Project Genesis: The Neural Un-Mixer v2.0

Technical Specification for Inverse Signal Chain Estimation

Planning Document

February 2026

1 System Objective

The **Neural Un-Mixer** is an end-to-end MLOps system designed for *Inverse Parameter Estimation*. It decomposes a "Wet" audio sample x into its constituent synthesizer parameters θ_{synth} and effect chain configurations θ_{fx}, such that a re-rendering of these parameters in Ableton Live 12 minimizes the perceptual distance to the original source.

2 Mathematical Framework

The project treats audio reconstruction as a Non-Linear Inverse Problem. We define the forward mapping $G(\theta)$ as the Ableton Signal Chain. Our goal is to find the inverse mapping $F(x) \approx G^{-1}(x)$.

2.1 Multi-Task Architecture

The model employs a shared feature extractor (Encoder) with N specialized heads:

- **MDN Head**: Outputs parameters π_i, μ_i, σ_i for the mixture distribution of continuous knobs.
- **Classification Head**: $P(y|x)$ for discrete wavetable and filter indices.
- **Hyperbolic Sequence Decoder**: Unlike standard Euclidean decoders, this module embeds the effect chain $S = (fx_1, fx_2, ..., fx_8)$ into **Hyperbolic space (Poincaré ball)**. This geometry naturally captures the hierarchical and non-commutative nature of signal paths (where order matters), resolving the combinatorial explosion of effect permutations [1].

2.2 Hybrid Loss Function

To address the multimodal nature of the parameter space (Non-Identifiability), we replace standard MSE with Negative Log-Likelihood (NLL). Crucially, to maintain end-to-end differentiability through the proxy, we employ the *Reparameterization Trick* during the spectral loss calculation.

$$L_{total} = \lambda_p \underbrace{-\log\left(\sum_{k=1}^{K} \pi_k(x)\mathcal{N}(\theta_{gt}|\mu_k(x), \sigma_k(x))\right)}_{\text{NLL Loss (MDN)}} + \lambda_s \underbrace{L_{Spectral}(G_{proxy}(\hat{\theta}_{sample}), x)}_{\text{Perceptual Loss}} \tag{1}$$

Where $\hat{\theta}_{sample}$ is drawn from the predicted distribution $p(\theta|x)$ using a differentiable sampling strategy (e.g., Gumbel-Softmax for discrete, Gaussian reparameterization for continuous) to allow gradient flow back to the encoder.

2.3 Inference-Time Finetuning (ITF)

To address the "Proxy Gap" (discrepancy between the neural proxy and the real Ableton engine), we implement an ITF stage during inference [2]. After the initial prediction $\hat{\theta}_{init}$, we freeze the proxy decoder weights ϕ^* and refine the input parameters θ for the specific test sample x_t:

$$\theta_{final} = \text{argmin}_\theta \left(L_{Spectral}(G_{proxy}(\theta), x_t) + \lambda_B L_{regularization} \right) \tag{2}$$

This allows the model to "overfit" the specific audio sample at runtime, significantly reducing reconstruction error.

3 Statistical Guardrails & Optimization

Given the high-dimensional and non-convex nature of the parameter space, we implement the following:

1. **Mixture Density Networks (MDN)**: The model predicts a probability distribution $p(\theta|x)$ rather than a point estimate to handle the one-to-many mapping problem where multiple settings create identical sounds.
2. **GCWD Optimization**: Standard Adam optimizers often fail on Spectral Loss landscapes due to ill-conditioning (gradients ranging from 10^{40} to 10^{-40}). We utilize **Gradient Clipping with Weight Decay (GCWD)** to prevent floating-point overflow and stabilize convergence [3].
3. **Curriculum Learning**: Sequential training phases: Dry Synth $\rightarrow$ Non-Linear FX $\rightarrow$ Full Chain.

4 Technical Stack

- **Engine**: PyTorch, Torchaudio, AbletonOSC.
- **Differentiable DSP**: Google Magenta DDSP (ported to PyTorch).
- **Optimization**: Geoopt (for Riemannian/Hyperbolic optimization).
- **Data/Ops**: DVC (Versioning), MLflow (Tracking), FastAPI (Inference).
- **Domain**: Ableton Live 12 Native Devices (Wavetable + Big 8 FX).

5 References

1. Wada, A., et al. "Hyperbolic Embeddings for Order-Aware Classification of Audio Effect Chains." *DAFx25*, 2025.
2. Barkan, O., et al. "InverSynth II: Sound Matching Via Self-Supervised Synthesizer-Proxy and Inference-Time Finetuning." *ISMIR*, 2023.
3. Combes, P., et al. "Gradient Clipping Improves Neural Network Optimization for Perceptual Sound Matching." *Eusipco*, 2025.