Differentiable Short-Time Fourier Transform: A Time-Frequency Layer with Learnable Parameters

Maxime Leiber [1] **Yosra Marnissi** [2] **Axel Barrau** [3]

Abstract

We present a differentiable version of the short-time Fourier transform (STFT), enabling gradient-based optimization of its parameters. This approach integrates with neural networks, allowing joint learning of both STFT and network parameters. Tests on simulated and real data demonstrate an improved time-frequency representation and enhanced performance on downstream tasks, illustrating the potential of our method as a standard for setting spectrogram parameters automatically.

1. Introduction

Context: The short-time Fourier transform (STFT) is a widely used technique for analyzing non-stationary signals in a variety of fields, including audio processing [1] and medical applications [2]. STFT-based representations such as spectrograms are popular tools for visualizing and processing signals. They are often used to visualize the frequency content of music [3] or speech [4]. They can also be combined with other processing methods such as neural networks to perform advanced tasks. They have been extensively utilized in various tasks, including speech recognition [5], [6], music detection [7], and data augmentation [8] etc. The combination of neural networks and spectrograms has shown remarkable success in these applications.

Motivation: Nevertheless, accurate STFT-based representations require careful parameter selection. The *window length* determines the trade-off between temporal and frequency resolution: shorter windows offer better time resolution but suffer in frequency resolution, and vice versa. The selection of an appropriate window length depends on the specific characteristics of the signal and the application requirements. (e.g., transient events favor shorter windows and slowly varying components favor longer windows). Similarly, the *hop or overlap length* (window shift between frames) controls the trade-off between smooth frequency evolution and computational cost. It also affects frame positioning relative to signal components. Misalignment can lead to energy leakage in the spectrogram, highlighting the importance of aligning frames with signal components to capture the signal's temporal dynamics. These parameters are crucial and influence the accuracy and interpretability of the STFT representation [9], [10].

Contributions: This paper gathers and extends previous works [11]–[15] by proposing a gradient-based optimization approach for tuning STFT parameters, paving the way for a standard approach in neural networks (STFT as a learnable layer). Our gradient-based approach avoid exhaustive search of traditional discrete approaches. In addition, continuous parameters allow for more precise optimization of time-frequency resolution and window positioning. Our key contribution lies in modifying the STFT definition to make window length and hop size continuous and the STFT representation differentiable with respect to these parameters. The key ideas are to decompose the window length L into an integer *numerical window support* N and a continuous *time resolution* θ, and to use the continuity of the tapering function to differentiate with respect to the temporal position of the frames.

Outline: The paper is organized as follows. In Section 2 we present the STFT, introduce the differentiable STFT with respect to both the window and hop lengths. In Section 3 we propose two optimization approaches for the STFT parameters: representation learning and end-to-end learning and we demonstrate the effectiveness of our approach through several applications.

2. Differentiable short-time Fourier transform

2.1. Short-time Fourier transform

The STFT is a two-step operation that consists of windowing an input signal into segments and computing the discrete Fourier Transform (DFT) on each of these chunks, resulting in a two-dimensional complex matrix $\mathcal{S}_{\omega_L}[i, f]$, where i is an integer expressing the starting frame index and f the associated frequency. The concept involves sliding a window

[1]Safran Tech, Châteaufort, France [2]Capital Fund Management, Paris, France [3]Offroad, Alfortville, France. Correspondence to: Maxime Leiber <maxime.leiber0@gmail.com>.

Published at the 2nd Differentiable Almost Everything Workshop at the 41st International Conference on Machine Learning, Vienna, Austria. July 2024. Copyright 2024 by the author(s).

over the signal $s[t]$ to be analyzed, selecting a specific time interval within the signal. At each position of the window, a Fourier transform is computed, representing the frequency content within that particular time span. It is common to multiply these signal slices by an analysis window of unit norm, which is a smooth function called *tapering function*. Let ω_L denote the tapering function, the STFT of $s[t]$ can be then explicitly written as:

$$\begin{aligned}\mathcal{S}_{\omega_L}[i,f] &= \mathcal{F}\left(\omega_L s\left[t_i : t_i + L - 1\right]\right)[f] \\ &= \sum_{k=t_i}^{t_i+L-1} \omega_L[k-t_i] s[k] e^{\frac{-2j\pi kf}{L}}\end{aligned} \tag{1}$$

In (1), each column $\mathcal{S}_{\omega_L}[i,:]$ is the DFT $\mathcal{F}(.)[f]$ of a tapered chunk $\omega_L s\left[t_i : t_i + L - 1\right]$ of length L of the signal s, starting from an index t_i to an index $t_i + L - 1$. The time indices t_i of signal intervals on which spectra are computed are usually equally spaced, so we only have to set the first index t_0 and the spacing $H = t_{i+1} - t_i$, known as the *hop length*, between t_i and t_{i+1}.

2.2. Differentiable STFT with respect to window and hop lengths

Defining a differentiable STFT (DSTFT) with respect to its parameters (i.e., the window length and window temporal position) involves modifying the definition of the STFT operator to obtain a formula similar to Equation (1), where L and t_i are continuous parameters and $\mathcal{S}_{\omega_L}[i,f]$ is differentiable with respect to L and t_i. Note that differentiating the STFT with respect to the frame temporal position (or window temporal position) is equivalent to differentiating $\mathcal{S}_{\omega_L}[i,f]$ with respect to the hop length, which is the difference between two consecutive frames.

There are some challenges to overcome in order to make the STFT differentiable with respect to these parameters. In fact, L appears in the formula (1) in the lower bound of the sum and as denominator in the exponential. In both cases, L must be an integer as we can only sum over integers along the length of the tapering window and the values ranging from t_i to $t_i + L - 1$ must be uniformly distributed on the unit circle in the complex exponential.

To overcome these challenges, i.e., window length upper bound of the sum and denominator of the complex exponential, we decompose L into two parameters:

- An integer *window support* N (upper bound),
- A continuous *time resolution* θ (learnable parameter).

The window support will be fixed and serve to define an upper bound of the tapering window length. We can now have a sum over integers namely over an integer valued support of the analysis window. The next step is to introduce a tapering function that is differentiable with respect to the window lengths and temporal positions of the windows.

Let $\omega(x,\theta) : \mathbb{R}\times]0,+\infty[\to \mathbb{R}_+$ be a non-negative function that is continuous and differentiable with respect to both of its parameters (x,θ) and with compact support such that:

$$\forall x \notin [0,\theta], \quad \omega(x,\theta) = 0 \tag{2}$$

An example of such a function is the common Hann function given by (see Figure 1):

$$\omega(x,\theta) = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi x}{\theta}\right)\right) 1_{0\le x\le\theta} \tag{3}$$

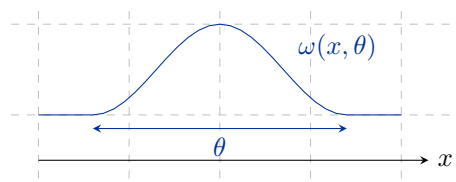

Figure 1. An example of tapering function : the Hann window

Another common example is the Gaussian function centred on $\theta/2$ and with standard deviation $\theta/3$:

$$\omega(x,\theta) = \frac{3}{\sqrt{2\pi\theta^2}} \exp\left(-\frac{9(x-\theta/2)^2}{2\theta^2}\right) 1_{0\le x\le\theta} \tag{4}$$

In the latter expression of the Gaussian window, we set the standard deviation to be equal to $\theta/3$ because we assumed that the Gaussian function is approximately zero outside of the interval centered around the mean and with a half-length of three times the standard deviation i.e, $3\times\theta/3 = \theta$. Then, θ can be interpreted as the length of the window function.

Let $N > 0$ be a positive integer and define the translated version of ω on the first variable x denoted by:

$$\omega_N(x,\theta) = \omega\left(x + \frac{1}{2}(\theta - N + 1), \theta\right) \tag{5}$$

From Equation (2), we obtain:

$$\begin{aligned}\forall\theta \in]0, N-1[, \forall x \notin]\frac{1}{2}(N-1-\theta), \frac{1}{2}(N-1+\theta)[, \\ \omega_N(x,\theta) = 0\end{aligned} \tag{6}$$

We can then rewrite the STFT in Equation (1) as follows:

$$\mathcal{S}_{\omega_N}[i,f] = \sum_{k\in\mathbb{Z}} \omega_N(k - t_i, \theta_{i,f}) s[k] e^{-\frac{2j\pi}{N}kf} \tag{7}$$

where $\theta_{i,f}$ and t_i are continuous window length and temporal positions variables respectively.

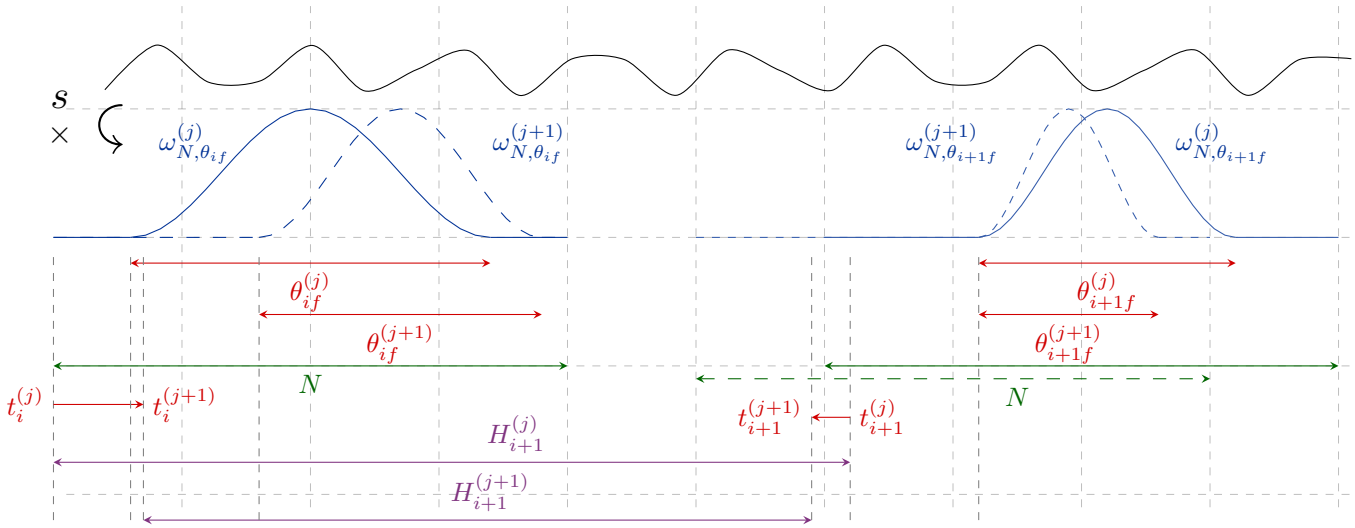

Figure 2. Differentiable STFT: The window support N of the subsignal on which DFT is computed is fixed, while on other hand, the temporal resolution θ_{if} of the tapering function $\omega_N(k - \{t_i\}, \theta\tau f)$, which actually determines time resolution is allowed to vary. Additionally, the position of the tapering windows can smoothly shift along the time axis, while the window supports start at the integer part of the temporal position of the tapering windows. The exponent j denotes the iteration in the gradient descent optimizer.

Proposition 2.1. *The modified STFT defined in Equation (7) is differentiable with respect to both the window and hop lengths (or window temporal positions).*

Proof. The modified STFT in Equation (7) is a finite sum since function $x \to \omega_N(x, \theta)$ has support contained in $]\frac{1}{2}(N - 1 - \theta), \frac{1}{2}(N - 1 + \theta)[\subset [0, N - 1]$. We can thus differentiate $\mathcal{S}_{\omega_N}$ with respect to $\theta_{i,f}$ and t_i as all terms involved are differentiable, namely tapering function is differentiable with respect to its first and second parameter, and exponential is differentiable too. Since the hop lengths H_i are defined as the difference between the starting points of two consecutive window temporal positions $H_i = t_i - t_{i-1}$, the STFT is also differentiable with respect to the hop lengths. □

The time resolution parameter $\theta_{i,f}$ has a meaning similar to that of L: it sets the time length of the signal chunk on which a local spectrum is computed. The difference with the classical STFT is that the slice is filled with zeros in order to always be of size N. As a result, the maximum frequency resolution (number of frequency bins) of the local spectra is no longer controlled by $\theta_{i,f}$ but by the second parameter $N > \theta$. This frequency resolution is unrealistic because the signal was padded with zeros, which amounts to interpolating in the frequency domain. Increasing N does not bring more information. In fact, N should only be considered as an upper bound on the temporal resolution $\theta_{i,f}$ that allows differentiation (see Figure 2).

In our differentiable STFT definition, all frames and frequencies have their own window length value, and all frames have their own hop length. It is however possible to share same values of parameters around one or both axes in the time-frequency plane. For instance, one can assume frequency-only varying window θ_f as for the S-transform, time-only varying window θ_i or time-and-frequency varying θ as for the Gabor transform or in the classical STFT case. The same remark is valid for the window temporal position. Note that the classical STFT defined in (1) can be obtained by setting $\theta_{i,f} = L$ and $H_i = H \in \mathbb{N}$.

In Appendix A we provide all the gradient and backpropagation formulas. In Appendix B we discuss computational complexity and specify numerical implementation.

3. Applications

Our proposed DSTFT can be of particular interest in two types of applications.

3.1. Representation Learning

The objective is to find optimal STFT parameters (window and hop lengths) for a specific signal, in order to adapt STFT parameters to each different signal. To define optimality, we establish criteria translating desired representation properties into objective functions. Common criteria from literature include energy concentration i.e. sparsity such as norm ratios ℓ_p-over-ℓ_q [16] and the Renyi entropy [17]. In Appendix C we performed two experiments for optimal representation. The first experiment optimized a time- and

frequency-varying window length. The second experiment optimized a time-varying window and hop lengths.

3.2. End-to-End Learning

A promising advantage of DSTFT is the capability to optimize window and hop lengths within neural networks. Unlike traditional methods with fixed parameters, our approach treats the DSTFT as a learnable layer with trainable weights representing these parameters. This allows for joint optimization of all parameters alongside the network for tasks like classification or regression. Notably, this is applicable beyond neural networks to any learning algorithm using STFT-based representations. It is worth to note that the optimal representation for learning tasks might differ from what's ideal for visualization. In the following, we will investigate the joint optimisation of the window length with the weights of a convolutional neural network for a spoken digit classification task.

Joint optimization with a neural network: In this example, we show how DSTFT can be easily integrated into existing neural networks. We train a convolutional neural network to classify spoken digits from the Free Spoken Digit Dataset (FSDD), also called the audio MNIST. The goal is to find the best window size for the spectrogram and the best network weights together in a joint optimisation by minimizing the cross-entropy with the ground truth. To evaluate the impact of window size on network performance, we trained a 2-layer convolutional neural network (CNN) with 16 filters per layer. Depthwise separable convolutions were employed to achieve a balance between efficiency and accuracy. The network utilized ReLU (REctified Linear Unit) activation functions for hidden layers, dropout for regularization to prevent overfitting, and the Adam optimizer for efficient training. We see in Table 1, that the window size parameter is very important because the accuracy of the same trained network for different values of the window size varies strongly. Also, whatever the initial value of time resolution is, during a joint optimization with network weights, the time resolution window length parameter converges to an optimal value as shown in the first row of Figure 3. Indeed, losses decrease by jointly optimizing the weights of the neural network and the time resolution continuous parameter θ that converged to an optimal value: at the end of the gradient descent, the window length reached the value 34.9 whereas we started the training with a window size of 200. The second row of Figure 3 shows the spectrogram of a sample at the end of the convergence. We note that the obtained spectrogram is not the best for visualization but the best representation for the neural network.

This simple simulation proves the effectiveness of our backpropagation procedure based on a differentiable version of STFT. It illustrates a general window length tuning methodology applicable to any existing signal processing algorithm or neural network involving spectrograms: replace the spectrogram computation step by the computation of our differentiable spectrogram, then optimize the window length by gradient descent based on the backpropagation formulas we have introduced.

Table 1. Training, validation and testing losses of CNN per fixed window length.

window length	train loss	val loss	test loss
10	0.3672	1.1442	1.0391
20	0.0304	0.4443	0.3177
30	0.0089	0.2408	0.0306
40	0.0064	0.2550	0.1249
50	0.0061	0.4859	0.2868

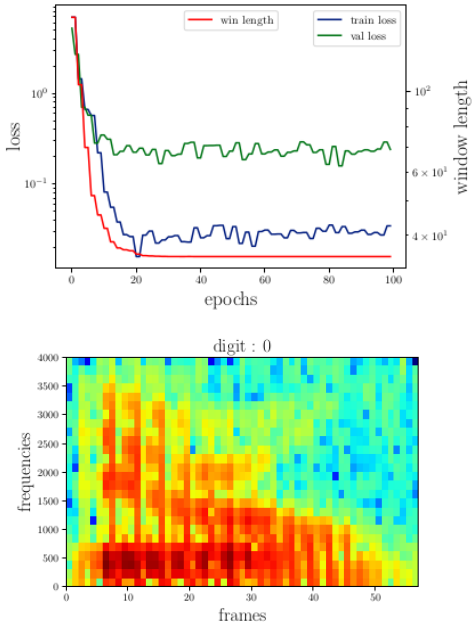

Figure 3. Training loss, validation loss and window length per epoch (left) and Spectrogram of a sample obtained at the end of the training (right)

4. Conclusion

We have defined a differentiable version of the STFT, which is differentiable with respect to its parameters, enabling to optimize by gradient descent the window lengths and the window temporal positions. This capability has the potential to establish this approach as a standard method for parameter selection in STFT and STFT-based representations. This contribution aligns with the ongoing efforts within the machine learning community to provide differentiable relaxations of various traditional tools for efficient optimization with gradient-based algorithms.

References

[1] K. M. Stafford, C. G. Fox, and D. S. Clark, "Long-range acoustic detection and localization of blue whale calls in the northeast pacific ocean," *The Journal of the Acoustical Society of America*, vol. 104, no. 6, pp. 3616–3625, 1998.

[2] J. Huang, B. Chen, B. Yao, and W. He, "Ecg arrhythmia classification using stft-based spectrogram and convolutional neural network," *IEEE access*, vol. 7, pp. 92 871–92 880, 2019.

[3] A. Klapuri and M. Davy, "Signal processing methods for music transcription," 2007.

[4] B. E. Kingsbury, N. Morgan, and S. Greenberg, "Robust speech recognition using the modulation spectrogram," *Speech communication*, vol. 25, no. 1-3, pp. 117–132, 1998.

[5] Y. Gong, Y.-A. Chung, and J. Glass, "Ast: Audio spectrogram transformer," *arXiv preprint arXiv:2104.01778*, 2021.

[6] A. M. Badshah, J. Ahmad, N. Rahim, and S. W. Baik, "Speech emotion recognition from spectrograms with deep convolutional neural network," in *2017 international conference on platform technology and service (PlatCon)*, IEEE, 2017, pp. 1–5.

[7] J. Schlüter and S. Böck, "Improved musical onset detection with convolutional neural networks," in *2014 ieee international conference on acoustics, speech and signal processing (icassp)*, IEEE, 2014, pp. 6979–6983.

[8] D. S. Park, W. Chan, Y. Zhang, *et al.*, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[9] L. R. Rabiner, R. W. Schafer, *et al.*, "Introduction to digital speech processing," *Foundations and Trends® in Signal Processing*, vol. 1, no. 1–2, pp. 1–194, 2007.

[10] C. García Ruíz, J. M. Martín Doñas, Á. M. Gómez García, *et al.*, "The role of window length and shift in complex-domain dnn-based speech enhancement," 2022.

[11] M. Leiber, A. Barrau, Y. Marnissi, and D. Abboud, "A differentiable short-time fourier transform with respect to the window length," in *European Signal Processing Conference (EUSIPCO)*, 2022, pp. 1392–1396.

[12] M. Leiber, Y. Marnissi, A. Barrau, and M. El Badaoui, "Differentiable adaptive short-time fourier transform with respect to the window length," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[13] M. Leiber, Y. Marnissi, A. Barrau, and M. El Badaoui, "Differentiable short-time fourier transform with respect to the hop length," in *IEEE Workshop on Statistical Signal Processing (SSP)*, 2023.

[14] M. Leiber, A. Barrau, Y. Marnissi, D. Abboud, and M. El Badaoui, "Optimisation de la longueur de fenêtre du spectrogramme au sein d'un réseau de neurones," 2022.

[15] A. Zhao, K. Subramani, and P. Smaragdis, "Optimizing short-time fourier transform parameters via gradient descent," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 736–740.

[16] S. Pei and S. Huang, "STFT with adaptive window width based on the chirp rate," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4065–4080, 2012.

[17] S. Meignen, M. Colominas, and D. Pham, "On the use of rényi entropy for optimal window size computation in the short-time fourier transform," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 5830–5834.

[18] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[19] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[20] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *Journal of Marchine Learning Research*, vol. 18, pp. 1–43, 2018.

[21] E. Stevens, L. Antiga, and T. Viehmann, *Deep learning with PyTorch*. Manning Publications, 2020.

[22] G. Gilboa and S. Osher, "Nonlocal operators with applications to image processing," *Multiscale Modeling & Simulation*, vol. 7, no. 3, pp. 1005–1028, 2009.

A. Gradients and backpropagation formulas

We introduced the DSTFT that admit continuous parameters namely the window length and the frame position (or equivalently the hop length). We showed that the DSTFT is differentiable with respect to its parameters. This means that $\frac{\partial \mathcal{S}_{\omega_N}[i,f]}{\partial \theta_{i,f}}$ and $\frac{\partial \mathcal{S}_{\omega_N}[i,f]}{\partial t_i}$ exist and are finite. In this section, we start by giving the analytical expression of these derivatives and show how these formulas can be integrated in more general applications. Finally, we specify numerical implementation.

A.1. Analytical formulas for DSTFT gradients

In the following, we will denote by $\partial_x \omega_N$ and $\partial_\theta \omega_N$ the derivatives of ω_N in Equation 5 with respect to the first and second variables, respectively. Let us compute the differential of our proposed DSTFT with respect to the window length $\theta_{i,f}$, frame temporal position t_i, and hop length H_i. $\mathcal{S}_{\omega_N}[i,f]$ being complex, we apply the term-by-term differentiation by considering complex numbers as vectors with two real components, in particular $\exp(jx) \equiv [\cos(x), \sin(x)]$.

A.1.1. Gradient with respect to window length

Time-and-frequency varying window length First, we consider the case of a time-and-frequency varying window length $\theta_{i,f}$. By compute the partial derivative of each value in the time-frequency plane with respect to the window length $\theta_{i,f}$, we obtain the Jacobian of size (2, 1) as follows :

$$\frac{\partial \mathcal{S}_{\omega_N}[i,f]}{\partial \theta_{i,f}} = \sum_{k \in \mathbb{Z}} \frac{\partial \omega_N(k - t_i, \theta_{i,f})}{\partial_{\theta_{i,f}}} s[k]\, e^{-\frac{2j\pi kf}{N}} \tag{8}$$

where we recognize the STFT of s where the tapering function $\omega_N(k, \theta_{i,f})$ is replaced by the function

$$\bar{\omega}_N(k, \theta_{i,f}) = \partial_\theta \omega_N(k, \theta_{i,f}) \tag{9}$$

Then, we can write:

$$\frac{\partial \mathcal{S}_{\omega_N}[\eta,f]}{\partial \theta_{i,f}} = \mathcal{S}_{\bar{\omega}_N}[i,f] \mathbf{1}_i(\eta) \mathbf{1}_f(f) \tag{10}$$

where $\mathbf{1}_a(.)$ is the Dirac function in a.

Shared window length In the case of shared window length, the derivative of the STFT with respect to window length θ writes

$$\frac{\partial \mathcal{S}_{\omega_N}[\eta,f]}{\partial \theta} = \mathcal{S}_{\bar{\omega}_N}[\eta,f] \tag{11}$$

A.1.2. Gradient with respect to window position and hop length

The derivative of the STFT with respect to window position writes

$$\frac{\partial \mathcal{S}_{\omega_N}[i,f]}{\partial t_i} = \sum_{k \in \mathbb{Z}} \tilde{\omega}_N(k - t_i, \theta_{i,f}) s[k]\, e^{-\frac{2j\pi kf}{N}} \tag{12}$$

where

$$\tilde{\omega}_N(k, \theta_{i,f}) = \partial_x \omega_N(k, \theta_{i,f}) \tag{13}$$

This results in:

$$\frac{\partial \mathcal{S}_{\omega_N}[\eta,f]}{\partial t_i} = \mathcal{S}_{\tilde{\omega}_N}[i,f] \mathbf{1}_i(\eta). \tag{14}$$

Time-varying hop length Using the formula of a time-varying hop length $t_i = \sum_{t=0}^{i} H_t$, we can deduce that the derivatives for the hop length are given by:

$$\frac{\partial \mathcal{S}[\eta,f]}{\partial H_i} = \mathcal{S}_{\tilde{\omega}_N}[\eta,f] \mathbf{1}_{[0,\eta]}(\tau) \tag{15}$$

where $\mathbf{1}_{[0,\eta]}$ is the indicator function defined on the interval $[0, \eta]$.

Shared hop length Using the formula of a shared hop length $t_i = (i+1)H$, we can deduce that the derivatives for the hop length are given by:

$$\frac{\partial \mathcal{S}[\eta, f]}{\partial H} = (i+1)\mathcal{S}_{\tilde{\omega}_N}[\eta, f] \tag{16}$$

Remark A.1. One interesting feature of our DSTFT is that all derivative expressions share the same form as the forward pass. In other words, they can always be represented as a DSTFT transform step, by using modified tapering functions.

A.2. Backpropagation formulas

The above derivative expressions enable us to optimize any almost everywhere smooth differentiable scalar loss function with respect to DSTFT parameters. This is fulfilled by employing gradient descent and the backpropagation algorithm [18] and automatic differentiation (AD) [19], [20] two fundamental tools in machine learning.

Let $\mathcal{L}$ be the loss to be optimized. With the STFT and automatic differentiation tools, such as PyTorch [21], which allow for the automatic calculation of gradients, we can optimize a loss the DSTFT parameters using gradient descent. In this section, we provide the analytical expressions for backpropagation through the STFT and then we compare them to the ones computed numerically using PyTorch.

A.2.1. Analytical expressions

Let $\mathcal{L}$ be the loss function to be differentiated with respect to our DSTFT parameters.

Variable parameters In the general case of time-and-frequency varying window length and time-varying hop-length, we obtain:

$$\frac{\partial \mathcal{L}}{\partial \theta_{i,f}} = \frac{\partial \mathcal{L}}{\partial \mathcal{S}_{\omega_N}[i,f]} \frac{\partial \mathcal{S}_{\omega_N}[i,f]}{\partial \theta_{i,f}} = \frac{\partial \mathcal{L}}{\partial \mathcal{S}_{\omega_N}[i,f]} \mathcal{S}_{\tilde{\omega}_N}[i,f] \tag{17}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial t_i} &= \sum_{f=0}^{N-1} \frac{\partial \mathcal{L}}{\partial \mathcal{S}_{\omega_N}[i,f]} \frac{\partial \mathcal{S}_{\omega_N}[i,f]}{\partial t_i} \\ &= \sum_{f=0}^{N-1} \frac{\partial \mathcal{L}}{\partial \mathcal{S}_{\omega_N}[i,f]} \mathcal{S}_{\tilde{\omega}_N}[i,f] \end{aligned} \tag{18}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial H_i} &= \sum_{\eta=0}^{T-1} \sum_{f=0}^{N-1} \frac{\partial \mathcal{L}}{\partial \mathcal{S}_{\omega_N}[\eta,f]} \frac{\partial \mathcal{S}_{\omega_N}[\eta,f]}{\partial H_i} \\ &= \sum_{\eta=0}^{\tau} \sum_{f=0}^{N-1} \frac{\partial \mathcal{L}}{\partial \mathcal{S}_{\omega_N}[\eta,f]} \mathcal{S}_{\tilde{\omega}_N}[\eta,f] \end{aligned} \tag{19}$$

where $\frac{\partial \mathcal{L}}{\partial \mathcal{S}_{\omega_N}(i,f)}$ is the vector of derivatives with respect to real and imaginary parts of size (1, 2) and $\mathcal{S}$ is understood as a vector of size (2,1).

Shared parameters In the classical STFT case of window and hop length sharing, we obtain:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{i=0}^{T-1} \sum_{f=0}^{N-1} \frac{\partial \mathcal{L}}{\partial \mathcal{S}_{\omega_N}[i,f]} \frac{\partial \mathcal{S}_{\omega_N}[i,f]}{\partial \theta} = \left\langle \frac{\partial \mathcal{L}}{\partial \mathcal{S}_{\omega_N}}, \mathcal{S}_{\tilde{\omega}} \right\rangle, \tag{20}$$

$$\frac{\partial \mathcal{L}}{\partial H} = \sum_{i=0}^{T-1} \sum_{f=0}^{N-1} \frac{\partial \mathcal{L}}{\partial \mathcal{S}_{\omega_N}[i,f]} \frac{\partial \mathcal{S}_{\omega_N}[i,f]}{\partial H} = \left\langle \frac{\partial \mathcal{L}}{\partial \mathcal{S}_{\omega_N}}, \mathcal{S}_{\tilde{\omega}} \right\rangle \tag{21}$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius scalar product of two matrices : $\langle A, B \rangle = \sum_{i,j} \mathrm{Re}(A_{j,i})\mathrm{Re}(B_{i,j}) + \mathrm{Im}(A_{j,i})\mathrm{Im}(B_{i,j})$ with $\mathrm{Re}(X)$ and $\mathrm{Im}(X)$ the real part and imaginary part of X, respectively.

Remark A.2. These analytical expressions are highly flexible and enable differentiation of any scalar function $\mathcal{L}$ of the STFT's outputs with respect to its tuning parameters. In fact, all backpropagation computation can be performed efficiently

using matrix multiplications, where the matrices involved have the same dimension as the forward propagation, since individual derivatives are just DSTFT operations. In the following section, we will show that this approach reduces computational complexity over automatic differentiation tools, is conducive to faster gradient calculations, and use exact gradient values.

B. Computational complexity and numerical implementation

B.1. Computational complexity

Our proposed approach ($\mathcal{S}_{\omega_N}$) shares a computational space and time complexity of $O(N^2)$ with the classical STFT using the DFT. However, by leveraging the FFT computation, our approach can achieve a reduced complexity of $O(N \log N)$. This improved complexity is attainable when using only a shared window length $\theta_{i,f} = \theta$ or a time-varying window length $\theta_{i,f} = \theta_i$. This efficient computational complexity makes the time-varying window length approach particularly advantageous for real-time applications where fast processing is essential. On the other hand, for applications that do not have memory or time constraints but require higher precision with respect to the temporal positions of windows or the lengths of windows in frequency, the complexity is $O(N^2)$.

Regarding the gradient computation in Appendix A, the backward pass has a computational complexity equivalent to the forward pass, i.e., O(N log N) when using the FFT, otherwise, the complexity is $O(N^2)$.

B.2. Numerical implementation

To implement the DSTFT transformation, we need to modify the infinite sum to a finite one because the tapering function has limited support. The tapering function is nonzero only within the support interval $[0, N-1]$, so we perform the sum only over this interval. To achieve this, we make a variable transformation, so that k is an integer within the interval $[0, N-1]$. More specifically, we execute the following transformation: $k \to k + \lfloor t_i \rfloor$, taking advantage of the fact that $t_i = \lfloor t_i \rfloor + \{t_i\}$. Here, $\{t_i\}$ represents the fractional part of t_i, and $\lfloor t_i \rfloor$ is the integer part of t_i. Consequently, we obtain the following expression:

$$
\begin{aligned}
\mathcal{S}_{\omega_N}[i,f] &= \sum_{k\in\mathbb{Z}} \omega_N(k - t_i, \theta_{i,f}) s[k] e^{-\frac{2j\pi kf}{N}} \\
&= \sum_{k=0}^{N-1} \omega_N(k - \{t_i\}, \theta_{i,f}) s\left[\lfloor t_i \rfloor + k\right] e^{-\frac{2j\pi}{N}(k+\lfloor t_i \rfloor)f} \\
&= e^{-\frac{2j\pi \lfloor t_i \rfloor f}{N}} \sum_{k=0}^{N-1} \omega_N(k - \{t_i\}, \theta_{i,f}) s\left[\lfloor t_i \rfloor + k\right] e^{-\frac{2j\pi kf}{N}}
\end{aligned}
\tag{22}
$$

Therefore, to overcome the constraint of starting on integer values, we start the window at $\lfloor t_i \rfloor$ and compensate for the effect of the integer part by the combination of a small shift, $\{t_i\} = t_i - \lfloor t_i \rfloor$, in the argument of ω_N and a factor $e^{-2j\pi \lfloor t_i \rfloor f/N}$ in the exponential. Note that the above equation, we set the lower bound of the sum to be 0 and the upper bound to be $N-1$ because $0 \leq \{t_i\} < 1$, and the tapering function ω_N is zero at the boundaries of its support.

C. Additional experiments

C.1. Representation-driven optimization

In this section, we will demonstrate, through simulated and real signals, the immediate interest and significance of DSTFT, highlighting its potential for finding a good representation. The optimal settings for STFT, such as the choice of the best window and hop lengths, should be adapted to the specific input signal and its unique features. The DSTFT allows the window lengths to flexibly adapt locally to the signal's frequency content. This improves the ability to capture both time and frequency information accurately, enhancing the overall time-frequency resolution of the transformation. Second, it fine-tunes the hop lengths, which determine the temporal positions of the analysis windows. By aligning these positions with the frequency components of the input signal, energy leakage can be then minimized and short-time signals can be well localized in the spectrogram. That is why, the DSTFT can be seen as an adaptive time-frequency transform, automatically adjusting its parameters to track the unique characteristics of the input signal. To establish an optimal representation.

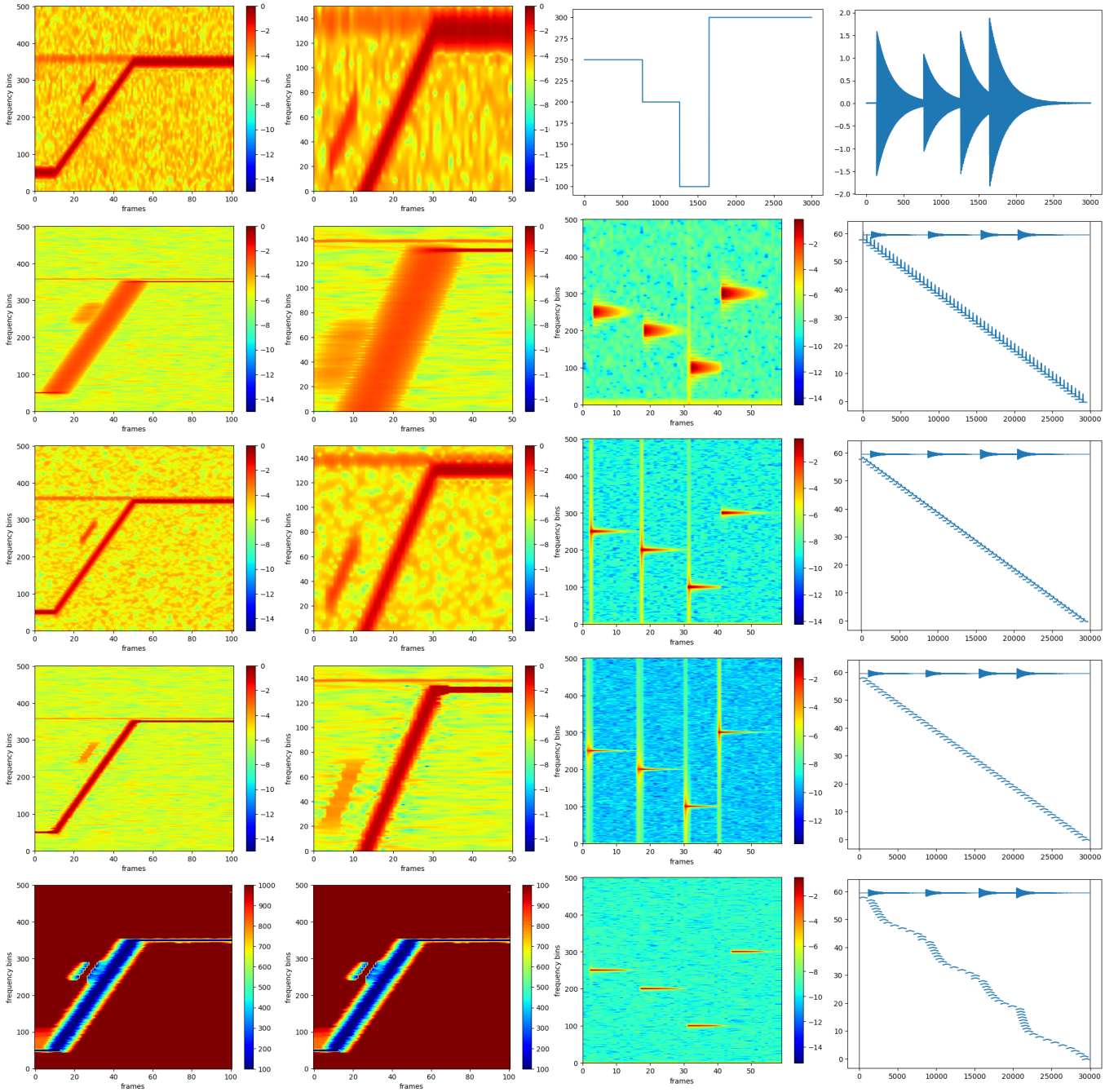

Figure 4. Spectrograms (left) and Zoomed-in spectrograms (right) with respectively from top to bottom small window of length 100, long window of length 1000, single-window DSTFT with a time-and-frequency-varying window length without (middle) and with (bottom) regularisation. Spectrograms (left) and distributions of window lengths (right).

Figure 5. Spectrograms (left) and Zoomed-in spectrograms (right) with respectively from top to bottom small window of length 100, long window of length 1000, single-window DSTFT with a time-and-frequency-varying window length without (middle) and with (bottom) regularisation. Spectrograms (left) and distributions of window lengths (right).

C.1.1. TIME AND FREQUENCY VARYING WINDOW LENGTH

We consider a simulated signal composed of 3 components: a main non-stationary component, a close and stationary component, and a transient signal. In this example, we focus on optimizing the window length. A small window length gives fine temporal resolution to localize the transient event in time but a coarse frequency resolution to distinguish nearby frequencies as displayed in the first row of Figure 4. In contrast, a long window length (in second row of Figure 4) enables to distinguish the main component from the stationary frequency but a poor temporal resolution of the transient event. We now optimize DSTFT with a single window length by minimizing the entropy loss $\mathcal{H}(\mathcal{S}_{\omega_N};\theta) = -\sum_{i,f} p_{if}\log(p_{if})$ where $p_{if} = \frac{\|\mathcal{S}_{\omega_N}[i,f]\|}{\sum_{k,l}\|\mathcal{S}_{\omega_N}[k,l]\|}$. At the end of the optimization, we obtain a spectrogram, displayed in the third row of Figure 4 that is optimal given the criterion. Our optimization is done in a number of iterations much lower than the grid-search case. In this specific case, for grid-search, we have tested all integer values between 100 and 1000, which is a fixed computational cost proportional to 900 forward steps. In the case of DSTFT, the number of iterations depends on the choice of hyperparameters, including the gradient descent optimizer hyperparameters (e.g. learning rate, stopping condition). It is therefore possible to perform the optimization between 10 and 100 iterations. Knowing that one iteration consists of a forward pass and a backward pass, and that a backward pass is exactly equivalent to a forward pass in terms of computational cost as we saw in the previous section, we obtain a variable computational cost between 20 and 200 iterations. This is much lower than traditional grid-search. We will see that the gap is even greater in the case of a window length that varies in time and frequency. In this case of single window length, DSTFT gives an interesting compromise between time and frequency resolution. But since we use only one window length for the whole spectrogram, we are not able to localize precisely in time transient event, and in frequency close frequencies. As we can see in this specific simulated example, it is difficult to find an optimal unique window length for all different components in the spectrogram. We need for different window lengths, more precisely a time-and-frequency-varying window length, to adapt to the time-varying spectral structure of the signal.

We now optimize DSTFT using time-and-frequency-varying window length by minimizing the following criteria:

$$\mathcal{L}(\Theta) = \mathcal{H}(\mathcal{S}_{\omega_N};\Theta) + \lambda\mathcal{R}(\Theta) \tag{23}$$

where $\mathcal{R}(\Theta)$ a regularization term that constrains nearby windows in the time-frequency plane to have close values, which smooths the distribution of window lengths and improves robustness to noise and λ is an hyperparameter controlling trade-off between these two terms. Particularly, we consider $\mathcal{R}(\Theta) = \sum_{i,f}\sqrt{\sum_{(j,k)\in V_{i,f}}\gamma_{(j,k);(i,f)}(\theta_{i,f}-\theta_{j,k})^2}$ where $V_{i,f}$ is a neighboring index set for the bin (i,f) and $\gamma_{j,k}$ is a weight given to the neighbor $\theta_{j,k}$ to measure the similarity between the frequency content of the regions (i,f) and (j,k). These weights can be equal or set from a previously computed spectrogram with a constant window length. The above regularization is related to the non-local total variation penalization used in image processing [22]. Results with DSTFT using time-and-frequency-varying window length are shown in fourth and fifth row of Figure 4 for spectrograms and window length distribution respectively. We see that we can localize precisely all the components of our signal both in time and in frequency, thanks to time-and-frequency-varying window length. In fact, the latter adapts to each component as expected e.g. small windows for transient events and long windows for stationary components.

C.1.2. TIME-VARYING WINDOW AND HOP LENGTHS

In this section, we will show on a simulated signal that differentiable STFT with respect to both window length and hop length (or frame temporal position) can be of immediate interest and deserves more attention. We consider a signal that simulates shocks of different frequencies and durations. We also added noise to the signal as shown in the first row of Figure 5. Classical STFT uses frames that are uniformly spread along the signal which may not be the optimal positioning to localize frequency changes as frequencies have a variable length. In fact, using small, medium, or long window size, where the frames are uniformly spaced along the signal, produce a spectrogram with a lot of energy leakage ad shown in the the second, third and fourth rows of Figure 5 respectively. We then consider optimizing the window positions and lengths by gradient descent using our DSTFT. More specifically, our goal is to find windows that are well-distributed over the overall signal, ensuring that there are no gaps and loss of information, and that allow for minimal energy leakage while achieving a better concentration of energy in the time-frequency plane. To promote energy concentration in each frame, we maximize the kurtosis of frame spectrum which is defined as $\mathcal{K}(\mathcal{S}_{\omega_N};\Omega) = \frac{1}{\sum_{i=0}^{T-1}\lambda_i}\sum_{i=0}^{T-1}\lambda_i\frac{\mathbb{E}_f[\|\mathcal{S}_{\omega_N}[i,f]\|^4]}{\mathbb{E}_f[\|\mathcal{S}_{\omega_N}[i,f]\|^2]^2}$ where λ_i is a weight used to minimize the contribution of windows that share the same segment of the signal. More specifically, we set $\lambda_0 = t_1 - t_0$, $\lambda_{T-1} = t_{T-1} - t_{T-2}$ and $\lambda_i = \frac{t_{i+1}-t_{i-1}}{2}$ for every $i \in \{1,\dots,T-2\}$. To avoid information loss, we also

want to maximize the spectrogram coverage defined as $\mathcal{C}(\Omega) = \frac{1}{M}\sum_{i=0}^{T-1} \min\left(\theta_i, \tilde{H}_{i+1}\right)$ with $\tilde{H}_T = +\infty$ and M is the signal length. Let us recall that in cases where the window's support extends beyond the signal, we zero-pad the signal for the calculation of the STFT. However, it is important to note that the coverage metric only takes into account the part of the window that overlaps with the signal. While we observe that the spectrogram obtained at the end of the optimization has a better concentration of energy due to reduced spectral leakage resulting from the proper positioning of frames, which are no longer straddling two different frequencies, as shown in the fifth row of Figure 5. However, caution must be exercised when reading the time-axis of the resulting time-frequency representation as the distribution of frames is no longer uniform.