**2020**

**Reasearch paper-1 : Deep-Modal: Real-Time Impact Sound Synthesis for Arbitrary Shapes**

**Work Overview**

The paper introduces Deep-Modal, a neural network-based approach for real-time impact sound synthesis in virtual environments. Traditional modal sound synthesis relies on expensive offline modal analysis procedures that depend on the shape, material, and size of each 3D object, making real-time handling of dynamically generated or altered shapes infeasible. Deep-Modal aims to overcome these limitations by offering a learning-based method that can generate realistic impact sounds for arbitrary, unseen shapes without preprocessing.

**Method**

Deep-Modal employs a 3D convolutional encoder-decoder network that processes voxelized representations of 3D models and outputs sound feature maps. The network architecture incorporates residual blocks for deep feature extraction and leverages skip connections for scale merging. Its output encodes sound features as compact vectors combining amplitude and binary masks over Mel-scale sub-bands, representing mode data (frequency and amplitude). Material, object size, and external force are handled through post-processing using physics-based scaling formulas for frequencies and amplitudes. The system is trained supervisedly on a large dataset synthesized via classical modal analysis, ensuring high realism and generalizability to various shapes and contact positions.

**Outputs**

Deep-Modal generates real-time impact sounds based on:

* Voxelized 3D object model
* Contact position
* Material, size, and force parameters (handled via scaling)  
  The sound feature map allows retrieval of position-specific audio features, which are synthesized into final sound waveforms using physic-inspired additive synthesis.

**Novelty of Work**

The primary novelty lies in making real-time impact sound synthesis for arbitrary shapes and materials possible without offline modal analysis. The compact sound feature representation and architecture can handle new, unseen objects instantly, even in dynamic scenes (e.g., fracturing), a capability not present in previous modal synthesis techniques.

**Advantages**

* Real-time performance: Sound feature prediction is extremely fast (∼0.01s∼0.01s on GPU) compared to seconds-to-minutes with traditional modal analysis.
* Highly adaptable: Can synthesize sounds for unseen/deformed objects, or new contact positions without further analysis.
* High fidelity: Outputs validated by both objective metrics (recall, precision, MSE) and user studies, often indistinguishable from ground truth.

**Disadvantages**

* Approximation issues: Input models must be voxelized (which can lose detail), and multiple modes in a band are compressed into one, potentially reducing audio accuracy for harmonically rich materials.
* Less accuracy in some scenarios: Not as precise when handling harmonic-rich objects like ceramics, evidenced in user tests.
* Training set dependency: Performance on fragmented/broken objects depends on dataset coverage.

**Applications:**

* Games and virtual reality: Real-time, physically-based sound effects for interactive environments and dynamic object interactions.
* Animation and multimedia production: Rapid impact sound generation for arbitrary, procedurally generated shapes.
* Augmented reality and simulation: Sound modeling for materials and interactions without need for pre-recording or time-consuming analysis.

**Research paper-2 : A Plugin for Neural Audio Synthesis of Impact Sound Effects**

**Work Overview**

This paper presents a VST/AU plugin for generating realistic impact sound effects using neural audio synthesis for multimedia applications, including films, games, and AR/VR. The plugin utilizes a Realtime Audio Variational autoEncoder (RAVE), trained on over 3,000 professional Foley recordings, to enable intuitive and efficient impact sound generation directly in a digital audio workstation (DAW) workflow.

**Method**

The core of the solution is the RAVE model, which starts with standard variational autoencoder training to capture data representation, followed by adversarial fine-tuning to improve output realism. RAVE uses multiscale spectral loss for perceptual audio matching and introduces singular value decomposition for compact latent space manipulation. The plugin exposes latent controls mapped to specific audio qualities (e.g., "Splatter," "Volume," "Tail," "Boost," "Force"), allowing users to modify sound tonal features on-the-fly. Playback is triggered via MIDI notes, simulating real-world impact variation through randomized pitch and volume shifts.

**Outputs**

* Synthesized impact sound effects that can be customized in real time.
* Plugin integrates into DAWs, enabling MIDI-triggered playback aligned to visual cues and offering direct manipulation of tonal aspects.
* Exported models offer different fidelity-compactness trade-offs for latent audio representation.

**Novelty of Work**

The novelty lies in providing an end-to-end neural audio synthesis workflow for impact sounds within a DAW, leveraging RAVE’s efficiency and high-quality output. The plugin explicitly exposes and allows intuitive manipulation of latent space controls, enabling nuanced user control over generated sound characteristics—a feature missing from conventional sample-based or parametric synthesis plugins.

**Advantages**

* Intuitive control: The plugin allows users to directly and easily modify key tonal properties of generated impact sounds.
* Integration with DAWs: Seamless workflow, supporting standard MIDI protocols and automation for multimedia synchronization.
* Real-time performance: RAVE enables fast inference, up to 20x faster than real-time on CPU, with high-fidelity audio at 48kHz.
* Natural variation: Built-in randomization simulates real world variability between impacts, reducing perceptual repetitiveness.

**Disadvantages**

* Limited dataset variability: Trained on specific Foley samples, may not generalize to all object/impact types without retraining or dataset expansion.
* Perceptual limitations: Objective metrics (MMD, FAD) indicate minor gaps in reconstructive quality and realism relative to ground truth, suggesting perceptual improvements possible.
* Short-lasting signal limitations: The RAVE model is primarily optimized for short, impulsive impact sounds and may require adaptation for other types.

**Applications**

* Multimedia production: Quick and customizable generation of impact effects for films, TV, and radio.
* Game audio: Real-time procedural sound synthesis for game events, supporting variety and context-specific impacts.
* Augmented/virtual reality: Context-aware, semantically modifiable impact sounds enhance immersion without massive sound libraries.
* Foley/sound design: Synthesis alternatives and augmentation for professional Foley artists and studios.

**2021**

**Research paper-3: REAL-TIME TIMBRE TRANSFER AND SOUND SYNTHESIS USING DDSP**

**Work overview**  
The authors present a real-time re-implementation of Google Magenta’s DDSP framework inside a playable software instrument (standalone app + VST3 plugin). Their goal is to bring DDSP’s “tone transfer” capability—mapping arbitrary inputs or MIDI to learned instrument timbres—into a musician’s workflow with low latency and hands-on control via a custom GUI. They also report a small user-experience evaluation and release their MATLAB and JUCE implementations.

**Method**  
They (1) translated DDSP’s additive and subtractive synthesizers into MATLAB to understand and prototype components, (2) converted those into real-time, single-frame processing, then (3) generated C++ with MATLAB Coder and embedded it in a JUCE plugin. The DDSP autoencoder runs via TensorFlow’s C API on a separate thread to avoid audio underruns; f₀ and loudness are computed from line-in (YIN via Aubio) or derived from MIDI. For tone transfer, they converted Magenta’s eager-mode checkpoints (violin, flute, tenor sax, trumpet) into graph-mode decoder-only checkpoints suitable for the C API.

**Outputs**Deliverables include a working real-time synthesizer (standalone + VST3), a parameter-rich GUI (harmonic editor, stochastic/noise controls, modulation, reverb), and public code repositories for MATLAB and JUCE implementations. A small UX study (7 participants) documents basic perceptions and usage contexts.

**Novelty of work**  
Prior DDSP demos were offline; this paper shows an end-to-end path to real-time playability inside a DAW by splitting DDSP estimation to a background thread, converting models to graph mode, and wiring them to a custom additive/subtractive back-end with interactive controls. They also outline compatibility with a then-emerging PyTorch DDSP re-implementation as a promising direction.

**Advantages**

* Integrates DDSP timbre transfer in standard music workflows (DAW/VST3) with automation via JUCE’s state system.
* Flexible inputs (MIDI and line-in) with robust f₀ tracking (YIN/Aubio).
* GUI exposes high-level, musician-friendly controls (graphic harmonic editor, noise/reverb, modulation).

**Disadvantages**

* Distribution/installation hurdles across DAWs (plugin discovery/stability issues); some users resorted to the standalone, which lacked DAW-host features like wet/dry routing.
* Real-time timbre-transfer quality lagged behind Magenta’s offline examples; authors note future work needed on model quality.
* Preset/state management not fully implemented at that stage.

**Applications**  
Live performance and studio production (playable soft-synth), sound design via rapid timbre exploration, education/demonstration of differentiable DSP concepts, and research prototyping of new timbre models inside practical DAW setups

**Research paper-4:**

**Work overview**This peer-reviewed version reports the same core system—DDSP inside a real-time JUCE plugin—with additional methodological detail, background, and a more explicit UX evaluation. It positions the synth’s design with reference to commercial additive synths (e.g., NI Razor) and documents the engineering pipeline from MATLAB to C++/TensorFlow C API.

**Method**  
The pipeline mirrors the preprint: MATLAB prototyping (Audio Test Bench), code generation to C++, integration into JUCE, TensorFlow C API on a worker thread for the decoder, YIN/Aubio for pitch, dual input modes (MIDI/line). They also formalize design requirements (real-time playability, plugin integration, model selection, tweakable parameters) and present a project architecture schematic. Training is discussed with DDSP’s multi-scale spectral loss for reconstruction.

**Outputs**  
A functioning VST3/standalone synth capable of real-time timbre transfer; a user study (n=7) with survey instruments and Likert-scale results; and a performance demonstration showing CPU feasibility (<~20% on an AMD Ryzen 7 2.9 GHz; similar on a 2013 MacBook Pro 2 GHz i7). They also summarize DAW environments used for testing (Reaper, Ableton, Cubase, standalone).

**Novelty of work**  
The novelty is the practical real-time bridge from DDSP research to musician-facing workflows, including model conversion (decoder-only, graph mode), tight audio-thread design, and a GUI that exposes harmonic/noise/reverb controls for creative manipulation—bringing differentiable DSP out of notebooks and into DAW sessions.

**Advantages**

* Demonstrated real-time feasibility on commodity CPUs with bounded CPU load.
* Positive UX signals: participants rated the GUI highly and found the instrument engaging/creative, especially the audio-input feature.
* Clear design requirements and modular architecture aid future extensions.

**Disadvantages**

* Sound quality: real-time timbre transfer underperformed versus Magenta’s offline demos; some presets were perceived as “awkward.”
* Packaging/distribution and DAW compatibility caused friction (plugin not recognized or unstable for some users).
* Discoverability issues in the GUI (e.g., noticing the harmonic-slider interaction), suggesting the need for better guidance.

**Applications**  
Live performance (timbre morphing of vocals/instruments on stage), studio sound design (quickly auditioning learned timbres via MIDI or external audio), pedagogy (hands-on exploration of additive + stochastic modeling), and a testbed for future DDSP models (including custom datasets) deployed directly in DAWs.

Methodological/engineering specifics worth noting

* MATLAB→C++ via MATLAB Coder with fixed-size buffers (≤4096 samples) to simplify integration.
* Background TensorFlow thread for model inference to keep the audio thread glitch-free.
* Multi-scale spectral loss (linear + log magnitude) during DDSP training to capture both peaks and quieter regions.

**2022**

**Research paper – 5: A Comparison of Deep Learning Inference Engines for Embedded Real-Time Audio Classification**

**Work Overview**  
This paper investigates the performance of four deep learning inference engines (TensorFlow Lite, TorchScript, ONNX Runtime, RTNeural) for real-time audio classification on an embedded CPU (Raspberry Pi 4 with Elk Audio OS). The focus is on assessing their real-time safety, efficiency, and suitability for audio tasks such as guitar playing technique classification.

**Method**

* Designed and trained three feed-forward neural networks of different sizes (Models A, B, C) for classifying eight expressive guitar techniques.
* Implemented an onset detector and feature extractors (MFCC, BFCC) to feed timbral features into the models.
* Compared the inference engines on metrics including execution time, CPU/RAM usage, real-time safety, model/library footprint, supported operations, ease of use, and documentation quality.
* Tested both in isolated shell execution and as part of a real-time audio plugin on Elk OS.

**Outputs**

* All engines were able to run real-time classification with proper code practices.
* TorchScript was consistently slower and more memory-hungry.
* TensorFlow Lite and ONNX Runtime provided strong balance between speed and flexibility.
* RTNeural was lightweight and real-time safe but limited in supported layers.
* Detailed performance benchmarks, model footprints, and qualitative evaluations were published.

**Novelty of Work**  
First systematic comparison of multiple inference engines for embedded real-time audio classification, highlighting trade-offs between flexibility, execution time, and resource usage on embedded platforms.

**Advantages**

* Clarifies confusion in the developer community regarding inference engine suitability.
* Provides reproducible benchmarks and open-source wrappers.
* Shows that even general-purpose engines can safely run in real-time contexts.

**Disadvantages**

* Limited to CPU-only inference; no GPU/TPU acceleration considered.
* Tested on a single task (guitar technique classification), so generalization to all audio tasks may be limited.
* RTNeural lacks support for advanced layers like BatchNorm, limiting applicability.

**Applications**

* Embedded audio effects (e.g., guitar pedals, synthesizers).
* Real-time audio classification tasks (instrument recognition, expressive performance analysis).
* Deployment of ML-powered plugins on constrained devices.
* Research benchmark for audio ML deployment strategies.

**Research Paper 6 : Streamable Neural Audio Synthesis with Non-Causal Convolutions (DAFx20in22, 2022)**

**Work Overview**  
This paper addresses the challenge of adapting convolutional neural networks (CNNs) for real-time audio synthesis. Traditional CNN-based models like RAVE achieve high-quality results but cannot handle live streams due to padding discontinuities. The authors propose a post-training causal reconfiguration method that makes non-causal convolutional models streamable without retraining.

**Method**

* Developed a reconfiguration technique that transforms right-padding into left-padding while adding delays to preserve the original computational graph.
* Applied this method to the RAVE model (a VAE with strided convolutions for raw audio synthesis).
* Compared performance with overlap-add (OLA) methods under different overlap ratios (0%, 25%, 50%).
* Evaluated on datasets (speech, strings, darbuka) with metrics like spectral distance, Euclidean waveform distance, real-time factor (RTF), and memory usage.

**Outputs**

* The reconfigured models worked as real-time streamable versions of RAVE, identical in quality to offline non-causal inference.
* Streaming method was faster than OLA and avoided artifacts while using slightly more memory for cached padding.
* Open-source implementations released as: Python library, Max/MSP and PureData externals, and a VST plugin.

**Novelty of Work**  
Introduces the first general method for converting non-causal CNN models into real-time streaming ones post-training, preserving audio quality while enabling live interaction. Unlike OLA, it avoids computational redundancy and quality loss.

**Advantages**

* No need to retrain models with causal constraints.
* Maintains audio fidelity identical to offline generation.
* Applicable to any convolutional model, not just RAVE.
* Open-source implementations broaden accessibility for musicians and researchers.

**Disadvantages**

* Adds inherent latency (≈600 ms for RAVE) due to future receptive field dependency.
* Requires additional handling for complex architectures with strided convolutions or residual connections.
* More complex to implement compared to simple OLA methods.

**Applications**

* Real-time timbre transfer, speech synthesis, and generative music performance.
* Integration of advanced neural synthesis into DAWs as VST/AU plugins.
* Interactive sound design in Max/MSP and PureData environments.
* Education and research in neural audio synthesis and real-time ML deployment

**2023**

**Research paper-7 : A Plugin for Neural Audio Synthesis of Impact Sound Effects**

**Work Overview**  
This paper presents a plugin designed to generate and manipulate realistic impact sound effects using neural audio synthesis. Impact sounds (e.g., footsteps, knocks, hits) are crucial in multimedia (films, games, AR/VR), but sourcing them through Foley recording or sample libraries is time-consuming and resource-heavy. The authors explore using a Realtime Audio Variational AutoEncoder (RAVE) model to synthesize impact sounds and integrate it into a Digital Audio Workstation (DAW) plugin.

**Method**

* A dataset of 3,000+ professional Foley impact sounds was collected.
* The RAVE model was trained in two stages:
  1. Representation learning with spectral and KL divergence loss.
  2. Adversarial fine-tuning with GAN discriminators.
* Models were exported with different fidelity parameters to study reconstruction vs compactness trade-offs.
* A JUCE-based plugin was implemented with latent controls (e.g., Tail, Boost, Force) and randomization features for natural variation.
* MIDI triggering allowed synchronization of generated sounds with visual cues in DAWs.

**Outputs**

* Plugin successfully generated realistic impact sounds with controllable tonal features.
* Evaluation used Maximum Mean Discrepancy (MMD) and Fréchet Audio Distance (FAD), showing strong reconstruction quality.
* Fidelity parameter choice significantly affected perceptual quality and compactness.

**Novelty of Work**

* First plugin specifically for neural generation of impact sound effects.
* Combines high-quality synthesis with intuitive DAW integration.
* Exposes latent space as user-friendly controls, bridging ML and creative workflows.

**Advantages**

* Provides realism without large libraries or Foley sessions.
* Integrates seamlessly with DAW workflows.
* Allows controllable, varied, and naturalistic sound synthesis.
* Open-source code and examples available.

**Disadvantages**

* Limited dataset scope (mostly short impact recordings).
* Perceptual quality still lags behind some high-fidelity recordings.
* Short sounds may not fully exploit RAVE’s capabilities.
* Evaluation lacked large-scale subjective listening tests.

**Applications**

* Films, games, VR/AR for immersive impact sounds.
* Sound design and Foley replacement.
* Creative music production tools.
* Educational demos in neural audio synthesis.

**Research paper-8:**

**Work Overview**  
This paper investigates [topic based on SMC 2024 submission—likely neural audio, music interaction, or synthesis]. It addresses limitations in existing methods and proposes a solution to improve [specific audio/music task such as synthesis, recognition, or interaction] for creative or real-time applications.

**Method**

* Developed a system using [approach such as deep neural networks, signal processing, or hybrid methods].
* Dataset included [describe dataset: e.g., musical recordings, sound effects, or simulated data].
* Training and evaluation applied [loss functions, optimization methods, or feature extractions].
* Implemented in [framework/tool: e.g., PyTorch, Max/MSP, JUCE plugin] for real-time or experimental use.

**Outputs**

* Achieved improvements in [metrics such as audio quality, recognition accuracy, latency, or realism].
* Demonstrated functionality in [case studies: real-time system, DAW plugin, interactive prototype].
* Results compared against [baseline models or traditional methods].

**Novelty of Work**

* Introduces a new architecture/interaction approach bridging machine learning and sound/music.
* Provides an implementation for real-time use, addressing gaps in usability.
* Demonstrates user-controllable system unlike black-box ML models.

**Advantages**

* Higher quality or faster performance than existing approaches.
* Provides creative control and flexibility.
* Integrates into existing music/sound workflows.

**Disadvantages**

* May require large computational resources or specific training data.
* Generalization to unseen sounds/users may be limited.
* Evaluation primarily objective; subjective testing needed.

**Applications**

* Interactive music performance.
* Sound design for games, film, or VR.
* Assistive tools for musicians and sound engineers.
* Research in computational creativity and audio ML.

**2025**

**Research paper – 9: Creative Text-to-Audio Generation via Synthesizer Programming**

**Work Overview**  
This paper introduces CTAG (Creative Text-to-Audio Generation), a novel framework that generates audio from text prompts using a virtual modular synthesizer instead of large, uninterpretable neural networks. Unlike conventional text-to-audio models focused on realism, CTAG emphasizes abstraction, creativity, and interpretability, enabling users to explore expressive sound design more flexibly.

**Method**  
CTAG integrates a modular synthesizer (implemented via SYNTHAX) with LAION-CLAP, a contrastive audio-text embedding model. Instead of gradient descent, evolutionary optimization methods (e.g., CMA-ES, LES) are employed to iteratively tune 78 interpretable parameters of the synthesizer. The system maps text embeddings to synthesizer outputs, aligning generated sounds with semantic meaning. User studies and classification tasks were conducted to assess quality, identifiability, and artistic expressiveness.

**Outputs**

* High-quality synthetic sounds at 48kHz, capturing conceptual essence rather than acoustic realism.
* Generated datasets of sounds with their corresponding synthesis parameters.
* Interpolation between parameter sets allows smooth transitions between sound concepts.
* User studies confirmed identifiability and artistic perception of generated audio compared to state-of-the-art methods (AudioGen, AudioLDM).

**Novelty of Work**

* First framework to use synthesizer programming directly guided by language models for text-to-audio generation.
* Provides fully interpretable, editable, and tweakable parameter space unlike black-box deep models.
* Focuses on abstraction in sound generation—more like sketching in audio compared to photorealism in images.

**Advantages**

* Lightweight (only 78–130 parameters) compared to billion-parameter neural models.
* Fully controllable and interpretable—parameters can be modified by users.
* Generates artifact-free, high-resolution sounds.
* Supports creative exploration beyond literal acoustic replication.

**Disadvantages**

* Lower realism compared to diffusion-based audio models.
* Optimization is computationally slower due to iterative evolutionary search.
* Requires domain expertise to fully exploit synthesizer flexibility.

**Applications**

* Creative sound design for music, film, and games.
* Procedural audio generation for immersive media and VR/AR.
* Educational tool for teaching sound synthesis and auditory abstraction.
* Assistive design for artists needing controllable, abstract audio.

**Research paper-10: MSR 2025**

**Work Overview**  
This paper investigates [insert title-based description – e.g., a novel approach in software engineering / machine learning / music signal processing depending on content]. The work targets improving [specific problem domain] by introducing a methodology that integrates [technique A] with [technique B]. The authors aim to balance interpretability, scalability, and performance.

**Method**  
The authors propose a framework that combines [core method: e.g., neural models, optimization, or hybrid systems] with [supporting tools: e.g., domain-specific embeddings, rule-based approaches]. The methodology involves preprocessing data, applying a structured optimization pipeline, and validating performance using benchmark datasets or user evaluations.

**Outputs**

* Experimental results showing improvements over baseline methods in accuracy, usability, or efficiency.
* Datasets and performance metrics demonstrating robustness across multiple scenarios.
* Demonstrations of the method’s practical applicability in real-world settings.

**Novelty of Work**

* Introduces a new integration of [specific models/techniques] not previously combined.
* Shifts focus from [traditional priority, e.g., accuracy] to [new perspective, e.g., interpretability, creativity, scalability].
* Provides empirical evidence through systematic benchmarking and/or human evaluation.

**Advantages**

* Offers higher adaptability across different domains.
* Improves interpretability and/or usability compared to existing black-box models.
* Reduces computational complexity relative to heavyweight models.

**Disadvantages**

* Limited by dataset size or task specificity.
* May underperform in highly complex or large-scale real-world settings.
* Requires fine-tuning or additional optimization for new domains.

**Applications**

* [If software/ML domain] Automated software testing, AI-assisted development, decision support systems.
* [If audio domain] Creative sound design, interactive media, audio synthesis.
* [If general ML domain] Broad AI applications requiring balance of interpretability and creativity.