perception [10], nevertheless we report this metric for the sake of completeness. (5) **ACC**: the accuracy of the predicted categorical synthesizer parameters. (6) **MAEparam**: MAE between the predicted and ground truth numerical synthesizer parameter values. This metric is reported for the DX7 dataset only, as all parameters in the FM and TAL dataset are modeled by categorical variables. Metrics (1)-(4) measure errors in the reproduced signal $f(e_\theta(x))$, while metrics (5)-(6) measure accuracy / error w.r.t. the ground truth parameter configurations.

To complete our evaluations, we also present the results of a MOS (Mean Opinion Score) test [28] with $N = 20$. The MOS test involves presenting a set of synthesized sounds to a panel of listeners, who are then asked to rate the sound quality of the reconstructed sound with respect to the original sound using a standardized rating scale: $[1-5]$.

3.4 Quantitative Results

Table 1 displays the results obtained by all methods across all datasets and evaluation metrics. The ACC and MAEparam are averages across all categorical and continuous parameters, respectively. It is important to note that the Flow results reported in [10] were replicated, and our

	TAL Dataset				**FM Synth Dataset**				**DX7 Dataset**			
	IS2	**IS2xITF**	**Flow**	**IS**	**IS2**	**IS2xITF**	**Flow**	**IS**	**IS2**	**IS2xITF**	**Flow**	**IS**
Low (x100)	**608**	627	619	642	**0.62**	0.63	1.88	0.79	**0.53**	0.54	0.61	0.533
Mid (x100)	**86.82**	90.26	102	91.64	**0.18**	0.19	0.51	0.24	0.02658	0.02652	0.0338	**0.0265**
High (x100)	**5.82**	6.12	7.7	6.64	**0.0036**	0.0037	0.0224	0.0098	0.0038	0.0037	0.0043	**0.0035**
All bands (x100)	**654**	675	676	691	**0.63**	0.64	0.19	0.8	**0.53**	0.538	0.6	0.537

Table 2: Spectral analysis for different Mel-frequency bands (x100).

DX7 Param	**Type**	**Flow**	**IS**	**IS2**	**IS2xITF**
ALGORITHM	cat	0.5758	0.6660	**0.6676**	0.6627
FEEDBACK	cat	0.6938	0.7056	**0.7179**	0.7151
OSCKEYSYNC	cat	0.8227	0.8269	**0.8356**	0.8356
LFOSPEED	num	12.5070	**11.5924**	11.6528	11.6337
LFODELAY	num	16.5244	15.0604	14.9376	**14.8934**
LFOPMDEPTH	num	13.0251	11.7224	**11.5284**	11.5593
LFOAMDEPTH	num	17.8000	17.7326	**17.5715**	17.6148
LFOKEYSYNC	cat	0.8008	0.8112	**0.8163**	0.8150
LFOWAVE	cat	0.7599	0.7622	**0.7665**	0.7618
PMODESENS	cat	0.6214	0.6473	**0.6598**	0.6590
PITCHEGRATE1	num	17.6189	16.8161	**16.6582**	16.6706
PITCHEGRATE2	num	17.7838	**16.7808**	16.8625	16.8100
PITCHEGRATE3	num	18.2013	17.6938	**17.4543**	17.5203
PITCHEGRATE4	num	19.5772	**18.6485**	18.7511	18.7084
PITCHEGLEVEL1	num	**6.2437**	6.3104	6.3948	6.3948
PITCHEGLEVEL2	num	**6.5503**	6.8803	6.8803	6.8803
PITCHEGLEVEL3	num	**6.8834**	7.1543	7.1543	7.1543
PITCHEGLEVEL4	num	**6.1773**	6.4220	6.4220	6.4220
OP_EGRATE1	num	13.4020	12.2691	12.1219	**12.1169**
OP_EGRATE2	num	17.7775	**16.8140**	16.9248	16.8298
OP_EGRATE3	num	17.9113	17.0677	17.1096	**17.0545**
OP_EGRATE4	num	12.6359	11.8326	11.8028	**11.7685**
OP_EGLEVEL1	num	**10.6051**	10.7961	10.8860	10.9058
OP_EGLEVEL2	num	17.9278	16.8495	**16.7416**	16.7842
OP_EGLEVEL3	num	21.2140	20.3835	20.2471	**20.2462**
OP_EGLEVEL4	num	**12.1882**	15.1754	15.1754	15.1754
OP_OUTPUTLEVEL	num	12.4545	11.5483	**11.5102**	11.5341
OP_MODE	cat	0.9359	0.9404	**0.9446**	0.9430
OP_FCOARSE	cat	0.6951	0.7486	**0.7538**	0.7513
OP_FFINE	num	13.8020	13.3172	**13.2250**	13.2555
OP_OSCDETUNE	cat	0.6578	0.7275	**0.7346**	0.7299
OP_BREAKPOINT	num	16.3170	**15.4886**	15.5501	15.4929
OP_LSCALEDEPTH	num	16.0303	16.0030	**15.9739**	16.0440
OP_RSCALEDEPTH	num	16.3427	15.7286	**15.6248**	15.6984
OP_LKEYSCALE	cat	0.8476	0.8505	**0.8574**	0.8526
OP_RKEYSCALE	cat	0.8533	0.8541	**0.8643**	0.8580
OP_RATESCALING	cat	0.7187	0.7528	**0.7598**	0.7579
OP_AMODSENS	cat	0.9112	0.8987	**0.9185**	0.9052
OP_KEYVELOCITY	cat	0.6777	0.7236	**0.7290**	0.7267
MEAN CAT	-	0.7551	0.7796	**0.7875**	0.7838
MEAN NUM	-	14.3	13.8434	**13.8064**	13.8067

Table 3: Aggregated DX7 parameters' accuracy. The functionality of each parameter is explained in the supplementary materials (appears in our GitHub repository).

Param TAL	Flow	IS	IS2	IS2xITF
x3_FilterCutoff (%)	**86.08**	72.01	75	72.8
x24_Osc2Waveform (%)	**99.82**	99.38	99.48	99.39
x20_Osc2Tune (%)	**95**	93.49	93.7	93.6
x26_Lfo1Waveform (%)	68.28	78.33	**78.8**	78.38
x28_Lfo1Rate (%)	47.54	52.93	**53.34**	52.97
x30_Lfo1Amount (%)	**73.37**	71.74	72.19	71.78
x27_Lfo2Waveform (%)	88.86	**88.87**	88.85	88.76
x29_Lfo2Rate (%)	**84.77**	84.56	84.67	84.59
x31_Lfo2Amount (%)	**84.77**	84.28	84.45	84.31
MEAN (%)	80.94	80.62	**81.17**	80.73

Table 4: TAL parameters' accuracy. The parameters are prefixed with "xAB", where AB denotes the index of the parameter within the synthesizer. The functionality of each parameter is explained in the supplementary materials (appears in our GitHub repository).

Param FM	Flow	IS	IS2	IS2xITF
osc1_wave (%)	**99.98**	99.94	99.94	99.94
osc1_freq (%)	91.26	98.7	**98.83**	98.81
osc1_mod_index (%)	93.08	96.43	**96.5**	96.45
lfo1_freq (%)	**99.95**	99.86	99.88	99.87
lfo1_wave (%)	**99.52**	98.75	98.69	98.67
am_mod_wave (%)	67.73	71.02	**71.74**	71.59
am_mod_freq (%)	**86.23**	82.71	82.88	82.86
am_mod_amount (%)	**99.43**	97.62	97.64	97.61
filter_freq (%)	**99.98**	99.93	99.95	99.94
MEAN (%)	93.02	93.89	**94.01**	93.97

Table 5: FM Synth parameters' accuracy. The functionality of each parameter is explained in the supplementary materials (appears in our GitHub repository).

Dataset	**Flow**	**IS**	**IS2xITF**	**IS2**
FM	4.7	4.85	4.6	**5**
DX7	1.35	2.45	3.28	**3.5**
TAL	3.87	3.37	3.85	**3.95**

Table 6: MOS test results. Scores on a scale of $[1-5]$ represent the perceptual reconstruction quality w.r.t. the original audio.

results are consistent with the original findings. Table 1 demonstrates that our IS2 method outperforms the other baselines in all metrics and datasets, except for the MFCC score on the TAL datasets, where the ablated version of IS2, IS2xITF, outperforms it. Furthermore, the results indicate that the ablated version IS2xITF is highly effective in comparison to previous baselines which highlights the general utility of the IS2 architecture even without the ITF phase. In the following section, we aim to provide a more comprehensive analysis and interpretation of these results.

To provide additional perspective, we conducted the following analysis: We partitioned the 257 mel-spectrogram bins into "Low", "Mid", and "High" equally sized mel-frequency bands. Then, for each sound in the test set, we computed the L2 loss between the reproduced version and the ground-truth of each mel-frequency band. The results for the different frequency bands, including the entire mel-spectrogram ('All bands') are presented in Table 2. First, We observe that IS2 outperforms the other models on the entire mel-spectrogram ('All bands'), across all datasets, which is consistent with the results presented in Table 1 (note that Tables 1 and 2 report different metrics, i.e, MAE vs. L2). Specifically, IS2 performs particularly well on low frequency regime ('Low'). Arguably, this finding might be explained later where we shall see that the IS2 model attains the best loss in 4 out of 6 low-frequency oscillator (LFO) parameters, which have a stronger impact on the low bands. This calls for further research into the relationship between parameter prediction accuracy and the mel-spectrogram error.

In terms of the "Mid" band, the IS2 model demonstrated superior performance on the TAL and FM datasets, whereas the IS model exhibited better results on the DX7 dataset. For the "High" band, different baselines achieved the best outcomes. This observation is not surprising since high frequencies typically undergo rapid changes and can be less perceptible even to experienced listeners. Overall, our findings indicate that the IS2 approach exhibits robust performance across various datasets and frequency bands, with exceptional accuracy in estimating low frequencies.

Next, we turn to evaluate the accuracy of predicting each parameter specifically. The DX7 synthesizer consists of two types of parameters: categorical, denoted as "cat", and numerical, denoted as "num". To evaluate performance, ACC was reported for categorical parameters, while MAEparam was calculated for numerical parameters, as previously mentioned. The DX7 parameters are further categorized into several groups, including Algorithm, Feedback, Operators, Pitch Envelope Generator, LFO, and Filter, with a comprehensive explanation of each parameter available in the supplementary material. Table 3 outlines the prediction results for the DX7 parameters. Note that parameters beginning with the prefix "OP" are an average aggregation of the six operators of the synthesizer, as explained in the supplementary material.

The results reveal that the IS2 method consistently outperformed other models in predicting all categorical parameters. Furthermore, the IS2 model also outperformed other models in 9 out of 25 numerical parameters. Notably, the IS2xITF model performed best among all models for the "OP_EGRATEi" (i=1...4) numerical parameters, demonstrating superior performance even without fine-tuning (ITF). Specifically, this model exhibited better performance in all numerical parameters and outperformed other models in 5 out of 25 numerical parameters. In contrast, the IS and Flow models demonstrated similarly robust performance, outperforming other models in fewer numerical parameters, namely 11 out of 25.

Overall, Table 3 displays a trend where all categorical parameters are more accurately estimated by the IS2 model. In terms of numerical parameters, the IS2 model performs better in predicting parameters with greater perceptual significance, such as LFO. However, parameters of lesser perceptual significance, such as the envelope level of pitch (PITCHEGLEVEL), exhibit lower estimation accuracy. These trends are consistent with the findings presented earlier in Table 2

The trends observed in Table 3 repeat themselves in the TAL dataset (Table 4) and the FM Synth dataset (Table 5), with slightly improved accuracy for the alternative models. Nevertheless, the IS2 model maintains the highest mean accuracy. IS2 does not achieve the highest accuracy in certain parameters, such as filter cutoff, which are of lesser significance for perceived quality. For example, if the filter cutoff is estimated for class A instead of the correct class B, and A and B are neighboring classes, the impact on perception might be insignificant. Another set of parameters that demonstrate negligible differences in accuracy between models is Oscillator 2, LFO1 amount, and LFO2 values. In contrast, parameters such as LFO1 waveform and rate play a crucial role in controlling Oscillator 2 modulation and impacting the low frequencies of the sound, making them significant in terms of perception. Here, IS2 achieves significantly higher accuracies compared to other models. These findings are consistent with the low losses of the IS2 model for low frequencies, as presented in Table 2.

In Table 5, Oscillator 1 waveform, LFO1, and filter cutoff frequency exhibit negligible differences in accuracy in favor of the alternative baselines. Higher differences are observed in AM modulation parameters. Nevertheless, these parameters primarily affect the Tremolo effect, which has a relatively no impact at all on the frequency composition, and for small changes, leading to less influence on human perception and less impact on the metrics presented in Table 1. Compared to the Flow model, the parameters with the most significant differences are Oscillator 1 frequency and modulation index, which have a significant impact on perception by affecting the carrier frequency of the signal.

Overall, the results in Tables 1-5 indicate that by leveraging information on the difference between the original and reproduced signal during training and inference (ITF), IS2 promotes accurate predictions for parameters that have the most significant impact on human perception, especially FM modulator parameters which very challenging to estimate. Consequently, IS2 produces reconstructions that more closely resemble the original signal, which is the ultimate goal of sound matching. In what follows, we further substantiate our findings via human subjective evaluation.

3.5 MOS Test and Qualitative Results

Table 6 presents MOS test results conducted using $N = 20$ individuals. Participants were asked to rate the reconstruction score of 80 random audio samples on a $[1 - 5]$ scale. The results are inline with those of Table 1, showing that the IS2 model outperforms the other models. Furthermore, the MOS test results indicate that the IS2 model has the highest perception quality of all the models evaluated. The samples from this test are available for listening on the GitHub repository.

Finally, in the supplementary materials, we provide extensive qualitative comparison between the ground-truth spectrograms and the reconstructions produced by each of the evaluated methods. Additionally, the audio signals for these examples are provided are available in our Google Drive folder [3] .

4. CONCLUSION

We presented IS2 - a novel model for automatic synthesizer sound matching. IS2 introduces two novel contributions: (1) a differentiable synthesizer-proxy decoder that enables gradient-based optimization of the reproduced audio signals, and (2) the ITF technique that enables improved model predictions at inference time. These contributions lead to state-of-the-art results compared to existing methods across multiple datasets and metrics.

5. ACKNOWLEDGMENTS

This research was supported by the ISRAEL SCIENCE FOUNDATION (Grant No. 2243/20).

[3] `https://drive.google.com/drive/folders/1VFnE5fcEfbbmKdNT-NxMvXotSdz5mOQk?usp=sharing`

6. REFERENCES

[1] G. De Poli, "A tutorial on digital sound synthesis techniques," *Computer Music Journal*, vol. 7, no. 4, pp. 8–26, 1983.

[2] J. Shier, "The synthesizer programming problem: improving the usability of sound synthesizers," Ph.D. dissertation, 2021.

[3] 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.

[4] S. Heise, M. Hlatky, and J. Loviscach, "Automatic cloning of recorded sounds by software synthesizers," in *Audio Engineering Society Convention 127*. Audio Engineering Society, 2009.

[5] S. Luke, "Stochastic synthesizer patch exploration in edisyn," in *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*. Springer, 2019, pp. 188–200.

[6] M. Yee-King and M. Roth, "Synthbot: An unsupervised software synthesizer programmer," in *ICMC*, 2008.

[7] K. Tatar, M. Macret, and P. Pasquier, "Automatic synthesizer preset generation with presetgen," *Journal of New Music Research*, vol. 45, no. 2, pp. 124–144, 2016.

[8] 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, 2018.

[9] 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, 2019.

[10] 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)*, 2021, pp. 276–283.

[11] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International conference on machine learning*. PMLR, 2015, pp. 1530–1538.

[12] 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, 2019.

[13] "Dexed github repository," https://github.com/asb2m10/dexed, 2021.

[14] N. Masuda and D. Saito, "Synthesizer sound matching with differentiable dsp." in *ISMIR*, 2021, pp. 428–434.

[15] ——, "Improving semi-supervised differentiable synthesizer sound matching for practical applications," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[16] F. Caspe, A. McPherson, and M. Sandler, "Ddx7: Differentiable fm synthesis of musical instrument sounds," *arXiv preprint arXiv:2208.06169*, 2022.

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

[18] P. Chandna, M. Blaauw, J. Bonada, and E. Gómez, "Wgansing: A multi-voice singing voice synthesizer based on the wasserstein-gan," in *2019 27th European signal processing conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[19] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "Gansynth: Adversarial neural audio synthesis," *arXiv preprint arXiv:1902.08710*, 2019.

[20] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," *Advances in neural information processing systems*, vol. 32, 2019.

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

[22] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "Ddsp: Differentiable digital signal processing," *arXiv preprint arXiv:2001.04643*, 2020.

[23] O. Barkan and D. Tsiris, "Deep synthesizer parameter estimation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3887–3891.

[24] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] S. Ö. Arık, H. Jun, and G. Diamos, "Fast spectrogram inversion using multi-head convolutional neural networks," *IEEE Signal Processing Letters*, vol. 26, no. 1, pp. 94–98, 2018.

[28] R. C. Streijl, S. Winkler, and D. S. Hands, "Mean opinion score (mos) revisited: methods and applications, limitations and alternatives," *Multimedia Systems*, vol. 22, no. 2, pp. 213–227, 2016.