A STATISTICS-DRIVEN DIFFERENTIABLE APPROACH FOR SOUND TEXTURE SYNTHESIS AND ANALYSIS

Esteban Gutiérrez

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
esteban.gutierrezc@upf.edu

Frederic Font

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
frederic.font@upf.edu

Xavier Serra

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
xavier.serra@upf.edu

Lonce Wyse

Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
lonce.wyse@upf.edu

ABSTRACT

In this work, we introduce `TexStat`, a novel loss function specifically designed for the analysis and synthesis of texture sounds characterized by stochastic structure and perceptual stationarity. Drawing inspiration from the statistical and perceptual framework of McDermott and Simoncelli, `TexStat` identifies similarities between signals belonging to the same texture category without relying on temporal structure. We also propose using `TexStat` as a validation metric alongside Frechet Audio Distances (FAD) to evaluate texture sound synthesis models. In addition to `TexStat`, we present `TexEnv`, an efficient, lightweight and differentiable texture sound synthesizer that generates audio by imposing amplitude envelopes on filtered noise. We further integrate these components into `TexDSP`, a DDSP-inspired generative model tailored for texture sounds. Through extensive experiments across various texture sound types, we demonstrate that `TexStat` is perceptually meaningful, time-invariant, and robust to noise, features that make it effective both as a loss function for generative tasks and as a validation metric. All tools and code are provided as open-source contributions and our PyTorch implementations are efficient, differentiable, and highly configurable, enabling its use in both generative tasks and as a perceptually grounded evaluation metric.

1. INTRODUCTION

Defining audio textures is a complex problem that has been explored by various authors. The concept originated as an analogy to visual textures. In [1], Julesz, one of the pioneers in this field, proposed the so-called "Julesz conjecture," suggesting that humans cannot distinguish between visual textures with similar second-order statistics. This hypothesis was later disproven in [2], but whose statistical perspective remains influential in texture analysis (see [3]) and synthesis (see [4], [5], and [6]). This perspective is also foundational for this work.

Regarding the auditory domain, an early definition of audio textures was introduced in [7]. The authors described them as patterns of basic sound elements, called atoms, occurring in structured but potentially random arrangements. These high-level patterns must remain stable over time while being perceivable within a short time span. Rosenthal et al. [8] later expanded this idea, defining audio textures as a two-level structure: atoms forming the core elements and probability-based transitions governing their organization. Recent research has refined these ideas, but the balance between short-term unpredictability and long-term stability remains a common thread. In this matter, Wyse et al. [9] points out that sound texture's complexity and unpredictability at one level is combined with the sense of eternal sameness at another.

Many approaches have been taken regarding the synthesis and re-synthesis of texture sounds. These approaches can be classified in various ways. For example, Sharma et al. [10] separates them into three categories: granular synthesis, which corresponds to the concatenation of already existing sounds, as in the pioneering works [11], [12], and [13]; model-based synthesis, which relies on either improved time-frequency representation models, as in [14], physical models, as in [15], and/or physiological/statistical models, as in the foundational series of articles [4], [5], and [6]; and finally, deep learning-based models, encompassing various contemporary machine learning techniques, as in [16], [9], and [17].

The sound resynthesis task involves recreating a given sound through analysis and synthesis. While sound reconstruction can be viewed as an exact replication of the original sound, in the case of texture-based resynthesis, it is often sufficient to generate a sound that is perceptually similar to the original, and thus, the resulting sound may differ significantly in its detailed time structure from the original one. Moreover, Caracalla et al. [16] state that typically the texture sound resynthesis goal is to "create sound that is different from the original while still being recognizable as the same kind of texture." In the context of deep learning, this goal can be approached in various ways. For instance, models with inherent stochasticity naturally generate a "similar enough" sound, drawing from the set of sounds they can produce, which is influenced by their biases. Conversely, even in the absence of stochasticity, if the model's loss function is perceptually grounded, it will inevitably focus solely on that instead of exact replication.

An example of a model that performs resynthesis by accurately matching the spectrum of a texture sound is NoiseBandNet [17], a Differentiable Digital Signal Processing (DDSP)-based ar-

chitecture that indirectly follows this approach by using a multi-scale spectrogram loss applied to small time scales (starting at approximately 0.7 ms) and small sound windows (around 180 ms). On the side of perceptual/features-based resynthesis using deep learning, [16] employed a loss function that compares the Gram matrices of two sets of features computed from pretrained Convolutional Neural Networks (CNNs). Another example comes from [9] and its use of Generative Adversarial Networks (GANs), where the objective of the loss function is to train the model to generate sounds that can deceive a classifier, thereby biasing the training in a more nuanced manner than direct signal comparison.

In the context of texture sound analysis and resynthesis, this paper has three main goals. First, we introduce `TexStat`, a loss function grounded in human perception of texture sounds. This function builds upon the pioneering work of McDermott and Simoncelli [4], [5], [6], and is made available as an efficient, open-source implementation in PyTorch. Second, we present `TexEnv`, a simple and efficient noise-based texture sound synthesizer developed in synergy with `TexStat`. Finally, to evaluate both `TexStat` and `TexEnv`, we introduce `TexDSP`, a DDSP-based architecture trained across multiple scenarios, demonstrating the effectiveness of our proposed components.

In Section 2, we formally define the loss function `TexStat` together with its potential variations. In Section 3, we briefly introduce the `TexEnv` synthesizer, highlighting both its capabilities and its limitations. In Section 4, we present the `TexDSP` architecture as a showcase for the tools introduced earlier. In Section 5, a series of experiments demonstrating the tools introduced in this paper are presented. Finally, in Section 6, we summarize the tools and results discussed in this work and outline some future directions for research.

2. `TEXSTAT`: A LOSS FUNCTION SPECIFICALLY TAILORED FOR TEXTURE SOUNDS

In this section, we discuss some desirable properties of a texture sound loss function and then introduce `TexStat`, a loss function specifically designed for texture sounds that fulfills, to some extent, the properties outlined here.

2.1. What Should a Texture Loss Be?

In the context of deep learning, solving a task—whatever the task may be—is framed as an optimization problem. Such an optimization problem typically involves minimizing a loss function, with the expectation that doing so will lead to solving the proposed task. In this scenario, the loss function must be able to represent, to some extent, the task to be solved.

The texture sound generation task can be described as a process in which a sound is created so that it is recognized as belonging to a particular type of texture. However, it does not need to be—and in fact, it is preferable that it is not—identical to the input or dataset examples. Additionally, depending on the context, it may also be desirable to generate non-repetitive sound over time, with the sound generation process allowing for some degree of temporal control.

This task description, together with our current understanding of texture sounds, leads us to the following desirable properties for a loss function specifically tailored for texture sounds.

Overall Structure Focus: Texture sounds resemble filtered noise over short periods but are stable and recognizable over time. Thus, a loss function should prioritize long-term characteristics, like rhythm, pitch variations, or granular shifts, rather than specific and detailed temporal structure.

Stochastic Focus: Following [1] and [8], texture sounds arise from stochastic processes in time and timbre. A suitable loss function should capture these statistical properties to recognize similarity between sounds corresponding to the same type of texture.

Perceptual Foundation: The loss function must capture perceptual similarity, ensuring that two sounds considered perceptually similar are treated as such. This involves leveraging psychoacoustic principles to align with human auditory perception.

Time Invariance: Texture sounds can be chopped or rearranged with minimal perceptual impact. The loss function should thus exhibit time invariance, allowing rearrangement without significantly altering the sound's characteristics.

Noise Stability: Texture sounds can tolerate noise, so the loss function should be robust to subtle noise variations, preserving core texture features even with low to mid level disturbances.

Flexibility: A highly restrictive loss function risks overfitting, leading to audio too similar to the training data. It should encourage creative variation, generating novel yet perceptually consistent sounds that retain defining textural characteristics.

2.2. `TexStat` Formal Definition

According to Julesz's conjecture and McDermott and Simoncelli's synthesis approach, the nature of textures can be understood through direct comparison of their statistics. McDermott and Simoncelli's synthesis approach [4], [5], and [6] involves the imposition of a set of statistics precomputed for a given texture sound using numerical methods, suggesting the possibility of using that exact set of statistics as a feature vector for comparison.

In this work, we introduce `TexStat`, a loss function that operates by directly comparing a slight variation of a superset of the summary statistics used by McDermott and Simoncelli.

In the following, we formally introduce the necessary tools, preprocessing steps, and sets of summary statistics related to the computation of `TexStat`. For further details on summary statistics' perceptual importance, we refer to [5] and [6].

Preliminary Tools: At the core of `TexStat` there are a series of subband decompositions. Following the original work of McDermott and Simoncelli, we consider the following: a Cochlear Filterbank, which is an Equivalent Rectangular Bandwidth (ERB) filterbank [18], [19] $F = \{f_j\}_{j=1}^{N_F}$ made up of N_F filters; and a Modulation Filterbank, which is a Logarithmic Filterbank $G = \{g_k\}_{k=1}^{N_G}$ made up of N_G filters.

Preprocessing: Given a signal s, the Cochlear filterbank F is used to compute a subband decomposition of it by $s_j = s * f_j$, $j = 1, \ldots, N_F$. Then, the amplitude envelope of each one of these subbands is computed using the Hilbert Transform $e_j = |s_j + iH(s_j)|$, $j = 1, \ldots, N_F$. Finally, the Modulation Filterbank G is used to compute a subband decomposition of each one of the envelopes e_j, that is, $m_{j,k} = g_k(e_j), j = 1, \ldots, N_F, k = 1, \ldots, N_G$.

Statistics sets: The first set of statistics is comprised of the first L normalized moments of the signals e_j, that is, $S_{1,l,j} = M_l(e_j)$ for $j = 1, \ldots, N_F$ and $l = 1, \ldots, L$, where $M_1(X) = \mathrm{E}(X) = \mu$, $M_2(X) = V(X)/\mu^2 = \sigma^2/\mu^2$ and

$$M_l(X) = \frac{\mathrm{E}(X-\mu)^l}{\sigma^l}, \quad l = 3, \ldots, L.$$

Note that the vectors $S_{1,l} \in \mathbb{R}^{N_F}$ correspond to different moments and hence they are prone to have values that vary in orders of magnitude. In order to fix this issue, the first set of statistics correspond to a weighted concatenation of the vectors corresponding to different moments

$$S_1(s) = \text{concat}(\alpha_1 S_{1,1}, \ldots, \alpha_N S_{1,L}) \in \mathbb{R}^{L \cdot N_F}.$$

The second set of statistics corresponds to the Pearson correlation between different amplitude envelopes e_j, that is,

$$S_2(s) = \text{vech}(\text{corr}(e_1, \ldots, e_{N_F})) \in \mathbb{R}^{T_{N_F - 1}},$$

where $\text{vech}(\cdot)$ corresponds to the half-vectorization operator, $\text{corr}(\cdot)$ to the correlation matrix and T_n to the n-th triangular number.

The third set of statistics corresponds roughly to the proportion of energy in each modulation band, that is,

$$\begin{aligned} S_{3,j} &= \frac{\left(\text{V}(m_{j,1})^{1/2}, \ldots, \text{V}(m_{j,N_G})^{1/2}\right)}{\text{V}(e_j)^{1/2}} && \in \mathbb{R}^{N_G} \\ S_3(s) &= \text{concat}(S_{3,1}, \ldots, S_{3,N_F}) && \in \mathbb{R}^{N_G \cdot N_F}. \end{aligned}$$

The fourth set of statistics corresponds to the Pearson correlations between modulation subbands corresponding to the same amplitude envelope, that is,

$$\begin{aligned} S_{4,j} &= \text{vech}(\text{corr}(m_{j,1}, \ldots, m_{j,N_G})) && \in \mathbb{R}^{T_{N_G - 1}} \\ S_4(s) &= \text{concat}(S_{4,1}, \ldots, S_{4,N_F}) && \in \mathbb{R}^{N_F \cdot T_{N_G - 1}}. \end{aligned}$$

Finally, the fifth set of statistics corresponds to the Pearson correlation between modulation subbands corresponding to the same band but different amplitude envelopes, that is,

$$\begin{aligned} S_{5,k} &= \text{vech}(\text{corr}(m_{1,k}, \ldots, m_{N_F,k})) && \in \mathbb{R}^{T_{N_F - 1}} \\ S_5(s) &= \text{concat}(S_{5,1}, \ldots, S_{5,N_G}) && \in \mathbb{R}^{N_G \cdot T_{N_F - 1}}. \end{aligned}$$

Now that we have the five sets of statistics well-defined, we can finally define the `TexStat` loss function.

Definition 1. (`TexStat` loss function) The `TexStat` loss function is defined as

$$\mathcal{L}_{\alpha,\beta}(x, y) = \sum_{j=1}^{5} \beta_j \cdot \text{MSE}(S_j(x), S_j(y)),$$

where $\alpha \in \mathbb{R}^L$ and $\beta \in \mathbb{R}^5$ are parameters, x, y are a pair of signals, and $S_1(x), S_2(y), \ldots, S_5(x), S_5(y)$ are the summary statistics vectors defined above.

Before continuing, it is important to mention that although the `TexStat` loss function is strongly based on the work of McDermott and Simoncelli, several changes and additions were made to make it more suitable for machine learning tasks. Most of these changes involve adaptations that ensure control over the number of statistics and their weights during training. The number of statistics is important, as it would be counterproductive to compute more statistics than the size of the window used. Furthermore, weighting the statistics provides a way to balance their contribution to the loss. The main changes and their impact are outlined in Table 1.

	McDermott and Simoncelli's Summary Statistics	**`TexStat` Summary Statistics**
Filterbanks	Fixed type, frequency range, and size of filterbanks for their experiments.	Variable type, frequency range, and size of filterbanks for controllability.
Statistical Moments	Fixed number of statistical moments.	Variable number of statistical moments. Useful for compensating small filterbanks.
Modulation Band Correlations	Only some correlations are computed.	Multiple correlations were avoided in the original work for computational efficiency; however, with modern GPU computations, this doesn't make a significant difference.
Compressive Non-linearity	Applies one to the amplitude envelopes following past auditory models.	Removes compressive non-linearity for gradient stability.
Weights	Summary statistics are not weighted.	Variable weights to control the importance of certain sets of statistics during training and to avoid overflow (especially in S_1).

Table 1: Summary statistics comparison between McDermott and Simoncelli's work, and those used by `TexStat`.

2.3. `TexStat` Properties

As discussed in Subsection 2.1, there are several desirable properties that a texture sound loss function should have, and we argue that, when used correctly, `TexStat` addresses them all.
First, `TexStat` can be utilized on arbitrarily large frames of audio, and in any case, all summary statistics are directly influenced by the entire signal, which we argue implies a focus on the overall structure. Moreover, since it is built on a statistical and perceptual model, we also argue that its focus is on the stochastic properties of the signal, and that it has a strong foundation in perception.
Overall, one could argue that the operations involved in the computation of summary statistics are quite stable with respect to the addition of low-amplitude white noise. This will be empirically demonstrated in Subsection 5.1. Moreover, if $s \in C(\mathbb{R})$ is a continuous infinite signal whose summary statistics exist (for example, if it belongs to the Schwarz class $s \in \mathcal{S}(\mathbb{R})$), there are no operations in the process of computing $S_1(s), \ldots, S_5(s)$ that can be affected by time shifting, i.e., $\hat{s}(t) = s(t - t_0)$. This is, of course, not the case for discrete finite signals, where time shifting, $\hat{s}[t] = s[t - t_0 \mod \text{len}(s)]$, might introduce a click sound at the beginning, which would affect the spectrum and, consequently, the subband decomposition. However, this is not a significant issue for noisy signals, as adding a click will not strongly impact their spectrum. This will also be further explored in Subsection 5.1.

Finally, given a choice of parameters that does not generate an excessive number of summary statistics in relation to the samples used in the original signal, there are generally many signals that, when compared using `TexStat`, will result in low loss values. For example, all chopped and reordered versions of the same signal will typically yield similar summary statistics, and hence would be close in the sense of the `TexStat` distance. We claim that this suggests flexibility for the `TexStat` loss function.

2.4. Capabilities and Limitations

The `TexStat` loss function is based on the direct comparison of summary statistics, meaning that two sounds whose summary statistics are similar will not be recognized as different by this loss function. In their original work, McDermott and Simoncelli imposed the summary statistics of a series of sounds onto white noise and found that, although they successfully synthesized a range of sounds with good results, this process was unable to generate convincing sounds for certain types of timbres. This was because the summary statistics were not sufficient to fully characterize those types of sounds. Regarding the sounds that couldn't be fully captured by the summary statistics, McDermott and Simoncelli stated that "they fall into three general classes: those involving pitch (e.g., railroad crossing, wind chimes, music, speech, bells), rhythm (e.g., tapping, music, drumming), and reverberation (e.g., drumbeats, firecrackers)." These capabilities and limitations are naturally inherited by the `TexStat` loss function, and both effective and ineffective examples are shown in Subsection 5.5 to make transparent this loss' limitations.

2.5. Usage as a Loss and/or Evaluation Metric

The `TexStat` loss function introduced here is differentiable, making it suitable for use as a loss function in machine learning applications. Moreover, together with this article we release an efficient and open-source PyTorch implementation.

The number of computations required to run both `TexStat` and its gradients are relatively large compared to other loss functions, such as the Multi-Scale Spectral (MSS) Loss function. However, this can be mitigated by adjusting the frame size, as summary statistics are intended to be used on large audio frames. For example, most models evaluated in Section 5.5 used a frame size corresponding to approximately 1.5 seconds of audio at a framerate of 44100 Hz. In contrast, architectures like NoiseBandNets [17] use the MSS on frames of up to 0.18 seconds. Additionally, the increased computational cost also comes with higher memory usage, especially when increasing N_F and N_G, which can significantly impact memory requirements.

Given the timbre limitations of the summary statistics discussed earlier, we believe that, in order to fully guide the learning process of a general generative model, additional losses should be introduced to exert full control over different types of sounds, such as pitched or rhythmic sounds. In such cases, `TexStat` can be regarded as a regularizer that aids in guiding the timbre learning of texture sounds.

In some instances, `TexStat` may not be suitable as a loss function for the reasons mentioned earlier. In these cases, one can use other losses to guide the learning process and, if appropriate for the task, employ either `TexStat` as an evaluation metric. To facilitate this, we propose a fixed set of parameters, including filterbank types, filterbank sizes, and values for α and β, which have been proven useful for texture sound synthesis in the past and that were tested in this article's experiments. These parameters ensure comparability, and we also provide a list of precomputed values for interpretability. All of this can be found in the repository for the `TexStat` loss function. Moreover, since the `TexStat` loss function is essentially a direct comparison of summary statistics, one could view this set of statistics as a fully interpretable feature vector. This feature vector can thus be used as the basis for other evaluation metrics, such as Frechet Audio Distance (FAD) [20].

Our repository comes with a simple to use implementation of this method that uses a subset of the summary statistics here proposed and extensive experimentation to prove this concept can be found in Subsections 5.3 and 5.5.

3. TEXENV: A DIFFERENTIABLE SIGNAL PROCESSOR TAILORED FOR TEXTURE SOUNDS

The foundations of `TexStat` are based on the idea that summary statistics of amplitude envelopes derived from a low-size filterbank subband decomposition are sufficient to compare certain types of texture sounds. Implicit in this concept is the fact that directly imposing amplitude envelopes on a subband decomposition of white noise is one method of resynthesizing texture sounds while preserving their summary statistics. In the context of this work, two questions arise: How can we efficiently and differentiably create amplitude envelopes from a sequence of parameters? How can we efficiently and differentiably impose amplitude envelopes?

Creating amplitude envelopes from scratch can be done in multiple ways. For this synthesizer, we chose to use the Inverse Fast Fourier Transform (IFFT) because it is differentiable and can be computed efficiently to generate cyclic functions. For the imposition process, we fixed precomputed white noise, decomposed it using a filterbank, and then normalized the amplitude envelope of each subband. This procedure generates an object we call the *seed*, which serves as a "source of deterministic randomness" and can be used to impose amplitude envelopes via simple multiplication. Algorithm 1 outlines this synthesis method.

Algorithm 1 The `TexEnv Synth`

Input: Let F be a filterbank of size N_F, $(s_1, \dots, s_{N_F})$ a *seed* generated from F, $p_1, \dots, p_{N_F} \in \mathbb{C}^{N_P}$ a set of complex vector parameters and N the length of the signal to be generated.
Output: A texture sound synthesized $y \in \mathbb{R}^N$.

1: For each $j = 1, \dots, N_F$ construct a spectrum as

$$A_j = \text{concat}(p_{j,0}, \dots, p_{j,N_P-1}), \mathbf{0}, (\overline{p}_{j,N_P-1}, \dots, \overline{p}_{j,1})),$$

where $\mathbf{0}$ is a null vector of size $N - 2N_P + 1$.

2: For each $j = 1, \dots, N_F$ construct a real signal using the Inverse Discrete Fourier Transform (IDFT)

$$a_j = \text{IDFT}(A_j).$$

3: Impose the signals $a_1, \dots, a_{N_F}$ as amplitude envelopes on the seed and sum up to generate the final signal

$$y = \sum_{j=1}^{N_F} s_j \odot a_j.$$

4: **return** The synthesized signal y.

4. TEXDSP: A DDSP-BASED ARCHITECTURE TAILORED FOR TEXTURE SOUNDS

In this section we introduce TexDSP, a relatively simple texture sound generative model based on DDSP [21] that showcases the capabilities of TexStat and TexEnv.

The original DDSP model requires an encoder, decoder, signal processor, and loss function, with each component designed to generate small frames of pitched sounds based solely on pitch and loudness features. However, in the context of texture sound generation, the task is quite different, necessitating several modifications. The changes we propose ensure that the model's objective is to learn high-level statistical patterns, rather than aiming for perfect (frequency domain) reconstruction, as is the case with the original DDSP architecture. The final architecture is shown in Figure 1, and the changes made are briefly outlined here.

4.1. Encoder and Decoder

As in the original DDSP architecture, the encoding process involves both feature extraction and a more complex transformation of these features. In this case, we used different pairs of features that can be more informative for texture sounds than pitch and loudness. These features include spectral mean (frequency centroid), spectral standard deviation, energy in each band of a subband decomposition, and onset rate. To increase the complexity of the feature extraction process, we followed the original approach: first, we applied a Multi-Layer Perceptron (MLP) to each feature $F_j = \text{MLP}(f_j \mid \varepsilon_j^{\text{enc}})$, and then concatenated the results with the output of a Gated Recurrent Unit (GRU) applied to the same features $Z = \text{GRU}(F_1, F_2 \mid \varphi)$, yielding the latent representation $L = \text{concat}(F_1, F_2, Z)$. The decoding process involves transforming the latent representation L through another MLP and an additional layer to obtain a polar representation of the parameters used by the signal processor, i.e.,

$$\rho = \sigma(A \cdot \text{MLP}(L \mid \varepsilon^{\text{dec}}))$$
$$\theta = 2\pi\sigma(B \cdot \text{MLP}(L \mid \varepsilon^{\text{dec}}))$$

where A, B are real-valued matrices to be learned, and $\sigma(\cdot)$ denotes the sigmoid function applied component-wise.

4.2. Signal Processor and Loss Function

The original DDSP model used a Spectral Modeling Synthesis (SMS) [22] synthesizer, which is well-suited for pitched sounds, and a Multi-Scale Spectrogram (MSS) Loss, which is effective for perfect reconstructions. Since the goal of this architecture is to test the capabilities of TexStat, we opted for our signal processor, as it was designed to work synergistically with our loss function. Both TexEnv and TexStat are built around a filterbank, and to optimize synergy, this filterbank is shared between the two.

5. EXPERIMENTS AND RESULTS

In this section, we briefly explain a series of experiments conducted to provide proof of concept for the models proposed in this work. For these experiments, we hand-curated MicroTex[1], a dataset made from a selection of texture sounds from the following sources: the BOReilly dataset, containing textures made using analog synthesizers; Freesound [23], which contains environmental sounds; and synthetically generated data using Syntex [24]. All experiments can be found in their respective repositories and are fully replicable[2].

5.1. TexStat Properties Tests

Two desirable properties proposed for a loss function tailored to texture sounds are stability under time shifting and robustness to added noise. To test these properties in the TexStat loss function, we computed the loss between the sounds in the Freesound class of MicroTex and their corresponding transformations using various parameters. We focused on the Freesound class of MicroTex, as it contains the most representative examples of environmental texture sounds and includes recordings that are long and dynamic enough to allow meaningful time shifting and noise addition. The other two classes were excluded, as their sounds are either too short or too silent, making such transformations impractical without introducing substantial alterations. The experiment was also run with the MSS loss for comparison and some of the results can be found in Table 2.

	TexStat			MSS		
Transformation	10%	30%	50%	10%	30%	50%
Time-Shift	0.04 ± 0.03	0.04 ± 0.03	0.04 ± 0.03	6.09 ± 1.22	6.27 ± 1.38	6.29 ± 1.41
Noise-Add	2.08 ± 1.99	2.51 ± 2.21	2.65 ± 2.27	11.79 ± 4.91	16.84 ± 5.92	19.57 ± 6.26

Table 2: Loss measurements ($\mu \pm \sigma$) between sounds in the Freesound class of MicroTex and their corresponding time-shifted and noise-added transformations. Time shift is expressed as a percentage of the total signal duration, and noise percentage are defined by their maximum amplitude relative to the original signal. All measurements were computed over one-second segments for each of the sounds mentioned above. For reference, all satisfactory models trained using TexStat converged to loss values below 3, whereas evaluations using MSS typically yield reasonable values below 10.

The results demonstrate that TexStat exhibits strong stability under both time shifting and noise addition, incurring a consistent penalty for time shifts and a sublinear increase in penalty as noise levels increases.

5.2. TexStat Benchmarks

To benchmark the computational requirements of TexStat, we evaluated its computation time, gradient descent time, and GPU memory usage. These measurements were conducted multiple times, recording the time taken for loss computation and optimization while tracking memory allocation. The results are presented in Table 3, along with the values for other typical losses.

The results show that, as expected, the TexStat loss function is slower than other less specific losses, but it uses a similar amount of memory.

[1] MicroTex HuggingFace repository: cordutie/MicroTex.

[2] Find access to all repositories, experiments and sound examples in this article's webpage: cordutie.github.io/ddsp_textures/.

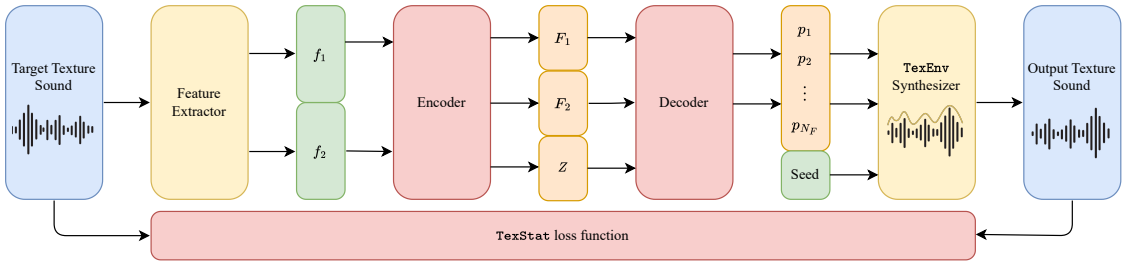

Figure 1: `TexDSP` architecture. Prechosen features are computed and are used to run the model. The encoder adds complexity and entangles this features into the latent representation $L = (F_1, F_2, Z)$. The decoder transforms this representation into a set of complex parameters that are used to run the `TexEnv` synthesizer. Finally, the output signal is compared to the original one using the `TexStat` loss function.

Loss	Forward pass time (ms)	Backward pass time (ms)	Memory usage (mb)
`TexStat`	93.5 ± 0.4	154.6 ± 0.4	0.84 ± 2.5
MSS	3.9 ± 0.3	8.5 ± 0.3	0.85 ± 2.6
MSE	0.2 ± 0.3	0.2 ± 0.1	1.7 ± 5.0
MAE	0.1 ± 0.0	0.2 ± 0.1	0.8 ± 2.5

Table 3: Measurements regarding computation time, gradient computation time, and memory usage ($\mu \pm \sigma$) in batches of 32 signals of size 65536 (around 1.5s at a sample rate of 44100 Hz). The losses studied were `TexStat`, Multi-Scale Spectrogram (MSS), Mean Squared Error (MSE), and Mean Absolute Error (MAE). All measurements were done using CUDA on an RTX 4090 GPU.

5.3. `TexStat` as an Evaluation Metric

In order to test `TexStat` summary statistics as a powerful representation that can be used in metrics like FAD, we conducted the following experiment. First, all data in the three selections of the `MicroTex` dataset were segmentated and both their summary statistics and VGGish [25] embeddings were computed. Then, a downstream classifier (MLP with hidden layers 128, 64) was trained in both cases. A summary of the results can be found in Table 4.

Model	Selection	Accuracy	Precision	Recall	F1
`TexStat`	BOReilly	**0.94**	**0.94**	**0.94**	**0.94**
VGGish	BOReilly	0.71	0.73	0.71	0.71
`TexStat`	Freesound	**0.99**	**0.99**	**0.99**	**0.99**
VGGish	Freesound	0.98	0.99	0.98	0.98
`TexStat`	Syntex	**1.0**	**1.0**	**1.0**	**1.0**
VGGish	Syntex	0.95	0.95	0.95	0.94

Table 4: Classification performance of equivalent models trained on both our proposed feature vector and VGGish embeddings.

The results indicate that in the context of texture sounds, summary statistics are strictly more informative than general-purpose embeddings such as VGGish.

5.4. Texture resynthesis using `TexEnv`

Extensive exploration using the `TexEnv` synthesizer in resynthesis tasks employing a signal processing-based parameter extractor was conducted to better understand its limitations and overall behavior. A summary of sound examples can be found on this article's webpage. Some of our key findings were as follows: water-like sounds such as flowing water, rain, and continuous bubbling do not benefit from larger parameter sets, but do benefit from larger filterbanks. In contrast, crackling sounds like fireworks or bonfires benefit from larger parameter sets, but not as much from larger filterbanks. These insights were crucial in determining the optimal parameters for the models trained later.

5.5. `TexDSP` Trained Models

To showcase the capabilities of `TexStat`, we trained a set of `TexDSP` models using different parameters, with `TexStat` as the sole loss function to guide the learning process. The details and results for some of these models are presented below.

Training Details: A curated set of sounds representing different classes of texture sounds from Freesound was used for each model. Each model employed different parameters tailored to the specific texture type. These parameters were chosen based on the resynthesis exploration discussed in Subsection 5.4. The number of layers in the MLPs for both the encoder and decoder was limited to a maximum of 3, with the number of parameters capped at 512. This configuration ensured that the resulting models, even when combined with the seed used for the filterbank, remained under 25 MB and could, if necessary, be ported to a real-time environment. The `TexStat` α and β parameters were set to the default values proposed in our repository, and all models used the same optimizer, training for up to 1500 epochs with early stopping enabled. Additionally, for each `TexDSP` model trained, a corresponding NoiseBandNet model was also trained using default parameters for comparison.

Validation Method: To evaluate model performance, we resynthesized a subset of the dataset, excluded from training, for both the `TexStat` and NoiseBandNet models. We then segmented the original and resynthesized signals, and measured Fréchet Audio Distance (FAD) using both VGGish embeddings and our custom summary statistics, along with frame-level `TexStat` and MSS losses. For the latter, we report the mean and standard deviation across all segments. Results are presented in Table 5.

	FAD				Loss metrics			
Texture Sound	VGGish TexDSP	VGGish NoiseBandNet	Ours TexDSP	Ours NoiseBandNet	TexStat TexDSP	TexStat NoiseBandNet	MSS TexDSP	MSS NoiseBandNet
Bubbles	35.20	**21.37**	1.86	**1.15**	1.2 ± 0.3	$\mathbf{0.7 \pm 0.1}$	6.6 ± 0.3	$\mathbf{4.7 \pm 0.1}$
Fire	11.86	**2.53**	6.14	**1.52**	2.8 ± 2.1	$\mathbf{1.7 \pm 1.0}$	9.6 ± 1.3	$\mathbf{4.5 \pm 0.2}$
Keyboard	13.02	**9.70**	**16.64**	277.12	$\mathbf{5.7 \pm 2.0}$	20.0 ± 7.7	$\mathbf{9.1 \pm 0.7}$	13.8 ± 0.6
Rain	**9.09**	11.31	**0.98**	6.19	$\mathbf{0.5 \pm 0.2}$	2.4 ± 2.0	$\mathbf{9.0 \pm 0.2}$	9.1 ± 0.4
River	**43.66**	49.85	**0.80**	1.75	$\mathbf{0.5 \pm 0.1}$	0.6 ± 0.1	$\mathbf{6.0 \pm 0.6}$	6.7 ± 0.3
Shards	4.64	**1.36**	**3.79**	7.58	$\mathbf{1.0 \pm 0.2}$	1.1 ± 0.3	$\mathbf{7.9 \pm 0.2}$	8.8 ± 0.2
Waterfall	**18.23**	25.88	**0.53**	1.06	$\mathbf{0.3 \pm 0.0}$	0.4 ± 0.0	$\mathbf{5.0 \pm 0.0}$	6.3 ± 0.0
Wind	**9.66**	31.35	**1.95**	8.48	$\mathbf{0.8 \pm 0.5}$	1.1 ± 0.7	$\mathbf{5.6 \pm 0.1}$	5.8 ± 0.2

Table 5: Validation metrics for both a TexDSP and a NoiseBandNet model trained on different textures. FAD metrics (lower is better) use VGGish and our proposed feature vector. Additionally, TexStat and MSS loss metrics are reported with means and standard deviations. In all metrics smaller is better and the best performer is highlighted in bold. (1)Energy bands were imposed post-resynthesis. (2)A loudness tracker was added post-resynthesis.

Results: The results highlight three key observations. First, performance varied across models, reflecting patterns observed in McDermott and Simoncelli's work and aligning with the limitations discussed in Subsection 2.4. Second, although some models performed adequately, their scores remained lower than those of the reconstruction focused model NoiseBandNet. This outcome is expected, as our approach prioritizes the disentanglement of higher-level sound structures over precise reconstruction, an aspect favored by the evaluation metrics used. The latter being said, surprisingly some TexDSP models managed to beat its counterpart even in these metrics. Third, the metrics derived from our models appear to align more closely with our own perception of sound quality. However, to support this claim more robustly, a subjective evaluation would be necessary—an analysis that was beyond the scope of this work.

Additional Comments: A widely recognized application of the original DDSP model was timbre transfer [21], where features such as pitch and loudness from an unexpected input sound are used to drive a model trained on a different instrument's timbre. For instance, applying the features of a voice recording to a violin-trained model will generate output that sounds like a violin playing the same melody. This effect is primarily due to the strong inductive bias of the model, which is trained exclusively on violin sounds and thus can only synthesize violin-like audio. Since the model operates on extracted features rather than raw audio, it naturally generates ("transfer") the timbre it was trained to pitch and loudness content extracted from the source sound, though this effect diminishes when pitch is less relevant.

In the models developed in this article, the same principle applies, though with less clear-cut results. These models retain the DDSP architecture's bias but are designed for a wider range of sounds. For example, a fire sound can be processed by a water-trained model, resulting in a form of timbre transfer. However, unlike the pitched case, the results are harder to interpret because pitch and loudness are more meaningful for musical sounds. For texture-based sounds, meaningful timbre transfer occurs only when the input and output share a key feature from the training data. While spectral centroid and rate might seem like viable features, they lack pitch's distinctiveness in musical contexts. Many examples of this effect are available on the article's webpage.

6. CONCLUSIONS

This paper introduced a novel framework for advancing the analysis and synthesis of texture sounds through deep learning. Central to our contribution is TexStat, a loss function grounded in auditory perception and statistical modeling. By explicitly encoding key properties such as time invariance, perceptual robustness, and long-term structural focus, TexStat provides a formulation that is better aligned with the inherent nature of texture sounds than conventional alternatives.

In addition to TexStat, we presented two complementary tools: TexEnv, a signal processor that efficiently generates texture audio via amplitude envelope imposition, and TexDSP, a DDSP-based model that demonstrates the effective integration of our proposed loss into a synthesizer framework.

Our experiments validated the theoretical motivations and practical utility of TexStat. Specifically, when used as a feature vector, the summary statistics derived from TexStat outperformed general-purpose embeddings like VGGish in classification tasks. Moreover, TexStat exhibited improved stability against transformations such as time shifting and noise addition, providing a perceptually coherent metric for evaluating resynthesis timbre similarity.

We also demonstrated the successful application of TexStat as a loss function in training the TexDSP model, where it guided the learning process to generate indefinitely long sequences of controllable texture sounds. Although the synthesized textures differed from the input sounds, they maintained the essential perceptual qualities that define their type. Despite its strengths, we acknowledge limitations in handling pitched or rhythmically structured content, suggesting that TexStat is most effective when combined with other losses as a regularization component or used as an evaluation metric.

Future work should extend TexStat to hybrid tasks by incorporating additional loss terms for pitched and/or rhythmic sounds, and explore its applications in generative modeling, texture transformation, and neural sound design. Overall, this framework represents a promising step toward achieving better perceptual alignment in machine listening and synthesis tasks.

7. ACKNOWLEDGMENTS

This work has been supported by the project "IA y Música: Cátedra en Inteligencia Artificial y Música (TSI-100929-2023-1)", funded by the "Secretaría de Estado de Digitalización e Inteligencia Artificial and the Unión Europea-Next Generation EU".

8. REFERENCES

[1] Bela Julesz, "Visual pattern discrimination," *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 84–92, Feb. 1962.

[2] Terry Caelli and Bela Julesz, "On perceptual analyzers underlying visual texture discrimination: Part i," *Biological Cybernetics*, vol. 28, no. 3, pp. 167–175, Sep. 1978.

[3] Anne Humeau-Heurtier, "Texture feature extraction methods: A survey," *IEEE Access*, vol. 7, pp. 8975–9000, 2019.

[4] Josh H. McDermott, Andrew J. Oxenham, and Eero P. Simoncelli, "Sound texture synthesis via filter statistics," in *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October 2009.

[5] Josh H. McDermott and Eero P. Simoncelli, "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis," *Neuron*, vol. 71, no. 5, pp. 926–940, 2011.

[6] Josh H. McDermott, Michael Schemitsch, and Eero P. Simoncelli, "Summary statistics in auditory perception," *Nature Neuroscience*, vol. 16, no. 4, pp. 493–498, 2013.

[7] Nicholas Saint-Arnaud, "Classification of sound textures," M.S. thesis, Massachusetts Institute of Technology, Cambridge, MA, Sep. 1995.

[8] D.F. Rosenthal, H.G. Okuno, H. Okuno, and D. Rosenthal, *Computational Auditory Scene Analysis: Proceedings of the Ijcai-95 Workshop*, CRC Press, 1st edition, 1998.

[9] Lonce Wyse, "An audio texture lutherie," *Annual Art Journal*, vol. 43, no. 54, 2022.

[10] Garima Sharma, Karthikeyan Umapathy, and Sridhar Krishnan, "Trends in audio texture analysis, synthesis, and applications," *J. Audio Eng. Soc.*, vol. 70, no. 3, pp. 108–127, March 2022.

[11] Nicolas Saint-Arnaud and Kris Popat, "Analysis and synthesis of sound textures," in *Readings in Computational Auditory Scene Analysis*, pp. 125–131. 1995.

[12] S. Dubnov, Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman, "Synthesizing sound textures through wavelet tree learning," *IEEE Computer Graphics and Applications*, vol. 22, no. 4, pp. 38–48, 2002.

[13] Diemo Schwarz, *Data-Driven Concatenative Sound Synthesis*, Ph.D. thesis, Université Paris 6 – Pierre et Marie Curie, Paris, France, 2004.

[14] Agostino Di Scipio, "The synthesis of environmental sound textures by iterated nonlinear functions, and its ecological relevance to perceptual modeling," *Journal of New Music Research*, vol. 31, no. 2, pp. 109–117, 2002.

[15] James F. O'Brien, Chen Shen, and Christine M. Gatchalian, "Synthesizing sounds from rigid-body simulations," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2002, pp. 175–181.

[16] Hugo Caracalla and Axel Roebel, "Sound texture synthesis using convolutional neural networks," in *Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19)*, Birmingham, UK, September 2019.

[17] Adrián Barahona-Ríos and Tom Collins, "Noisebandnet: Controllable time-varying neural synthesis of sound effects using filterbanks," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 32, pp. 1573–1585, Feb. 2024.

[18] Brian C. Moore and Brian R. Glasberg, "Suggested formulae for calculating auditory-filter bandwidths and excitation patterns," *The Journal of the Acoustical Society of America*, vol. 74, no. 3, pp. 750–753, 1983.

[19] Roy Patterson, Ian Nimmo-Smith, John Holdsworth, and Peter Rice, "An efficient auditory filterbank based on the gammatone function," in *Speech-Group Meeting of the Institute of Acoustics on Auditory Modelling*, RSRE, Malvern, Dec. 1987, Institute of Acoustics, Meeting held on December 14–15, 1987.

[20] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi, "Frechet audio distance: A reference-free metric for evaluating music enhancement algorithms," in *INTERSPEECH 2019*, Graz, Austria, September 2019, ISCA.

[21] Jesse Engel, Lalit Hantrakul, Chenjie Gu, and Adam Roberts, "Ddsp: Differentiable digital signal processing," in *International Conference on Learning Representations*, 2020.

[22] Xavier Serra and Julius Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[23] Frederic Font, Gerard Roma, and Xavier Serra, "Freesound technical demo," in *Proceedings of the 21st ACM International Conference on Multimedia*, New York, NY, USA, 2013, MM '13, p. 411–412, Association for Computing Machinery.

[24] Lonce Wyse and Prashanth Thattai Ravikumar, "Syntex: parametric audio texture datasets for conditional training of instrumental interfaces.," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, The University of Auckland, New Zealand, jun 2022.

[25] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson, "Cnn architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 131–135.