DiffImpact: Differentiable Rendering and Identification of Impact Sounds

Samuel Clarke **Negin Heravi** **Mark Rau** **Ruohan Gao**
Jiajun Wu **Doug James** **Jeannette Bohg**
Stanford University
{spclarke, nheravi, mrau, rhgao, jiajunw, djames, bohg}@stanford.edu

Abstract: Rigid objects make distinctive sounds during manipulation. These sounds are a function of object features, such as shape and material, and of contact forces during manipulation. Being able to infer from sound an object's acoustic properties, how it is being manipulated, and what events it is participating in could augment and complement what robots can perceive from vision, especially in case of occlusion, low visual resolution, poor lighting, or blurred focus. Annotations on sound data are rare. Therefore, existing inference systems mostly include a sound renderer in the loop, and use analysis-by-synthesis to optimize for object acoustic properties. Optimizing parameters with respect to a non-differentiable renderer is slow and hard to scale to complex scenes. We present DiffImpact, a fully differentiable model for sounds rigid objects make during impacts, based on physical principles of impact forces, rigid object vibration, and other acoustic effects. Its differentiability enables gradient-based, efficient joint inference of acoustic properties of the objects and characteristics and timings of each individual impact. DiffImpact can also be plugged in as the decoder of an autoencoder, and trained end-to-end on real audio data, so that the encoder can learn to solve the inverse problem in a self-supervised way. Experiments demonstrate that our model's physics-based inductive biases make it more resource efficient and expressive than state-of-the-art pure learning-based alternatives, on both forward rendering of impact sounds and inverse tasks such as acoustic property inference and blind source separation of impact sounds. Code and videos are at https://sites.google.com/view/diffimpact.

Keywords: Differentiable Sound Rendering, Auditory Scene Analysis

1 Introduction

The sound we perceive during rigid object contact is a function of many factors, including noise in the environment, position of the listener, reflecting surfaces in the surrounding environment, etc. The most defining of these factors are the vibrations the object makes, a product of its intrinsic acoustic properties and the way it was contacted. Extracting information about the acoustic properties of an object could inform a robot about the object's size, shape, material, and where and how it was struck. Extracting information about the way the object was contacted could inform a robot of the velocity, mass, shape, and hardness of the tool that was used to strike the object. Therefore, the ability to interpret impact sounds could augment and complement what robots can perceive from vision, especially in cases of occlusion, low visual resolution, poor lighting, or blurred focus.

Decomposing the sound an object makes into the contact forces that excited it and the idealized acoustic impulse response is generally an ill-posed problem. As annotations on sound data are rare, existing inference systems mostly include a sound renderer in the loop, and use analysis-by-synthesis to optimize for object acoustic properties. Optimizing parameters with respect to a non-differentiable renderer is slow and hard to scale to complex scenes. We propose DiffImpact, a fully differentiable, generative sound model that uses physics-based priors as a bias. Specifically, we introduce fully differentiable models of the object's impulse response and the contact force during impact. We combine these with differentiable models of environment acoustics such as ambient noise and reverberation into a fully differentiable generative sound model. With this physics-based model, we can now infer physically interpretable parameters of rigid body impact sounds through analysis-by-synthesis, even

5th Conference on Robot Learning (CoRL 2021), London, UK.

from real-world recordings, including from real YouTube videos or from a noisy and reverberant robotics lab.

Impacts in the real world are not always clean impulses, nor do they occur in controlled acoustic environments. We show how DiffImpact's differentiability allows us to learn meaningful sound models that are fit to the sounds of multiple imperfect impact sounds over one or multiple recordings. We first validate our approach on single impact sounds. We show that our model can fit interpretable physical parameters of an object's sound model and of the contact forces to audio recordings. We then show how DiffImpact enables a robot to passively learn object sound models from a real-world dataset (ASMR YouTube videos [1]). Our model not only extracts physically interpretable parameters, but also outperforms model-based and model-free learning baselines in rendering realistic impact sounds for held-out data. Finally, we show how a robot can learn object sound models from data it collected by striking these objects with a tool. Specifically, we show how the robot can perform blind source separation of the collected sound and thereby achieves 140% higher accuracy on classifying the material of the objects it strikes.

2 Related Work

Audio-frequency vibrations have been shown to be an effective modality for robots to ascertain characteristics of physical events in their environment, including instance level classification of everyday objects [2, 3] and estimates of their motion [4], terrain classification by legged robots [5], masses of granular materials being poured from a scoop the robot is holding [6], and amounts of liquid poured by a robot [7]. Auditory scene analysis also has applications for dynamic inference of stochastic events, enabling creation of audio based reactive robotic trackers [8].

Many of these works have a common characteristic with more recent visual perception approaches, that learning-based neural methods are subsuming many classical approaches by enabling automatic feature extraction rather than requiring hand engineered features [9, 10]. However, whereas differentiable image renderers have empowered new approaches in visual perception [11, 12, 13, 14], differentiable audio renderers have been introduced only recently. Engel et al. [15] recently debuted the Differential Digital Signal Processing (DDSP) library, showing applications in the musical domain on tasks such as disentangling the pitch and timbre of instrument sounds. Rather than the musical domain, we focus on modeling modal impact sounds of rigid objects, and show applications of our physics-based model for robotics tasks.

Model-free neural audio representation methods are powerful in synthesizing realistic human speech [16, 17] but lack interpretability and flexibility for transfer to different categories such as transient environment sounds. Model-based methods often require less data and training time to produce interpretable learned parameters, which can be reused and applied to other tasks more intuitively. They are also capable of synthesizing realistic object sounds [18, 19, 20] but depend on hand-engineered analytical methods for parameter estimation with structured audio samples. Ren et al. [21] use real audio clips of striking objects to infer properties such as Young's modulus and the modes of the object, then attempt to reproduce the sound based on a modal model with a residual. However, they assume knowledge of the 3D geometry of the object as well as impact location. Zhang et al. [22] infer object properties such as Young's modulus and Rayleigh damping from sounds of objects falling on a surface by using a physics-based modal sound generation engine with a black box forward model. But their model is not fully differentiable, making it challenging to learn from unconstrained in-the-wild audio or to optimize over many dimensions at once. While we use a model conceptually similar to that of [23, 22], our fully differentiable structure empowers learning a best estimate of an object's modal model from the sound of multiple uncontrolled, imperfect, heterogeneous impacts, rather than requiring strictly controlled recordings for analysis.

3 Differentiable Rendering of Impact Sounds

Our DiffImpact models the sound that a rigid object makes when struck in a real environment as a function of 1) the impact force's magnitude over time; 2) the brief click of a sound pressure wave caused by the rapid deceleration of the striking object, referred to as the "acceleration sound"; 3) the struck and striking objects' impulse responses, or their idealized theoretical vibrational response to a perfect unit impulse; 4) any environment, background, or recording noise; and finally 5) the impulse response of the local environment, reflecting and transmitting reverberations of any sound being directly emitted by the object. DiffImpact attempts to decompose real impact sounds into each of these contributions, using physics-based models in order to regularize this decomposition.

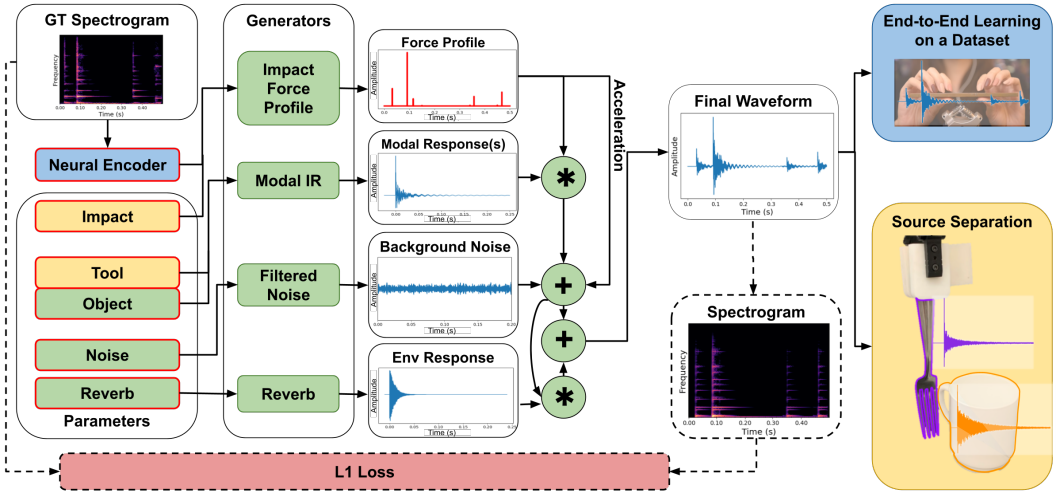


Figure 1: Overview of proposed approach, DiffImpact. Blue elements are only used for the end-to-end learning task, yellow elements are only used in source separation, and green elements are used in both. Dashed components are only used during training, and elements with a red outline have learnable parameters.

3.1 Overview

Figure 1 provides a high-level overview of our approach. There are some model variations dependent on which inference task we are considering. For clarity, here we will give a brief overview of the model when used for the end-to-end learning task of predicting an audio waveform from a magnitude spectrogram of the ground truth audio sample, similar to the models in [15, 24, 16]. The magnitude spectrogram lacks phase information and is therefore not directly invertible to an audio waveform. This spectrogram is input to an encoder network that outputs a coarsely temporally discretized time series of parameters that represent impact characteristics. These parameters are the input to the *Impact Force Profile Generator* that synthesizes an impact force profile as described in more detail in Section 3.2. Given object-specific, learnable parameters of a modal model of rigid body impact sound, the *Modal Impulse Response Generator* synthesizes an impulse response as described in more detail in Section 3.3. This impulse response is convolved with the force profile to synthesize object vibrations. To model the contribution of the sound pressure wave caused by the rapid deceleration of the impacting object, this impact force profile is also multiplied by a learnable scalar and added to the object vibration sounds.

We theoretically now have all the sounds emitted directly from the point of impact on the object. To model sound that is specific to the environment, we introduce a noise and reverb generator as detailed in Section 3.4. Given learnable noise parameters, the *Filtered Noise Generator* synthesizes recording or background noise that is then added to the current audio. Finally, given learnable reverb parameters, the *Reverb Generator* synthesizes an impulse response of the environment. This response is then convolved with the current audio to model the sounds that have transmitted through the environment and reflected off nearby surfaces, rather than directly from the object. This result is added to the current audio to produce the final waveform.

For training the aforementioned learnable parameters of the generators and the encoder, we use a loss that compares spectrograms of the generated waveform with those of the ground truth (Section 3.5).

The final waveform $\hat{W}(t)$ produced by our model can be expressed as

$$\hat{W}(t) = [F(t;\mathbf{m},\mathbf{t},\boldsymbol{\tau}) \circledast [\alpha + IR_o(t_{IR};\mathbf{f},\mathbf{g_o},\mathbf{d_o})] + N(t;\mathbf{g_n})] \circledast (1 + IR_e(t_{IR};\mathbf{g_e},\mathbf{d_e})), \quad (1)$$

where t is the time in the final waveform; F is the force profile function that takes as input the impulse magnitudes $\mathbf{m}$, impulse timings $\mathbf{t}$, and contact time scales $\boldsymbol{\tau}$; $\circledast$ denotes a convolution; α is a scalar for the acceleration sound; IR_o is the modal object impulse response function parameterized by modal frequencies $\mathbf{f}$, gains $\mathbf{g_o}$, and dampings $\mathbf{d_o}$, and t_{IR} is time within each impulse response; N is the filtered noise parameterized by frequency band gains $\mathbf{g_n}$; and IR_e is the impulse response of the environment parameterized by frequency band gains $\mathbf{g_e}$ and dampings $\mathbf{d_e}$. We will now detail the equations and physics-based models for each of these components we propose, and how we scale a low dimensional set of normalized parameters into input parameters for each of these functions, producing an audio frequency signal in a differentiable manner.

3.2 Differentiable impact force profile and acceleration sound

To model the force profiles of multiple impact events occurring over the time interval of an audio clip, our proposed module converts the two time series outputs from the encoder into a contact force profile at audio frequency. This profile is defined by the time scales $\boldsymbol{\tau}$ and magnitudes $\mathbf{m}$ of the impulses.

First, we define a physical model for each impulse event. Since objects will generally have convex curvature at the points at which they strike, they can be locally approximated as spheres in contact. Therefore, we adopt a Hertz half-sine contact model based on collisions between frictionless rigid spheres similar to that of [25]. To ensure smoothness, we use a Gaussian approximation $C_i(t;t_i,\tau_i)$ of this contact model force impulse profile [26, 27]:

$$C_i(t;t_i,\tau_i) = \exp\left(-\frac{6}{\tau_i^2}\left(t - t_i - \frac{\tau_i}{2}\right)^2\right), \tag{2}$$

where t_i represents the timing of the onset of the unsmoothed impact, and τ_i is the time scale of the contact force, which depends on the velocity of the impact and material properties of the object [28]. Sharper and faster contact points as well as harder materials will each contribute to reducing the value of τ_i, while softer, slower, and duller contact points will have higher τ_i values.

Equation 2 provides a model for an individual contact event. But in a natural audio clip, multiple such contact events, each with different characteristics, can occur over a given interval, such that a natural force profile over time will be the sum of multiple such contact impulses, each with different parameters and magnitudes, and each occurring at different instants during the interval. We model the force profile, the scalar value of the normal force over time, for multiple contacts as a weighted sum of such Gaussian impulses from Equation 2:

$$F(t;\mathbf{m},\mathbf{t},\boldsymbol{\tau}) = \mathbf{m}^T \mathbf{C}(t;\mathbf{t},\boldsymbol{\tau}), \tag{3}$$

where $\mathbf{t} = [t_0,...,t_M]^T$, $\boldsymbol{\tau} = [\tau_0,...,\tau_M]^T$ and $\mathbf{m} = [m_0,...,m_M]^T$. Each magnitude m_i serves as the weight for the corresponding impact. $\mathbf{C}(t;\mathbf{t},\boldsymbol{\tau})$ stacks the output of Equation 2 into a vector.

Equation 2 is naturally differentiable, but a challenge to making Equation 3 fully differentiable is that the number of impacts occurring during an interval may not be be known a priori, and may vary between examples. To address this, we propose an algorithm that finds M peaks in a time series of impact magnitude estimates over time, and places model-based impulses at those peaks based on the parameters at each peak's point in the time series. If there are fewer than M peaks, the impact magnitudes m_i at the extraneous peaks should be estimated as zero. We detail this algorithm in Appendix A.1.

To model the acceleration sound, we must know the acceleration profile of the striking tool's tip. Since the striking tool's mass is fixed, then the deceleration is proportional to the impulse force, which is already estimated in Equation 3. Therefore, we model the contribution of the acceleration sound as the force profile from Equation 3 multiplied by the learnable scalar variable α.

3.3 Object impulse response

For the object impulse response, we adopt a modal model by modeling objects as spring-damper systems, where the impulse response is a linear combination of exponentially decaying sinusoids, with each sinusoid having a different frequency f_k, gain g_k, and damping d_k [18, 23]. We model the K most salient modes of the object through constructing a finite impulse response (FIR) as $IR_o(t;\mathbf{f},\mathbf{g_o},\mathbf{d_o}) = \mathbf{g_o}^T[\exp(-\mathbf{d_o}t)\circ\sin(2\pi\mathbf{f}t)]$, where $\mathbf{f} = [f_0,\dots,f_K]$, $\mathbf{g_o} = [g_0,\dots,g_K]$, $\mathbf{d_o} = [d_0,...,d_K]$ and $\circ$ is the Hadamard product. Note that we make the simplifying assumption that the gains of each object are fixed across the entire body of the object, whereas in reality for general objects, they vary significantly depending on the location at which the object is impacted. We made this assumption to make the optimizations in our experiments more tractable since the inputs to our frameworks were blind to the 3D models of the objects as well as the locations of each contact. We detail how we scale normalized inputs to our module to produce the parameters in Appendix A.2.

3.4 Environment noise and reverberation

We approximate background and measurement noise as static filtered noise, whereas we approximate reverberation from the measurement environment with exponentially decaying filtered noise. For background and measurement noise, our filtered noise function $N(t;\mathbf{g_n})$ filters random white noise with time-constant gains for each frequency band given by learnable noise parameters: $N(t;\mathbf{g_n}) = \mathbf{g_n}^T filt(\mathcal{N}(t),B_n)$, where $filt(\mathcal{N}(t),B_n) \in \mathbb{R}^{B_n \times T}$ is a function that filters white noise $\mathcal{N}(t)$ with

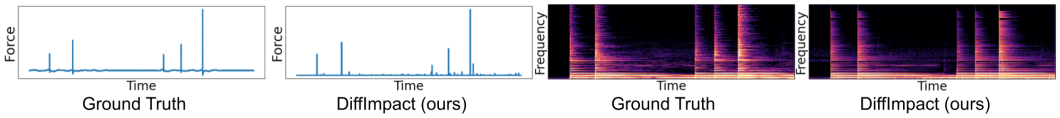


Figure 2: Comparing our model's estimates to ground truth time series of normalized contact forces and spectrograms, for our analysis by synthesis task on short recordings. Results are from the steel bowl.

B_n band pass filters spaced linearly over the interval of frequencies to the Nyquist frequency. The output matrix is multiplied with the vector $\mathbf{g_n}$ weighting each frequency band.

For generating reverberations, we model the local environment's response to sounds with an impulse response with exponentially decaying filtered noise, where each noise band has its own decay rate, similar to [23]: $IR_e(t_{IR};\mathbf{g_e},\mathbf{d_e}) = \mathbf{g_e}^T[\exp(-\mathbf{d_e}t)\circ filt(\mathcal{N}(t),B_e)]$. Here $\mathbf{g_e}$ and $\mathbf{d_e}$ are the gains and damping factors per frequency band, respectively, and have dimension B_e. We describe how these parameters are derived from scaling our learnable normalized reverb parameters in Appendix A.3.

3.5 Loss function

As a loss function for all tasks, we use the sum of the $\mathcal{L}_1$ distance between the spectrograms and log spectrograms of multiple window sizes from the ground truth W and synthesized audio $\hat{W}$ [15, 24]:

$$\mathcal{L}(W,\hat{W}) = \sum_{s_w \in \{128,\ldots,2048\}} \lambda_1 |S(W,s_w,\beta s_w) - S(\hat{W},s_w,\beta s_w)| + \lambda_2 |\log(S(W,s_w,\beta s_w)) - \log(S(\hat{W},s_w,\beta s_w))|,$$

where s_w is the window size, β is the overlap ratio, λ_1 and λ_2 are weights, and S is the short-term Fourier transform (STFT) or spectrogram. Using multiple spectrogram scales ensures that the loss characterizes an error signal in both high frequency and temporal resolution. The log spectrograms also ensure that higher frequency components, which tend to have lower energy magnitude than low frequency components, are also weighted more strongly. We simultaneously fit estimates of parameters for both the sound the object makes when impacted, and the impacts that produce those sounds.

4 Results

We first validate that DiffImpact is able to not only learn expressive sound models, but also to extract accurate physically interpretable parameters from controlled recordings of impacting different household objects. Then we show how our contributed modules can be used in two different real-world tasks, one showing how a robot could use our models to passively learn sound models from data, and another showing how our robot can interactively learn sound models and use them for useful downstream tasks.

4.1 Analysis by synthesis: Fitting to single real impact sounds

To test whether our DiffImpact can extract contact forces from sound, we collected 3-second clips of striking four different household objects (bottom right of Figure 4) in a recording studio with an impact hammer (PCB 086C01). The impact hammer had a force transducer in its tip, providing ground truth contact forces synchronized with the audio recorded by a microphone (PCB 378A06). We randomly initialized the parameters for our model, and fit them to each recording with respect to our loss function which only compares synthesized and ground truth spectrograms. We use Adam as the optimizer [29]. Our results are shown in Figure 2. Though our model's estimates of contact force have no sense of scale or calibration to a physical quantity (see Appendix E), we show that the force profiles our model synthesized were quite similar in shape to those of the ground truth. Note that we do not supervise directly on force profiles. Furthermore, the spectrograms demonstrate that our model is able to synthesize audio very similar to the ground truth. Audio examples are included in the Supplementary Materials, and further details of this experiment and its setup are in Appendix B.

4.2 End to end learning: Learning rigid body impact sound models from in-the-wild audio

We demonstrate the utility of our modules in the end to end learning task of autoencoding real world impact audio. Similar to previous work [16, 15], the input to each model is the ground truth magnitude spectrogram, and the output is the wave form of the audio. Our experiments in this section are driven by evaluating perceptual realism and similarity of our model's predictions, measured both qualitatively by human user studies and quantitatively by perceptual loss metrics.

Dataset We curated a dataset of audio clips extracted from videos on the ASMR Bakery YouTube Channel [1]. We selected clips from segments in which the creator is repeatedly tapping five objects

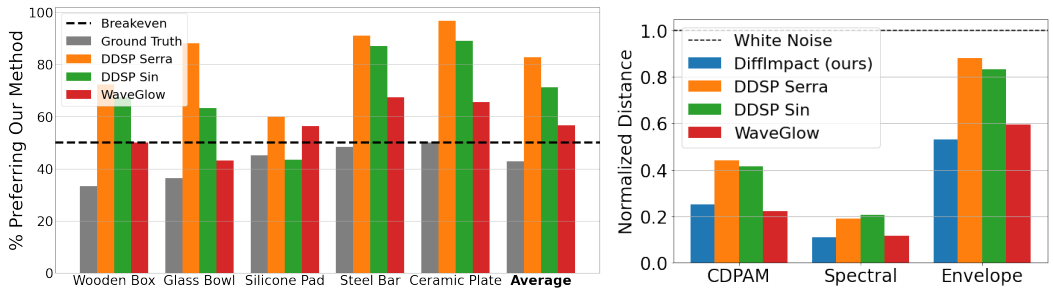


Figure 3: Results from end-to-end learning task evaluations on ASMR dataset. (**Left**) Human study ratings of realism of different ASMR objects' generated audio from test set. (**Right**) Average computational audio distance metrics of test set output from different models.

of different materials: a wooden box, a glass bowl, a silicone pad, a steel bar, and a ceramic plate. Screenshots from clips of each of the objects we selected are shown Appendix C.1. With this dataset, we show that a robot could use our framework to learn meaningful, physically interpretable models of rigid objects from a rich real world dataset. This dataset presents many challenges to the task of estimating a modal sound model, challenges which prior works have not addressed. First, whereas other works record the sound of a single controlled impact by a small pellet or specialized instrument [23, 18], many clips in our dataset include impacts randomly alternating between tapping with hard fingernails and soft fingertips within short timespans. Some of these objects are being held in the creator's hand while they are being tapped, and other objects are lying on a table (see Appendix C.1). Holding each object or resting it on a surface will dampen certain modes of vibration. Therefore, at no point in the entire dataset does an object emit an ideal undamped modal response. The ASMR dataset consists of around 3-5 minutes of continuously recorded audio of tapping for each object. We validated our model and its hyperparameters on a held out validation set of the steel bar clips, then split remaining data for each object in approximately a 90-10 train-test split. Thus, each model had 200 seconds of training data for each object, and at least 20 seconds of unseen audio samples per object.

Baselines We implemented two baseline models based on DDSP [15, 24], which we refer to as "DDSP Serra" and "DDSP Sin." DDSP Serra is based on a time-varying harmonic model developed for musical instruments [30]. To test the influence of the harmonic model bias, DDSP Sin replaces the harmonic oscillator with an unrestricted time-varying sinusoidal oscillator. We also tested WaveGlow [16], a state of the art model-free approach for generating human speech. Further details on baselines are in Appendix C.2. The WaveGlow baseline took more than two orders of magnitude more GPU time to train ($\sim$1 hour vs. $\sim$ 300 hours of training time for a single object on an Nvidia Quadro P5000 GPU). To demonstrate the necessity of this training time, we report results from training this model for only $\sim$1 order of magnitude more than the other models in Appendix C.4. For our evaluations, we trained a separate instance of each baseline and our model per object.

Qualitative evaluation through human studies To test DiffImpact's capability of producing realistic impact sounds on held-out audio samples of each object compared to baselines, we re-dubbed the original videos with each model's output given the video's original audio as input. We then presented participants on Amazon Mechanical Turk with a two-alternative forced choice to pick the best matching audio for 5-second videos. The comparisons were between our model and either the baselines or ground truth (1200 questions total). Our model significantly outperforms all the baselines on average ($p<0.05$), as shown on the left of Figure 3. Per object category, DiffImpact outperforms all baselines for the steel bar and ceramic plate. It outperforms DDSP Serra and DDSP Sin models for all objects except for the silicone pad ($p<0.05$). We suspect our model struggles with modeling the impact sound of the silicone pad due to the damping effect and roughness of the material as well as its high deformability that violates our model's rigid body assumption. The other models without these biases may better model its sound. Furthermore, the user's hand sometimes rubs against the surface during consequent tapping, resulting in unmodeled components in the audio which affect the performance of our method. On average, our participants chose our model's audio over ground truth 42.9% of the time, suggesting that our generated audio is often realistic enough to be confused with ground truth audio.

Quantitative evaluation using distance metrics We also compared each model's test outputs with the ground truth audio using different computational distance metrics, including CDPAM, a learning-based perceptual distance metric designed for human speech applications [31], the multi-scale spectrogram loss used in [15, 24], and the envelope distance used in [32]. Since each of these metrics have

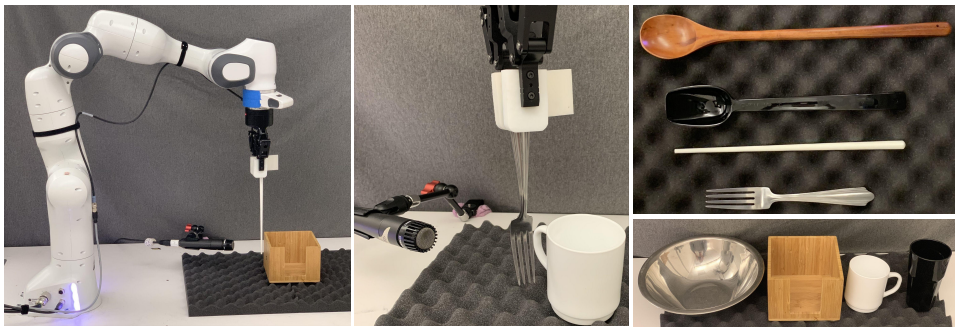


Figure 4: Physical setup and tools for robotic impact dataset collection. **(Left)** A Franka Panda holds and rotates a ceramic chopstick to swing at the wood napkin holder, with audio recorded by the bottom-left microphone. **(Middle)** Close up of a fork swung at a mug, with microphone at left. Objects were rested on a soft foam pad to prevent sliding and mitigate contact damping and acoustic reflection from the hard tabletop. **(Right top)** The tools used for striking (from top): wooden spoon, polycarbonate spoon, ceramic chopstick, and steel fork. **(Right bottom)** The struck objects (from left): steel bowl, wood napkin holder, ceramic mug, and polycarbonate cup.

widely different scales, we normalized each model's average distance on each metric by the average distance of white noise with respect to that metric. The results are shown on the right of Figure 3. These results align with our human studies, suggesting that our DiffImpact outperforms DDSP Serra and Sin, and it performs on par with WaveGlow, yet requires two orders of magnitude less training time.

Discussion We demonstrate that DiffImpact is expressive enough to characterize and generate real world impact sounds, with performance competitive or better than state of the art alternatives. But perhaps most importantly, while completing this end-to-end learning task, our model is forced to extract physically interpretable and transferable parameters of impact forces and acoustic properties of the object and environment, whereas the parameters learned by other models are either uninterpretable (in the case of WaveGlow), or not directly meaningful to extracting impact event and intrinsic acoustic properties of objects. These parameters can be used with a novel impact profile to render the sound the object would make when impacted as such, creating a multimodal object model for simulation environments [33, 34] merely from information learned from in-the-wild YouTube videos.

4.3 Robotic task: Source separation of kitchen utensil sounds

We use DiffImpact to perform a source separation task. Unlike traditional audio source separation, where the goal is to isolate sounds in passive, pre-recorded audios and videos [35, 36, 37], our goal is to enable robots to actively interact with objects in the environment and separate the sounds they make. Consider the scenario where the robot strikes an object with a tool. The perceived sound of the impact will be a combination of the sounds the tool makes and the sounds the struck object makes. We show that by separating them, a robot can more accurately classify object materials, as different materials have distinct acoustic properties. Knowing the material of an object can inform the robot about its durability, surface friction, and category, all key factors to consider when planning grasps and actions.

Dataset We collected a dataset of a robot swinging four household tools ("tools") at each of four household objects ("objects") of four different materials, with each distinct material represented in exactly one element of each category. We recorded each impact with a Shure SM57 microphone. Photos of the robotic setup and the tools and objects are shown in Figure 4. We filtered out any samples where the recording was too loud and clipped, which would obfuscate the frequency content, then used the two loudest remaining recordings of each pairwise interaction for our experiments, to ensure a high signal to noise ratio.

Optimization setup We used the yellow elements of the model shown in Figure 1. Compared to the description in Section 3, we modified the model as follows: 1) instead of using the neural encoder to control impact force, the impact force profile for each instance was a single impulse parameterized by the Impulse trainable parameters, and 2) the Tool parameters were also used to produce a modal impulse response which was convolved with the force profile along with the Object modal impulse response. The parameters for each tool and each object were optimized with respect to each of the recordings for which they were present. All recordings shared the same set of Noise parameters, with each having a single gain scalar for the noise. We then performed analysis by synthesis on the entire batch of 32 recordings at once, optimizing all of our trainable variables with the Adam optimizer with respect to the loss that only compares ground truth with generated spectrograms of the non-separated

sound. For producing the separated object sound, we turned off the background noise, reverberation, acceleration sound, and the tool impulse response, then generated the model output for each recording. For producing the separated tool sound, we turned off the background noise and the object impulse response, then generated the model output for each recording. See Appendix D for more details.

Baselines We compared with non-negative matrix factorization (NMF) clustering [38] and the more recent "Deep Audio Prior" (DAP) [39] networks. Similar to our framework, both baselines require no pre-training. Note that these baselines do not have a built-in way of specifying which separated source corresponds to the tool and which to the object, whereas ours does.

Metrics We provide qualitative examples of the separated sources in our Supplementary Materials. Ground truth separated object sounds are not available to compare our results against, as there is no realistic way to even collect them in the real world. Thus, for a quantitative comparison, we compare the performance of material classification on each framework's outputs. Specifically, we train an audio-based material classifier on balanced data of only the relevant materials from the Sound-20K dataset, which consists of only *synthetic* sounds of objects of different materials bouncing on surfaces in a virtual environment [40]. We tested this trained classifier on the original sound as well as the source-separated output of each framework, to test whether the separated audio can be more accurately classified by this trained material classifier.

Results We report results in Table 1. Because our baselines have no constraint on which separated source is the tool and which is the object, we show the results of two evaluation schemes. Under the "Dataset Max" scheme, we compute the object material accuracy twice for each baseline: first treating their first output as the object, then treating the second as the object. We use the higher between the two as their result. We do the same for tools. Under the "Inst. Max" scheme, we take the maximum for each instance instead of across the whole dataset. While both schemes offer the baselines an advantage, our DiffImpact still outperformed them, while also explicitly differentiating between the object and the tool. All models, including ours, performed better for classifying objects than tools. Object sounds dominated the recordings over tool sounds, since they were much larger, with more surface area for dispersing vibrations. Their size also gave them stronger low frequency modes with lower dampings, lasting longer in recordings, and thus easier to capture and characterize. See Appendix D for confusion matrices from this task, as well as results from a similar experiment where our DiffImpact outperforms the same baselines for estimating Young's modulus of each object and tool from separated sounds.

Table 1: Classifying materials from source-separated audio.

	Object Material Acc. (%)		Tool Material Acc. (%)	
Model	Dataset Max	Inst. Max	Dataset Max	Inst. Max
Raw audio	37.5	37.5	12.5	12.5
NMF [38]	40.6	43.8	21.9	21.9
DAP [39]	50.0	68.8	28.1	50.0
DiffImpact (ours)	**90.6**	**96.9**	**43.8**	**53.1**

5 Conclusion

We have proposed DiffImpact, a fully differentiable physics-based framework for modeling rigid object impact sounds. This model enables robots to learn valuable physically interpretable parameters from sounds of objects in their everyday environments. We first validated that our method successfully simulates impact sounds of different everyday objects, and that it can be used in an analysis-by-synthesis approach to estimate impact forces that generated the analysed sound. We also showed how our framework can be used as a neural rendering layer, learning physically interpretable parameters of object sound models during an end-to-end learning task from a real-world dataset. We evaluated its performance against state of the art audio learning baselines with both a human study and distance metrics, and found our model to be more expressive and resource efficient than alternatives for modeling real-world impact sounds. Finally, we showed how a robot can use our model to learn sound models of everyday objects in its environment by striking them in a noisy environment with everyday tools. The robot used our model to differentiate between the sounds of the tools it used and the objects it struck with them and could classify the materials of the objects it strikes from sound 140% more accurately than from the original, unseparated audio or from generic audio source separation baselines. The knowledge robots can use our model to ascertain about objects, materials, and contact forces will be valuable to future applications which use our approach in robot decision making and policy learning. As future work, we aim to expand the scope of our model to other contact events, such as rubbing and scratching, or other object acoustic events, such as bouncing on different surfaces or fracturing. We also intend to use our contributions as a foundation for models and applications which use other multimodal inputs, such as vision and tactile data.

Acknowledgments

We thank Christopher Atkeson, Ante Qu, Oliver Kroemer, Jacky Liang, and Leonid Keselman for valuable discussions, Kevin Zhang, Michael A. Lin, Hojung Choi, Brian Roberts, and Toki Migimatsu for assistance in setting up robotic experiments, and Matt Wright for assisting with audio recordings. This work is in part supported by the Toyota Research Institute (TRI), NSF CCRI #2120095, Amazon Research Award (ARA), Autodesk, IBM, and the Stanford Institute for Human-Centered AI (HAI). N. Heravi was supported by the NSF Graduate Research Fellowship.

References

[1] Asmr bakery. URL https://www.youtube.com/channel/UCDH70xAv1bBeaCt5Cc7a96A.

[2] J. Sinapov, C. Schenck, K. Staley, V. Sukhoy, and A. Stoytchev. Grounding semantic categories in behavioral interactions: Experiments with 100 objects. *Robotics and Autonomous Systems*, 62(5):632–645, 2014. ISSN 0921-8890. doi:https://doi.org/10.1016/j.robot.2012.10.007. URL https://www.sciencedirect.com/science/article/pii/S092188901200190X. Special Issue Semantic Perception, Mapping and Exploration.

[3] J. Sinapov, M. Wiemer, and A. Stoytchev. Interactive learning of the acoustic properties of household objects. In *2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12-17, 2009*, pages 2518–2524. IEEE, 2009. doi:10.1109/ROBOT.2009.5152802. URL https://doi.org/10.1109/ROBOT.2009.5152802.

[4] D. Gandhi, A. Gupta, and L. Pinto. Swoosh! rattle! thump!–actions that sound. *arXiv preprint arXiv:2007.01851*, 2020.

[5] J. Christie and N. Kottege. Acoustics based terrain classification for legged robots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3596–3603, 2016. doi:10.1109/ICRA.2016.7487543.

[6] S. Clarke, T. Rhodes, C. G. Atkeson, and O. Kroemer. Learning audio feedback for estimating amount and flow of granular material. *Proceedings of Machine Learning Research*, 87, 2018.

[7] H. Liang, S. Li, X. Ma, N. Hendrich, T. Gerkmann, F. Sun, and J. Zhang. Making sense of audio vibration for liquid height estimation in robotic pouring. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[8] C. Matl, Y. S. Narang, D. Fox, R. Bajcsy, and F. Ramos. Stressd: Sim-to-real from sound for stochastic dynamics. *CoRR*, abs/2011.03136, 2020. URL https://arxiv.org/abs/2011.03136.

[9] T. N. Sainath and C. Parada. Convolutional neural networks for small-footprint keyword spotting. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[10] H. Lee, P. Pham, Y. Largman, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009.

[11] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency, 2017.

[12] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 459–468, 2018.

[13] M. Bao, M. Cong, S. Grabli, and R. Fedkiw. High-quality face capture using anatomical muscles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10802–10811, 2019.

[14] A. Palazzi, L. Bergamini, S. Calderara, and R. Cucchiara. End-to-end 6-dof object pose estimation through differentiable rasterization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

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

[16] R. Prenger, R. Valle, and B. Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE, 2019.

[17] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[18] D. K. Pai, K. v. d. Doel, D. L. James, J. Lang, J. E. Lloyd, J. L. Richmond, and S. H. Yau. Scanning physical interaction behavior of 3d objects. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 87–96, 2001.

[19] K. Van Den Doel, P. G. Kry, and D. K. Pai. Foleyautomatic: physically-based sound effects for interactive simulation and animation. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 537–544, 2001.

[20] M. Rau, O. Das, and E. Canfield-Dafilou. Improved carillon synthesis. *in 22nd Int. Conf. Digital Audio Effects (DAFx-19), Birmingham, England*, Sep. 2–6, 2019.

[21] Z. Ren, H. Yeh, and M. C. Lin. Example-guided physically based modal sound synthesis. *ACM Transactions on Graphics (TOG)*, 32(1):1–16, 2013.

[22] Z. Zhang, Q. Li, Z. Huang, J. Wu, J. Tenenbaum, and B. Freeman. Shape and material from sound. In *Advances in Neural Information Processing Systems*, pages 1278–1288, 2017.

[23] J. Traer, M. Cusimano, and J. H. McDermott. A perceptually inspired generative model of rigid-body contact sounds. In *The 22nd International Conference on Digital Audio Effects (DAFx-19)*, 2019.

[24] J. Engel, R. Swavely, L. H. Hantrakul, A. Roberts, and C. Hawthorne. Self-supervised pitch detection by inverse audio synthesis. 2020.

[25] J. N. Chadwick, C. Zheng, and D. L. James. Precomputed acceleration noise for improved rigid-body sound. *ACM Transactions on Graphics (TOG)*, 31(4):1–9, 2012.

[26] J. Reed. Energy losses due to elastic wave propagation during an elastic impact. *Journal of Physics D: Applied Physics*, 18(12):2329, 1985.

[27] D. L. James. Real-time rigid-body dynamics with acceleration noise. `http://graphics.stanford.edu/courses/cs448z/HW_Sp21/HW1/CS448Z_2021sp_HW1_RBD_AccelNoise.pdf`, 2021.

[28] K. L. Johnson and K. L. Johnson. *Contact mechanics*. Cambridge university press, 1987.

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

[30] X. Serra and J. Smith. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, 1990.

[31] P. Manocha, Z. Jin, R. Zhang, and A. Finkelstein. Cdpam: Contrastive learning for perceptual audio similarity, 2021.

[32] P. Morgado, N. Vasconcelos, T. Langlois, and O. Wang. Self-supervised generation of spatial audio for 360° video. In *NeurIPS*, 2018.

[33] C. Gan, J. Schwartz, S. Alter, M. Schrimpf, J. Traer, J. De Freitas, J. Kubilius, A. Bhandwaldar, N. Haber, M. Sano, et al. Threedworld: A platform for interactive multi-modal physical simulation. *arXiv preprint arXiv:2007.04954*, 2020.

[34] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[35] R. Gao, R. Feris, and K. Grauman. Learning to separate object sounds by watching unlabeled video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 35–53, 2018.

[36] H. Zhao, C. Gan, A. Rouditchenko, C. Vondrick, J. McDermott, and A. Torralba. The sound of pixels. In *Proceedings of the European conference on computer vision (ECCV)*, pages 570–586, 2018.

[37] A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, and M. Rubinstein. Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation. *arXiv preprint arXiv:1804.03619*, 2018.

[38] M. Spiertz and V. Gnann. Source-filter based clustering for monaural blind source separation. In *Proceedings of the 12th International Conference on Digital Audio Effects*, volume 4, 2009.

[39] Y. Tian, C. Xu, and D. Li. Deep audio prior. *arXiv preprint arXiv:1912.10292*, 2019.

[40] Z. Zhang, J. Wu, Q. Li, Z. Huang, J. Traer, J. H. McDermott, J. B. Tenenbaum, and W. T. Freeman. Generative modeling of audible shapes for object perception. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[41] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, A. Ragni, V. Valtchev, P. Woodland, and C. Zhang. *The HTK Book (version 3.5a)*. 12 2015.

[42] B. C. Moore and B. R. Glasberg. A revision of zwicker's loudness model. *Acta Acustica united with Acustica*, 82(2):335–345, 1996.

[43] J. M. Ide. Comparison of statically and dynamically determined young's modulus of rocks. *Proceedings of the National Academy of Sciences of the United States of America*, 22(2):81, 1936.

[44] E. J. Chen, J. Novakofski, W. K. Jenkins, and W. D. O'Brien. Young's modulus measurements of soft tissues with application to elasticity imaging. *IEEE Transactions on ultrasonics, ferroelectrics, and frequency control*, 43(1):191–194, 1996.

[45] J. Wang, W. Li, C. Lan, and P. Wei. Effective determination of young's modulus and poisson's ratio of metal using piezoelectric ring and electromechanical impedance technique: A proof-of-concept study. *Sensors and Actuators A: Physical*, 319:112561, 2021.

[46] B. Frank, R. Schmedding, C. Stachniss, M. Teschner, and W. Burgard. Learning the elasticity parameters of deformable objects with a manipulation robot. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1877–1883. IEEE, 2010.

[47] P. Boonvisut and M. C. Çavuşoğlu. Estimation of soft tissue mechanical parameters from robotic manipulation data. *IEEE/ASME Transactions on Mechatronics*, 18(5):1602–1611, 2012.

[48] A. Davis, K. L. Bouman, J. G. Chen, M. Rubinstein, F. Durand, and W. T. Freeman. Visual vibrometry: Estimating material properties from small motion in video. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 5335–5343, 2015.

[49] D. L. James, J. Barbič, and D. K. Pai. Precomputed acoustic transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Trans. Graph.*, 25(3): 987–995, July 2006. ISSN 0730-0301. doi:10.1145/1141911.1141983. URL `https://doi.org/10.1145/1141911.1141983`.

[50] W. v. Schaik, M. Grooten, T. Wernaart, and C. v. d. Geld. High accuracy acoustic relative humidity measurement induct flow with air. *Sensors*, 10(8):7421–7433, 2010.

[51] C. Zheng and D. L. James. Toward high-quality modal contact sound. In *ACM SIGGRAPH 2011 papers*, pages 1–12. 2011.

[52] V. Valimaki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel. Fifty years of artificial reverberation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(5):1421–1448, 2012.

[53] J. S. Abel, S. Coffin, and K. Spratt. A modal architecture for artificial reverberation with application to room acoustics modeling. In *Audio Engineering Society Convention 137*. Audio Engineering Society, 2014.

Appendix A Method Details

A.1 Differentiable impact force profile

ALGORITHM 1: Differentiable Impact Force Profile Computation

Input: Normalized magnitude estimate time series vector $\mathbf{m_{in}} \in \mathbb{R}^{T_{in}}$, normalized contact time scale estimate time series vector $\boldsymbol{\tau_{in}} \in \mathbb{R}^{T_{in}}$, final number of time samples T_{out}, and expected number of peaks M

Output: A predicted impulse profile over time, with T_{out} samples sampled at rate f_s Hz

$\mathbf{m_{out}} \leftarrow MagScaleFn(\mathbf{m_{in}})$;
$\boldsymbol{\tau_{out}} \leftarrow TauScaleFn(\boldsymbol{\tau_{in}})$;
$\mathbf{m_{out}} \leftarrow HannWindowResample(\mathbf{m_{out}}, T_{out})$;
$\boldsymbol{\tau_{out}} \leftarrow HannWindowResample(\boldsymbol{\tau_{out}}, T_{out})$;
$w \leftarrow \lfloor T_{out}/M \rfloor$;
$\mathbf{Ind} \leftarrow ArgmaxPool(\mathbf{m_{out}}, Kernel = w, Stride = w)$;
$\mathbf{t} \leftarrow ([\mathbf{Ind}-\delta] \circ \mathbf{m_{out}}[\mathbf{Ind}-\delta] + \mathbf{Ind} \circ \mathbf{m_{out}}[\mathbf{Ind}] + [\mathbf{Ind}+\delta] \circ \mathbf{m_{out}}[\mathbf{Ind}+\delta])/3f_s$;
return $F([0, 1/f_s, ..., T_{out}/f_s]; \mathbf{m_{out}}[Ind], \mathbf{t}, \boldsymbol{\tau_{out}}[Ind])$

Here, we describe the details of the algorithm we use in Section 3.2 for generating a physics-based series of impact impulses at audio sample rate over time, using as inputs two coarsely temporally discretized and normalized time series. These inputs are the outputs of an encoder in the case of our End-to-End experiment, or learnable time series parameter vectors in the case of our Analysis-by-Synthesis experiment. We will describe how we convert these normalized inputs into inputs for Equation 3 in a fully differentiable manner.

To model Equation 3, we must choose an integer value for M in the equation, corresponding to the number of impacts occurring during the interval. To do this, we fix M as a hyperparameter based on a reasonably tight upper bound of our expectation of the overall frequency of impacts, with the impacts spaced somewhat regularly across the interval. Because M is intentionally chosen to be an *overestimate* of the number of impacts occurring during most time intervals, our algorithm for Equation 3 will frequently instantiate more impacts than actually occurred in an interval. Impacts that have been placed in subintervals where no impact occurred should ideally approach magnitude $m_i = 0$.

Our differentiable impact profile algorithm, with pseudocode shown in Algorithm 1, takes a coarsely discretized time series of network outputs or variables corresponding to continuous normalized input parameters for estimating magnitude and τ values for M different impact impulses, each modeled by Equation 2. We pass the input parameters for the magnitude and the τ values through respective scaling functions, $MagScaleFn$ and $TauScaleFn$ in Algorithm 1. $MagScaleFn$ and $TauScaleFn$ are both exponential sigmoid functions. Each vector has T_{in} temporal elements, but with T_{out} being the number of samples in the final output audio, at an audio sample rate f_s Hz, the input time series are coarsely discretized such that $T_{in} << T_{out}$. Given an input vector of a time series of magnitudes $\mathbf{m_{in}} = [m_{in,0}, ..., m_{in,T_in}]$ with T_{in} temporal elements, $MagScaleFn$ bounds the magnitude estimates from 0 to 2:

$$MagScaleFn_i(m_{in,i}) = u_{mag} * sigmoid^{\lambda_{mag}}(m_{in,i} + b_{mag}) + l_{mag} \tag{4}$$

$$\mathbf{m_{out}} = \mathbf{MagScaleFn}(\mathbf{m_{in}}), \tag{5}$$

where b_{mag} is a hyperparameter bias for controlling initialization conditions, hyperparameters u_{mag} and l_{mag} control the upper and lower bounds respectively of the scaling function, and $\mathbf{MagScaleFn}$ stacks the output of Equation 4 into a vector. We set hyperparameter l_{mag} to a very small constant (10^{-7}), ensuring the scaled magnitude never fully collapses to 0 during optimization. The exponent λ_{mag} on the exponential sigmoid function is a hyperparameter controlling the slope of the function about $x = 0$.

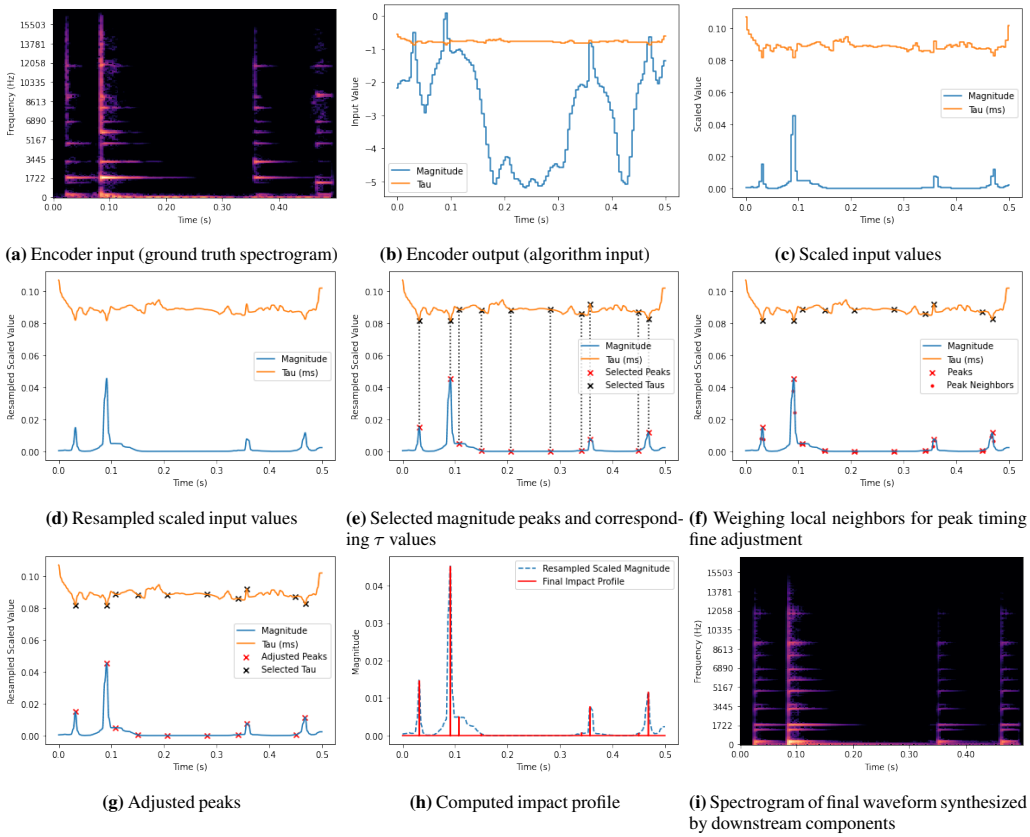
Given a similar input vector for the τ values, $\boldsymbol{\tau_{in}} = [\tau_{in,0}, ..., \tau_{in,T_in}]$, we produce the $\boldsymbol{\tau_{out}}$ scaled vector as follows:

$$TauScaleFn_i(\tau_{in,i}) = u_\tau * sigmoid^{\lambda_\tau}(\tau_{in,i} + b_\tau) + l_\tau / f_s \tag{6}$$

$$\boldsymbol{\tau_{out}} = \mathbf{TauScaleFn}(\boldsymbol{\tau_{in}}) \tag{7}$$

where f_s is the audio sample rate, b_τ is a bias learned for each material, hyperparameter λ_τ controls the slope on the sigmoid, hyperparameters u_τ and b_τ control the respective upper and lower bounds

of the scaling function, and **TauScaleFn** stacks the output of Equation 6 into a vector. We have the learnable bias b_τ because each material is likely to have a different distribution of τ, depending on softness of the object's material or contact surface. Harder materials should have smaller τ values, and softer materials should have larger τ values. The upper bound u_τ on this scaling function ensure that contact time scales τ remain within physically reasonable ranges (below 3ms for our experiments), and the lower bound l_τ ensures that the resulting impulse force profile's curve shape does not degrade too much as the width of the curve approaches the sample rate.



(a) Encoder input (ground truth spectrogram) **(b)** Encoder output (algorithm input) **(c)** Scaled input values

(d) Resampled scaled input values **(e)** Selected magnitude peaks and corresponding τ values **(f)** Weighing local neighbors for peak timing fine adjustment

(g) Adjusted peaks **(h)** Computed impact profile **(i)** Spectrogram of final waveform synthesized by downstream components

Figure 5: Steps of the Differentiable Impact Force Profile algorithm

We next resample both time series to the resolution of the audio sample rate (*e.g.*, 44100 Hz) using Hann windows. We select peaks in the resampled magnitude time series using an $ArgmaxPool$ operation, with window sizes based on the length of the time interval divided by M. Since selecting the peaks with $ArgmaxPool$ can lose any gradient with respect to the timing of the peaks, to recapture this gradient, we next adjust the selected peak timings by taking a magnitude-weighted average of the magnitude at the peak timings and a preceding and a succeeding neighboring magnitude of each selected peak. Finally, we select the magnitude and τ values from the time instant of each peak in the series. Through these differentiable steps, we now have the magnitude vector $\mathbf{m}$, the timing vector $\mathbf{t}$, and contact time scale vector $\boldsymbol{\tau}$, providing us with all the input parameters for Equation 3, producing a fully differentiable impact impulse profile with M peaks over the time interval. For an example walkthrough of each of these steps, see Figure 5.

A.2 Modal impulse response

For our modal impulse response, we will explain how we take normalized input parameters and convert them into input parameters of our modal impulse response equation from Section 3.3:

$$IR_o(t;\mathbf{f},\mathbf{g_o},\mathbf{d_o}) = \mathbf{g_o}^T[\exp(-\mathbf{d_o}t)\circ\sin(2\pi\mathbf{f}t)], \tag{8}$$

where $\mathbf{f} = [f_0,...,f_K]$, $\mathbf{g_o} = [g_0,...,g_K]$, $\mathbf{d_o} = [d_0,...,d_K]$ and $\circ$ is the Hadamard product.

We begin with vectors $\mathbf{a_{in}} = [a_0, ..., a_K]$, $\mathbf{b_{in}} = [b_0, ..., b_K]$, and $\mathbf{f_{in}} = [f_{in,0}, ..., f_{in,K}]$ which are normalized (unscaled) input vectors corresponding to the gains $\mathbf{g_o}$, dampings $\mathbf{d_o}$, and frequencies $\mathbf{f}$, respectively. For scaling input vector $\mathbf{a_{in}}$ into $\mathbf{g_o}$, we use a softmax scaling function :

$$ModalGainScaleFn_i(a_i) = \frac{\exp(a_i)}{\sum_j \exp(a_j)} \tag{9}$$

$$\mathbf{g_o} = \mathbf{ModalGainScaleFn}(\mathbf{a_{in}}) \tag{10}$$

where $\mathbf{ModalGainScaleFn}$ stacks the output of Equation 9 into a vector. This softmax formulation maintains a normalized distribution of relative gains of each frequency component in the modal impulse response, ensuring that the overall magnitude of the impulse response is not a free parameter during optimization, but rather is completely deferred to the impact impulse force profile.

To convert the input vector $\mathbf{b_{in}}$ into the damping vector $\mathbf{d_o}$, we use an exponential sigmoid function:

$$DampingScaleFn_i(b_i) = u_{damp} * sigmoid^{\lambda_{damp}}(b_i + b_{damping}) + l_{damp} \tag{11}$$

$$\mathbf{d_o} = \mathbf{DampingScaleFn}(\mathbf{b_{in}}) \tag{12}$$

where $b_{damping}$ is a hyperparameter bias, hyperparameter λ_{damp} controls the slope of the sigmoid, u_{damp} and l_{damp} are hyperparemeters for the respective upper and lower bounds of the function, and $\mathbf{DampingScaleFn}$ stacks the output of Equation 11 into a vector. We set the value of the upper bound u_{damp} in Equation 11 to 10000 to ensure that the damping values do not cause the contribution of their corresponding mode to disappear from the final impulse response at the audio sample rate.

Finally, for scaling input vector $\mathbf{f_{in}}$ into frequencies vector $\mathbf{f}$, we use a scaling function from the authors of [24], where input parameters are used to fine-tune a somewhat even distribution of frequencies over a range of frequencies (*e.g.* the range of human audible frequencies from ∼20Hz - 20kKhz), with respect to a Mel scale for human audio perception. First, we will define some equations important to the Mel scale [41]:

$$MelToHz(x) = 700 * (10.0^{(x/2595.0)} - 1.0) \tag{13}$$

$$HzToMel(x) = 2595 * \log_{10}(1.0 + x/700.0), \tag{14}$$

And the equivalent rectangular bandwidth (ERB) equation attempts to approximate theoretical frequency bandwidth filters in human hearing [42]:

$$HzToERB(x) = 0.108x + 24.7, \tag{15}$$

Now using these equations, we scale $\mathbf{f_{in}}$ into $\mathbf{f}$ as so:

$$CenterFreq(i) = (1 - \frac{i}{K-1}) * HzToMel(f_{min}) + \frac{i}{K-1} * HzToMel(f_{max}) \tag{16}$$

$$FrequencyScaleFn_i(f_{in,i}) = CenterFreq(i) + \tanh(f_{in,i}) * HzToERB(CenterFreq(i)) \tag{17}$$

$$\mathbf{f} = \mathbf{FrequencyScaleFn}(\mathbf{f_{in}}) \tag{18}$$

where f_{min} and f_{max} are hyperparameters corresponding to the lower and upper bound of the frequency range, respectively. $\mathbf{FrequencyScaleFn}$ stacks the output of Equation 18 into a vector.

A.3 Reverb

Here we describe how we scale normalized input parameter vectors into parameters for our room impulse response described in Section 3.4, which we use to model reverberation effects,

$$IR_e(t_{IR}; \mathbf{g_e}, \mathbf{d_e}) = \mathbf{g_e}^T [\exp(-\mathbf{d_e} t) \circ filt(\mathcal{N}(t), B_e)], \tag{19}$$

where $\mathbf{g_e}$ and $\mathbf{d_e}$ are the gains and damping factors per frequency band, respectively, and have each have dimension B_e.

Given input vector $\mathbf{g_{in}} = [g_{in,0}, ..., g_{in,B_e}]$, we produce $\mathbf{g_e} = [g_{e,0}, ..., g_{e,B_e}]$ using a similar scaling function as that from Equation 5:

$$ReverbGainScaleFn_i(g_{in,i}) = 2.0 * sigmoid^{\log(10)}(m_{in,i} + b_{reverb_gain}) + 10^{-7} \tag{20}$$

$$\mathbf{g_e} = \mathbf{ReverbGainScaleFn}(\mathbf{g_{in}}), \tag{21}$$

where b_{reverb_gain} is a hyperparameter bias, and $\mathbf{ReverbGainScaleFn}$ stacks the output of Equation 20 into a vector.

For the dampings, we scale input vector $\mathbf{d_{in}} = [d_{in,0},...,d_{in,B_e}]$ to produce $\mathbf{d_e} = [d_{e,0},...,d_{e,B_e}]$.

$$ReverbDampingScaleFn_i(d_{in,i}) = 2.0 + \exp(d_{in,i} + b_{decay}) \tag{22}$$

$$\mathbf{d_e} = \mathbf{ReverbDampingScaleFn}(d_{in}) \tag{23}$$

where b_{decay} is a hyperparameter bias, and $\mathbf{ReverbDampingScaleFn}$ stacks the output of Equation 22 into a vector.

With these differentiable scaling functions, we convert input parameter vectors $\mathbf{g_{in}}$ and $\mathbf{d_{in}}$ into parameter vectors $\mathbf{g_e}$ and $\mathbf{d_e}$, respectively, to be used as the inputs for Equation 19.

Appendix B Analysis by Synthesis Experiment Details

Here we describe some further details about the data collection and methods used for our experiment in Section 4.1, then show some additional results.

Figure 6: Physical setup for Analysis by Synthesis dataset collection. A microphone at the top left records audio from the suspended ceramic mug being struck by the impact hammer on the right.

Dataset To collect our dataset, we first suspended each of the four everyday objects shown at the bottom right of Figure 4 on a string to minimize the effects of contact dampings on each object's vibrations. We then struck each of them with a PCB 086C01 impact hammer with a force transducer and a hard nylon tip. We took synchronized recordings of audio with a PCB 378A06 microphone and this impact hammer while striking the objects repeatedly in roughly the same place. We adjusted gains on the microphone to maximize volume but prevent clipping for each object. We collected at least 60 seconds of data per object, then reviewed each object's data to find a 3 second clip for each object where the audio did not clip, and the impact forces were frequent and varied enough to be interesting to try to capture. Our data collection setup is shown in Figure 6.

Optimization Setup Our optimization used all the green elements in Figure 1, with the nuance that the inputs to the *Impact Force Profile Generator*, instead of coming from the encoder, were two variable vectors for the two normalized time series input parameters $\mathbf{m_{in}}$ and $\boldsymbol{\tau_{in}}$ required by the scaling functions of the generator (see Equations 5 and 7 in Appendix A.1). For this experiment, in order to mitigate the risk of falling into local minima early in the optimization, we also added zero-mean Gaussian noise to the magnitude variable vector. We controlled the standard deviation of the noise with a learnable variable vector representing a time series of standard deviations, with the same temporal resolution as the magnitude time series variable vector and all elements initialized to 0.1. We randomly initialized all other parameters including these two variable vectors, then optimized our loss function comparing the output of our model with the ground truth original audio with respect to all parameters, using the Adam optimizer [29]. To be clear, we did not provide our model with supervision for the impact force profile.

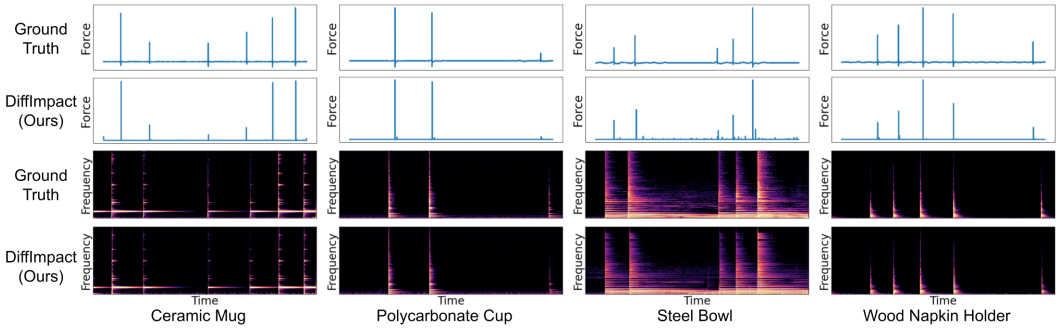


Figure 7: Comparing our model's estimates to ground truth time series of normalized contact forces and spectrograms, for our analysis by synthesis task on short recordings of each everyday object.

Full Results The results of using our model to fit parameters to each of the four objects are shown in Figure 7, and we include examples in our Supplementary Video for qualitative comparison. As we can see in Figure 7, each spectrogram demonstrates that the audio our model is able to produce for each object is very similar to its ground truth recorded audio. Furthermore, for each of the objects, our model fits an impact force profile which will most effectively produce audio approximating the ground truth recording, and as we can see, the timings and relative magnitudes of each of the impact impulses that our model has derived during the optimization are remarkably similar to the ground truth hammer force transducer recordings. This validates our hypothesis that our model is able to both generate realistic impact audio and extract physically interpretable parameters along the way, through this unsupervised analysis-by-synthesis process.

Appendix C End-to-End Learning Experiment Details

C.1 ASMR Bakery Dataset

(a) Wooden box **(b)** Glass bowl **(c)** Silicone pad

(d) Steel bar **(e)** Ceramic plate

Figure 8: Objects selected for our dataset from the ASMR Bakery YouTube channel

In searching for in-the-wild audio recordings of impacts, we discovered the ASMR Bakery YouTube Channel [1]. At time of writing, the channel features hundreds of YouTube videos of the creator

manipulating different everyday objects without speaking, providing an "unpolluted" dataset of manipulation sounds. We narrowed our search to videos with titles including "tapping", and used the annotations on each video to find interesting objects to select for our dataset. For the sake of our experiments, we assumed that each of the clips that were labeled as being of "tapping," consisted of only tapping sounds, *i.e.* were all real world impact sounds. However, in some of our clips, there were multiple occasions that the creator's in-hand manipulations of the objects inadvertently caused rubbing sounds in the course of tapping. These rubbing sounds were unfortunately unmodeled in our framework, causing our model to try to fit impact sounds to them. This demonstrates the importance of future work to expand our model with models for non-impact contacts such as rubbing and scratching. The creator used only fingertips and fingernails to tap each object in the clips we selected. We assumed that the fingernails and fingertips were small and damped enough that their modal vibrations' contribution to the sound was negligible, and therefore did not include their impulse responses as a contribution to the final sound in our framework's formulation. We only assumed that the fingertips and fingernails could cause acceleration sounds when they struck the object.

C.2 Our Model and Baselines

The high-level structure of our model used for this experiment is diagrammed in Figure 9. We first convert the spectrogram to Mel-frequency cepstrum coefficient (MFCC) features, which we fed into an encoder almost identical to that of [15]. The encoder consisted of a Gated Recurrent Unit (GRU), followed by a 1D temporal convolution with leaky ReLU activation, four fully connected layers with a leaky ReLU activation, followed by a GRU, then three more fully connected layers with leaky ReLU activation. Our autoencoder structure diverges from [15] after the encoder. We fed the outputs of this encoder directly to our model as the normalized input parameter vectors to our generators which we had combined in a directed acyclic graph (DAG) as summarized in Figure 1. We will refer to this section of our model as the *Synthesizing DAG* (see Figure 10), and compare it to two of our baselines. We passed the time series of outputs from the encoder as input to the Impact Force Profile generator, and used randomly initialized parameter variables as inputs for the other generators. To ensure that our model can effectively capture sharp impacts with short time scales without being too constrained by the lower bound of the scaling function in Equation 6, each of our generators produces its output at twice the final audio sample rate. Therefore, our final operation in our Synthesizing DAG performs a linear resampling to take our model's output back to final audio sample rate.

Both the DDSP Serra and DDSP Sin baseline models used the exact same encoder structure as our model but only vary the structure of the Synthesizing DAG by using pre-existing generators. This allows us to isolate the effects of our contributions. The DDSP Serra model used the same Synthesizing DAG as [15] (Figure 11), based on the Serra-Smith model[30], using encoder outputs to control both a harmonic oscillator and time-varying filtered noise, summed to make the final output. Since the modal vibrations of general objects are not necessarily harmonic like the musical instruments for which the Serra-Smith model was designed, we constructed the Synthesizing DAG for the DDSP Sin by replacing the harmonic oscillator of DDSP Serra with an unrestricted time-varying sinusoidal oscillator (Figure 12). These two models were trained with the same loss as our model, with the addition of spectrograms of lower window sizes (128 and 64) to mirror the loss used in the work which introduced these models [15, 24] (details in Appendix C.2).

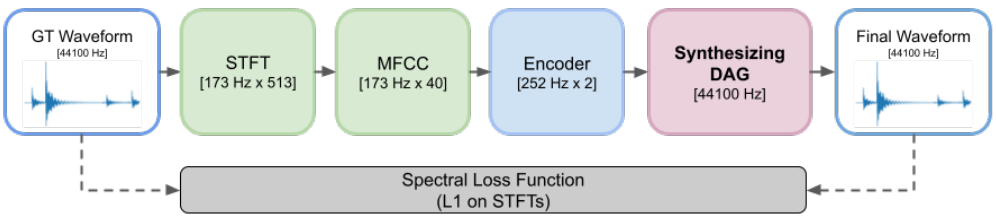


Figure 9: High-level Autoencoder Diagram

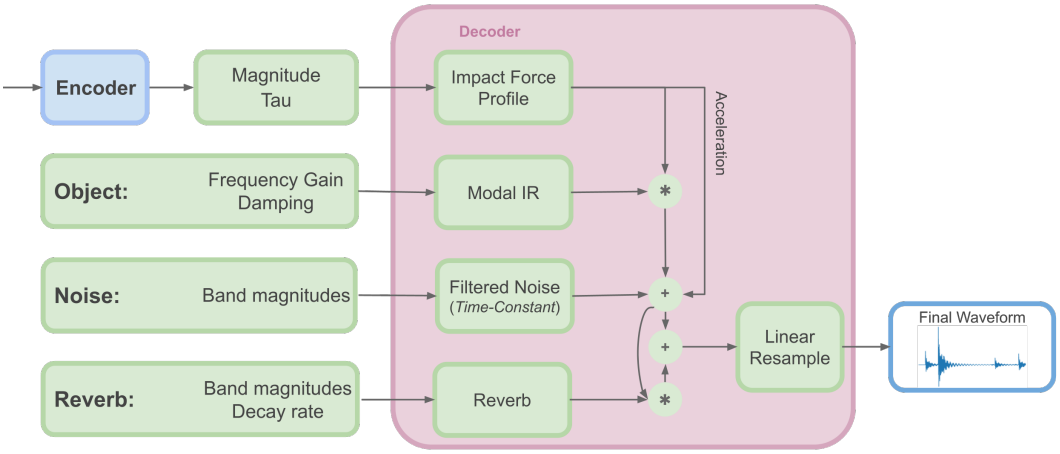


Figure 10: Proposed Synthesizing DAG.

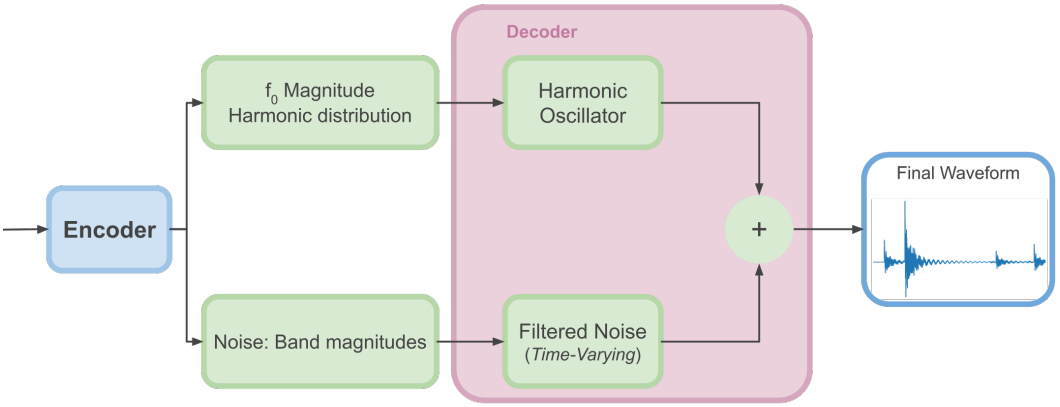


Figure 11: "DDSP Serra" baseline model Synthesizing DAG.

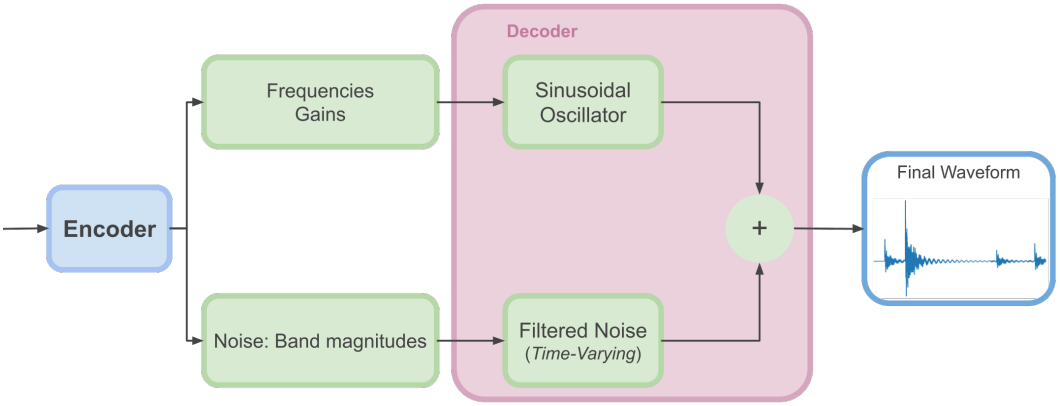


Figure 12: "DDSP Sinusoidal" baseline model Synthesizing DAG.

C.3 Human Study

For each of our published Human Intelligence Task (HIT) tasks on Amazon Mechanical Turk, we presented the workers with 12 questions containing randomly chosen tapping ASMR videos from our test set of each of the 5 different objects: wooden box, glass bowl, silicone pad, steel bar, and ceramic plate. Each of these videos had been redubbed with either our model's output or the output of

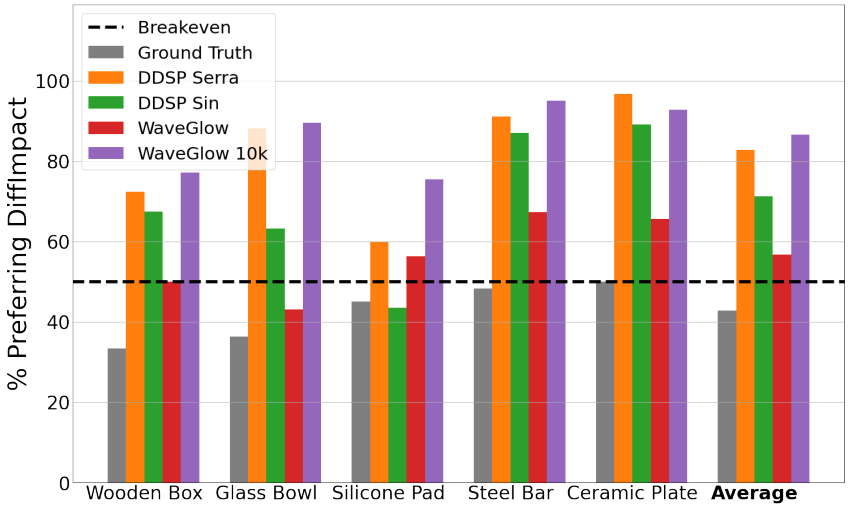


Figure 13: Comparing human study ratings of realism of different ASMR objects' test set outputs from each model to the output of the undertrained WaveGlow model ("WaveGlow 10k").

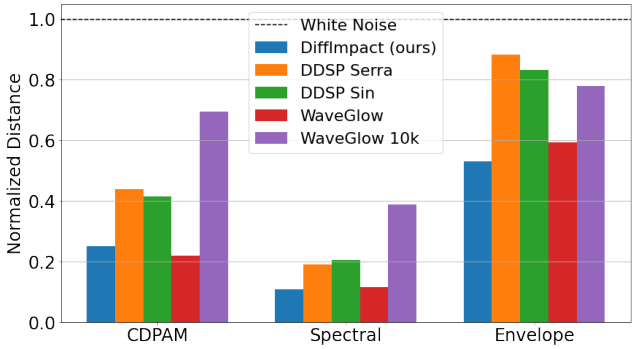


Figure 14: Comparing average computational audio distance metrics of test set output from different models to the output of the undertrained WaveGlow model ("WaveGlow 10k").

a baseline model, where the input to each model was the spectrogram computed from the ground truth original audio, or we left the video with its ground truth audio. In each question, we presented the participants with two videos, one from each treatment, and asked them to choose the one that better matches the sound they would expect from the object in the video. We conducted 30 HITs per model resulting in a total of 120 HITs. To detect fraudulent responses from users who were not following the instructions, we randomly selected 2 out of 12 questions in each HIT as checkpoints where a video redubbed with white noise was presented as one of the options. We then removed the responses of the users who chose white noise in those questions from our study to produce the results we show. 15 out of 120 HITs were determined as fraudulent based on this metric. Our final results included answers from the remaining 105 HITs containing a total of 1050 questions across all models.

C.4 Effects of Training Time on WaveGlow Performance

In Section 4.2, we claim that our model is able to learn to generate audio which is realistic enough to be narrowly rated as more realistic on average than audio from the WaveGlow model [16], as shown in Figure 3. However, WaveGlow requires much more computing resources to train, requiring about 300 hours to train a single object's model on a desktop computer with an Nvidia Quadro P5000 GPU, versus our model and other baselines requiring about 1 hour. To demonstrate the necessity of training the WaveGlow model for this long, here we show the results of our experiments from training each WaveGlow model for only 10,000 steps, requiring about 30 hours on our single GPU desktop, whereas the results we have presented as "WaveGlow" in all figures are from models which have been trained for 70,000 steps for each object, requiring about 300 hours on our desktop. Results are shown in Figures 13 and 14.

Appendix D Source Separation Experiment Details

In this experiment, the only supervision used during the optimization were the sound clips from each pairwise impact, and the labels of which object and tool are present in each clip. These labels can be considered as weak supervision.

ALGORITHM 2: Modal Impact Sound Source Separation Optimization

Input: Batch of recorded impact waveforms $\mathbf{W} \in \mathbb{R}^{B \times T}$, weak labels of N_o distinct object IDs present in each waveform $\mathbf{L_o} \in \mathbb{W}^B, N_o \leq B$, and weak labels of N_t distinct tool IDs present in each waveform $\mathbf{L_t} \in \mathbb{W}^B, N_t \leq B$

Output: Object modal parameters $\mathbf{M_o} \in \mathbb{R}^{N_o \times 3F_o}$, tool modal parameters $\mathbf{M_t} \in \mathbb{R}^{N_t \times 3F_t}$, tool reverb parameters $\mathbf{R_t} \in \mathbb{R}^{B \times 3F_R}$, contact force parameters $\mathbf{C_f} \in \mathbb{R}^{3B}$, acceleration sound scalars $\mathbf{A} \in \mathbb{R}^B$, environment noise parameters $\mathbf{N} \in \mathbb{R}^{F_N}$, and noise scalars $\mathbf{N_s} \in \mathbb{R}^B$

$\mathbf{M_o}, \mathbf{M_t}, \mathbf{R_t}, \mathbf{C}, \mathbf{N}, \mathbf{N_s} \leftarrow InitParameters()$;

for $i \leftarrow 1$ *to* S **do**

$\mathbf{M_{o,temp}} \leftarrow Replicate(\mathbf{M_o}, \mathbf{L_o})$;

$\mathbf{M_{t,temp}} \leftarrow Replicate(\mathbf{M_t}, \mathbf{L_t})$;

$\mathbf{F} \leftarrow F([0,...,T]; \mathbf{C_f})$;

$\mathbf{W_o} \leftarrow \mathbf{F} \circledast IR_o([0,...,T]; \mathbf{M_o})$;

$\mathbf{W_t} \leftarrow \mathbf{F} \circledast IR_o([0,...,T]; \mathbf{M_t})$;

$\mathbf{W_N} \leftarrow \mathbf{N_s} \circ N([0,...,T], \mathbf{N})$;

$\hat{\mathbf{W}} \leftarrow \mathbf{W_o} + \mathbf{W_t} + \mathbf{W_N} + \mathbf{A} \circ \mathbf{F}$;

$\hat{\mathbf{W}} \leftarrow \hat{\mathbf{W}} + \hat{\mathbf{W}} \circledast IR_e([0,...,T]; \mathbf{R_t})$;

$\mathbf{M_o}, \mathbf{M_t}, \mathbf{R_t}, \mathbf{C}, \mathbf{N}, \mathbf{N_s} \leftarrow GradientStep(\mathcal{L}(\mathbf{W}, \hat{\mathbf{W}}), \{\mathbf{M_o}, \mathbf{M_t}, \mathbf{R_t}, \mathbf{C}, \mathbf{N}, \mathbf{N_s}\})$;

end

return $\mathbf{M_o}, \mathbf{M_t}, \mathbf{R_t}, \mathbf{C}, \mathbf{N}, \mathbf{N_s}$

Algorithm details We show abstracted pseudocode for the impact sound source separation optimization algorithm in Algorithm 2. After iteratively optimizing all the parameter variables with this algorithm, we generate separated object and tool sounds by using the generation code in the inner loop, but eliminate components from the generation process which are irrelevant to each separated source we are synthesizing. For example, we can synthesize a "denoised" version of the impact sound by omitting the contribution of the noise $\mathbf{W_N}$ from all summations within the generation code in the inner loop. For synthesizing the object modal sounds, we take $\mathbf{W_o}$ alone as the final sound. To produce the separated tool sounds, including the acceleration sound, we take $\mathbf{W_t}$, add acceleration sound $\mathbf{A} \circ \mathbf{F}$, finally add this result with the convolution of itself and the reverberation filter $IR_e([0,...,T]; \mathbf{R_t})$. For further details, we have included our source code for this in our Supplementary Materials.

Material classification model For training a material classification model which can classify object materials from audio, we first begin with the Sound-20K dataset [40], which has thousands of synthesized audio recordings of different virtual objects of different materials falling on surfaces in a virtual environment. We first filter the available audio samples down to those from virtual objects made of materials we tested for our experiments ("ceramic", "polycarb" "steel", and "wood"), and then balance our dataset by taking the same amount of recordings from each material. We then compute the spectrogram of the audio sample with a window size of 512, to ensure a balance of frequency and temporal resolution, so that the input can capture and characterize the rapidly decaying modal frequency components in each impact. We split our data with a 90-10 train-validation ratio.

Our model takes this input, passes it through a Gated Recurrent Unit (GRU) layer, followed by two fully-connected layers with ReLU activations, and a final fully-connected layer with a softmax activation. During training, we use dropout on the outputs of the GRU and hidden layers to prevent overfitting, and train using the Adam optimizer [29] until a local minimum is reached in the loss on the validation data. Note that this model has not been trained on any real audio data.

Additional classification results In Figure 15, we show the confusion matrices comparing how well our pre-trained material classification model can classify the material of the objects and tools from the original audio versus the audio that our model generates as their "separated" impact audio. The

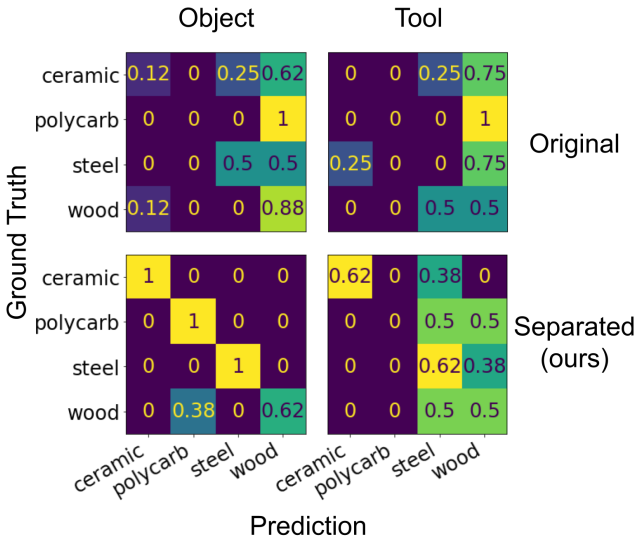


Figure 15: Confusion matrices for classifying material of the struck object and the tool with a classifier trained on Sound20k. (**Top Left**) Classifying the object with the original audio. (**Bottom Left**) Classifying the object with our separated audio. (**Top Right**) Classifying the tool with the original audio. (**Bottom Right**) Classifying the tool with our separated audio.

material classification model clearly struggles to classify the original audio with the correct material. We would expect that the sound the tool makes during the impact *pollutes* the sound that the object makes when impacted. Therefore, we would expect the classifier to occasionally waver between classifying the audio with either the object or the tool material. However, our confusion matrix at the far left of Figure 15 shows that the model classifies the material of every audio sample with the polycarbonate cup as wood, though only two of the eight recordings of striking the polycarbonate cup use the wooden spoon. This suggests that this pollution of the tool and object sound together can cause the classifier to classify the audio as the material of *neither* of the objects.

Our trained classifier model generally struggles with classifying wood sounds correctly. This is especially evident in the tool classification confusion matrices on the right of Figure 15. Wood sounds are very broad depending on the type and structure of the wood, *e.g.*, hardwoods sound very different than soft woods. Therefore, differentiating between wood and other materials with higher dampings, such as polycarbonate, using only sound may be especially challenging.

Young's modulus metrics Young's modulus is a material property which quantifies the stiffness of a material. Knowing an object or tool's material stiffness is important for a robot to reason about that object or tool's affordances. For hard materials, precisely measuring Young's modulus traditionally requires invasive or destructive measurement of precisely fashioned samples [43] or requires highly specialized measurement equipment and dedicated sensors with which robots are generally not equipped [44, 45]. Robots have been shown to be capable of using force sensors to measure soft objects [46] and soft human tissues [47], but these objects have Young's modulus values which are many orders of magnitude lower than those of hard materials such as ceramics and steel. In a modal model of object vibrations, the sound of an object's modal response is highly influenced by the Young's modulus value of its material [21]. Vibrations and sound have been successfully used to infer Young's modulus of materials of rods of known geometry [48] and of simulated objects [22]. Inspired by this work, we determine how accurately we can predict Young's modulus values for the material of each object and tool from the impact sounds separated by our approach and compare to the same source separation baselines we used for the material classification experiment in Section 4.3. We train a regression model on only the Sound20k dataset of synthetic impact sounds of objects of different materials [40], using their Young's modulus values as labels. We then use this model, trained only on synthetic audio, to predict Young's modulus of the materials of the real objects and tools from each approach's source-separated impact sounds. We measure error in terms of percent error of the estimate relative to the ground truth value.

Table 2: Estimating Young's modulus from source-separated audio.

	Object Regression Error (%)		Tool Regression Error (%)	
Model	Dataset Min	Instance Min	Dataset Min	Instance Min
Raw audio	263.4	263.4	243.3	243.3
NMF [38]	174.4	93.4	153.3	97.8
DAP [39]	161.0	71.9	227.7	104.1
DiffImpact (ours)	**59.6**	**46.1**	**118.0**	**86.6**

Young's modulus regression model We used a slightly different model architecture and training regimen for the regression task of estimating Young's modulus from sound versus our model for material classification from sound. We used the entire Sound20k dataset for training and validation, without filtering based on materials, in order to increase the training set size, splitting the dataset with a 90-10 training-validation split. We obtained labels for Young's modulus from the material configuration files accompanying the dataset. Each clip in the dataset is a $\sim$2-second simulated recording of an object bouncing on a surface, so each clip includes the sound from multiple impact events. Because the clips from our source separation audio were each 1 second long and included the sound of exactly one impact event, we designed our architecture to be more invariant to the number of impact events or length of clip. We thus replaced the recurrent unit of our classification model with three dilated 1D temporal convolution layers, each with ReLU activations, followed by a maximum pooling operation across the entire temporal dimension for each feature channel. Similar to our material classification model, we followed this with two fully-connected layers with ReLU activations, with a final fully connected layer to output the final Young's modulus regression estimate. To mitigate overfitting, we trained this model with dropout regularization on the output of the maximum pooling layer and each of the hidden fully connected layers. Because the Young's modulus values for different materials spanned multiple orders of magnitude, we trained against a mean squared logarithmic error loss with the Adam optimizer [29] until the validation error had reached a minimum. Note that this model was not trained on any real audio data.

Young's modulus results The results of this regression task are shown in Table 2. Similar to the results of the classification experiments in Section 4.3, we evaluate the performance of our regression model on the outputs of our framework against its performance on the outputs of baselines using two different schemes, since our baselines do not explicitly differentiate which output comes from which source, object or tool. In the "Dataset Min", we take the minimum between the mean percent error of treating the first output and treating the second output as the object sound. We do the same for tools. Under the "Instance Min" scheme, we take the minimum percent error between the two outputs for each instance instead of across the whole dataset, then take the mean across all instances. Similar to the classification results, both schemes offer the baselines an advantage, but our DiffImpact outperformed them, while also explicitly differentiating between the object and the tool. Also similar to our classification results, our framework performs better on the object sounds than the tool sounds, perhaps for the same reasons which could explain the difference in performance between tool and object classification, detailed in Section 4.3.

Appendix E Model Assumptions

Assumptions about the quantities and features influencing our model are organized in Table 3 by whether they are known, observable and estimated by our model, or unobservable. We either assume unobservable quantities to be constant, select them as a hyperparameter, or estimate them to a relative scale. Priors are induced on estimated quantities by the hyperparameters selected for bounds of the scaling functions (see Appendix A) and those selected for initialization and biases (see Appendix F).

For the recording, we have the ground truth waveform, but for recordings from the wild, the microphone's position, gain, and directionality (*e.g.*, cardioid or omnidirectional) are assumed to be unobservable but constant for each recording. Because these quantities are unobservable, we do not model acoustic transfer, which would depend on the microphone's position relative to the object [49] and properties of air in the environment [50]. We instead assume any effects of acoustic transfer to be part of the impact magnitude and object modal response. For the impact, because we do not observe the microphone position and gains, we cannot estimate absolute quantities of impact force magnitudes without calibration. For the count of impacts, we know exactly one impact occurs in recordings

Table 3: Model assumptions of information and physical quantities.

	Known	Observable (Estimated)	Unobservable (Assumed)
Recording	• Waveform		• Microphone position • Microphone gain • Microphone directionality
Impact	• Count (Sec 4.3)	• Magnitude (relative) • Time scale • Timing	• Count (Sec 4.1 and 4.2) • Magnitude (absolute) • Location
Object	• Instance ID • Tool instance ID (Sec 4.3)	• Mode frequencies • Mode gains (relative distribution) • Mode dampings	• Mode count • Mode gains (absolute) • Contact conditions
Environment		• Static noise gains • Reverb response gains • Reverb response decays	• Geometry

of Section 4.3, but we do not have ground truth counts of the number of impacts in recordings of Sections 4.1 and 4.2 and select this as a hyperparameter as described in Appendix A.1. Our model estimates the time scales (sharpnesses) and timings of impacts (shifted by the unobserved sound travel time from object to microphone), though estimated magnitudes are proportional rather than absolute.

For objects, we only know the object's instance ID (as well as the tool instance ID for Section 4.3). We select a reasonable number of salient modes as a hyperparameter, then estimate the frequencies and dampings of those modes. For mode gains, because we do not have absolute impact magnitudes as previously explained, we estimate a normalized relative distribution. We also do not know the exact contact conditions of the object, which will dampen certain frequencies depending on the region, force, and material properties of each contact on the object [51]. We instead assume contact damping to be negligible or part of the object's intrinsic modal impulse response. Striking an object in different locations excites each mode of the object at a different gain [18], but since we do not observe the location of each impact, we assume the location of impact to be constant for our all our models.

Finally, while we estimate static measurement noise, we have measured neither the geometry nor the standalone impulse response of our environment for reverberations. Room reverb responses are often specifically measured with a small explosion of air to record the independent response of the room [52]. A popular model for a room's reverb response is a modal model, which can require 1000-2000 separate modes to characterize [53]. Because we have no such isolated measurement or knowledge of the room geometry (nearby surfaces' distances, angles, and absorptions) from recordings in the wild, and we cannot separate out reverb mode responses from object mode responses, we instead model only the late stage reverb response of the room with exponentially decaying filtered noise [52].

Appendix F Hyperparameters and Initialization

Here we make some high-level observations about the effects of different choices of hyperparameters and initialization. Specific values can be found in our released code.

Loss function Perhaps the most influential hyperparameters are those of the loss function, the multi-scale spectrogram loss detailed in Section 3.5. With spectrograms, there is an inherent tradeoff between the time resolution and the frequency resolution, which are inversely and directly proportional to the window size, respectively. For the object modal response, frequency resolution is important for precisely estimating modal response frequencies, and time resolution is important for precisely characterizing the decay of each mode to estimate damping. Time resolution is especially important for precisely localizing the onset of the impact sound, which is upstream of characterizing every other feature of the impact sound. For example, mistakenly estimating the timing of the impact to be slightly after the ground truth timing could cause the model to characterize the object's modal impulse response as only the tail end of the ground truth response, where many modes will already have decayed significantly.

The intention of using multiple scales of spectrograms is to attain the best of both worlds in terms of both frequency and time resolution. However, natural spectrograms can present an optimization landscape which is far from smooth. Spectrograms with high time resolution are finer-grained and

less smooth on their temporal axis, causing much more potential for local minima in optimizing the temporal alignment of the synthetic impact sound with the ground truth impact sound. Spectrograms with lower time resolution present a much smoother optimization landscape with respect to timing. This same concept can be an important consideration with respect to the risks of optimizing frequency values of the modal response using spectrograms with large window sizes and high frequency resolution. Even though the final loss is summed across the spectrograms of different resolutions, a given spectrogram could dominate this loss and resulting gradients, such that the other spectrograms could not rescue the optimization from a local minimum. For these reasons, we eliminated the two spectrograms with the smallest window sizes from the loss function of [15] for our loss function on impact sounds. Also for these reasons, whenever we could assume only one impact occurred during a clip as we could in Section 4.3, we initialized the timing of the impact to the location of the maximum of the absolute value of the waveform across the recorded clip.

Generator biases Tuning bias hyperparameters for the different generator functions is important for ensuring faster convergence to reasonable values. During the optimization, each generator is essentially competing to explain different aspects of the sound. Starting the optimization with too high of gains for one generator could cause that generator's output to dominate the sound. For example, the background noise generator could overfit its filter gains to match the sounds of impacts as closely as possible with its time-constant noise.

Frequency resolutions Our Modal IR, Filtered Noise, and Reverb generator functions each could compute outputs for inputs with varying resolutions of frequencies. More specifically, for the Modal IR, the number of modes could be varied, and for the Filtered Noise and Reverb generators, the number of noise bands for each could be varied. The frequency resolutions of these generators presented a tradeoff between expressiveness and robustness to overfitting. For example, in our source separation experiment with a very small dataset, choosing too high of frequency resolutions for both the object and tool modal models in the Modal IR increased the likelihood that an object's separated sound would include faint artifacts of sound from one or more of the tools which struck it. Choosing a lower frequency resolution for each Modal IR forces our model to fit the more salient modal components of each object and tool.