Parameter Optimization for Audio Synthesis Matching

Luke Nelson
Towson University
Department of Computer Science
lnelson2@students.towson.edu

***Abstract*—This paper presents a system for automatically matching synthesizer parameters to target audio samples using an evolutionary algorithm for black box optimization. The system uses a 37-parameter subtractive synthesizer with CMA-ES optimization and PANNs CNN used as a perceptual similarity metric. Given a target audio file, the system finds synthesizer parameters that recreate the sound with similarity depending on target complexity in about 10 to 15 minutes. The approach addresses the challenge of optimizing a non-differentiable objective function across a parameter space without using supervised learning, which can be applied to any synthesizer setup. Parameter validation using sensitivity analysis and phase cancellation confirms all 37 parameters work correctly, with nearly all also being detected by PANNs CNN 14. The system demonstrates that black box optimization with deep perceptual embeddings can automate sound design tasks that normally require hours of manual parameter tweaking.**

***Index Terms*—audio synthesis, parameter optimization, CMA-ES, evolutionary algorithms, PANNs, perceptual similarity, automatic sound design, sound matching, parameter matching**

I. Introduction

Synthesizers are electronic or virtual instruments that generate audio through parameter manipulation. A typical subtractive synthesizer has dozens of parameters controlling oscillators, filters, envelopes, and effects. Finding the right parameter settings to recreate a target sound is time consuming and requires expertise in both sound design and the specific synthesizer architecture. This paper presents a system that automates this process by treating it as an optimization problem.

The system synthesizes audio based on parameters, then measures perceptual similarity using CNN 14 from pretrained audio neural networks (PANNs) embeddings [1]. PANNs CNN 14 is a pre-trained neural network that produces 2048-dimensional audio embeddings, but there's no clear gradient path from the embedding similarity back to the synthesizer parameters so a differentiable approach wasn't used. The synthesis pipeline itself has discrete filter coefficient updates that happen every 1024 samples, which breaks differentiability. This means gradient based optimization methods cannot be used.

Instead, the system uses CMA-ES [2], a type of evolutionary algorithm designed for black-box optimization in continuous spaces; however some discrete parameters were used regardless and happened to be effective. CMA-ES maintains a population of candidate solutions and adapts a covariance matrix to learn parameter correlations during the search. The algorithm typically converges in 60,000 to 100,000 evaluations, which is run in batches on highly parallelized code on a GPU.

A. System Overview

The complete system has several components working together. First, the target audio is analyzed to detect pitch using the probabilistic-YIN (as in Yin-yang, not an acronym) algorithm [6] and estimate noise level through high-frequency energy ratios. The detected pitch fixes the note parameter, reducing the search space from 38 to 37 dimensions. If the target has low noise, the noise parameter is locked to zero during optimization to prevent wasting evaluations. This not only allows a 2 parameter reduction but also increase matching accuracy since this match is done with known, deterministic algorithms that are more effective rather than relying on PANNs embedding comparison to notice a difference. It would be prudent to do this with all parameters where possible to turn the problem into a multi-objective optimization problem, however in this paper only the pitch and noise level were treated in this way.

The synthesizer itself is a 37 parameter subtractive architecture implemented in PyTorch with GPU acceleration for batch mode optimization. It has two oscillators with configurable waveforms, a biquad filter with four types, two ADSR envelopes for amplitude and filter modulation, and effects including distortion and delay. All operations are vectorized to synthesize 256 parameter sets in parallel, which is necessary for CMA-ES to evaluate populations efficiently.

For similarity measurement, the system uses PANNs CNN 14 embeddings trained on the data set called AudioSet. The embeddings are 2048-dimensional that capture perceptual audio characteristics including timbre, texture, and envelope. Cosine distance in embedding space serves as the objective function that CMA-ES minimizes. Alternative metrics including MFCC, spectral distance, and temporal envelope matching were tested but PANNs consistently produced better perceptual quality in the matched sounds.

The optimization runs for up to 60,000 evaluations or until reaching 99% similarity, whichever comes first, which can be modified in the synthesizer interface. A restart strategy with geometrically increasing populations (20, 40, 80, and higher) as well as alternat restarts (40, 80, 40, 80) and changing step sizes is used to escape local optima. This simple approach was

empirically validated on the PANNs-based objective function, which creates a relatively smooth fitness landscape.

B. Contributions

The main contribution is a complete working system that solves the synthesizer parameter problem without supervised learning. While individual components like PANNs and CMA-ES exist, the integration of GPU batch synthesis, black-box optimization, perceptual metrics, and audio pre-analysis into a functional system represents the novelty. Specific technical features include:

1) GPU batch synthesis architecture enabling practical CMA-ES convergence in 10-15 minutes depending on target sound complexity
2) Use of PANNs embeddings as an objective function for synthesis parameter optimization
3) Noise optimization strategy that locks parameters based on algorithmic audio analysis
4) Parameter validation methodology combining sensitivity analysis and phase cancellation

The system demonstrates that black-box optimization is viable for synthesis matching when the objective function is not differentiable. The 10-15 minute optimization time depending on target sound complexity makes it practical for sound design workflows, reducing what would normally take hours of manual tweaking or expertise.

C. Applications

The system can be used for several purposes. The most direct application is preset generation from audio examples, where a user provides a target sound and receives synthesizer parameters that recreate it. This enables timbre transfer where sounds from one source can be recreated on the synthesizer. The system can also continue optimizing from existing presets, allowing iterative refinement toward a desired sound. For sound designers, this automates the exploration of large parameter spaces that would be impractical to search manually.

II. System Architecture

The system architecture follows a straightforward pipeline. Target audio enters the system, gets analyzed for pitch and noise, then optimization runs with the synthesizer and similarity metric in a loop until convergence. The main components are the synthesizer, similarity metrics, the CMA-ES optimizer, and audio pre-analysis.

A. 37-Parameter Subtractive Synthesizer

The synthesizer is a subtractive architecture with 37 continuous parameters normalized to [0,1]. The parameters control oscillators, filters, envelopes, and effects. The system uses the same core synthesis algorithms in two different operational modes: batch mode for optimization processes batches of 256 parameter sets in parallel, while streaming mode provides low-latency polyphonic playback for the GUI.

1) Oscillator Stage: The oscillator stage has two primary oscillators (OSC1 and OSC2), a sub-oscillator one octave below OSC1, and white noise. Each primary oscillator has controls for waveform (sine, saw, square, triangle), octave offset, fine detune, and level. Waveforms use PolyBLEP anti-aliasing [3] to prevent aliasing artifacts from discontinuous waveforms like square and saw waves.

Advanced oscillator features include hard sync where OSC1 resets OSC2's phase, ring modulation multiplying OSC1 and OSC2, pulse width modulation for square waves, and frequency modulation. The oscillators output at the sample rate of 44.1kHz.

2) Filter Stage: The filter is a biquad IIR design with four types: low-pass, high-pass, band-pass, and notch. The transfer function is:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{1}$$

Filter parameters include cutoff frequency, resonance, envelope amount, drive for soft-clipping saturation, key tracking to make cutoff follow pitch, and LFO modulation. The filter state is updated every 1024 samples during synthesis, which makes the pipeline non-differentiable since these updates are discrete rather than continuous.

3) Envelope Stage: Two independent ADSR envelopes control amplitude and filter cutoff. The envelopes use exponential curves rather than linear because they sound more musical:

$$y(t) = y_0 \cdot e^{-t/\tau} \tag{2}$$

The amplitude envelope shapes overall volume over time. The filter envelope modulates the filter cutoff, creating timbral evolution. Each envelope has the typical four parameters for attack, decay, sustain level, and release time (ADSR)

4) Effects and Other Parameters: Additional parameters include distortion with soft clipping, delay with time/feedback-/mix controls, an LFO with rate and filter modulation amount, noise level, unison detune, and note hold time. The note hold time parameter controls synthesis duration but turns out to be problematic because PANNs embeddings lose temporal length information, which is discussed in the results section.

5) Parameter Scaling: Frequency and time parameters use logarithmic scaling to match human perception:

$$f_{\text{Hz}} = f_{\min} \cdot \left(\frac{f_{\max}}{f_{\min}} \right)^{p} \tag{3}$$

where $p \in [0, 1]$ is the normalized parameter value. For example, the filter cutoff maps $p = 0.0$ to 20Hz, $p = 0.5$ to approximately 632Hz, and $p = 1.0$ to 20kHz. This makes the optimization landscape more uniform because equal parameter steps correspond to equal perceptual steps.

6) GPU Batch Synthesis: All operations are vectorized in PyTorch to process multiple parameter sets simultaneously. The synthesis happens in batches of 256, where each parameter set generates 0.5 seconds of audio (22,050 samples at

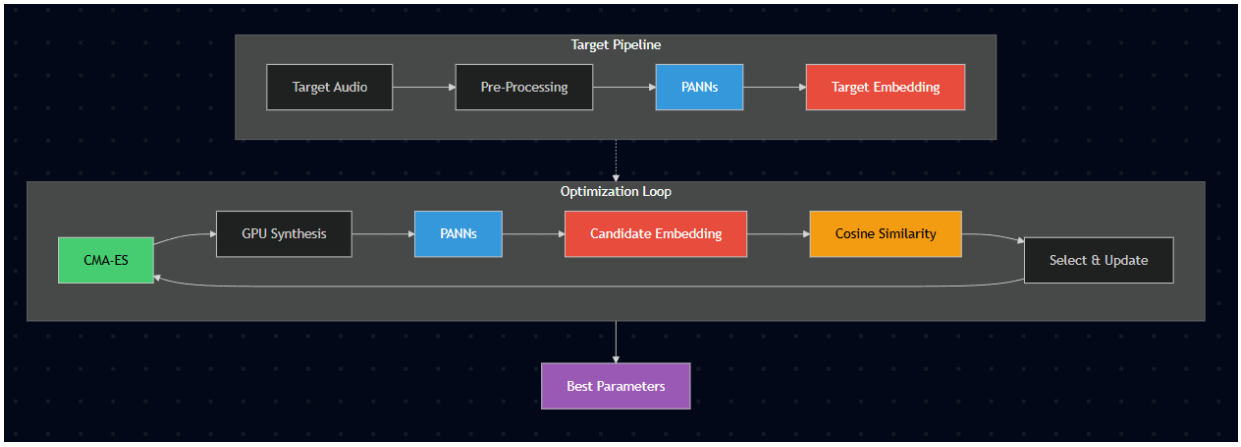

Fig. 1. System architecture showing the complete optimization pipeline from target audio to matched synthesizer parameters. Target audio is analyzed for pitch and noise characteristics, then CMA-ES iteratively generates parameter sets that are batch-synthesized on GPU and compared to the target using PANNs embeddings until convergence.

44.1kHz). GPU acceleration makes optimization practical in a reasonable time.

Batch synthesis is stateless, meaning each call generates a fixed duration of audio without maintaining state between invocations. The batch size of 256 was chosen to maximize GPU utilization while fitting in memory. The fixed 0.5 second duration provides enough audio for PANNs embeddings to capture timbral characteristics while keeping memory usage bounded during optimization.

7) Pipeline Implementation Pattern: The synthesizer implements a serial function composition where each stage transforms the audio tensor sequentially. The complete pipeline flow is:

Listing 1. Batch synthesis pipeline implementation

```
def synthesize_batch(self, param_array, duration=
    None, note=60, velocity=0.8):
    params = self._parse_parameters(param_array,
        note)
    audio = self._generate_oscillators(params,
        batch_size, num_samples)
    if self.use_filter:
        audio = self._apply_filter(audio, params,
            batch_size, num_samples)
    audio = self._apply_amplitude_envelope(audio,
        params, batch_size, num_samples, velocity)
    if self.use_effects:
        audio = self._apply_effects(audio, params)
    audio = self._normalize(audio)
    return audio  # Shape: (batch_size, num_samples)
```

This architecture is fully vectorized with no Python loops. Each transformation operates on batch_size, num_samples tensors simultaneously, enabling efficient GPU parallelization across the entire batch.

Using the filter stage with additive modulation, here is a concrete example one transformation. The filter stage demonstrates the additive modulation architecture used throughout the synthesizer. All modulation sources sum in logarithmic (octaves in this case) space before being applied to the base cutoff frequency:

```
# Sum all modulations in octaves
total_mod_octaves = 0
total_mod_octaves += filter_env * ENV_DEPTH_OCTAVES
    * env_amount
total_mod_octaves += lfo_bipolar * LFO_DEPTH_OCTAVES
     * lfo_amount
total_mod_octaves += (midi_note - 60) / 12 *
    filter_keytrack

# Apply modulation in log space: f_new = f_base * 2^
    mod
cutoff_modulated = base_cutoff * (2.0 **
    total_mod_octaves)
```

This additive approach in logarithmic space prevents multiplication order dependencies and matches the behavior of analog synthesizers where modulation sources sum at the modulation input. This is a typical implementation of filters for this type of synthesizer.

For batch mode optimization, oscillators and envelopes run on GPU using PyTorch operations while the biquad filter uses SciPy's C-optimized `lfilter()` function on CPU. This hybrid approach exists because profiling revealed that the overhead of transferring small coefficient arrays to GPU and launching kernels makes CPU filtering 10-100 times faster for this use case. The filter coefficients are updated every 1024 samples to balance computational efficiency with time-varying modulation accuracy. Audio tensors are transferred between GPU and CPU only for the filter stage, with transfers pipelined to minimize overhead.

8) Batch vs. Streaming Synthesis Modes: The system supports two distinct synthesis modes optimized for different use cases:

Batch Mode (Optimization): Processes 256 parameter sets in parallel with stateless synthesis using GPU acceleration. Each call generates a fixed 0.5 second duration of audio. The GPU/CPU hybrid architecture described above enables high throughput: oscillators and envelopes run on GPU while filters run on CPU. This mode is used by CMA-ES during optimization, where throughput is critical and the same pa-

rameters never need to be synthesized multiple times. Batch synthesis with GPU acceleration enables practical throughput for optimization workflows.

Streaming Mode (GUI): Uses stateful voice pooling for polyphonic real-time playback running entirely on CPU. Each voice maintains persistent oscillator phase, filter state variables, and envelope stage tracking across multiple audio callback invocations. The voice pool supports 6 simultaneous voices with voice stealing when the pool is exhausted (oldest voice is preempted). This mode is used by the GUI for interactive keyboard playback, where latency matters more than batch throughput. CPU-only execution avoids GPU context switching latency and allows the optimization process to run concurrently without resource contention.

Both modes use identical algorithms and parameter mappings, ensuring that sounds synthesized during optimization match those played back in the GUI. The core difference is execution context: batch mode prioritizes parallel throughput via GPU acceleration, while streaming mode prioritizes low-latency interactive playback via CPU execution and stateful voices. Without these different modes, using the synthesizer would be impractical.

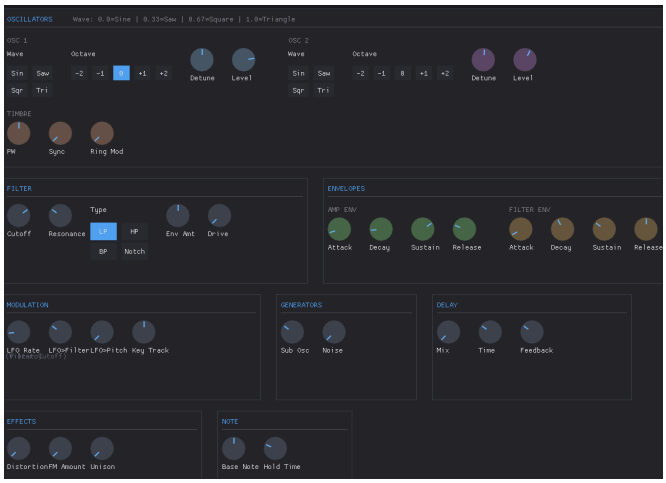

Fig. 2. Interactive synthesizer GUI built with DearPyGui showing parameter controls and keyboard input. The interface provides visual feedback for all controllable parameters and algorithm settings.

B. Similarity Metrics

Multiple similarity metrics were tested during development. The system needed a metric that captures perceptual similarity rather than raw waveform differences, since two sounds can have very different waveforms but sound nearly identical to the human ear. An extreme example of this is noise, where a litany of frequencies have the same characteristic sound. Alternatively, two waveforms may appear to be similar to PANNs, but not sound similar to the average listener.

1) PANNs (Primary Metric): The primary metric uses CNN14, a model from the PANNs (Pretrained Audio Neural Networks) family [1]. The network was trained on AudioSet, which contains over 2 million audio clips across 527 sound categories with human annotated labels. It produces 2048-dimensional embeddings that capture audio characteristics.

The similarity is found using cosine similarity:

$$\text{similarity}(\mathbf{e}_1, \mathbf{e}_2) = \frac{\mathbf{e}_1 \cdot \mathbf{e}_2}{\|\mathbf{e}_1\|\|\mathbf{e}_2\|} \tag{4}$$

where $\mathbf{e}_1$ and $\mathbf{e}_2$ are the 2048-dimensional PANNs embeddings of the target and synthesized audio respectively.

CMA-ES tries to minimize the distance. PANNs uses log-mel spectrogram features as input, which are magnitude based representations that discard phase information. This makes the embeddings phase insensitive, meaning identical waveforms are not determined to be different from having an altered starting point. Additionally, the CNN architecture with global pooling provides robustness to temporal shifts in the audio. The embeddings are also perceptually aligned since they were trained on human labeled audio, so features that humans consider important for sound classification are emphasized in the representation.

2) Alternative Metrics Tested: Several other metrics were explored during development:

MFCC (Mel-Frequency Cepstral Coefficients) uses 13 coefficients that capture timbral characteristics on a mel scale. The problem is that MFCCs were designed for speech recognition and are less effective for matching musical synthesis timbres. They do however also use log-mel spectrogram to take into account human hearing similar to PANNs, which made it a promising candidate that unfortunately was not effective.

Spectral Distance compares log-magnitude Short Term Fourier Transorm (STFT) representations in the frequency domain. This is phase insensitive but gives equal weight to all frequencies, which does not match human perception where some frequency ranges matter more than others.

Temporal matches the Root Mean Square (RMS) envelope over time, focusing on volume. This completely misses spectral and timbral content, so two sounds can have similar envelopes but sound completely different.

MSSL (Multi-Scale Spectral Loss) [5] was originally designed as a training loss for neural vocoders but was adapted as a perceptual similarity metric. The method operates in the frequency domain using STFT magnitude spectrograms at multiple window sizes. The implementation uses window sizes of [128, 512, 2048] samples to capture transients, general timbre, and harmonic content respectively. The similarity is computed as the L1 distance on magnitude spectrograms averaged across scales. Preliminary tests showed MSSL does not work as well as PANNs for this application. Neither MMSL nor the log-scaled MSSL version were effective.

PANNs consistently produces the best perceptual quality in matched sounds. The main advantage is that it captures timbre, texture, and envelope characteristics together in a learned representation rather than attempted hand crafted from traditional measures.

C. CMA-ES Optimization

CMA-ES (Covariance Matrix Adaptation Evolution Strategy) [2] is the optimization algorithm used. The choice of

CMA-ES is central to why this system works, so understanding the alternatives and why they were rejected is important.

1) The Non-Differentiable Problem: The objective function cannot be differentiated with respect to synthesizer parameters. There are several reasons for this:

1) PANNs embeddings come from a black-box neural network with no gradient path to synthesizer parameters
2) Filter coefficients update discretely every 1024 samples rather than continuously
3) The synthesis pipeline has operations that break gradient flow

This eliminates all gradient-based optimization methods like SGD, Adam, or any other approach that requires differentiating the similarity function with respect to the synthesizer parameters.

2) Alternative Approaches Considered: **Differentiable DSP** in the style of differential Digital Signal Processor (DSP) [4] could theoretically make the entire synthesis pipeline differentiable. This would require replacing discrete filter updates with continuous approximations, implementing differentiable oscillators, and ensuring all operations maintain gradient flow. The resulting system would be architecture-specific and wouldn't generalize to other synthesizers. This is discussed more in the future work section.

Reinforcement Learning treats optimization as a sequential decision problem where an agent learns a policy for setting parameters. This adds overhead of policy network training and treats the problem as more complex than it actually is. CMA-ES is more direct for continuous parameter optimization.

Grid Search or **Random Search** does not scale to 37 dimensions. Grid search would need millions of evaluations even with coarse spacing, and random search does not exploit the structure of the parameter space.

Genetic Algorithms could work but CMA-ES is essentially an evolved genetic algorithm with better convergence properties for continuous optimization. CMA-ES learns parameter correlations through the covariance matrix, which generic genetic algorithms do not do.

3) Why CMA-ES is Ideal: CMA-ES has several properties that make it the right choice:

- **Derivative-free**: Works with black-box objective functions
- **High-dimensional**: Handles 37 continuous parameters effectively, and can realistic work for up to 100
- **Learns correlations**: The covariance matrix adapts to parameter dependencies (for example if cutoff and resonance interact)
- **Self-adaptive**: Step size (σ) adjusts automatically during search
- **Rotation-invariant**: The search does not depend on coordinate system choice
- **Proven track record**: State-of-the-art for black-box optimization in robotics and engineering
- **Sample-efficient for derivative-free optimization**: Typically converges in 60,000 to 100,000 evaluations for this 37 dimension problem, which is efficient compared to other black-box methods though gradient-based approaches would require fewer evaluations if gradients were available

4) CMA-ES Mathematical Details: The CMA-ES algorithm works by sampling parameter sets from a multivariate normal distribution and updating the distribution parameters based on which samples worked best. The core equations are as follows:

Sampling: Each candidate parameter set is generated by:

$$\mathbf{x}_i = \mathbf{m} + \sigma \cdot \mathbf{L} \cdot \mathbf{z}_i \tag{5}$$

where $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is random noise, $\mathbf{L}$ is the Cholesky decomposition of covariance matrix $\mathbf{C}$, σ controls step size, and $\mathbf{m}$ is the current search center.

Mean Update: After evaluating all candidates, the mean moves toward the best ones:

$$\mathbf{m}_{\text{new}} = \sum_{i=1}^{\mu} w_i \cdot \mathbf{x}_i \tag{6}$$

where μ is the number of parents (set to half the parents in this implementation) and the weights are:

$$w_i = \frac{\log(\mu + 0.5) - \log(i)}{\sum_{j=1}^{\mu} [\log(\mu + 0.5) - \log(j)]} \tag{7}$$

This weighting assigns higher importance to better ranked solutions.

Covariance Update: The covariance matrix learns which parameter directions tend to improve fitness. It combines the evolution path (successful direction over multiple generations) with the empirical covariance of the selected parent steps. This adaptation is what makes CMA-ES so effective: it learns parameter correlations rather than treating dimensions independently.

5) Algorithm Configuration: The default CMA-ES configuration settled on is:

- Population size: 20 up to 80 with restarts. This is larger than the CMA-ES often suggested default of 14 for about 37 dimensions which was found to work better
- Max evaluations: 100,000
- Target similarity: 99%
- Initial step size (σ): 0.25
- Stop criteria: Reach 99% similarity OR 100,000 evaluations OR manual stop

6) BIPOP Restart Strategy: The system uses BIPOP (Bi-Population) restart strategy, which alternates between large population exploration and smaller population exploitation regimes. Each restart uses a fresh covariance matrix and mean, preventing premature convergence.

The pattern alternates between:

- **Large regimes** (even restarts): Population doubles and σ doubles. Explores broadly with larger step sizes to escape local optima and search distant regions.

- **Small regimes** (odd restarts): Population resets to base size (20) and σ resets to half the initial value. Refines search with smaller step sizes for local exploitation.

Example sequence for 6 restarts:

1) Restart 0: pop=20, $\sigma = 0.25$ (initial)
2) Restart 1: pop=40, $\sigma = 0.50$ (explore)
3) Restart 2: pop=20, $\sigma = 0.125$ (exploit)
4) Restart 3: pop=40, $\sigma = 0.25$ (explore)
5) Restart 4: pop=20, $\sigma = 0.125$ (exploit)
6) Restart 5: pop=40, $\sigma = 0.25$ (explore)

Empirical testing on audio synthesis targets showed BIPOP converged fastest among tested strategies. Compared to simpler approaches like increasing population with decreasing step size (inverted IPOP) or standard IPOP, BIPOP required 6x fewer evaluations (21,900 vs 139,160) to reach 99% similarity while achieving about the same final quality. The alternating regime structure balances global exploration and local refinement more effectively than monotonic population or step size progressions. BIPOP is particularly effective for this problem because the PANNs objective has multiple local optima despite being relatively smooth.

D. Audio Pre-Analysis and Noise-Aware Optimization

Before optimization starts, the target audio is analyzed to extract information that can constrain the search space.

1) Pitch Detection: The pYIN algorithm (probabilistic YIN) detects the fundamental pitch in the target audio. The algorithm searches for frequencies between 50Hz and 2000Hz and returns the MIDI note number, frequency in Hz, and confidence score. The detected pitch is used to fix the note parameter during synthesis, which reduces the degrees of freedom by one and prevents octave errors where the optimizer might match the timbre but at the wrong pitch.

2) Noise Estimation: Noise level is estimated by computing the ratio of high-frequency energy (above 8kHz) to low-frequency energy. The method is:

$$\text{noise\_ratio} = \frac{\text{RMS}_{>8kHz}}{\text{RMS}_{<8kHz}} \tag{8}$$

For clean sounds like pure tones or instruments, most energy is in lower frequencies and the ratio is small. For noisy sounds, high-frequency energy is significant and the ratio is larger. If the estimated noise level is below 0.1, the noise_level parameter is locked to zero during optimization. This is implemented in the objective function and prevents the optimizer from wasting evaluations varying a parameter that does not affect the match. The value of 0.1 may not be optimal.

The noise strategy is used because PANNs embeddings are somewhat insensitive to noise level. Without locking, the optimizer would randomly vary noise on clean targets without improving similarity.

III. Parameter Validation

With 37 parameters, it is easy to have a broken implementation. Broken or ineffective parameters waste optimization time by giving the algorithm dimensions to search that don't improve the objective. Two complementary validation approaches were used to verify all parameters work correctly, and that those changes can actually be detected by the chosen similariy metric, PANNs.

A. PANNs Sensitivity Sweep

The sensitivity sweep measures how much each parameter affects PANNs similarity when varied. The method uses multiple reference points to avoid bias from any single parameter setting.

1) Method: For each parameter, four pin values are chosen at 0.2, 0.4, 0.6, and 0.8 in the normalized parameter range. At each pin value:

1) Generate 50 random parameter contexts (random settings for all other parameters)
2) Set the test parameter to the pin value as the reference
3) Sweep the test parameter through [0.0, 0.05, 0.10, ..., 1.0]
4) For each sweep value, synthesize audio and compute PANNs distance to the reference
5) Average the distances across the 50 contexts

This creates V-shaped curves centered at each pin value. The sensitivity is quantified as the average range of these V-curves across all four pin values. High sensitivity means the parameter causes large PANNs distance changes when varied.

2) Sensitivity Ranges: Parameters show varying sensitivity levels. While these thresholds are somewhat arbitrary, they provide useful categorization for understanding which parameters matter most to the similarity metric:

- HIGH (greater than 0.3): Strong perceptual effect
- MEDIUM (0.1-0.3): Moderate effect
- LOW (0.05-0.1): Subtle effect
- VERY LOW (less than 0.05): Minimal PANNs response

The distribution found was 12 HIGH, 15 MEDIUM, 9 LOW, and 1 VERY LOW parameter.

3) Statistical Validation: The methodology itself was validated using identical parameter pairs. Since osc1_level and osc2_level have identical roles in the synthesis architecture, they should have identical sensitivity values. The same applies to osc1_waveform and osc2_waveform, the four amp envelope parameters versus the four filter envelope parameters, and so on.

With 50 random contexts per pin value, all identical pairs converged to within 2% of each other. This confirms the measurement is statistically valid and not dominated by noise. This method was absolutely critical to debugging and confirming that the pipeline works as intended.

B. Phase Cancellation Test

The phase cancellation test provides mathematical proof that parameters change the waveform. This is independent of PANNs and verifies the synthesizer implementation itself.

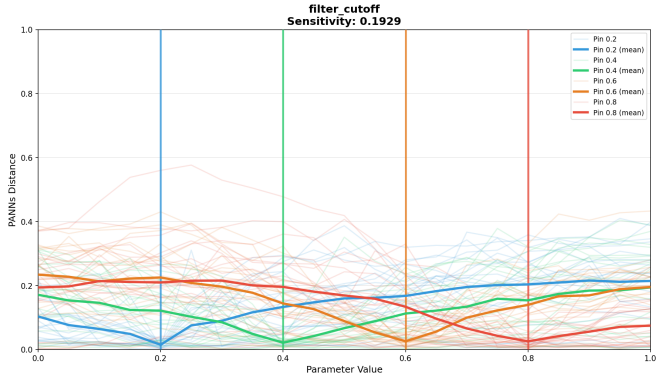

Fig. 3. PANNs sensitivity sweep for filter_cutoff parameter. Four V-shaped curves show PANNs distance vs parameter value at pin values 0.2, 0.4, 0.6, and 0.8. Each curve is averaged over 50 random parameter contexts. The filter_cutoff parameter shows HIGH sensitivity (>0.3 average range), meaning PANNs embeddings are strongly affected by filter cutoff changes. This demonstrates the validation methodology for assessing perceptual parameter importance.

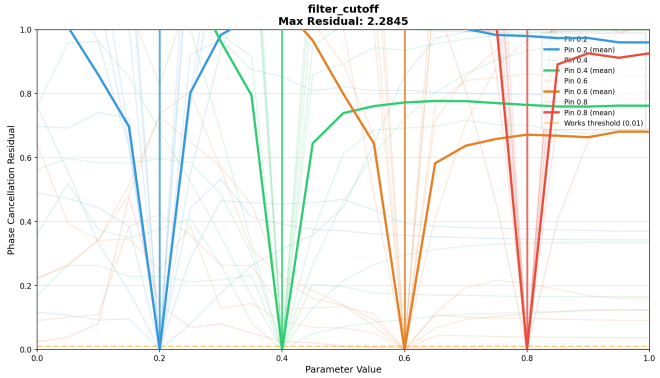

Fig. 4. Phase cancellation test for filter_cutoff parameter. Four V-shaped curves show residual ratio (non-cancelled energy / original energy) vs parameter value at pin values 0.2, 0.4, 0.6, and 0.8. Each curve is averaged over 50 random contexts. High residual values (>1%) prove the parameter affects the waveform. The filter_cutoff parameter shows strong waveform changes confirming it is functionally implemented in the synthesizer.

1) Method: For each parameter at each pin value:

1) Generate reference audio with the parameter at pin value
2) Generate test audio with the parameter at a sweep value
3) Invert the test audio phase: $x_{\text{inv}} = -x_{\text{test}}$
4) Sum the signals: $r = x_{\text{ref}} + x_{\text{inv}}$
5) Compute residual ratio:

$$\text{residual} = \frac{\text{RMS}(r)}{\text{RMS}(x_{\text{ref}})} \tag{9}$$

The residual represents the ratio of the non-cancelled energy to the original signal energy. A residual near 0% means the signals cancel completely (parameter has no effect), while a residual near 100% means the signals are completely different (parameter has strong effect). Values above 1% indicate the parameter measurably affects the waveform. This technique is possible due to sound waves ability to completely destructively interfere.

If the parameter does not affect the waveform, the test audio equals the reference and the residual is near zero because they cancel. If the parameter changes the waveform, the residual is large because they don't cancel.

2) Results: A threshold of 1% residual was used to determine if a parameter works. Nearly all timbre parameters showed residuals well above 1%, confirming they change the waveform. The note_hold_time parameter was excluded from this test because it changes duration rather than timbre, so phase cancellation isn't meaningful.

The phase test confirms there are no implementation bugs. Combined with the PANNs sensitivity sweep, this shows all parameters work correctly and most are detectable by the similarity metric.

C. Key Findings

The validation revealed several important results:

- All 36 timbre parameters are functional (phase cancellation proof)
- PANNs detects 35 of 36 timbre parameters (only note_hold_time is blind)
- The note_hold_time blindness is expected because PANNs embeddings lose temporal length information
- Parameter sensitivity varies widely, with filter cutoff and oscillator waveforms having the strongest effect
- No implementation bugs were found in the synthesizer

The one parameter PANNs cannot detect is note_hold_time, which controls synthesis duration. This is a known limitation of using PANNs embeddings for this task and is discussed more in the results section as a challenge for matching sounds with specific duration requirements.

See Appendix for complete parameter classification table with sensitivity values and phase cancellation results for all 37 parameters (Table I).

IV. Experiments and Results

The system was tested on various monophonic audio targets including bell sounds, bass tones, and synthesized effects. Both quantitative metrics and human perceptual evaluation were used to assess performance.

A. Experimental Setup

GPU acceleration for synthesis and PANNs inference combined with CPU-based CMA-ES updates enables practical optimization. The optimization configuration was 60,000 max evaluations (budget limit), 99% target similarity, and 3 restarts. CMA-ES typically converges in 20-40k evaluations for simpler targets like bells and pure tones, with more complex targets requiring up to the full 60,000 budget. All test sounds were monophonic single notes.

Sample duration was fixed at 0.5 seconds to match the synthesizer's envelope characteristics. This duration was chosen because it's long enough to capture attack, decay, and sustain portions of typical sounds, though it misses longer release tails.

Multiple similarity metrics were tested including PANNs, MFCC, Spectral, Temporal, MSSL (linear), and MSSL_log.

PANNs was used for most experiments as it consistently produced the best perceptual quality.

B. Matching Performance

1) Quantitative Results: The optimizer typically converged in 60,000 to 100,000 evaluations depending on target complexity, which is more a qualitative statement about how ”un-synthesized” a sounds timbre is. However, the PANNs similarity percentages reported during optimization do not reliably correlate with perceptual match quality which is a very critical message in these experiments. Listening tests revealed that similarity scores below 95% consistently failed to produce perceptually similar results. Even scores in the 95-99% range showed wide variation in perceived quality: some sounded nearly identical while others had obvious timbral differences. The percentage alone cannot determine match quality; human listening is the only reliable evaluation method. Convergence time was typically 10-15 minutes depending on target sound complexity.

These numbers represent what the PANNs metric reports, which doesn't always align with human perception. The metric provides a rough guide but the true test is how the matched sounds actually sound to human ears.

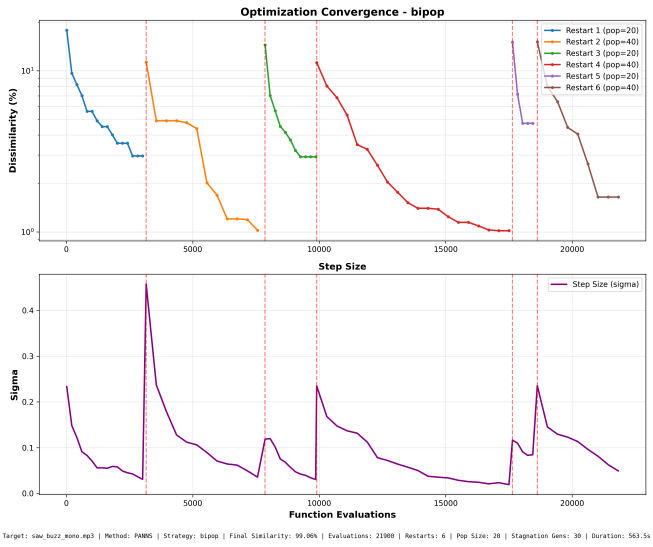

Fig. 5. BIPOP restart strategy convergence plot showing optimization progress for a test target. The algorithm converged to 99.06% similarity in 21,900 evaluations, achieving 6x faster convergence compared to alternative restart methods. Vertical lines indicate restart points where population size and step size are adjusted.

2) Qualitative Results: The main insight from listening tests is that metrics do not always match perception. Some matched sounds measure at 95% but don't sound similar at all.

Listening evaluation revealed that the PANNs metric's limitations prevent it from serving as a reliable quality indicator. Most optimization runs failed to achieve perceptually acceptable matches, with the majority of results sounding noticeably different from target audio despite reporting similarity scores. Target sounds with simpler harmonic structures occasionally produced acceptable matches when reaching similarity scores above 95%, but complex timbres and time-varying characteristics remained challenging regardless of reported similarity percentage.

The conclusion from listening evaluation is that PANNs similarity percentages do not predict perceptual match quality. Even at 95-99% similarity, match quality varies unpredictably. Human auditory evaluation is essential and the metric alone cannot determine whether a match sounds similar.

C. GPU Acceleration

GPU acceleration provides speedup over CPU synthesis, making the optimization practical. Without GPU, the maximum budget of 60,000 evaluations would take estimated hours which is impractical for interactive use. With GPU batching 256 parameter sets in parallel, completes in 10-15 minutes depending on target sound complexity, making the approach viable for sound design workflows where this optimization time is acceptable for generating presets.

D. Challenges and Limitations

1) Duration and Note Hold Time Problem: The main challenge is that PANNs embeddings lose temporal length information. The note_hold_time parameter is essentially ignored by the optimizer because changing duration doesn't affect PANNs similarity meaningfully. This makes it difficult to match sounds with specific duration requirements, like release tails, silence after the note, or long echoes (effects from the sustain and release portions of ADSR).

The current approach uses fixed 0.5 second synthesis duration, but the optimal hold time for playback is unclear. When the matched sound is played in the GUI, how long should the note be held? Should there be silence after the release? These questions don't have clear answers because duration wasn't part of the optimization objective.

2) Other Limitations: The system is designed for monophonic targets only. Polyphonic audio with multiple simultaneous notes doesn't work because the synthesizer generates single notes and the matching would be ambiguous about which note to match. This was mostly mitigated by the pYIN fundamental frequency detection, but it is not foolproof.

Some sounds are outside the synthesizer's capability. If a target requires specific waveforms or effects that the 37-parameter architecture cannot produce, the match quality will be limited by the synthesizer's expressiveness rather than the optimizer's effectiveness.

Heavy reverb or other effects in the target audio can confuse the similarity metric. PANNs was trained on diverse audio but not specifically on heavily processed synthetic sounds, so the embeddings may not capture these characteristics well.

Optimization time of 10-15 minutes depending on target sound complexity is practical for offline preset generation but not real-time. The system cannot be used for live sound matching during performance.

3) Failure Cases: The system fails on polyphonic targets with multiple simultaneous notes. It also struggles with sounds outside the synthesizer's frequency range like deep sub-bass below 50Hz. Highly noisy or heavily distorted sources where the fundamental pitch cannot be detected also cause problems because the pre-analysis would fail.

V. Discussion

A. Key Findings

The main finding is that the complete end-to-end system works for its intended purpose of matching synthesizer parameters to target audio. The system typically completes in 10-15 minutes depending on target complexity for monophonic targets, though similarity percentages do not reliably indicate perceptual match quality. This timing makes it practical for sound design workflows.

GPU acceleration is essential for making CMA-ES practical. Without GPU batch synthesis, the evaluations needed for convergence would take hours instead of minutes. The ability to synthesize 256 parameter sets in parallel directly enables the optimization approach.

PANNs embeddings are effective as an objective function for this task. The phase-invariant and perceptually-aligned characteristics make it better than traditional metrics like MFCC or spectral distance. The learned representation captures timbral aspects that other features miss.

Parameter validation confirmed all 37 parameters work correctly. The sensitivity analysis shows PANNs detects 35 of 36 timbre parameters, and phase cancellation proves all parameters affect the waveform. This validation was important because broken parameters would waste optimization time.

Noise-aware optimization improves efficiency. Locking parameters based on audio analysis prevents wasting evaluations on irrelevant dimensions, but more importantly one that PANNs does not detect well.

B. Design Trade-offs

The choice between black-box and gradient-based optimization is fundamental. CMA-ES is slower per iteration than gradient descent would be, but it works with non-differentiable objectives. The system is also generalizable because it could be adapted to different synthesizers without rewriting the entire pipeline to maintain differentiability. More parameters can be added and taken away and it would still be able to run without many changes or go through a training phase.

PANNs versus other metrics is another trade-off. PANNs produces better perceptual quality but is blind to duration. A multi-objective approach combining PANNs for timbre with temporal metrics for duration could address this, though it would add complexity to the optimization. Many combinations were tried and they did not work well.

Fewer parameters would be easier to optimize and less expressive, but not be able to capture a richer set of sounds. The 37 parameters seem to be a reasonable balance for subtractive synthesis.

C. Broader Impact

The system demonstrates that audio synthesis parameter optimization is practical with current hardware and algorithms. Sound designers can use it to automate tasks that would normally take hours of manual tweaking.

The time savings are substantial. Where a sound designer might spend an hour or more manually adjusting 37 parameters to match a target, the system does it in 10-15 minutes depending on target complexity. This enables exploration of larger parameter spaces and more experimentation.

The approach also enables new creative workflows. Users can provide audio examples from any source and get back synthesizer presets that recreate those sounds. This enables timbre transfer and inspiration from external audio rather than just preset libraries.

Results are automated and reproducible. The same target audio and optimizer settings will produce similar results each time, unlike manual parameter tweaking which depends on the sound designer's skill and experience.

VI. Conclusion

This paper presented a complete system for automatically matching synthesizer parameters to target audio using black-box optimization. The system addresses the challenge of optimizing a non-differentiable objective across 37 continuous parameters by combining CMA-ES evolutionary optimization with PANNs-based perceptual similarity metrics. GPU batch synthesis makes the approach practical, enabling convergence in 10-15 minutes depending on target sound complexity. Parameter validation using both phase cancellation and PANNs sensitivity analysis confirmed all 36 timbre parameters function correctly.

While the system demonstrates that automated parameter matching is feasible, an important limitation must be emphasized: PANNs similarity percentages do not directly translate to perceptual match quality as judged by human listeners. During testing, multiple sounds measuring 95-98% similarity exhibited wide variation in perceived match quality. Some sounded nearly identical to the target while others were clearly different to most listeners. Sounds below 95% similarity rarely sounded like accurate matches despite the seemingly high percentage. This discrepancy occurs because the PANNs CNN14 embeddings, while capturing many important aspects like general timbre, texture, and envelope characteristics, do not fully represent every perceptual dimension that matters for human listening. The metric provides sufficient guidance to make the optimization system work, but human listening remains essential for evaluating true match quality. This is why the results section relies on qualitative listening assessments rather than tables of quantitative metrics - the numbers alone cannot determine whether one match is better than another.

Additional limitations include PANNs embeddings not understading note duration, losing temporal length information which makes the note_hold_time parameter difficult to optimize. The system is designed for monophonic targets only and requires 10-15 minutes per optimization depending on

target sound complexity, making it unsuitable for real-time applications.

Future work should prioritize human listener evaluation studies. Gathering perceptual assessments from multiple listeners would build ground truth data for match quality beyond what automated metrics can provide. The human evaluation would enable meaningful comparison of different optimization strategies and similarity metrics. This could also allow interactive refinement where listeners guide the optimization based on which perceptual aspects need improvement.

Despite the metric limitations, this system demonstrates that black-box optimization with perceptual embeddings can solve high-dimensional synthesis matching problems in practical timeframes. It provides a foundation for future work that combines automated optimization with human evaluation to achieve both efficiency and perceptual accuracy.

REFERENCES

[1] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880-2894, 2020.

[2] N. Hansen and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159-195, 2001.

[3] V. Välimäki and A. Huovilainen, "Antialiasing Oscillators in Subtractive Synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 116-125, 2007.

[4] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," *International Conference on Learning Representations (ICLR)*, 2020.

[5] W. Jang, D. Lim, J. Yoon, B. Kim, and J. Kim, "UnivNet: A Neural Vocoder with Multi-Resolution Spectrogram Discriminators for High-Fidelity Waveform Generation," *Interspeech*, 2021.

[6] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917-1930, 2002.

APPENDIX

This appendix presents validation results for all 37 synthesizer parameters. The complete grids show both PANNs sensitivity analysis and phase cancellation tests across the entire parameter space.

A. PANNs Sensitivity - All Parameters

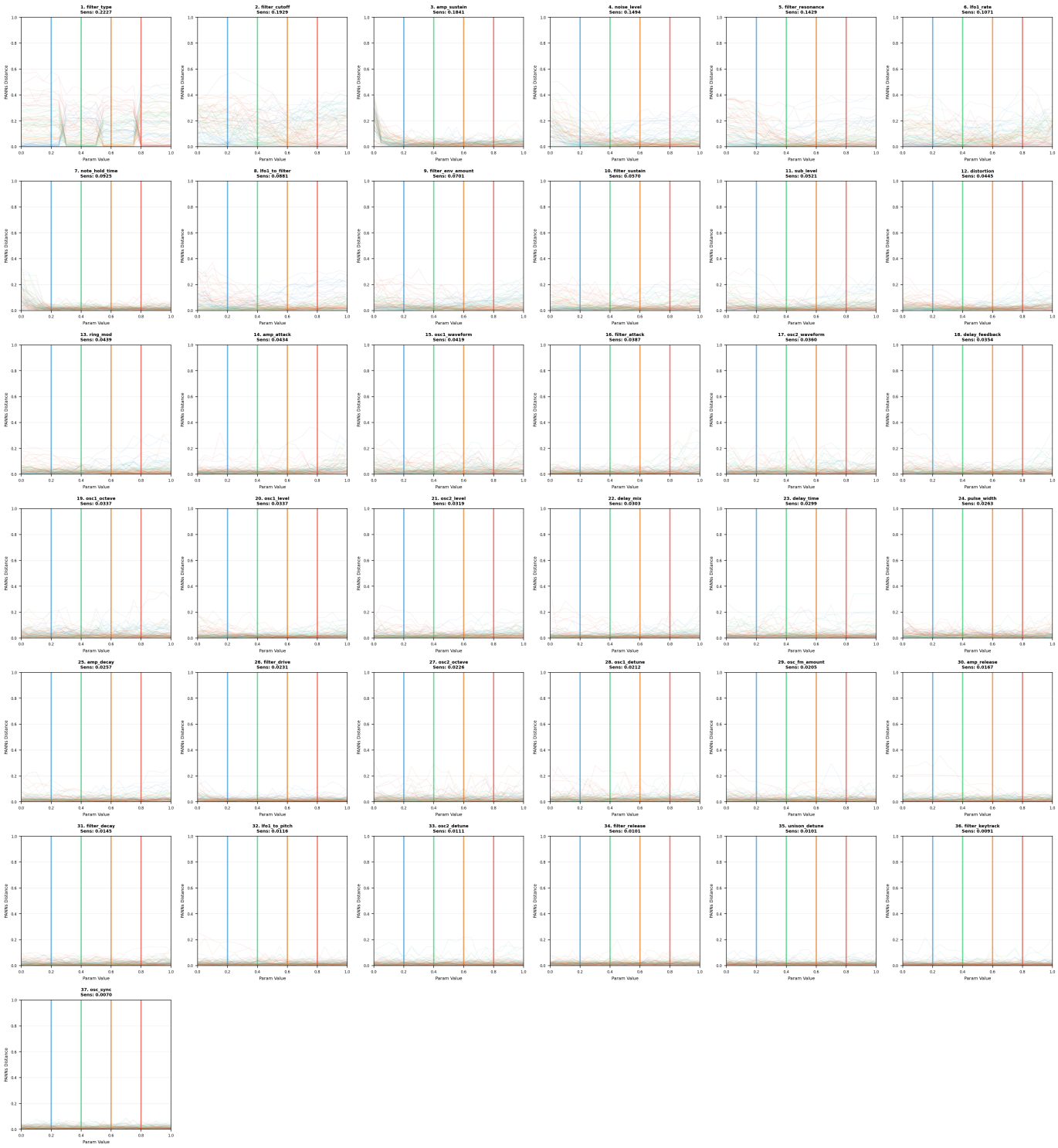

Fig. 6. PANNs sensitivity validation for all 37 parameters. Each subplot shows four V-curves at pin values 0.2, 0.4, 0.6, 0.8. Sensitivity ranges: HIGH (greater than 0.3), MEDIUM (0.1-0.3), LOW (0.05-0.1), VERY LOW (less than 0.05). Only note_hold_time shows very low sensitivity.

B. Phase Cancellation - All Parameters

Fig. 7. Phase cancellation validation for all 36 timbre parameters (note_hold_time excluded). Each subplot shows four residual curves at pin values 0.2, 0.4, 0.6, 0.8. Threshold at 1% indicates functional parameters. All 36 parameters show residuals well above threshold, confirming correct implementation.

C. Parameter Classification Reference

Table I provides sensitivity and validation data for all 37 synthesizer parameters. PANNs Sensitivity values indicate perceptual importance, with higher values showing stronger parameter effects. Phase Residual values quantify waveform impact (values greater than 1% indicate functional parameters).

TABLE I
COMPLETE PARAMETER CLASSIFICATION BY PANNS SENSITIVITY AND PHASE CANCELLATION VALIDATION. PARAMETERS ARE ORDERED BY CATEGORY AND PANNS SENSITIVITY WITHIN EACH CATEGORY.

Parameter	**PANNs Sensitivity**	**Phase Residual**
Oscillators		
osc1_waveform	0.042	117%
osc2_waveform	0.036	118%
osc1_octave	0.034	108%
osc2_octave	0.023	71%
osc1_level	0.034	62%
osc2_level	0.032	73%
osc1_detune	0.021	98%
osc2_detune	0.011	51%
sub_level	0.052	98%
pulse_width	0.026	75%
osc_sync	0.007	7%
ring_mod	0.044	86%
osc_fm_amount	0.021	89%
Filter		
filter_type	0.223	369%
filter_cutoff	0.193	228%
filter_resonance	0.143	119%
filter_env_amount	0.070	95%
filter_drive	0.023	51%
filter_keytrack	0.009	0%
Envelopes		
amp_sustain	0.184	85%
filter_sustain	0.057	67%
amp_attack	0.043	83%
filter_attack	0.039	40%
amp_decay	0.026	20%
filter_decay	0.015	42%
amp_release	0.017	2%
filter_release	0.010	1%
Effects & Modulation		
noise_level	0.149	126%
lfo1_rate	0.107	108%
lfo1_to_filter	0.088	73%
lfo1_to_pitch	0.012	89%
distortion	0.045	100%
delay_time	0.030	25%
delay_feedback	0.035	20%
delay_mix	0.030	20%
unison_detune	0.010	83%
note_hold_time	0.093	N/A