# Efficient Neural and Numerical Methods for High-Quality Online Speech Spectrogram Inversion via Gradient Theorem

*Andres Fernandez*[1,2], *Juan Azcarreta*[2], *Çağdaş Bilen*[2], *Jesus Monge Alvarez*[3]

[1]Tübingen AI Center, University of Tübingen, Germany
[2]Reality Labs Research, Meta, UK
[3]Reality Labs Research, Meta, Spain

a.fernandez@uni-tuebingen.de, jsmalvarez@meta.com

## Abstract

Recent work in online speech spectrogram inversion effectively combines Deep Learning with the Gradient Theorem to predict phase derivatives directly from magnitudes. Then, phases are estimated from their derivatives via least squares, resulting in a high quality reconstruction. In this work, we introduce three innovations that drastically reduce computational cost, while maintaining high quality: Firstly, we introduce a novel neural network architecture with just 8k parameters, 30 times smaller than previous state of the art. Secondly, increasing latency by 1 hop size allows us to further halve the cost of the neural inference step. Thirdly, we propose a linear-complexity solver for the least squares step that leverages tridiagonality and positive-semidefiniteness, achieving a speedup of several orders of magnitude. We release samples online.

**Index Terms**: spectrogram inversion, online, gradient theorem, deep learning

## 1. Introduction

The Short-Term Fourier Transform (STFT) magnitudes of an audio waveform, also called *spectrograms*, are a widely used representation in speech processing tasks such as recognition [1], denoising [2], separation [3], enhancement [4], and synthesis [5]. In this paper, we focus on the task of converting a spectrogram into a waveform by first estimating the STFT phases directly from the magnitudes and then performing an Inverse STFT (ISTFT) —see e.g. [6, 7] for extended discussion on alternative strategies. Furthermore, we focus on the *online* setting, in which the phases at a given time frame $\tau_0$ can only be estimated from magnitudes at $\tau \leq \tau_0$, and more specifically a *real-time* setting, where computational resources are limited. We refer to this task as Real-Time Spectrogram Inversion (RSI).

RSI faces several challenges: many pipelines produce a *modified spectrogram* that may be inconsistent, thus a true phase may not even exist [8, 9]. And even if it exists, recovering the true phase is generally NP-complete [10], or attainable under conditions that are unsuited for RSI [11–13]. For this reason, most RSI methods resort to estimating the phase. Consistency-based RSI methods like RTISI [14] are signal-agnostic and straightforward, but also exposed to artifacts and require a potentially large number of iterations to work [15, 16]. Related gradient-based methods suffer from similar issues [17, 18]. Sinusoidal methods like SPSI [19] leverage knowledge about phases around local maxima [20], becoming iteration-free, but they are also exposed to artifacts due to the sinusoidal assumption [21]. The *Gradient Theorem (GT)* [22] brings together the best of all worlds by expressing the relation between all STFT magnitudes and phases in a signal-agnostic and online fashion, as efficiently leveraged by the RTPGHI algorithm [23, 24].

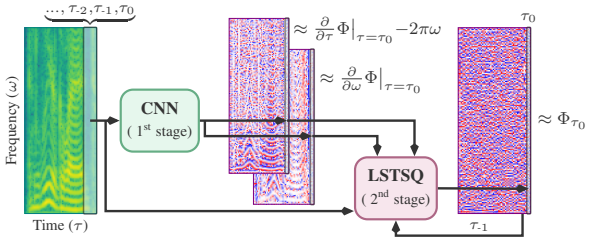



Figure 1: *Overview of the proposed pipeline, modified from [21]. A causal, lightweight CNN is trained to jointly map STFT log-magnitudes into FPD and BPD features (defined in Section 2) at time frame $\tau_0$. Then, phases ($\Phi$) are recursively estimated via complex least squares, computed in linear time w.r.t. number of frequency bins. The waveform can be then reconstructed via ISTFT. Additionally, the CNN can be modified to provide also the features for frame $\tau_{-1}$ at minimal overhead, thus requiring only half the forward passes at the price of 1 hop size of latency.*

In recent years, the introduction of Deep Learning (DL) techniques substantially increased output quality [25–27]. The main challenge here in the context of RSI is the increased computational cost and latency. To address the latter, [21] proposed an online framework in which a first stage approximates the GT equations via DL, followed by a second stage in which the phase is obtained from the DL output via complex least squares. This framework combines the low-latency, iteration-free strengths of GT with the high-quality output of DL, but is still affected by increased computation. In this work, we introduce three innovations that *drastically reduce the memory and arithmetic requirements of [21], at no cost in output quality*. Our contributions, highlighted in Figure 1, are the following:

1. A causal Convolutional Neural Network (CNN) architecture with only ∼8k parameters, ∼30× smaller and faster than [21] and with comparable performance.
2. A modified inference scheme to optionally reduce computation by an extra ∼2× at the cost of one hop in latency.
3. A characterization of the second stage as a tridiagonal and positive-semidefinite linear system, leading to orders of magnitude faster computation with guaranteed linear complexity.

These innovations allow us to achieve a speech spectrogram inversion solution that is *online, efficient and high-quality*. Section 2 provides further context. We describe our contributions in Section 3. In Section 4 we provide experiments to support our claims. Section 5 concludes discussing limitations and future work.

## 2. Background

### 2.1. Online Spectrogram Inversion via Gradient Theorem

Consider the STFT of a function $y(t)\in L^2(\mathbb{R})$ w.r.t. window $h(t)\in L^2(\mathbb{R})$, both real-valued:

$$Y_{y,h}(\omega,t) \coloneqq \int_{t'\in\mathbb{R}} y(t+t')\,h(t')\,e^{-2\pi i\omega t}dt', \quad \omega,t\in\mathbb{R} \quad (1)$$

If we take a Gaussian window in the form $\varphi_\lambda(t)\coloneqq e^{-\pi\frac{t^2}{\lambda}}$, the Gradient Theorem (GT) expresses the relation between log-magnitude and phase of $Y$ as follows [22, 23]:

$$\frac{\partial}{\partial\omega}\operatorname{Arg}(Y_{y,\varphi_\lambda}(\omega,t)) = -\lambda\frac{\partial}{\partial t}\log|Y_{y,\varphi_\lambda}(\omega,t)| \quad (2)$$

$$\frac{\partial}{\partial t}\operatorname{Arg}(Y_{y,\varphi_\lambda}(\omega,t)) = \frac{1}{\lambda}\frac{\partial}{\partial\omega}\log|Y_{y,\varphi_\lambda}(\omega,t)| + 2\pi\omega \quad (3)$$

Assuming knowledge of magnitudes, this provides dense information about phases. It is also particularly useful for *online* tasks, due to the eminently local nature of the derivatives. Consider now the following STFT of a time-discrete audio *waveform* $\boldsymbol{y}[n]\in\mathbb{R}^N$, window $\boldsymbol{h}[n]\in\mathbb{R}^{2L}$ and hop size $a\in\mathbb{N}_{>0}$:

$$\boldsymbol{Y}_{\boldsymbol{h},a}[\omega,\tau] \coloneqq \sum\nolimits_{l=-L}^{L}\boldsymbol{y}[a\tau+l]\,\boldsymbol{h}[l]\,e^{-2\pi i\frac{\omega l}{2L}} \begin{cases}\omega\in\{0,\dots,L\}\\ \tau\in\mathbb{N}_{\geq 0}\end{cases}$$

where $\omega$ represents a *frequency bin* and $\tau$ a *time frame*. We then have the log-magnitude spectrogram $\boldsymbol{M}\coloneqq\log|\boldsymbol{Y}|$ and phase $\boldsymbol{\Phi}\coloneqq\operatorname{Arg}(\boldsymbol{Y})$. In this setting, $\boldsymbol{\Phi}$ can still be estimated from $\boldsymbol{M}$ via numerical integration of the GT equations in an online fashion via the RTPGHI algorithm [24]. This is efficient and effective, but it introduces error due to discretization and use of non-Gaussian windows [23].

### 2.2. The Gradient Theorem and Deep Learning

It is possible to drastically improve RSI performance by incorporating prior knowledge about the $\boldsymbol{M}\to\boldsymbol{\Phi}$ mapping via DL [25–27]. However, the $2\pi$ periodicity and general irregularity of $\boldsymbol{\Phi}$ makes it a challenging target for end-to-end training. Instead, numerous works highlighted the effectiveness of using phase derivative features, which are easier to learn [28–31]. Here, we focus on the two-stage framework from [21], which involves the Frequency Phase Difference (FPD), Time Phase Difference (TPD), and Baseband Phase Delay (BPD) features[1]:

$$\boldsymbol{u}_{\tau_0} \coloneqq \mathcal{W}(\boldsymbol{\Phi}[\omega,\tau_0] - \boldsymbol{\Phi}[\omega\text{-}1,\tau_0]) \in [-\pi,\pi)^L \quad \text{(FPD)} \quad (4)$$

$$\boldsymbol{v}_{\tau_0} \coloneqq \mathcal{W}(\boldsymbol{\Phi}[\omega,\tau_0] - \boldsymbol{\Phi}[\omega,\tau_{\text{-}1}]) \in [-\pi,\pi)^{L+1} \quad \text{(TPD)} \quad (5)$$

$$\boldsymbol{w}_{\tau_0} \coloneqq \mathcal{W}\Big(\boldsymbol{v}_{\tau_0} - \frac{a\pi\omega}{L}\Big) \in [-\pi,\pi)^{L+1} \quad \text{(BPD)} \quad (6)$$

where $\mathcal{W}(x)\coloneqq\operatorname{Arg}(e^{ix})\in[-\pi,\pi)$ is a phase-wrapping operator. The key idea here is that $\boldsymbol{u}_{\tau_0}$ and $\boldsymbol{w}_{\tau_0}$ resemble the discrete derivatives of $\log|\boldsymbol{Y}|$ at time frame $\tau_0$, as presented in Equations 2-3 (see also [21] and Figure 1). Thus, *they provide dense information about $\boldsymbol{\Phi}$ while being DL-friendly*. This is leveraged in [21] by training two *causal* CNNs to learn the mappings $\hat{\boldsymbol{u}}_{\tau_0}\coloneqq f_{FPD}(\boldsymbol{M}[\omega,\dots\tau_0])$ and $\hat{\boldsymbol{w}}_{\tau_0}\coloneqq f_{BPD}(\boldsymbol{M}[\omega,\dots\tau_0])$ via supervised minimization of the *von Mises Loss*, which also circumvents the issue of the $2\pi$ periodicity [28, 32]:

$$\mathcal{L}(\boldsymbol{X},\hat{\boldsymbol{X}}) \coloneqq -\sum_{\omega}\sum_{\tau}\cos(\boldsymbol{X}[\omega,\tau] - \hat{\boldsymbol{X}}[\omega,\tau]) \quad (7)$$

[1] To minimize verbosity, we vectorize notation frame-wise, and implicitly ignore entries with negative $\omega,\tau$. See [21] for details.

Once the CNNs produce $(\hat{\boldsymbol{u}}_{\tau_0},\hat{\boldsymbol{w}}_{\tau_0})$, we can estimate $\hat{\boldsymbol{v}}_\tau\coloneqq\mathcal{W}(\hat{\boldsymbol{w}}_\tau + a\pi\omega/L)$. Then, a second stage estimates the phases numerically [21]. The method relies on complex ratios $(\mathfrak{u}_{\tau_0}\in\mathbb{C}^L, \mathfrak{v}_{\tau_0}\in\mathbb{C}^{L+1})$, defined as[1]:

$$|\mathfrak{u}_{\tau_0}| \coloneqq \frac{|\boldsymbol{Y}[\omega,\tau_0]|}{|\boldsymbol{Y}[\omega\text{-}1,\tau_0]|}, \quad \operatorname{Arg}(\mathfrak{u}_{\tau_0})\coloneqq\operatorname{Arg}\Big(\frac{\boldsymbol{Y}[\omega,\tau_0]}{\boldsymbol{Y}[\omega\text{-}1,\tau_0]}\Big)=\boldsymbol{u}_{\tau_0} \quad (8)$$

$$|\mathfrak{v}_{\tau_0}| \coloneqq \frac{|\boldsymbol{Y}[\omega,\tau_0]|}{|\boldsymbol{Y}[\omega,\tau_{\text{-}1}]|}, \quad \operatorname{Arg}(\mathfrak{v}_{\tau_0})\coloneqq\operatorname{Arg}\Big(\frac{\boldsymbol{Y}[\omega,\tau_0]}{\boldsymbol{Y}[\omega,\tau_{\text{-}1}]}\Big)=\boldsymbol{v}_{\tau_0} \quad (9)$$

These ratios satisfy $\boldsymbol{Y}[\omega,\tau_0]=\boldsymbol{Y}[\omega\text{-}1,\tau_0]\odot\mathfrak{u}_{\tau_0}$ as well as $\boldsymbol{Y}[\omega,\tau_0]=\boldsymbol{Y}[\omega,\tau_{\text{-}1}]\odot\mathfrak{v}_{\tau_0}$ (assuming all $\boldsymbol{Y}[\omega,\tau]\neq 0$). This allows us to express $\boldsymbol{Y}[\omega,\tau_0]$ as the optimum of the following quadratic objective [21]:

$$\arg\min_{\boldsymbol{z}} \underbrace{\|\boldsymbol{z} - (\boldsymbol{Y}[\omega,\tau_{\text{-}1}]\odot\mathfrak{v}_{\tau_0})\|^2_{\boldsymbol{\Lambda}_{\tau_0}}}_{\tau\text{-term}} + \underbrace{\|\boldsymbol{D}_{\tau_0}\boldsymbol{z}\|^2_{\boldsymbol{\Gamma}_{\tau_0}}}_{\omega\text{-term}} \quad (10)$$

where $\boldsymbol{D}_{\tau_0}\in\mathbb{C}^{L\times(L+1)}$ is a matrix with $-\mathfrak{u}_{\tau_0}$ in the main diagonal, ones in the diagonal above, and zeros elsewhere. Here, $\|\boldsymbol{a}\|^2_{\boldsymbol{X}} \coloneqq \boldsymbol{a}^{\mathsf{H}}\boldsymbol{X}\boldsymbol{a}$ is a weighted norm with *diagonal nonnegative* matrix $\boldsymbol{X}$, used in [21] to mitigate errors for small magnitudes. Equation 10 admits the following closed-form solution:

$$\boldsymbol{z}_0^{(\natural)} = (\boldsymbol{\Lambda}_{\tau_0} + \boldsymbol{D}_{\tau_0}^{\mathsf{H}}\boldsymbol{\Gamma}_{\tau_0}\boldsymbol{D}_{\tau_0})^{-1}\boldsymbol{\Lambda}_{\tau_0}(\boldsymbol{Y}[\omega,\tau_{\text{-}1}]\odot\mathfrak{v}_{\tau_0}) \quad (11)$$

Thus, the second stage recursively[2] estimates $\hat{\boldsymbol{\Phi}}[\omega,\tau_0]$ from $(|\boldsymbol{Y}[\omega,\tau_{\text{-}1}]|, |\boldsymbol{Y}[\omega,\tau_0]|, \hat{\boldsymbol{u}}_{\tau_0},\hat{\boldsymbol{w}}_{\tau_0})$ by solving Equation 11 and extracting the phase from $\boldsymbol{z}_0^{(\natural)}$. This retains causality and produces high-quality speech spectrogram inversions [21].

## 3. Efficient Neural and Numerical Methods

We now describe our innovations to [21], aimed at reducing computation without affecting performance.

### 3.1. An Efficient CNN for the First Stage

The CNNs introduced in [21] comprise 7 learnable layers, featuring a series of residual, sigmoid-gated convolutions. They span a total of 247.81k float parameters and run at[3] 7.95 GMAC/s. Our first main contribution, depicted in Figure 2, is a novel architecture with 8.46k parameters and 0.27 GMAC/s, i.e. *29.27× smaller while maintaining performance* (see Figures 3 and 4a). We highlight the following design principles:

*a)* A stem-body-head structure, with parameter-heavy but shallow stem and head. *b)* No residual layers. Instead, stem and body outputs are concatenated. *c)* Incorporated Batch Normalization (BN) layers [33]. *d)* Replacing gated convolutions with leaky ReLUs [34]. *e)* Reduced receptive field and computation via 1×1 convolutions. *f)* Jointly producing FPD and BPD.

Several resource-intensive components were replaced with more affordable, off-the-shelf ones. The locality of the GT supports the use of 1×1 convolutions. The resemblance between FPD and BPD supports the idea of sharing most of the network between them, for which 50-dimensional features suffice.

[2] To start recursion, $\hat{\boldsymbol{\Phi}}[\omega,0]$ can be set to zeros or random [21].

[3] Giga-Multiply-Accumulate operations per second, measured with the `ptflops` library for $L$=512, $a$=256 (`https://pypi.org/project/ptflops/`).

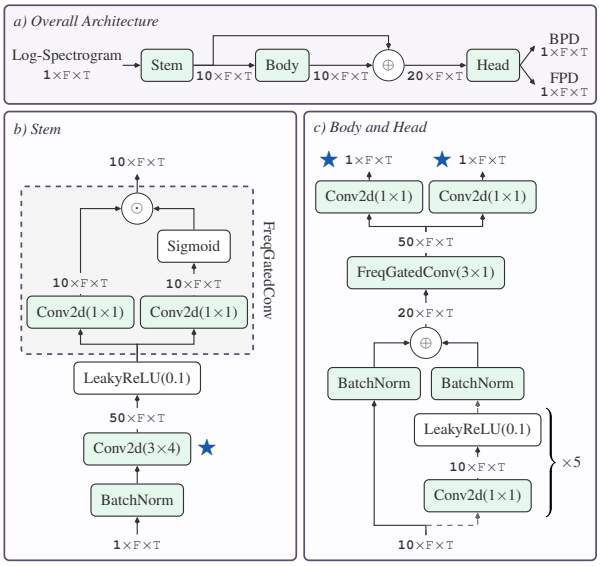



Figure 2: *Our proposed causal CNN for stage 1, featuring 8.46k parameters and 0.27 GMAC/s (Section 3.1). Tensor dimensions are* `Channels` × `Frequency bins` × `Time frames`. *The ⊙ nodes represent elementwise multiplication, and ⊕ concatenation across channel dimension. All convolutions are 2D, with unit stride. All padding is causal (left-sided) . Kernel sizes are detailed across* `(freq×time)` *dimensions.*

### 3.2. Reducing CNN Computation by Increasing Latency

The arithmetic cost of running the CNN is proportional to the number of frequency bins and time frames. Typically, lowering frequency and time resolution decreases output quality, exposing a trade-off between computation and performance. In this section, we show that *it is possible to keep full quality while halving computation by introducing a look-ahead frame*.

In a nutshell, the idea is to run a modified version of the CNN only once every 2 frames $(\ldots, \tau_{-2}, \tau_0)$, but training the network to also provide outputs for the skipped frames $(\ldots, (\tau_{-3}, \tau_{-2}), (\tau_{-1}, \tau_0))$. This can be implemented with the following minimal changes, marked with ★ in Figure 2: *i)* In the stem, the temporal stride of the initial convolution is set to 2, effectively skipping every other frame. *ii)* In the head, the last two convolutions are modified to produce 2 output channels, one for the skipped frame and one for the current frame. This allows to restore the original number of frames, so second stage and training work the same way as with the unmodified CNN.

With this technique, CNN computation is roughly halved (0.14 GMAC/s) without affecting memory (8.57k parameters) or output quality (Figure 4b). The tradeoff is that the refreshing rate of the online processing pipeline is also halved, thus latency is increased by one hop size[4]. This technique is suited for scenarios where a large hop size is hindering performance, or latency is not as important as energy or computation.

### 3.3. Orders of Magnitude Faster Second Stage

Equation 11 solves a linear system in the form $\boldsymbol{A}\boldsymbol{z}=\boldsymbol{b}$ for $\boldsymbol{A}=\boldsymbol{\Lambda}_{\tau_0}+\boldsymbol{D}_{\tau_0}^{\mathsf{H}}\boldsymbol{\Gamma}_{\tau_0}\boldsymbol{D}_{\tau_0}$ and $\boldsymbol{b}=\boldsymbol{\Lambda}_{\tau_0}(\boldsymbol{Y}[\omega,\tau_{-1}]\odot\mathfrak{v}_{\tau_0})$. In general, solvers require $\mathcal{O}(\kappa(L+1)^2)$ arithmetic for some $\kappa \in \{1,\ldots,L+1\}$ [35, ch. 6]. Here, we observe that the matrix

[4]More generally, $k$ hops correspond to $\frac{1}{k+1}$ speedup.

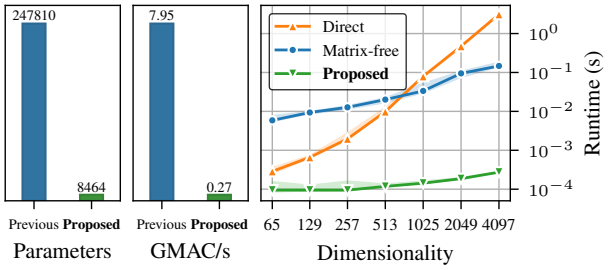



Figure 3: *Summary of computational improvements (see Section 4.2 for extended discussion).* ***Left:*** *Our proposed CNN is ∼30× smaller and faster than the previous one from [21].* ***Right:*** *Runtimes of different solvers for Equation 11, as a function of dimensionality $L+1$. Our proposed method is consistently fastest, and several orders of magnitude faster than other competitive matrix-free methods like LGMRES [37]. Lines show median and 99% confidence interval across 10 runs.*

$\boldsymbol{A}$ is tridiagonal, since $(\boldsymbol{\Lambda}_{\tau_0}\in\mathbb{R}_{\geq 0}^{(L+1)\times(L+1)}, \boldsymbol{\Gamma}_{\tau_0}\in\mathbb{R}_{\geq 0}^{L\times L})$ are diagonal and:

$$\begin{aligned}\boldsymbol{D}_{\tau_0}^{\mathsf{H}}\boldsymbol{\Gamma}_{\tau_0}\boldsymbol{D}_{\tau_0} &= \sum_{l=1}^{L}\gamma_l(\bar{d}_l\boldsymbol{e}_l+\boldsymbol{e}_{l+1})(d_l\boldsymbol{e}_l+\boldsymbol{e}_{l+1})^\top \\ &= \sum_{l=1}^{L}\gamma_l\big(|d_l|^2\underbrace{\boldsymbol{e}_l\boldsymbol{e}_l^\top}_{\text{diag.}}+\underbrace{\boldsymbol{e}_{l+1}\boldsymbol{e}_{l+1}^\top}_{\text{diag.}}\big) \\ &\quad+\sum_{l=1}^{L}\gamma_l d_l\underbrace{\boldsymbol{e}_{l+1}\boldsymbol{e}_l^\top}_{\text{subdiag.}}+\sum_{l=1}^{L}\gamma_l\bar{d}_l\underbrace{\boldsymbol{e}_l\boldsymbol{e}_{l+1}^\top}_{\text{superdiag.}}\end{aligned} \tag{12}$$

where $\gamma_l{:=}[\boldsymbol{\Gamma}_{\tau_0}]_{l,l}$ and $d_l{:=}[\boldsymbol{D}_{\tau_0}]_{l,l}{=}[\text{-}\mathfrak{u}_{\tau_0}]_l$ are scalar entries, and $\boldsymbol{e}_l{\in}\{0,1\}^{L+1}$ is the $l^{\text{th}}$ standard basis vector. Crucially, tridiagonal linear systems can be solved in $\mathcal{O}(L)$ arithmetic and memory [36, 4.3.2], which is optimal in the sense that we process $L{+}1$ individual entries. Additionally, $\boldsymbol{A}$ is positive semi-definite, since it is the sum of a nonnegative diagonal and a Gram matrix. This allows us to solve Equation 11 via more efficient methods like Thomas' Algorithm [36, 4.3.6]. Last but not least, Equation 12 provides a recipe to directly construct the diagonals of $\boldsymbol{A}$. These changes lead to a solver that is *orders of magnitude faster than other matrix-free or direct solvers (Figure 3), while remaining exact (Figure 4a).*

## 4. Experiments and Discussion

We ran several RSI pipelines to reconstruct clean speech waveforms from their STFT log-magnitudes. We first show that our proposed changes to [21] result in a drastic reduction of computation, both in terms of memory and runtime (Figure 3 and Section 4.2). We then show that our results are high-quality, with comparable performance to our own implementation of [21] (Figure 4 and Section 4.3), and subjectively absent of artifacts[5]. We also compare to direct ISTFT using the ground truth phases, as well as the official pretrained VOCOS implementation[6] [27] and an open-source implementation of RTISI[7] [14]. Finally, we discuss some ablation results and limitations that may guide further application and development of our model.

[5]`https://andres-fr.github.io/efficientspecinv`
[6]`https://github.com/gemelo-ai/vocos`
[7]`https://pypi.org/project/torch-specinv`

### 4.1. Setup

We used the Librispeech dataset[8] [38], containing multi-speaker (balanced female and male) speech utterances in English of up to 35 seconds long, in the form of waveforms sampled at 16kHz. For training, we used random 5-second segments from the `train-clean-360` split (363.6 hours across 921 speakers). For evaluation, we took 50 random utterances from the `test-clean` split (5.4 hours across 40 speakers).

We computed the STFTs with a Hann window of size 1024 (i.e. $L$=512) and a hop size of 256, following [21]. The log-magnitudes were then used as input to the CNN, and the phases were used to compute the FPD and BPD features, in order to train the CNN via Equation 7. Both our CNN and the one from [21] were initialized with uniform He distribution for weights [39], and 0 for biases. We trained both our model and the one from [21] using the RAdam optimizer [40] with a batch size of 64, weight decay of $10^{-5}$ and momentum parameters ($\beta_1$=0.9, $\beta_2$=0.999, $\varepsilon$=$10^{-8}$). For the learning rate, we used a ramp-up from 0 to $10^{-3}$ across 1000 batches, followed by cosine annealing in cycles of 1000 batches [41], decaying by a factor of 0.97 after each cycle. Training our CNN until convergence lasted ∼17 hours on 4 NVIDIA H100 GPUs. To run VOCOS we used the provided "copy-synthesis" function which, given an audio waveform, internally computes its spectrogram and the subsequent reconstruction using the pretrained model. For this, input waveforms had to be resampled to 24kHz. We ran RTISI using the aforementioned STFT settings, no lookahead, and a momentum of 0.99.

### 4.2. Drastic Reduction of Computation

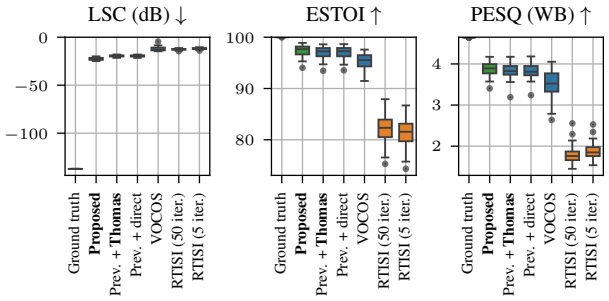
As discussed in Section 3.1 and showcased in Figure 3, our proposed CNN for the first stage is ∼30× smaller than the one from [21]. Note that most of the parameters come from convolutional layers, so computation and parameters scale very similarly. Since [21] does not specify how to solve Equation 11, we compare our approach to two other methods: *a) directly* constructing and inverting the $\boldsymbol{A}$ matrix (with $\mathcal{O}((L+1)^2)$ memory and $\mathcal{O}((L+1)^3)$ arithmetic), and *b)* implementing $\boldsymbol{A}$ as a linear operator and solving the system via LGMRES[9] [37], a competitive *matrix-free* method with $\mathcal{O}(L)$ memory and $\mathcal{O}(\kappa(L+1)^2)$ arithmetic for $\kappa \in \{1, \ldots, L+1\}$ iterations. Our *proposed* method directly constructs the diagonals following Equation 12, and solves the system via Thomas' Algorithm, in $\mathcal{O}(L)$ memory and arithmetic.

On a commodity laptop, we sampled $(\mathfrak{u}, \mathfrak{v}, \boldsymbol{\Lambda}, \boldsymbol{\Gamma})$ from i.i.d. standard Gaussian noise, built the linear system and ran each solver, measuring the wallclock time. We did this 10 times for each dimensionality $L$+1, corresponding to STFT window sizes ranging from 128 to 8192. Our method was fastest, achieving speedups of up to several orders of magnitude.

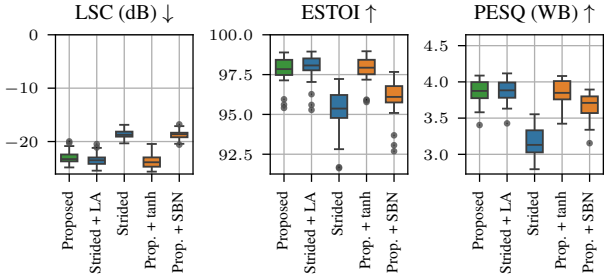
### 4.3. Maintaining High Quality

We follow the same metrics as [21] for evaluation: ESTOI [43] to measure speech intelligibility (0-100, higher is better), WB-PESQ [44] for speech quality (1.04-4.64, higher is better[10]) and Log-Spectral Convergence (LSC) [21] for Euclidean error (dB, lower is better).

In Figure 4a, we observe that our custom implementations (*Proposed* and *Prev.*) yield good scores, close to ground truth




(a) *Performance using ground truth phases, our proposed method, the previous method from [21], pretrained VOCOS [27], and RTISI [14] for 5 and 50 iterations. Our proposed changes from Sections 3.1 and 3.3 (in bold) maintain the competitive performance from [21].*




(b) *Variations of our proposed CNN using tanh activations or subspectral BN [42] exhibit comparable or worse performance. The strided variation introduced in Section 3.2 performs comparably when a lookahead (LA) of 1 hop is introduced, and worse (but still fairly) otherwise.*

Figure 4: *Performance comparison for different approaches and metrics. See Section 4.3 for more details.*

in terms of quality and intelligibility, supporting the validity of our setup (VOCOS is worse here, but this may be due to the upsampling from 16kHz to 24kHz). Comparing *Prev.+direct* with *Prev.+Thomas*, we verify that our proposed second stage does not affect performance negatively. Checking now *Proposed*, we see that further incorporating our CNN also maintains performance. Our smaller CNN seems to exhibit sufficient generalization ability: despite the multi-speaker nature of the data, boxplots are fairly tight and comparable to [21]. In Figure 4b, we see that using *tanh* activations or subspectral BN [42] did not help, despite their higher cost. We also verify that the strided variation introduced in Section 3.2 performs comparably to full inference when the discussed latency of 1 hop is introduced, and worse (but still fairly) otherwise.

## 5. Limitations and Future Work

In this work we introduced three innovations on top of the framework from [21] in order to achieve drastic reduction in computation, while maintaining low latency and high quality. While results are good for our settings, other window and hop sizes, as well as enhanced/inconsistent spectrograms, remain to be evaluated. It would also be informative to incorporate subjective metrics via participatory studies. Including noisy phase information as in [45], and further cost reduction via downsampling of frequency dimension, may also be explored. Our proposed second stage is particularly amenable for differentiable implementations, paving the way for single-stage, end-to-end approaches, including extensions of the loss function (e.g. by using $(\boldsymbol{\Lambda}, \boldsymbol{\Gamma})$ as $\ell_2$ regularizers) and ideas from [46].

[8] `https://www.openslr.org/12`

[9] `scipy` implementation: `https://scipy.org/`

[10] `https://github.com/ludlows/PESQ/issues/13`

## 6. Acknowledgements

The authors thank Buye Xu, Sanjeel Parekh, Michael McManus, Adrian Stepien and Kuba Rad for the constructive and helpful discussions. This work was done while AF was a research scientist intern at Meta Reality Labs.

## 7. References

[1] D. Amodei *et al.*, "Deep speech 2: end-to-end speech recognition in English and Mandarin," in *ICML*, 2016.

[2] L. Wang *et al.*, "Denoising speech based on deep learning and wavelet decomposition," *Scientific Programming*, vol. 2021, no. 1, p. 8677043, 2021.

[3] Z.-Q. Wang *et al.*, "End-to-end speech separation with unfolded iterative phase reconstruction," in *Interspeech 2018*, 2018.

[4] J. Kim, M. El-Khamy, and J. Lee, "T-GSA: Transformer with Gaussian-weighted self-attention for speech enhancement," in *ICASSP*, 2020.

[5] J. Engel *et al.*, "GANSynth: Adversarial neural audio synthesis," in *ICLR*, 2019.

[6] K. Paliwal, K. Wójcicki, and B. Shannon, "The importance of phase in speech enhancement," *Speech Communication*, vol. 53, no. 4, pp. 465–494, 2011.

[7] J. Le Roux *et al.*, "Phasebook and friends: Leveraging discrete representations for source separation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 370–382, 2019.

[8] J. Allen and L. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *IEEE Proceedings*, vol. 65, no. 11, pp. 1558–1564, 1977.

[9] N. Sturmel and L. Daudet, "Signal reconstruction from stft magnitude : A state of the art," in *DAFx*, 2011.

[10] H. Sahinoglou and S. Cabrera, "On phase retrieval of finite-length sequences using the initial time sample," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 8, pp. 954–958, 1991.

[11] S. Nawab, T. Quatieri, and J. Lim, "Signal reconstruction from short-time Fourier transform magnitude," *IEEE Transactions on Acoustics, Speech, and Sig. Proc.*, vol. 31, no. 4, 1983.

[12] E. J. Candès, T. Strohmer, and V. Voroninski, "PhaseLift: Exact and stable signal recovery from magnitude measurements via convex programming," *Communications on Pure and Applied Mathematics*, vol. 66, no. 8, pp. 1241–1274, 2013.

[13] D. L. Sun and J. O. S. III, "Estimating a signal from a magnitude spectrogram via convex optimization," *Journal of the Audio Engineering Society*, Oct 2012.

[14] G. T. Beauregard, X. Zhu, and L. Wyse, "An efficient algorithm for real-time spectrogram inversion," in *DAFx*, 2005.

[15] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," in *ICASSP*, 1983.

[16] T. Peer *et al.*, "A flexible online framework for projection-based STFT phase retrieval," in *ICASSP*, 2024.

[17] R. Decorsière *et al.*, "Inversion of auditory spectrograms, traditional spectrograms, and other envelope representations," *TASLP*, vol. 23, no. 1, pp. 46–56, 2015.

[18] P.-H. Vial *et al.*, "Phase retrieval with Bregman divergences and application to audio signal recovery," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 1, pp. 51–64, 2021.

[19] G. T. Beauregard, M. Harish, and L. Wyse, "Single pass spectrogram inversion," in *IEEE International Conference on Digital Signal Processing (DSP)*, 2015.

[20] M. Puckette, "Phase-locked vocoder," in *WASPAA*, 1995, pp. 222–225.

[21] Y. Masuyama *et al.*, "Online phase reconstruction via DNN-based phase differences estimation," *TASLP*, vol. 31, pp. 163–176, 2023.

[22] M. Portnoff, "Magnitude-phase relationships for short-time Fourier transforms based on Gaussian analysis windows," in *ICASSP*, 1979.

[23] Z. Průša, P. Balázs, and P. L. Søndergaard, "A noniterative method for reconstruction of phase from STFT magnitude," *TASLP*, vol. 25, no. 5, pp. 1154–1164, 2017.

[24] Z. Průša and P. L. Søndergaard, "Real-time spectrogram inversion using phase gradient heap integration," in *DAFx*, 2016.

[25] A. van den Oord *et al.*, "WaveNet: A generative model for raw audio," in *9th ISCA Speech Synthesis Workshop*, 2016.

[26] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: generative adversarial networks for efficient and high fidelity speech synthesis," in *NeurIPS*, 2020.

[27] H. Siuzdak, "VOCOS: Closing the gap between time-domain and Fourier-based neural vocoders for high-quality audio synthesis," in *ICLR*, 2024.

[28] S. Takamichi *et al.*, "Phase reconstruction from amplitude spectrograms based on Von-Mises-distribution deep neural network," *IWAENC*, 2018.

[29] ——, "Phase reconstruction from amplitude spectrograms based on directional-statistics deep neural networks," *Elsevier Signal Processing*, vol. 169, no. C, Apr. 2020.

[30] L. Thieling, D. Wilhelm, and P. Jax, "Recurrent phase reconstruction using estimated phase derivatives from deep neural networks," in *ICASSP*, 2021.

[31] N. B. Thien *et al.*, "Inter-frequency phase difference for phase reconstruction using deep neural networks and maximum likelihood," *TASLP*, vol. 31, 2023.

[32] ——, "Two-stage phase reconstruction using DNN and von Mises distribution-based maximum likelihood," in *APSIPA ASC*, 2021.

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[34] B. Xu *et al.*, "Empirical evaluation of rectified activations in convolutional network," in *arXiv*, no. 1505.00853, 2015.

[35] J. W. Demmel, *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

[36] G. H. Golub and C. F. Van Loan, *Matrix Computations*. The Johns Hopkins University Press, 2013.

[37] A. H. Baker, E. R. Jessup, and T. Manteuffel, "A technique for accelerating the convergence of restarted GMRES," *SIAM Journal on Matrix Analysis and Applications*, vol. 26, no. 4, 2005.

[38] V. Panayotov *et al.*, "Librispeech: An ASR corpus based on public domain audio books," in *ICASSP*, 2015.

[39] K. He *et al.*, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, 2015.

[40] L. Liu *et al.*, "On the variance of the adaptive learning rate and beyond," in *ICLR*, 2020.

[41] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *ICLR*, 2017.

[42] S. Chang *et al.*, "Subspectral normalization for neural audio data processing," *ICASSP*, 2021.

[43] J. Jensen and C. H. Taal, "An algorithm for predicting the intelligibility of speech masked by modulated noise maskers," *TASLP*, vol. 24, no. 11, pp. 2009–2022, 2016.

[44] *Rec. P.862.2: Wideband extension to Recommendation P.862 for the assessment of wideband telephone networks and speech codecs*, ITU-T, 2007.

[45] Y. Song and N. Madhu, "Phase reconstruction in single channel speech enhancement based on phase gradients and estimated clean-speech amplitudes," in *ICASSP*, 2024.

[46] S. Ö. Arık, H. Jun, and G. Diamos, "Fast spectrogram inversion using multi-head convolutional neural networks," *IEEE Signal Processing Letters*, vol. 26, no. 1, pp. 94–98, 2019.