Meta-AF: Meta-Learning for Adaptive Filters

Jonah Casebeer ⬤, *Graduate Student Member, IEEE*, Nicholas J. Bryan ⬤, *Member, IEEE*, and Paris Smaragdis, *Fellow, IEEE*

Abstract—Adaptive filtering algorithms are pervasive throughout signal processing and have had a material impact on a wide variety of domains including audio processing, telecommunications, biomedical sensing, astrophysics and cosmology, seismology, and many more. Adaptive filters typically operate via specialized online, iterative optimization methods such as least-mean squares or recursive least squares and aim to process signals in unknown or nonstationary environments. Such algorithms, however, can be slow and laborious to develop, require domain expertise to create, and necessitate mathematical insight for improvement. In this work, we seek to improve upon hand-derived adaptive filter algorithms and present a comprehensive framework for learning online, adaptive signal processing algorithms or update rules directly from data. To do so, we frame the development of adaptive filters as a meta-learning problem in the context of deep learning and use a form of self-supervision to learn online iterative update rules for adaptive filters. To demonstrate our approach, we focus on audio applications and systematically develop meta-learned adaptive filters for five canonical audio problems including system identification, acoustic echo cancellation, blind equalization, multi-channel dereverberation, and beamforming. We compare our approach against common baselines and/or recent state-of-the-art methods. We show we can learn high-performing adaptive filters that operate in real-time and, in most cases, significantly outperform each method we compare against – all using a single general-purpose configuration of our approach.

Index Terms—Adaptive filtering, meta learning, online optimization, learning to learn, deep learning.

I. INTRODUCTION

ADAPTIVE signal processing and adaptive filter theory are cornerstones of modern signal processing and have had a deep and significant impact on modern society. Applications of adaptive filters (AF) include audio processing, telecommunications, biomedical sensing, astrophysics and cosmology, seismology and more. Audio applications, in particular, are of exceptional importance and find utility for many problems such as single- and multi-channel system identification, echo cancellation, prediction, dereverberation, beamforming, noise cancellation, and beyond. AFs typically operate via online iterative optimization methods, such as least mean square filtering (LMS), normalized LMS (NLMS), recursive least-squares (RLS), or Kalman filtering (KF), to solve streaming or online optimization problems [1], [2], [3], [4], [5], [6] and process signals in unknown and/or nonstationary environments.

AF tasks are often grouped into one of four core categories: system identification, inverse modeling, prediction, and interference cancellation [4]. Each of these categories has numerous AF applications, and advances in one category or application can often be applied to many others. In the audio domain, acoustic echo cancellation (AEC) can be formulated as single- or multi-channel system identification and has been studied extensively [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18]. Equalization can be formulated as an inverse modeling problem, has been explored in single- and multi-channel formats, and is particularly useful for sound zone reproduction and active noise control [19], [20], [21], [22], [23], [24]. Dereverberation can be formulated as a prediction problem and has been studied considerably in this context [25], [26], [27], [28], [29], [30], [31]. Finally, multi-microphone enhancement or beamforming can be formulated as an informed interference cancellation task and also has a breadth of associated algorithms [32], [33], [34], [35], [36], [37], [38].

When we consider AFs in the context of deep neural networks (DNNs), we note two main observations. First, AFs continue to be used extensively, but mostly via hybrid approaches that combine neural networks with standard AF algorithms. Second, the underlying AF algorithms and tools for the development of new AFs have had little change in several decades. Hybrid approaches, however, have proven very successful. For example, in AEC, neural networks can be trained for residual echo suppression, noise suppression, reference estimation, and more [39], [40], [41], [42], [43], [44], [45], [46], [47]. In a similar vein, neural networks paired with AFs for active noise control tasks have shown impressive results [48], [49], [50], [51]. For dereverberation, DNNs have proven useful for both online and offline approaches by directly estimating statistics of the dereverberated speech [52], [53], [54], [55], [56]. This pattern has repeated itself for beamforming applications, where DNNs have led to many performance improvements [57], [58], [59], [60]. In many of these works, DNNs are trained to predict ratio-masks, or otherwise directly enhance/separate the desired signal. In essence, they act as a distinct module within a larger pipeline that also uses AFs.

Jonah Casebeer is with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: jonahmc2@illinois.edu).

Nicholas J. Bryan is with Adobe Research, San Francisco, CA 94103 USA (e-mail: nibryan@adobe.com).

Paris Smaragdis is with the Department of Computer Science and Department, University of Illinois, Urbana, IL 61801 USA (e-mail: smaragdis@ieee.org).

In contrast, there are a small number of works that more tightly couple neural networks and AFs and use DNNs for optimal control of AFs. Recently, it was shown that DNNs can estimate statistics to control step-sizes [61], [62] or estimate entire updates [63] for a single-channel AEC. Similarly, past work has used DNNs to predict updates for the internal statistics of multi-channel beamformers [64] and to learn source-models for multi-channel source separation [65]. These works differ from hybrid approaches in that they leverage neural networks to update or control AFs directly and thus focus on improving the performance of AFs themselves. Such improvement can be leveraged in isolation or, in theory, together with complementary hybrid approaches.

More broadly, the idea of controlling AFs via neural networks is related to several disciplines of signal processing and machine learning including optimal control, optimization, automatic machine learning, reinforcement learning and meta-learning. Relevant works within these areas include automatic selection of step sizes [7], [66], [67], the direct control of model weights [68], [69], [70], rapid fine-tuning [71], and meta-learning optimization rules [72]. Out of these works, learning optimization rules or *learned optimizers* is of critical relevance [73], [74], [75] and presents the idea of using one neural network as a function that optimizes another function. Such works, however, focus on creating learned optimizers for training neural networks in an offline setting, where the latter network is the final product, and the learned optimizer is otherwise discarded (or otherwise used to train additional networks). Moreover, this work has had little application to AFs, except for our own work [63], which we extend here.

In this work, we formulate the development of AF algorithms as a meta-learning problem. We learn adaptive filter update rules directly from data using self-supervision and call our approach meta adaptive filtering (Meta-AF). Using our method, we no longer need humans to hand-derive update equations, do not need any supervised labeled data for learning, and do not need exhaustive tuning. To showcase our approach, we systematically develop learned AFs for exemplary applications of each of the four canonical AF architectures. Then, for each AF task, we compare our work to a suite of baselines and/or state-of-the-art approaches for the problems of system identification, acoustic echo cancellation, equalization, multi-channel dereverberation, and multi-channel interference cancellation (beamforming). For all tasks, we use identical hyperparameters, significantly reducing engineering and design time. We evaluate performance using signal-to-noise ratio (SNR)-like signal metrics and perceptual- or task-specific metrics as well as specific qualitative comparisons. Our results show we can learn high-performing AF algorithms that operate in real-time and, in most cases, outperform all methods we compare against.

The contributions of our work are as follows: 1) we present the first general-purpose method of meta-learning AF algorithms (update rules) directly from data via self-supervision (no labels required) 2) we apply our approach to all canonical AF architecture categories including system identification, inverse modeling, prediction, and (informed) interference cancellation and 3) we show how a single hyperparameter configuration

TABLE I
SYMBOLS AND OPERATORS

Symbols	Description
$x \in \mathbb{R}$	A real-valued scalar
$x \in \mathbb{C}$	A complex-valued scalar
$\underline{x} \in \mathbb{R}$	A real-valued time-domain scalar
$\underline{\mathbf{x}} \in \mathbb{R}^N$	A time-domain N-dimensional column vector
$\mathbf{x} \in \mathbb{C}^N$	A complex-valued N-dimensional column vector
$\mathbf{X} \in \mathbb{C}^{M \times N}$	A complex-valued $M \times N$ matrix
$\mathbf{x}[\tau]$	A time-varying column vector
$\mathbf{X}[\tau]$	A time-varying matrix
$\mathbf{w}[\tau]$	FD AF linear-filter
$\mathbf{u}[\tau]$	FD AF input
$\mathbf{d}[\tau]$	FD AF target or desired response
$\mathbf{y}[\tau]$	FD AF estimated response
$\mathbf{s}[\tau]$	FD AF true desired signal
$\mathbf{I}_N$	An $N \times N$ identity matrix
$\mathbf{0}_{N \times R}$	$N \times R$ matrix of zeros
$\mathbf{1}_{N \times R}$	$N \times R$ matrix of ones
$\mathbf{F}_N$	N-point discrete Fourier transform (DFT) matrix

Operators	Description
$*$	Convolution
$.^{\top}$	Transpose
$.^{*}$	Complex conjugate
$.^{H}$	Hermitian transpose
$\text{diag}(\cdot)$	Vector to a diagonal matrix
$\text{cat}(\cdots)$	Column vector concatenation (vertical stack)
E	Expected value
$\frac{(\cdot)}{(\cdot)}$	Element-wise division
$\|\cdot\|$	Euclidean norm
$|\cdot|$	Element-wise magnitude of complex value
$\angle$	Phase of a complex-value
$\ln$	Natural logarithm
-1	Scalar or matrix inverse

of our approach, trained with different datasets and losses, can outperform all common AF baselines and/or several past state-of-the-art methods we compare against according to one or more evaluation metrics and are suitable for real-time operation on commodity hardware. Compared to our previous work on single-channel single-talk AEC [63], we present several new improvements including 1) an improved loss, 2) additional inputs to our learned optimizer and 3) an updated development for multi-block, multi-channel AFs with coupled updates and 4) extensive experimentation on four new applications. We release demos for each task and open source all code[1] including baselines.

II. BACKGROUND

A. Notation

We provide an overview of the symbols and operators we use in Table I. We denote scalars via lower-case symbols, column vectors via bold, lower-case symbols, and matrices via bold upper-case symbols. We use bracket indexing $[\tau]$ to denote time-varying signals and an underline to denote the time-domain counterpart of a frequency-domain (FD) symbol. We index FD rows via the subscript k, columns via the subscript m, and elements via the subscripts km.

B. Overlap-Save and Overlap-Add Filtering

We perform short-time (STFT) Fourier filtering using either overlap-save (OLS) or overlap-add (OLA) convolution. The OLS method uses block processing by splitting the input signal into overlapping windows and recombines complete non-overlapping components. We use $\underline{u}_\mathrm{m}[t] \in \mathbb{R}$ to represent the time-domain sample recorded by microphone m at discrete time t. We collect N such samples from microphone m to form the time-domain frame $\underline{\mathbf{u}}_\mathrm{m}[\tau] = [\underline{u}_\mathrm{m}[\tau R - N + 1], \dots, \underline{u}_\mathrm{m}[\tau R]] \in \mathbb{R}^N$, where τ is the frame index, N is the window length in samples, R is the hop size in samples, and $O = N - R$ is the overlap between frames in samples. Finally, we collect samples from all M microphones to form a multi-channel time-domain signal, $\underline{\mathbf{U}}[\tau] = [\underline{\mathbf{u}}_1[\tau], \dots, \underline{\mathbf{u}}_M[\tau]] \in \mathbb{R}^{N \times M}$. We compute the corresponding FD representation via $\mathbf{U}[\tau] = \mathbf{F}_N \underline{\mathbf{U}}[\tau] \in \mathbb{C}^{K \times M}$, where K is the number of frequency bins, set to $K = N$ for this work. We select the mth channel from a multi-channel FD representation using $\mathbf{u}_\mathrm{m}[\tau] \in \mathbb{C}^K$. We define the FD filter $\mathbf{w}_\mathrm{m}[\tau] \in \mathbb{C}^K$ and the frequency and time output for the mth channel as

$$\mathbf{y}_\mathrm{m}[\tau] = \mathrm{diag}(\mathbf{u}_\mathrm{m}[\tau])\mathbf{Z}_w \mathbf{w}_\mathrm{m}^*[\tau] \in \mathbb{C}^K \qquad (1)$$

$$\underline{\mathbf{y}}_\mathrm{m}[\tau] = \mathbf{Z}_y \mathbf{y}_\mathrm{m}[\tau] \in \mathbb{R}^R, \qquad (2)$$

where $\mathbf{Z}_w = \mathbf{F}_K \mathbf{T}_{K/2}^\top \mathbf{T}_{K/2} \mathbf{F}_K^{-1} \in \mathbb{C}^{K \times K}$ and $\mathbf{Z}_y = \bar{\mathbf{T}}_R \mathbf{F}_K^{-1} \in \mathbb{C}^{R \times K}$ are anti-aliasing matrices, $\mathbf{T}_{K/2} = [\mathbf{I}_{K/2}, \mathbf{0}_{K/2 \times K/2}] \in \mathbb{R}^{R \times K}$ trims the last $K/2$ samples from a vector and $\bar{\mathbf{T}}_R = [\mathbf{0}_{R \times O}, \mathbf{I}_R] \in \mathbb{R}^{R \times K}$ trims the first O samples.

The counterpart to OLS is OLA filtering, which computes the frequency output, time output, and buffer $after\underline{\mathbf{b}}_\mathrm{m}[\tau]$ as

$$\mathbf{y}_\mathrm{m}[\tau] = \mathrm{diag}(\mathbf{u}_\mathrm{m}[\tau])\mathbf{w}_\mathrm{m}^*[\tau] \in \mathbb{C}^K \qquad (3)$$

$$\underline{\mathbf{y}}_\mathrm{m}[\tau] = \mathbf{T}_R \mathbf{F}_K^{-1} \mathbf{y}_\mathrm{m}[\tau] + \bar{\mathbf{T}}_R \underline{\mathbf{b}}_\mathrm{m}[\tau - 1] \in \mathbb{R}^R \qquad (4)$$

$$\underline{\mathbf{b}}_\mathrm{m}[\tau] = \mathbf{F}_K^{-1} \mathbf{y}_\mathrm{m}[\tau] + \mathbf{T}_R^\top \bar{\mathbf{T}}_R \underline{\mathbf{b}}_\mathrm{m}[\tau - 1] \in \mathbb{R}^K. \qquad (5)$$

Here, $\mathbf{T}_R = [\mathbf{I}_R, \mathbf{0}_{R \times O}] \in \mathbb{R}^{R \times K}$. Typically, the forward and inverse DFTs are combined with analysis and synthesis windows and optionally zero-padded. We use Hann windows [76]. For multi-channel multi-block input, single-channel output FD filters, the OLS/OLA equations above are applied per channel, and anti-aliasing is applied per block. The per frequency (anti-aliased) filter is $\mathbf{w}_\mathrm{k}[\tau] \in \mathbb{C}^{BM}$ with B buffered frames and M channels all stacked. The filter input is similarly stacked and is $\mathbf{u}_\mathrm{k}[\tau] \in \mathbb{C}^{BM}$, thus requiring (1) and (3) to be modified.

C. Adaptive Filter Problem Formulation

We define an AF as an algorithm or optimizer that solves

$$\hat{\boldsymbol{\theta}}[\tau] = \arg\min_{\boldsymbol{\theta}[\tau]} \mathcal{L}(\cdots)[\tau] \qquad (6)$$

via an additive update rule of the form

$$\boldsymbol{\theta}[\tau + 1] = \boldsymbol{\theta}[\tau] + \boldsymbol{\Delta}[\tau], \qquad (7)$$

where $\hat{\boldsymbol{\theta}}[\tau]$ is a set of estimated time-varying filter parameters. In this work, we focus exclusively on linear FD adaptive filters (FDAFs), where $\boldsymbol{\theta}[\tau] = \{\mathbf{w}[\tau]\}$ without loss of generality.

Common losses include the mean-square error (MSE),

$$\mathcal{L}_{MSE}[\tau] = E[\|\mathbf{e}_\mathrm{m}[\tau]\|^2] = E[\|\mathbf{d}_\mathrm{m}[\tau] - \mathbf{y}_\mathrm{m}[\tau]\|^2], \qquad (8)$$

the instantaneous square error (ISE),

$$\mathcal{L}_{ISE}[\tau] = \|\mathbf{e}_\mathrm{m}[\tau]\|^2 = \|\mathbf{d}_\mathrm{m}[\tau] - \mathbf{y}_\mathrm{m}[\tau]\|^2, \qquad (9)$$

and the weighted least squares error (WSE),

$$\mathcal{L}_{WSE}[\tau] = \sum_{n=0}^{\tau} \gamma^{\tau-n} \|\mathbf{d}_\mathrm{m}[n] - \mathbf{y}_\mathrm{m}[n]\|^2, \qquad (10)$$

where γ is a forgetting factor, m is a reference mic, and the output $\mathbf{y}_m$ is computed via (1) or (3) for FD signals or losses and (2) or (4) for time-domain signals or losses.

D. Conventional Optimizers

For audio AFs, it is common to leverage OLS or OLA filtering and solve (6) via optimization methods per frequency bin. So, we modify (7) to be

$$\mathbf{w}_\mathrm{k}[\tau + 1] = \mathbf{w}_\mathrm{k}[\tau] + \boldsymbol{\Delta}_\mathrm{k}[\tau], \qquad (11)$$

where the update $\boldsymbol{\Delta}_\mathrm{k}[\tau]$ is per frequency k. We focus on three common conventional AF optimizers in this form as well as a machine learning optimizer to ground the development of our method to familiar, fundamental algorithms. For gradient methods, we use the partial derivative with respect to $\mathbf{w}_\mathrm{k}[\tau]^{\mathsf{H}}$.

1) Least Mean Square: The least mean square optimizer (LMS) is a stochastic gradient descent method that uses the ISE loss and gradient. The LMS update is

$$\boldsymbol{\Delta}_\mathrm{k}[\tau] = -\lambda \boldsymbol{\nabla}_\mathrm{k}[\tau], \qquad (12)$$

where λ is the step-size and $\boldsymbol{\nabla}_\mathrm{k}[\tau]$ is the gradient of the ISE. Note, LMS is stateless and only a function of the gradient.

2) Normalized Least Mean Square: The normalized LMS (NLMS) algorithm modifies LMS via a running normalizer based on the input power. The NLMS update is

$$\mathbf{v}_\mathrm{k}[\tau] = \gamma \mathbf{v}_\mathrm{k}[\tau - 1] + (1 - \gamma)\|\mathbf{u}_\mathrm{k}[\tau]\|^2 \qquad (13)$$

$$\boldsymbol{\Delta}_\mathrm{k}[\tau] = -\lambda \frac{\boldsymbol{\nabla}_\mathrm{k}[\tau]}{\mathbf{v}_\mathrm{k}[\tau]}, \qquad (14)$$

where the division is element-wise, λ is the step-size and γ is a forgetting factor.

3) Root Mean Squared Propagation: The root mean square propagation (RMSProp) optimizer [77] modifies NLMS by replacing $\mathbf{v}_\mathrm{k}[\tau]$ with a gradient-based per-element running normalizer, $\boldsymbol{\nu}[\tau]$ using forget factor γ as,

$$\boldsymbol{\nu}_\mathrm{k}[\tau] = \gamma \boldsymbol{\nu}_\mathrm{k}[\tau - 1] + (1 - \gamma)\|\boldsymbol{\nabla}_\mathrm{k}[\tau]\|^2 \qquad (15)$$

$$\boldsymbol{\Delta}_\mathrm{k}[\tau] = -\lambda \frac{\boldsymbol{\nabla}_\mathrm{k}[\tau]}{\sqrt{\boldsymbol{\nu}_\mathrm{k}[\tau]}}, \qquad (16)$$

The value, $1/\sqrt{\boldsymbol{\nu}_\mathrm{k}[\tau]}$ supplements the fixed step-size λ and acts as an adaptive learning rate, $\lambda/\sqrt{\boldsymbol{\nu}_\mathrm{k}[\tau]}$.

4) Recursive Least Squares: The aim of the recursive least squares (RLS) algorithm is to exactly solve the AF loss, most commonly the WSE error. This is accomplished by

expanding the weighted least squares error into a function of the weighted empirical signal covariance matrix, $\mathbf{\Phi}_k[\tau] = \sum_\tau \gamma^{N-\tau} \mathbf{u}_k[\tau] \mathbf{u}_k[\tau]^H$, where the summation time-indices are application dependent (e.g. causal vs. non-causal implementations), and the (weighted) empirical cross-correlation vector $\mathbf{z}_k[\tau] = \sum_\tau \gamma^{N-\tau} \mathbf{u}_k[\tau] \mathbf{d}_k[\tau]^H$ to compute the exact solution to the resulting normal equations, $\mathbf{\Phi}_k[\tau] \mathbf{w}_k[\tau] = \mathbf{z}_k[\tau]$. Running estimates of $\mathbf{\Phi}_k[\tau]$ and $\mathbf{z}_k[\tau]$ are also commonly used. However, instead of performing repeated matrix inversion, the matrix inversion lemma is used. Thus, RLS can be implemented using a time-varying precision (inverse covariance) matrix $\mathbf{P}_k[\tau]$ and the Kalman-gain $\boldsymbol{\kappa}_k[\tau]$,

$$\boldsymbol{\kappa}_k[\tau] = \frac{\mathbf{P}_k[\tau-1] \mathbf{u}_k[\tau]}{\gamma + \mathbf{u}_k[\tau]^H \mathbf{P}_k[\tau-1] \mathbf{u}_k[\tau]} \qquad (17)$$

$$\mathbf{P}_k[\tau] = \frac{\mathbf{P}_k[\tau-1] - \boldsymbol{\kappa}_k[\tau] \mathbf{u}_k[\tau]^H \mathbf{P}_k[\tau-1]}{\gamma} \qquad (18)$$

$$\mathbf{\Delta}_k[\tau] = \boldsymbol{\kappa}_k[\tau](d_{km}[\tau] - y_{km}[\tau])^*, \qquad (19)$$

where γ is a forgetting factor, $d_{km}[\tau]$ and $y_{km}[\tau]$ are the desired and estimated signal at frequency k and reference microphone m, and the initialization of $\mathbf{P}_k[\tau]$ is critical and commonly based on the input SNR. In the case of multi-block and/or multi-channel filters, there are multiple ways to formulate RLS, some of which differ from time-domain RLS. Common approaches include diagonalized RLS (D-RLS) and block diagonalized RLS (BD-RLS) as well as QR decomposition techniques [5], [78], [79] and differ in what terms of the covariance (precision) matrix are modeled. D-RLS makes an independence assumption and optimizes each k, m, b filter tap separately, forming K diagonal $BM \times BM$ precision matrices. BD-RLS couples across frames and channels by forming K separate $BM \times BM$ precision matrices. In the case of single block/channel filters, D-RLS, BD-RLS, and NLMS can be reduced to the same algorithm with different parameterizations. In our case, we use identical BD-RLS implementations across all tasks.

When conventional optimizers are compared to each other, the order of performance commonly follows LMS, RMSProp, NLMS, and RLS, while the order of computational complexity is reversed. These algorithms, however, can be sensitive to tuning, nonstationarities, nonlinearities, and other issues that require engineering effort to mitigate failure cases and stability. For multi-block BD-RLS filters, in particular, poor partition conditioning can also lead to degraded RLS performance [80] and/or stability issues compared to NLMS and other alternatives. For the purposes of this work, we exhaustively grid-search tune the hyperparemeters on held-out validation sets of signals per task.

Beyond these basic optimizers, we also compare against several additional methods. For the AEC task, we compare against the double-talk robust diagonal Kalman filter (D-KF) [11], the open-source double-talk robust Speex algorithm [12], a weighted-RLS (wRLS) algorithm [81], and WebRTC-AEC3 [82]. The D-KF algorithm is recommended over other variants [15] and is a common AEC baseline [61]. In addition, the Speex and wRLS [81] algorithms were the linear

AFs used (with a non-linear post-processor) in the first- [47] and second-place [81] winners of the ICASSP 2021 Acoustic Echo Cancellation Challenge [83], respectively. Since our work focuses on linear adaptive filters and can easily be combined with non-linear post-processors, we believe D-KF, wRLS, Speex, and WebRTC-AEC3 are reasonable baselines. For dereverberation, we compare against NARA-WPE [30], which is a highly effective normalized BD-RLS based optimizer [27], [30] and comparable to the original NTT implementation [84].

E. Related Work

Initial work on automatically tuning AFs has been explored in incremental delta-bar-delta [66], Autostep [67], and elsewhere. Recent machine learning work discusses the idea of using DNNs to learn entirely novel optimizer update rules from scratch [72], [73], [74], [75]. We take inspiration from this work, but make numerous advances specific to AFs. In particular, past learned optimizers [72] are element-wise, offline, real-valued, only a function of the gradient, and are trained to optimize general purpose neural networks. In contrast, we design *online* AF optimizers that use multiple input signals, are complex-valued, adapt block FD linear filters, and integrate domain-specific insights to reduce complexity and improve performance (coupling across channels and time). Moreover, we deploy learned optimizers to solve AF tasks as the end-goal and do not use them to train downstream neural networks. We also note recent work that uses a supervised DNN to control the step-size of a D-KF for AEC [61] and another that uses a supervised DNN to predict both the step-size and a nonlinear reference signal for AEC [62]. Compared to these, we replace the entire update with a neural network, do not need supervisory signals, and investigate many tasks.

III. PROPOSED METHOD

A. Problem Formulation

We formulate AF algorithm design as a meta-learning problem and train neural networks to learn AFs from data, creating meta-learned adaptive filters. This is in contrast to typical AFs that are manually created by human engineers. To do so, we define a learned optimizer, $g_\phi(\cdot)$, as a neural network with one or more input signals parameterized by weights ϕ that optimizes an AF loss or *optimizee* $\mathcal{L}(h_\theta(\cdot), \cdots)$ or $\mathcal{L}$ for short, using an additive update rule

$$\boldsymbol{\theta}[\tau+1] = \boldsymbol{\theta}[\tau] + g_\phi(\cdot). \qquad (20)$$

We then seek an optimal AF optimizer $g_{\hat{\phi}}$ over dataset $\mathcal{D}$

$$\hat{\phi} = \arg\min_\phi E_{\mathcal{D}}[\, \mathcal{L}_M(\, g_\phi, \mathcal{L}(h_\theta(\cdot), \cdots)\,)\,], \qquad (21)$$

where $\mathcal{L}_M$ is a functional that defines the meta (or optimizer) loss that is a function of g_ϕ and an AF loss $\mathcal{L}$ with one or more inputs and filtering function h_θ that itself has one or more inputs and parameters $\boldsymbol{\theta}$. Intuitively, when we solve (21), we learn a network $g_\phi(\cdot)$ that solves the AF loss $\mathcal{L}$ when applied repeatedly via an additive update.

TABLE II
RELATIONSHIP BETWEEN AF OPTIMIZERS

Optimizer	Inputs	State	Params	$\boldsymbol{\Delta}_\mathrm{k}[\tau]$
LMS	$\boldsymbol{\nabla}_\mathrm{k}[\tau]$	$\emptyset$	λ	(12)
NLMS	$\boldsymbol{\nabla}_\mathrm{k}[\tau], \mathbf{u}_\mathrm{k}[\tau]$	$\mathbf{v}_\mathrm{k}[\tau]$	λ, γ	(14)
RMSProp	$\boldsymbol{\nabla}_\mathrm{k}[\tau]$	$\boldsymbol{\nu}_\mathrm{k}[\tau]$	λ, γ	(16)
BD-RLS	$\mathbf{u}_\mathrm{k}[\tau], \mathbf{d}_\mathrm{k}[\tau], \mathbf{y}_\mathrm{k}[\tau]$	$\mathbf{P}_\mathrm{k}[\tau]$	γ	(19)
Ours	$\boldsymbol{\xi}_\mathrm{k}[\tau]$	$\boldsymbol{\psi}_\mathrm{k}[\tau]$	$\boldsymbol{\phi}$	g_ϕ

TABLE III
OPTIMIZER COMPLEXITY COMPARISON

Optimizer	Big-$\mathcal{O}$	$\approx \mathbb{C}$MACS
LMS	$\mathcal{O}(\mathrm{KMB})$	KMB
NLMS	$\mathcal{O}(\mathrm{KMB})$	5KMB
RMSProp	$\mathcal{O}(\mathrm{KMB})$	6KMB
BD-RLS	$\mathcal{O}(\mathrm{K(MB)}^2)$	$\mathrm{K(4(MB)}^2 + 5\mathrm{MB})$
Ours	$\mathcal{O}(\mathrm{K(H}^2 + \mathrm{MBH}))$	$\mathrm{K(12H}^2 + (21 + 10\mathrm{MB})\mathrm{H})$

B. Optimizee Architecture and Loss

The optimizee, or the AF loss $\mathcal{L}$ that is optimized via (20) is a function of the filter or architecture $h_{\boldsymbol{\theta}}(\cdot)$. The filter can be any reasonable differentiable filtering operator such as time-domain FIR filters, lattice FIR filters, non-linear filters, FD filters, multi-delayed block FD filters [3], etc. Similarly, the AF loss can be any reasonable differentiable loss such as the MSE, ISE, WSE, a regularized loss, negative log-likelihood, mutual information, etc. For our work, we focus on single- and multi-channel multi-frame linear block FD filters $h_{\boldsymbol{\theta}}$ applied via OLS or OLA with parameters $\boldsymbol{\theta}_\mathrm{k}[\tau] = \{\mathbf{w}_\mathrm{k}[\tau] \in \mathbb{C}^{BM}\}$ with B buffered frames and M channels per frequency k. We also set the FDAF loss $\mathcal{L}[\tau]$ to be the afterISE via after(9) with gradient computed with respect to $\mathbf{w}_\mathrm{k}[\tau]^\mathsf{H}$ as $\boldsymbol{\nabla}_\mathrm{k} after[\tau] = \mathbf{u}_\mathrm{k} after[\tau](y_{\mathrm{km}}[\tau] - d_{\mathrm{km}}[\tau])^*$.

C. Optimizer Architecture and Loss

1) Architecture: Our optimizer g_ϕ is inspired by conventional AF optimizers such as LMS, NLMS, and BD-RLS, but updated to have a neural network form. In particular, we focus on making a generalized, stochastic variant of an RLS-like optimizer that is applied independently per frequency k to our optimizee parameters, but coupled across channels and time frames to allow our approach to model interactions between channels and frames and vectorize across frequency. To do so, we use a recurrent neural network (RNN) where the weights ϕ are shared across all frequency bins, but we maintain separate state $\boldsymbol{\psi}_\mathrm{k}[\tau]$ per frequency. The inputs to our optimizer at frequency k are $\boldsymbol{\xi}_\mathrm{k}[\tau] = [\boldsymbol{\nabla}_\mathrm{k}[\tau], \mathbf{u}_\mathrm{k}[\tau], \mathbf{d}_\mathrm{k}[\tau], \mathbf{y}_\mathrm{k}[\tau], \mathbf{e}_\mathrm{k}[\tau]]$, where $\boldsymbol{\nabla}_\mathrm{k}[\tau]$ is the gradient of the optimizee with respect to $\boldsymbol{\theta}_\mathrm{k}$, and $\mathbf{e}_\mathrm{k}[\tau] = \mathbf{d}_\mathrm{k}[\tau] - \mathbf{y}_\mathrm{k}[\tau]$. Our optimizer outputs are the update $\boldsymbol{\Delta}_\mathrm{k}[\tau]$ and the internal state $\boldsymbol{\psi}_\mathrm{k}[\tau + 1]$, resulting in

$$(\boldsymbol{\Delta}_\mathrm{k}[\tau], \boldsymbol{\psi}_\mathrm{k}[\tau + 1]) = g_\phi(\boldsymbol{\xi}_\mathrm{k}[\tau], \boldsymbol{\psi}_\mathrm{k}[\tau]) \tag{22}$$

$$\boldsymbol{\theta}_\mathrm{k}[\tau + 1] = \boldsymbol{\theta}_\mathrm{k}[\tau] + \boldsymbol{\Delta}_\mathrm{k}[\tau]. \tag{23}$$

Our design is in contrast to LMS-, NLMS-, RMSProp-like optimizers that have no state (e.g. LMS) or minimal state dynamics (e.g. NLMS, RMSProp). In addition, these optimizers as well as other learned optimizers [72] typically apply updates independently per element. For a comparison of optimizer inputs, state, parameters, and gradients, please see Table II.

To define our RNN in more detail, we use a small network composed of a linear layer, nonlinearity, and two Gated Recurrent Unit (GRU) layers with hidden size $H = 32$, followed by two additional linear layers with nonlinearities, where all layers are complex-valued. We always re-scale the inputs element-wise

via

$$\ln(1 + |\boldsymbol{\xi}|)e^{j\angle \boldsymbol{\xi}}, \tag{24}$$

to reduce the dynamic range and facilitate training, but keep the phases unchanged. This pre-processing was found useful in several previous works [63], [72], although previous work used explicit clipping, which we found unnecessary.

2) Loss: We examine two meta losses $\mathcal{L}_M(\cdot)$ to learn our optimizer parameters ϕ. First, we define the FD *frame independent* loss

$$\mathcal{L}_M = \ln \frac{1}{L} \sum_{\tau}^{\tau+L} E[||\mathbf{d}_\mathrm{m}[\tau] - \mathbf{y}_\mathrm{m}[\tau]||^2], \tag{25}$$

where $\mathbf{d}_\mathrm{m}[\tau]$ and $\mathbf{y}_\mathrm{m}[\tau]$ are the desired and estimated FD signal vectors of the reference channel m (e.g. m $= 0$). Intuitively, to compute this loss for a given optimizer g_ϕ, we unroll (20) for a time horizon of L time frames, compute the FD mean-squared error over L frames, and then take the logarithm to reduce the dynamic range, which we found to empirically improve learning. This loss ignores the temporal order of AF updates and optimizes for filter coefficients that are unaware of any downstream STFT processing, but the idea of accumulating independent time-step losses is common [72].

Second, we define the time-domain *frame-accumulated* loss

$$\mathcal{L}_M = \ln E[||\bar{\mathbf{d}}_\mathrm{m}[\tau] - \bar{\mathbf{y}}_\mathrm{m}[\tau]||^2] \tag{26}$$

$$\bar{\mathbf{d}}_\mathrm{m}[\tau] = \mathrm{cat}(\underline{\mathbf{d}}_\mathrm{m}[\tau], \underline{\mathbf{d}}_\mathrm{m}[\tau + 1], \ldots, \underline{\mathbf{d}}_\mathrm{m}[\tau + L - 1]) \tag{27}$$

$$\bar{\mathbf{y}}_\mathrm{m}[\tau] = \mathrm{cat}(\underline{\mathbf{y}}_\mathrm{m}[\tau], \underline{\mathbf{y}}_\mathrm{m}[\tau + 1], \ldots, \underline{\mathbf{y}}_\mathrm{m}[\tau + L - 1]), \tag{28}$$

where $\underline{\mathbf{d}}_\mathrm{m}[\tau]$ and $\underline{\mathbf{y}}_\mathrm{m}[\tau]$ are the time-domain desired and estimated responses of reference channel m and $\bar{\mathbf{d}}_\mathrm{m}[\tau] \in R^{RL}$ and $\bar{\mathbf{y}}_\mathrm{m}[\tau] \in R^{RL}$. Intuitively, to compute this loss for a given optimizer g_ϕ, we run (20) for a time horizon of L frames, concatenate the sequence of time-domain outputs and target signals to form longer signals, then compute the time-domain MSE loss, and take the logarithm. While both losses use the same time-horizon, the frame accumulated loss allows us to model boundaries between adjacent updates and implicitly learn updates that are STFT consistent [85]. To the best of our knowledge, our frame accumulated loss is novel for AFs.

3) Computational Complexity: We compare the computational complexity of our proposed approach to conventional optimizers in Table III. We note that the complexity of our approach is dependent on the hidden state size H of our RNN, but is linear in channels M and blocks B, whereas BD-RLS is quadratic in M and B, but does not depend on H. Note

Algorithm 1: Training Algorithm.

function Forward$g_\phi, \psi, h_\theta, \underline{\mathbf{U}}, \underline{\mathbf{d}}_m$
  for $\tau \leftarrow 0$ to L **do**   ▷Unroll
    $\mathbf{U}[\tau], \mathbf{d}[\tau] \leftarrow \text{STFT}(\underline{\mathbf{U}}[\tau], \underline{\mathbf{d}}_m[\tau])$ ▷Forward STFT
    $\mathbf{y}_m[\tau] \leftarrow h_\theta(\mathbf{U}[\tau])$ ▷Save filter output
    $\underline{\mathbf{y}}_m[\tau] \leftarrow \text{STFT}^{-1}(\mathbf{y}_m[\tau])$ ▷Inverse STFT
    $\mathcal{L} \leftarrow ||\mathbf{d}_m[\tau] - \mathbf{y}_m[\tau]||^2$ ▷AF frame loss
    $\nabla[\tau] \leftarrow \text{GRAD}(\mathcal{L}, \theta)$ ▷Filter gradient
    for $k \leftarrow 0$ to K **do**   ▷Apply update per freq
      $\xi_k[\tau] \leftarrow [\nabla_k[\tau], \mathbf{u}_k[\tau], \mathbf{d}_k[\tau], \mathbf{y}_k[\tau], \mathbf{e}_k[\tau]]$
      $(\Delta_k[\tau], \psi_k[\tau+1]) \leftarrow g_\phi(\xi_k[\tau], \psi_k[\tau])$
      $\theta_k[\tau+1] \leftarrow \theta_k[\tau] + \Delta_k[\tau]$
  $\bar{\mathbf{y}} \leftarrow \text{CAT}(\underline{\mathbf{y}}[\tau] \, \forall \tau)$ ▷Concatenate accumulated frames
  return $\bar{\mathbf{y}}, \psi, h_\theta$
function Train$\mathcal{D}$
  $\phi \leftarrow$ [90] init
  while ϕ **not** CONVERGED **do**   ▷Train loop
    $\underline{\mathbf{U}}, \underline{\mathbf{d}}_m \leftarrow \text{SAMPLE}(\mathcal{D})$ ▷Sample signals
    $\theta, \psi \leftarrow \mathbf{0}, \mathbf{0}$ ▷Init filter and optimizer state
    for $n \leftarrow 0$ to end **do**   ▷Loop across long signal
      $\bar{\mathbf{U}}, \bar{\mathbf{d}}_m \leftarrow \text{NEXTL}(\underline{\mathbf{U}}, \underline{\mathbf{d}}_m)$ ▷Get next L frames
      $\bar{\mathbf{y}}, \psi, h_\theta \leftarrow \text{FORWARD}(g_\phi, \psi, h_\theta, \bar{\mathbf{U}}, \bar{\mathbf{d}}_m)$
      $\mathcal{L}_M \leftarrow$ via (26)   ▷Meta loss
      $\nabla \leftarrow \text{GRAD}(\mathcal{L}_M, \phi)$ ▷Optimizer gradient
      $\phi[n+1] \leftarrow \text{METAOPT}(\phi[n], \nabla)$ ▷Update opt
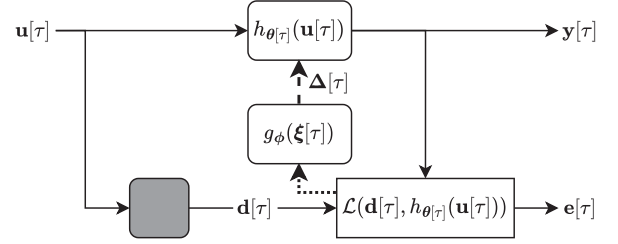 **return** $\hat{\phi}$ ▷Return best ϕ

Fig. 1.   System identification block diagram. System inputs are fed to both the adaptive filter and the true system (shaded box). The adaptive filter is updated to mimic the true system.

that prior work on learned optimizers, including our own [63], performs optimization completely element-wise, resulting in a larger complexity of $\mathcal{O}(\text{KMBH}^2)$.

D. Learning the Optimizer

To learn an optimizer g_ϕ from data, we solve (21) using standard deep learning tools (i.e. JAX [86], [87]), including the use of automatic differentiation for training and inference. In addition, we use truncated backpropagation through time (TBPTT) [88] with a stochastic gradient descent optimizer, Adam [89], that we call our meta optimizer. We show a simplified form of our training algorithm in Alg. 1 using our frame accumulated loss and a batch size of one, where STFT is an OLA or OLS processor, GRAD returns the gradient of the first argument with respect to the second, SAMPLE randomly samples signals from a dataset $\mathcal{D}$, and NEXTL grabs the next L time buffers from a longer signal. In practice, we use batching.

In more detail, we train g_ϕ until the application specific mean SNR metric on the validation fold of a dataset $\mathcal{D}$ has not improved for four epochs. For each of our five applications, we use (29), (30), (31), (36), and (45), respectively. We halve the step size after an epoch with no improvement and define an epoch as 10 passes through the dataset with a batch size of 32. We have a hard stop for training at 100 epochs. From initial experimentation, we note the choice of the meta-optimizer and meta-optimizer parameters have large impact on performance.

We initialize ϕ via [90] and, when using Adam[2], we found it was important to use a small learning of 10^{-4} and a large momentum term of .99. To help stabilize training, we use gradient clipping with a clipping value of 10. We use identical g_ϕ settings for each task with input and output sizes set to $M \cdot B \cdot 5$ and $M \cdot B$, respectively, all intermediate layer sizes set to 32, and separate ReLU nonlinearities for real and imaginary.

Compared to our past work on single-channel single-talk AEC [63], we 1) change our loss from (25) without the log to (26) 2) change the input of our learned optimizer from $\nabla_k[\tau]$ to $\xi_k[\tau]$ with updated pre-processing (24) and 3) develop multi-channel Meta-AFs or development multi-block Meta-AFs to perform time- and block-coupled updates instead of separate, independent updates per frequency, block, and channel. To validate our approach, we apply our algorithm to five audio tasks including system identification, acoustic echo cancellation, equalization, single/multi-channel dereverberation, and beamforming in Sections IV, V, VI, VII, and VIII, respectively and compare against conventional methods. For our first task, we also ablate key design decisions and then lock a single configuration for all remaining tasks and experiments.

IV. SYSTEM IDENTIFICATION ABLATIONS

A. Problem Formulation

For our first task, we train a Meta-AF to perform online system identification (ID). We seek to estimate the transfer function between an audio source and a microphone over time, as shown in Fig. 1. This is commonly done in room acoustics and head-related transfer function measurements for virtual and augmented reality. To do so, we model the unknown system (e.g. acoustic room) with a linear frequency-domain filter h_θ (optimizee architecture), inject input signal $\mathbf{u}$ into the system, measure the response $\mathbf{d}$, and adapt the filter weights $\theta = \{\mathbf{w}_k\}$ using our learned Meta-ID AF, g_ϕ. The AF loss is the ISE between the desired response, $\mathbf{d}_k[\tau]$, and the AF predicted response, $\mathbf{y}_k[\tau] = \mathbf{w}_k[\tau]^H \mathbf{u}_k[\tau]$.

---

[2]We use a custom implementation of Adam in our source code due to a complex-value error issue in the JAX implementation.
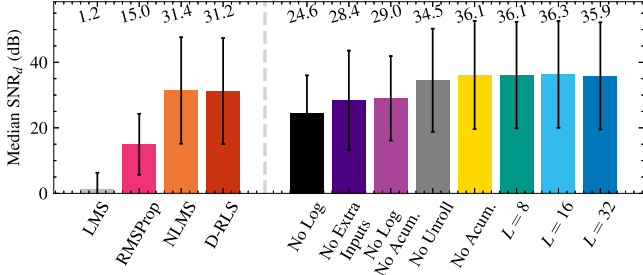
Fig. 2. Optimizer design decision ablation. Using an accumulated log-loss leads to our best model, particularly for more complex tasks we address later on. RLS-like inputs are also useful. The exact value of L is not critical, but larger is better.
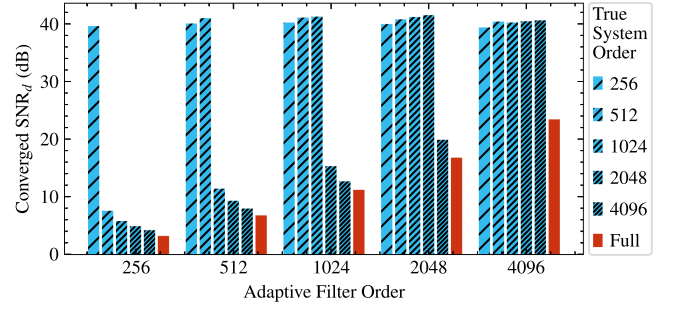

Fig. 3. Evaluating the effect of different true system orders and adaptive filter orders. Our learned AFs can operate well across a variety of linear system orders, even when training is restricted to systems of a fixed length (1024 taps).

B. Experimental Design

We 1) ablate key design decisions of our AF optimizer architecture and loss in Section IV-C and 2) study the robustness of our approach to modeling errors by changing the AF and true system order at test time in Section IV-D. For evaluation metrics, we use the segmental SNR (SNR_d) between the desired and estimated signals in dB. We compute this per-frame as,

$$\mathrm{SNR}_d(\underline{\mathbf{d}}[\tau], \underline{\mathbf{y}}[\tau]) = 10 \cdot \log_{10}\left(\frac{\|\underline{\mathbf{d}}[\tau]\|^2}{\|\underline{\mathbf{d}}[\tau] - \underline{\mathbf{y}}[\tau]\|^2}\right), \quad (29)$$

where higher is better. We train g_ϕ via Algorithm 1 using a single GPU to adapt an OLS filter with a rectangular window size $N = 2048$ and hop size $R = 1024$ on 16 kHz audio. For training data, we created a dataset by convolving the far-end speech from the single-talk portion of the ICASSP 2021 AEC Challenge data [91] with room impulse responses (RIRs) from [92] truncated to 1024 taps.

C. Optimizer Design Results and Discussion

In Fig. 2, we change one aspect of our final design at a time and show how each change negatively effects performance. Our final design, $L = 16$, is colored light blue, alternative configurations are colored differently, and our previous work [63] is approximately equivalent to a no log, no accumulation, no extra inputs setting. After this ablation, we fix these values for all remaining experiments and do not perform any further tuning except changing the dataset used for training and using different checkpoints caused by early stopping on validation performance. In contrast, we re-tune all conventional optimizer baselines for all subsequent tasks on held-out validation sets.

1) Loss Function: First, we compare our selected frame accumulated loss model (light blue) to the frame accumulated loss without log scaling (black) as well as the frame independent loss (yellow) and without log scaling (light-purple). As shown, the log-loss has the single largest effect on SNR_d and yields an astounding 11.7/7.3 dB improvement compared to the no-log losses. When we compare the independent vs. accumulated loss, the accumulation provides a .2 dB improvement. However, when we listen to the estimated response, especially in more

complex tasks, we found that the accumulated loss introduces fewer artifacts and perceptually sounds better. Thus, we fix the optimizer loss to be (26).

2) Input Features: Having selected the optimizer loss, we compare the model inputs. We compare setting the optimizer input to be only the gradient $\boldsymbol{\xi}_{\mathrm{k}}[\tau] = [\boldsymbol{\nabla}\mathrm{k}[\tau]]$ for an LMS-like learned optimizer (dark purple) and setting the optimizer to be the full signal set $\boldsymbol{\xi}_{\mathrm{k}}[\tau] = [\boldsymbol{\nabla}\mathrm{k}[\tau], \mathbf{u}_{\mathrm{k}}[\tau], \mathbf{d}_{\mathrm{k}}[\tau], \mathbf{y}_{\mathrm{k}}[\tau], \mathbf{e}_{\mathrm{k}}[\tau]]$ for our selected RLS-like learned optimizer (light blue). As shown, the inputs have the second largest effect on SNR_d and using the full signal set yields a notable 7.9 dB improvement. Thus, we set the input to be the full signal set.

3) AF Unroll: With the optimizer loss and inputs fixed, we evaluate four different values of AF unroll length, $L = 2, 8, 16, 32$, where L is the number of time-steps over which the optimizer loss is computed in (26). Intuitively, a larger unroll introduces less truncation bias but may be more unstable during training due to exploding or vanishing gradients. The case where $L = 2$ corresponds to no unroll, since for $L = 1$ the meta loss is not a function of the optimizer parameters and yields a zero gradient. As shown, for no unroll $L = 2$ (grey), we get a reduction of the SNR_d by 1.8 dB compared to our best model. When selecting the unroll between 8, 16, 32, however, there is a small ($<$1 dB) overall effect. That said, we find that longer unroll values work better in challenging scenarios but take longer to train. As a result, we select an unroll length of 16, as it represents a good trade-off between performance and training time. Note, the unroll length only effects training and is not used at test time.

D. System Order Modeling Error Results and Discussion

Given our fixed set of optimizer and meta-optimizer parameters, we evaluate the robustness of our Meta-ID AF to modeling errors by studying what happens when we use an optimizee filter that is too short or too long compared to the true system. We do so by testing a learned optimizer with 1) optimizee filter lengths between 256 and 4096 taps and 2) held-out signals with true filter lengths between 256 and 4096, as well as full length systems (up to 32,000 taps).

Results are shown in Fig. 3. We measure performance by computing the segmental SNR_d score via (29) at convergence.
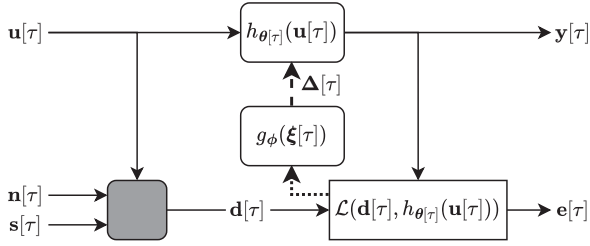
Fig. 4. AEC block diagram. System inputs are fed to the adaptive filter and true system (shaded box). The adaptive filter is updated to mimic the true system. The desired response can be noisy due to near-end noise and speech ($\mathbf{n}[\tau], \mathbf{s}[\tau]$).

Fig. 5. AEC single-talk performance. Meta-AEC converges rapidly and has better steady-state performance. We use this same legend for all AEC plots.

As expected, when the adaptive filter order is equal to or greater than the true system order, we achieve SNRs of ≈ 40 dB. It is interesting to note that our learned AF was only ever trained on optimizee filters with an order of 1024 and 1024 tap true systems. This experiment suggests our learned optimizers can generalize to new optimizee filter orders.

V. ACOUSTIC ECHO CANCELLATION ABLATION

A. Problem Formulation

In our second task, we train a Meta-AF for acoustic echo cancellation (AEC). The goal of AEC is to remove the far-end echo from a near-end signal for voice communication by mimicking a time-varying transfer function as show in Fig. 4. The far-end refers to the signal transmitted to a local listener and the near-end is captured by a local mic. We model the unknown transfer function between the speaker and mic with a linear multi-delay frequency-domain filter h_θ, measure the noisy response $\mathbf{d}$ which includes the input signal $\mathbf{u}$, noise $\mathbf{n}$, and near-end speech $\mathbf{s}$, and adapt the filter weights θ using our learned Meta-AEC AF, g_ϕ. The time-domain signal model is $\underline{\mathbf{d}}[t] = \underline{\mathbf{u}}[t] * \underline{\mathbf{w}} + \underline{\mathbf{n}}[t] + \underline{\mathbf{s}}[t]$. The AF loss is the ISE between the near-end and the predicted response. The FDAF near-end speech estimate is $\hat{\mathbf{s}}_k[\tau] = \mathbf{d}_k[\tau] - \mathbf{w}_k[\tau]^H \mathbf{u}_k[\tau]$.

B. Experimental Design

We compare our approach to LMS, RMSProp, NLMS, BD-RLS, a diagonal Kalman filter (D-KF) model [11], and Speex [12] for a variety of acoustic echo cancellation scenarios. Our scenarios, in increasing difficulty, include single-talk, double-talk, double-talk with a path change, and noisy double-talk with a path change and non-linearity. Single-talk refers to the case where only the far-end input signal $\mathbf{u}$ is active. Double-talk refers to the case where both the far-end signal $\mathbf{u}$ and near-end talker signal $\mathbf{s}$ are active at the same time. A path change refers to the case where the true system transfer function is abruptly changed (e.g. during a phone call). Nonlinearities refer to the case where the true system is not strictly linear (e.g. harmonic loudspeaker distortion). We train a single g_ϕ for AEC and then test it for each scene type against all hyperparameter-tuned baselines.

We measure echo cancellation performance using segmental echo-return loss enhancement (ERLE) [93] and short-time
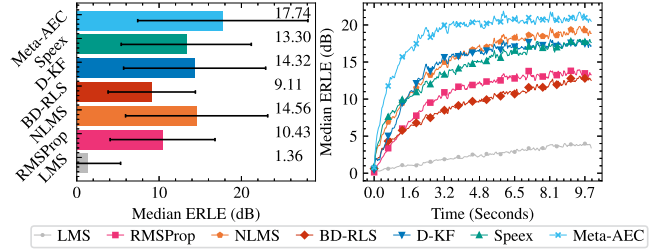
objective intelligibility (STOI) [94]. Segmental ERLE is

$$\text{ERLE}(\underline{\mathbf{d}}_{\mathbf{u}}[\tau], \underline{\mathbf{y}}[\tau]) = 10 \cdot \log_{10} \left(\frac{\|\underline{\mathbf{d}}_{\mathbf{u}}[\tau]\|^2}{\|\underline{\mathbf{d}}_{\mathbf{u}}[\tau] - \underline{\mathbf{y}}[\tau]\|^2} \right), \quad (30)$$

where $\underline{\mathbf{d}}_{\mathbf{u}}$ is the noiseless system response and higher values are better. When averaging, we discard silent frames using an energy-threshold VAD. In scenes with near-end speech, we use STOI $\in [0, 1]$ to measure the preservation of near-end speech quality. Higher STOI values are better. We train g_ϕ via Algorithm 1 using one GPU, which took ≈ 72 hours. We use a four-block multi-delay OLS filter (MDF) with window sizes of $N = 1024$ samples and a hop of $R = 512$ samples on 16 kHz audio. In double-talk scenarios, we use the noisy near-end, $\mathbf{d}$ as the target and do not use oracle cancellation results (such as $\underline{\mathbf{d}}_{\mathbf{u}}$).

With respect to datasets for single-talk, double-talk, and double-talk with path-change experiments, we re-mix the synthetic fold of the ICASSP 2021 AEC Challenge dataset (ICASSP-2021-AEC) [91] with impulse responses from [92]. We partition [92] into non-overlapping train, test, and validation folds and set the signal-to-echo-ratio randomly between $[-10, 10]$ with uniform distribution. To simulate a scene change, we splice two files such that the change occurs randomly between seconds four and six. For the noisy double-talk with nonlinearity experiments, we use the synthetic fold of [91]. We apply a random circular shift and random scale to all files, each ten seconds long. For each task, there are 9000 training, 500 validation, and 500 test files. Finally, we also use an unmodified version of the ICASSP-2021-AEC training, validation, and test set (does not include scene changes) to compare to other previously published works directly.

C. Results and Discussion

Overall, we find that our approach significantly outperforms all previous methods in all scenarios, but has a larger advantage in harder scenes—more details discussed below.

1) Single-Talk: Our approach (light blue, x) exhibits strong single-talk performance and surpasses all baselines by $> \approx 3$ dB in both median and converged ERLE, as shown in Fig. 5. Additionally, Meta-AEC converges fastest, reaching steady-state ≈ 4 seconds before other baselines.

2) Double-Talk: Our method (light blue, x) converges fastest in double-talk, and matches the D-KF in converged-performance, as shown in Fig. 6. Meta-AEC converges ≈ 5 seconds faster while scoring better in STOI. This result is striking as
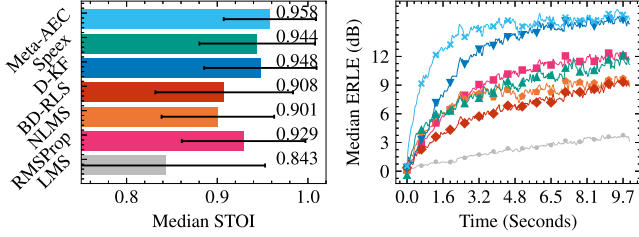
Fig. 6. AEC double-talk performance. Meta-AEC converges fastest and has similar peak performance to the D-KF, while preserving near-end speech quality.
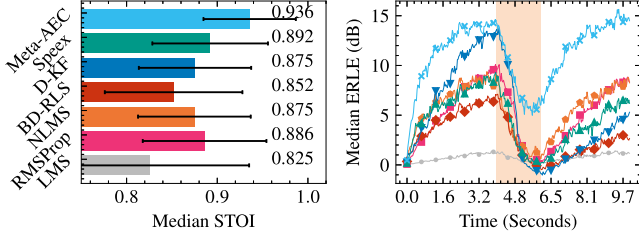

Fig. 7. AEC double-talk with a path change (shaded region) performance. Our approach re-converges rapidly with high speech quality.
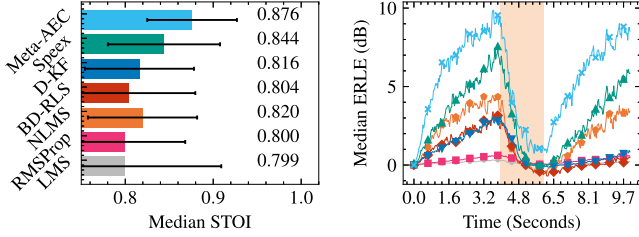

Fig. 8. AEC noisy double-talk with nonlinearities and a path change (shaded region) performance. Meta-AEC learns to compensate for the nonlinearity.

it is typically necessary to either explicitly freeze adaptation via double-talk detectors or implicitly freeze adaption via carefully derived updates as found in both the D-KF model (dark blue, down triangle) and Speex (green, up triangle). We hypothesize our method automatically learns how to adapt in double-talk in a completely autonomous fashion.

3) Double-Talk With Path Change: Our method (light blue, x) is able to more robustly handle double-talk with path changes compared to other methods as shown in Fig. 7. Similar to straight double-talk, our approach effectively learns how to deal with adverse conditions (i.e. a path change) without explicit supervision, converging and reconverging in ≈ 2.5 seconds, with. 044 better median STOI. All other algorithms similarly struggle, even Speex (green, up triangle), which has explicit self-resetting and dual-filter logic.

4) Noisy Double-Talk With Nonlinearities and Path Change: When we evaluate scenes with noise, nonlinearies to simulate loudspeaker distortion, and path changes, we find that our Meta-AEC approach (light blue, x) continues to perform well, as shown in Fig. 8. That is, our peak-performance is ≈ 2 dB above the nearest baseline. In STOI, Meta-AEC scores. 027 above

Method	STOI	STOI⋆	ERLE (dB)	ERLE⋆ (dB)
LMS	0.794	0.794	0.937	0.560
RMSProp	0.802	0.856	1.02	4.63
NLMS	0.854	0.861	4.81	5.96
BD-RLS	0.829	0.835	3.66	4.07
D-KF [11]	0.817	0.875	1.98	6.55
wRLS, $\beta = 0.2$ [81]	N.A	0.85	N.A	N.A
WebRTC-AEC3 [82]	0.82	N.A	N.A	N.A
Speex [12]	0.869	N.A	3.92	N.A
Meta-AEC	**0.881**	**0.899**	**7.73**	**9.13**

Speex (green, up triangle). We hypothesize that our approach effectively learns to compensate for the signal model inaccuracy, even if we only use a linear filter.

5) ICASSP 2021 AEC Challenge Results: In addition to testing with our own variant of the ICASSP-AEC-2021 dataset that includes scene changes, we test our work with an unmodified version of the test set in Table IV. Furthermore, we evaluate performance when we train (or tune) on our custom training dataset versus when we train on the original training dataset (denoted with ⋆). See also a similar Table in past work [81]. To the best of our knowledge, this dataset is the most recent and widely used dataset for benchmarking AEC algorithms. Here, results from WebRTC-AEC3 and wRLS, $\beta = 0.2$ come from past work [81]. All other methods have the same MDF filtering architecture as described above. Our approach outperforms all methods we compare against for both training datasets, including Speex and wRLS, which were the linear filters used in the first- and second-place winners of the ICASSP 2021 AEC Challenge [83]. Interestingly, there is a significant effect on training or tuning with data that includes scene changes (ours) vs. the original data (e.g. RMSProp and D-KF [61]). That is, because the ICASSP-2021-AEC train and test set does not include scene changes, most algorithms give better performance when trained/tuned on the matching, unmodified train set, even though such results are less realistic.

6) Computational Complexity: Our learned AF has a single CPU core real-time factor (RTF) (computation/time) of ≈ 0.36, and 32 ms latency (OLS hop size). Our optimizer network alone has $\approx 14 K$ complex-valued parameters and single CPU core RTF of ≈ 0.31. While this performance is already real-time capable, we suspect it could easily be improved with better optimized code.

VI. EQUALIZATION ABLATION

A. Problem Formulation

For our third task, we train a Meta-AF for the inverse modeling task of equalization (EQ). Here, our goal is to estimate the inverse of an unknown transfer function, while only observing input and outputs of the forward system, as shown in Fig. 9. This is a common component of loudspeaker tuning. We model
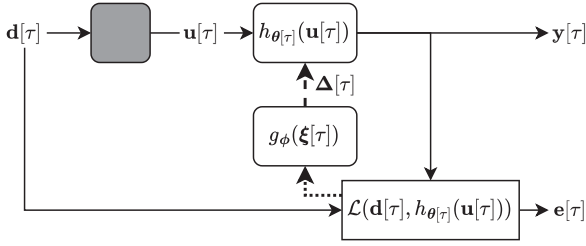
Fig. 9. Inverse modeling block diagram. System outputs are fed to the adaptive filter. The adaptive filter is continually updated to invert the unknown system (shaded box).

Fig. 10. Equalization results for signal (SNR_d) and system (SNR_w) SNR. Meta-EQ performance is the least impacted by constraints.

Fig. 11. Comparison of true and estimated systems over time. The Meta-EQ system rapidly fits to the correct inverse model. The top plot shows an example system and the bottom shows SNR_w over time across the test set.

the unknown inverse transfer function with a linear frequency-domain filter $h_{\boldsymbol{\theta}}$, measure the response $\mathbf{d}$ to an input signal $\mathbf{u}$, and adapt the filter weights $\boldsymbol{\theta}$ using our learned Meta-EQ AF $g_{\boldsymbol{\phi}}$. The AF loss is the ISE between the true and predicted responses. More precisely, the frequency-domain AF output is $\mathbf{y}_{\mathrm{k}}[\tau] = \mathbf{w}_{\mathrm{k}}[\tau]^{\mathsf{H}}\mathbf{u}_{\mathrm{k}}[\tau]$.

B. Experimental Design

We compare our Meta-EQ approach to LMS, RMSProp, NLMS, and D-RLS on the task of frequency equalization. Additionally, we ablate the equalization filtering mechanics for two cases: constrained and unconstrained filters (optimizee architecture modifications). In the constrained case, we set $h_{\boldsymbol{\theta}}$ to use standard OLS. However, in the unconstrained case, we set $h_{\boldsymbol{\theta}}$ to use aliased OLS where $\mathbf{Z}_w = \mathbf{I}_K$. This comparison lets us test if Meta-EQ is automatically learning constraint-aware update rules. We train a new $g_{\boldsymbol{\phi}}$ for each case (no separate tuning) and tune all baselines for each case.

We measure performance with signal SNR_d, and system SNR_w. We define these as

$$\mathrm{SNR}_d(\underline{\mathbf{d}}, \underline{\mathbf{y}}) = 10 \cdot \log_{10}\left(\frac{\|\underline{\mathbf{d}}\|^2}{\|\underline{\mathbf{d}} - \underline{\mathbf{y}}\|^2}\right) \tag{31}$$

$$\mathrm{SNR}_w(\hat{\mathbf{w}}^{-1}, \mathbf{w}^{-1}) = 10 \cdot \log_{10}\left(\frac{\||\mathbf{w}^{-1}|\|^2}{\||\mathbf{w}^{-1}| - |\hat{\mathbf{w}}^{-1}|\|^2}\right) \tag{32}$$

respectively, where higher is better. We compute SNR_w using the inverse system magnitude, which ignores the phase. We train $g_{\boldsymbol{\phi}}$ via Algorithm 1 on one GPU, which took at most 36 h. We use an OLS filter with a window size of $N = 1024$ samples and a hop of $R = 512$ samples on 16 kHz audio.

To construct the equalization dataset, we use speech from the DAPS dataset [95], take the *cleanraw* recordings as inputs, and apply random equalizer filters from the sox library to generate the outputs, where we randomly pick between $[5, 15]$ filters with settings $c \in [1, 8]$ kHz, $g \in [-18, 18]$, and $q \in [.1, 10]$. All values are sampled uniformly at random to produce 16,384 train, 2048 validation and 2048 test signals, all 5 seconds long. At train, validation, and test time we truncate the system response to 512 taps.
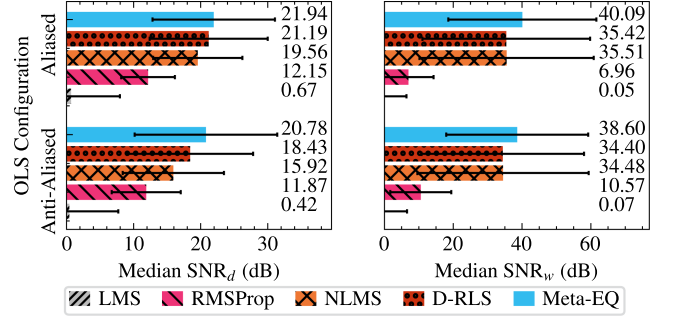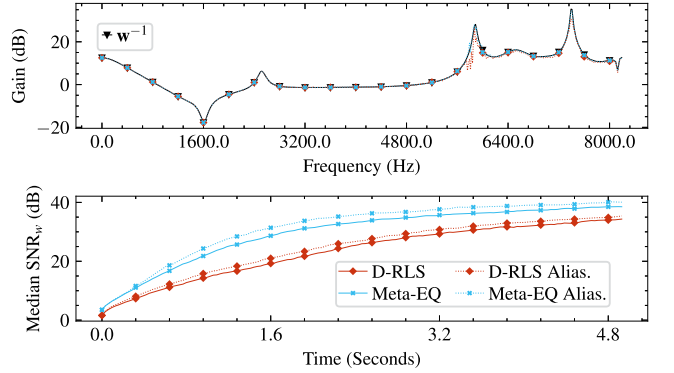
C. Results and Discussion

We find our approach (blue, solid) outperforms LMS, RMSProp, NLMS, and D-RLS for our equalization task by a noticeable margin as shown in Fig. 10 and further verify with a qualitative analysis plot in Fig. 11.

1) Constrained vs Unconstrained: For the unconstrained case, our method outperforms D-RLS in SNR_d by .75 dB and by 4.67 dB in SNR_w. When we look at the constrained case, the performance for all models is degraded. Interestingly, however, our performance is proportionally degraded the least. We hypothesize that our approach learns to perform updates which are aware of the constraint.

2) Temporal Performance Analysis: We display final system and convergence results in Fig. 11. Our Meta-EQ model finds better solutions more rapidly than D-RLS. D-RLS diverged $\approx$ 300 times but Meta-EQ never did.

3) Computational Complexity: Our learned AF has a single CPU core RTF of $\approx$0.24, and 32 ms latency. Our optimizer network alone has $\approx$14 K complex-valued parameters and single CPU core RTF of $\approx$0.19.

VII. DEREVERBERATION ABLATION

A. Problem Formulation

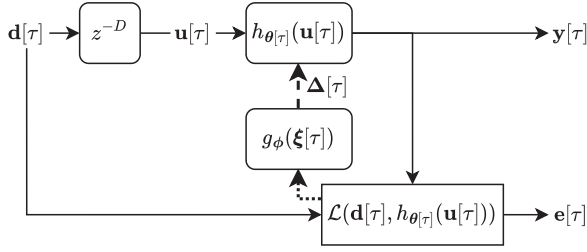For our fourth task, we train a Meta-AF to perform dereverberation via multi-channel linear prediction (MCLP) or the

Fig. 12. Prediction block diagram. A buffer of past inputs are used to estimate a future, unknown signal. The delay, z^{-D} signifies a delay by D frames.
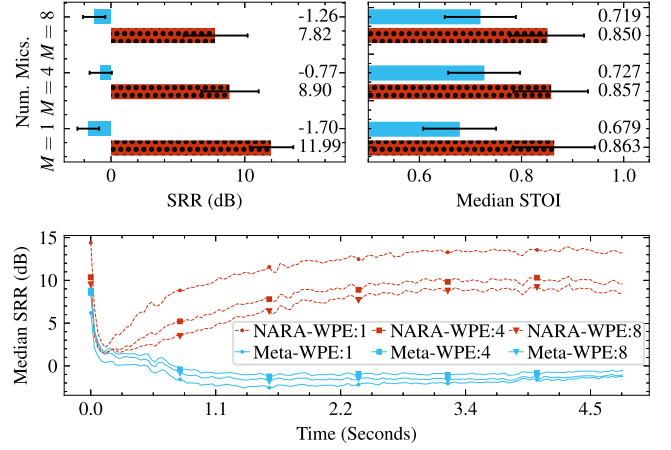

Fig. 13. Dereverberation performance in terms of SRR. Meta-WPE excels in SRR, a metric which measures energy removed. However, in STOI, Meta-WPE scores worse.

weighted prediction error (WPE) formulation, as is commonly used for speech-to-text pre-processing. The WPE formulation is based on the idea of being able to predict the reverberant part of a signal from a linear combination of past samples, most commonly in the frequency-domain [25], [26] and shown as a block diagram in Fig. 12. Using our method, we use a multi-channel linear frequency-domain filter h_θ and adapt the filter weights $\boldsymbol{\theta}$ using a learned AF g_ϕ to minimize the normalized ISE AF loss below.

Assuming an array of M microphones, we estimate a dereverberated signal with a linear model via

$$\hat{s}_{km}[\tau] = d_{km}[\tau] - \mathbf{w}_k[\tau]^H \mathbf{u}_k[\tau] \qquad (33)$$

where $\hat{s}_{km}[\tau] \in \mathbb{C}$ is the current dereverberated signal estimate at frequency k and channel m, $d_{km}[\tau] \in \mathbb{C}$ is the input microphone signal, $\mathbf{w}_k[\tau] \in \mathbb{C}^{BM}$ is a per frequency filter with B time frames and M channels flattened into a vector, and $\mathbf{u}_k[\tau] \in \mathbb{C}^{BM}$ is a running flattened buffer of $\mathbf{d}_k[\tau - D]$.

We then minimize a per channel and frequency loss

$$\mathcal{L}(\hat{s}_{km}[\tau], \boldsymbol{\lambda}_k[\tau]) = \frac{\|\hat{s}_{km}[\tau]\|^2}{\boldsymbol{\lambda}_k^2[\tau]}, \qquad (34)$$

$$\boldsymbol{\lambda}_k^2[\tau] = \frac{1}{M(B+D)} \sum_{n=\tau-B-D}^{\tau} \mathbf{d}_k[n]^H \mathbf{d}_k[n], \qquad (35)$$

where $\lambda_k^2[\tau]$ is a running average estimate of the signal power and $\mathbf{d}_k[\tau] \in \mathbb{C}^M$. We use this formation within our framework to perform online multi-channel dereverberation or Meta-WPE and focus on dereverberating a single output channel.

B. Experimental Design

We compare our Meta-WPE to frame-online NARA-WPE [30], a BD-RLS based AF which uses the WPE formulation. We ablate the filter size and inputs across: $M = 1, 4, 8$ microphones (optimizee size and input modification). We seek to test if our method can scale from single- to multi- channel tasks without modification. We train a new g_ϕ for each M (no tuning). We measure performance with two metrics, segmental speech-to-reverberation ratio (SRR) [31] and STOI. SRR is a signal level metric and measures how much energy was removed

from the signal. It is computed as,

$$\text{SRR}(\mathbf{d}_k[\tau], \hat{\mathbf{s}}_k[\tau]) = 10 \cdot \log_{10}\left(\frac{\|\hat{\mathbf{s}}_k[\tau]\|^2}{\|\mathbf{d}_k[\tau] - \hat{\mathbf{s}}_k[\tau]\|^2}\right), \quad (36)$$

where smaller values indicate more removed energy and better performance. STOI is computed between the dereverberated signal estimate and the ground truth anechoic signal. We train g_ϕ via Algorithm 1 on two GPUs, which took at most 24 h. We use an OLA filter with a Hann window size of $N = 512$ samples and a hop of $R = 256$ samples on 16 kHz audio. We fix the buffer size $B = 5$ taps and the delay to $D = 2$ frames.

We use the simulated REVERB challenge dataset [96]. The REVERB challenge contains echoic speech mixed with noise at 20 dB in small (T60 = .25 sec), medium (T60 = .5 sec) and large (T60 = .7 sec) rooms at near and far distances. The array is circular with a diameter of 20 cm. Background noises are generally stationary. The dataset has 7861 training files, 1484 validation files, and 2176 test files.

C. Results and Discussion

We find our approach (blue, solid) outperforms NARA-WPE in SRR across all filter configurations, but is worse in STOI as shown in Fig. 13. We discuss this below.

1) Overall and Temporal Performance: As shown in Fig. 13, Meta-WPE (blue, solid) scores strongly on SRR, where our single-channel Meta-WPE model scores better than 4 and 8 channel NARA-WPE (red, dotted) models. However, as shown by STOI, the perceptual quality is poor. While Meta-WPE is solving the prediction more rapidly, as shown by segmental SRR, it is not doing so in a perceptual pleasing manner. Previous studies [31], [97] have also encountered this phenomenon, and propose a variety of regularization tools to align the instantaneous optimization objective with perceptually pleasing processing. We re-ran these experiments with a buffer of size $B = 10$ as well as with larger and smaller optimizer network capacities and found this trend was consistent. As a result, we conclude our approach is very effectively improving the online
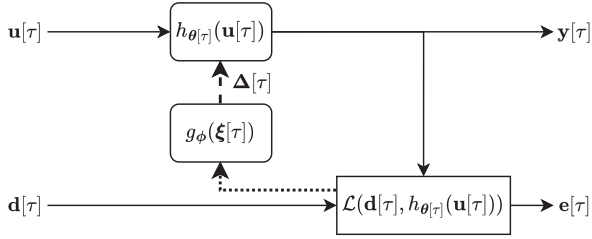
Fig. 14. Informed interference cancellation block diagram. An auxiliary signal is used as input to an adaptive filter which is fit to an alternate signal.

optimization of the target loss, but the instantaneous loss itself needs to be changed to better align with perception.

2) Computational Complexity: The 4 channel learned AF has a single CPU core RTF of ≈ 0.47, and 16 ms latency. Our Meta-WPE optimizer network alone has $\approx 17\,K$ complex-valued parameters and single CPU core RTF of ≈ 0.38.

VIII. BEAMFORMING ABLATION

A. Problem Formulation

For our fifth and final task, we train a Meta-AF for interference cancellation using the minimum variance distortionless response (MVDR) beamformer. The MVDR beamformer can be implemented as an AF using the generalized sidelobe canceller (GSC) [38] formulation and is commonly used for far-field voice communication and speech-to-text pre-processing. We depict a version of this problem setup in Fig. 14 and use a linear frequency-domain filter h_{θ}. We use the mixture $\mathbf{d}[\tau]$ as the target, informed input $\mathbf{u}[\tau]$, and adapt the filter weights $\theta[\tau]$ using our learned Meta-GSC AF g_{ϕ} and ISE AF loss.

Assuming an array of M microphones, we have the time-domain signal model for mic m via,

$$\underline{u}_{\mathrm{m}}[\mathrm{t}] = \underline{r}_{\mathrm{m}}[\mathrm{t}] * \underline{s}[\mathrm{t}] + \underline{n}_{\mathrm{m}}[\mathrm{t}] \tag{37}$$

where $\underline{u}_{\mathrm{m}}[\mathrm{t}] \in \mathbb{R}$ is the input signal, $\underline{n}_{\mathrm{m}}[\mathrm{t}] \in \mathbb{R}$ is the noise signal, $\underline{s}[\mathrm{t}] \in \mathbb{R}$ is the target signal, and $\underline{r}_{\mathrm{m}}[\mathrm{t}] \in \mathbb{R}$ is the impulse response from the source to mic m. In the time-frequency domain with a sufficiently long window, this can be reformulated as

$$u_{\mathrm{km}}[\tau] = r_{\mathrm{km}}[\tau] s_{\mathrm{k}}[\tau] + n_{\mathrm{km}}[\tau]. \tag{38}$$

The GSC beamformer also assumes access to a steering vector, $\mathbf{v}_{\mathrm{k}}$. While estimating the steering vector is well studied [38], it remains non-trivial for real-world applications. For our case, we assume access to a clean speech recording $\mathbf{s}_{\mathrm{k}}[\tau]$ and first compute

$$\Phi_{\mathrm{k}}^{ss}[\tau] = \gamma \Phi_{\mathrm{k}}^{ss}[\tau-1] + (1-\gamma)(\mathbf{s}_{\mathrm{k}}[\tau]\mathbf{s}_{\mathrm{k}}[\tau]^{\mathsf{H}} + \lambda \mathbf{I}_{\mathrm{M}}) \tag{39}$$

where $\Phi_{\mathrm{k}}^{ss}[\tau] \in \mathbb{C}^{M \times M}$ is a time-varying estimate of the target signal spatial covariance matrix, γ is a forgetting factor, and λ is a regularization parameter. We then estimate the steering vector by computing the normalized first principal component of the target source covariance matrix,

$$\tilde{\mathbf{v}}_{\mathrm{k}}[\tau] = \mathcal{P}(\Phi_{\mathrm{k}}^{ss}[\tau]) \tag{40}$$

$$\mathbf{v}_{\mathrm{k}}[\tau] = \tilde{\mathbf{v}}_{\mathrm{k}}[\tau]/\tilde{v}_{\mathrm{k}0}[\tau] \tag{41}$$

where $\mathcal{P}(\cdot)$ extracts the principal component and $\mathbf{v}_{\mathrm{k}}[\tau] \in \mathbb{C}^M$ is the final steering vector. We then use the steering vector to estimate a blocking matrix $\mathbf{B}_{\mathrm{k}}[\tau]$. The blocking matrix is orthogonal to the steering vector and can be constructed as

$$\mathbf{B}_{\mathrm{k}}[\tau] = \begin{bmatrix} -\frac{[v_{\mathrm{k}1}[\tau],\dots,v_{\mathrm{k}M}[\tau]]^{\mathsf{H}}}{v_{\mathrm{k}0}[\tau]^{\mathsf{H}}} \\ \mathbf{I}_{M-1 \times M-1} \end{bmatrix} \in \mathbb{C}^{M \times M-1}. \tag{42}$$

The distortionless constraint is then satisfied by applying the GSC beamformer as

$$\hat{s}_{\mathrm{k}}[\tau] = (\mathbf{v}_{\mathrm{k}}[\tau] - \mathbf{B}_{\mathrm{k}}[\tau]\mathbf{w}_{\mathrm{k}}[\tau])^{\mathsf{H}}\mathbf{u}_{\mathrm{k}}[\tau] \tag{43}$$

where $\mathbf{w}_{\mathrm{k}}[\tau] \in \mathbb{C}^{M-1}$ is the adaptive filter weight, and the desired response for the loss is $d_{\mathrm{k}}[\tau] = \mathbf{v}_{\mathrm{k}}[\tau]^{\mathsf{H}}\mathbf{u}_{\mathrm{k}}[\tau]$.

Our objective is to learn an optimizer g_{ϕ} that minimizes the AF ISE loss using this GSC filter implementation. By doing so, we learn an online, adaptive beamformer that listens in one direction and suppresses interferers from all others.

B. Experimental Design

We compare our Meta-GSC beamformer to LMS, RMSProp, NLMS, and BD-RLS beamformers, in scenes with either diffuse or directional noise sources. We seek to test if our method can learn to process scenes with different spatial characteristics without modification. We train a single g_{ϕ} and tune all baselines on a single dataset of all scene types. We measure performance using scale-invariant source-to-distortion ratio (SI-SDR) [98] and STOI. SI-SDR is computed as,

$$\mathbf{a} = (\hat{\underline{\mathbf{s}}}^{\top} \underline{\mathbf{s}})/\|\underline{\mathbf{s}}\| \tag{44}$$

$$\mathrm{SI\text{-}SDR}(\underline{\mathbf{s}}, \hat{\underline{\mathbf{s}}}) = 10 \cdot \log_{10}(\|\mathbf{a}\underline{\mathbf{s}}\|^2/\|\mathbf{a}\underline{\mathbf{s}} - \hat{\underline{\mathbf{s}}}\|^2), \tag{45}$$

where larger values indicate better performance. STOI is computed between the output and desired speech signal. We also compute the BSS eval metrics, source-to-distortion ratio (SDR), source-to-interference ratio (SIR), and source-to-artifact ratio (SAR) [99]. We train g_{ϕ} via Algorithm 1 on one GPU, which took ≈ 24 h. We use an OLA filter with a Hann window size of $N = 1024$ samples and a hop of $R = 512$ samples on 6 channel 16 kHz audio.

We use the CHIME-3 challenge proposed in [100], [101]. This dataset contains scenes with simulated speech and relatively diffuse noise sources in a multi-channel environment. The array is rectangular and has six microphones spaced around the edge of a smart-tablet. There are 7,138 training files, 1,640 validation files, and 1,320 test files. When running this dataset with directional sources, we mix spatialized speech from one mixture with the spatialized speech from a random other mixture. We do not mix speech across folds.

C. Results and Discussion

We find that Meta-GSC outperforms LMS, RMSProp, NLMS, and BD-RLS in median performance metrics as shown in Figs. 15 and 16 and in a qualitative analysis in Fig. 17.

1) Diffuse Interferers: The diffuse scenario tests the ability to suppress omnidirectional noise in a perceptually pleasant fashion. We show these comparisons in the "Diffuse Interferer" rows
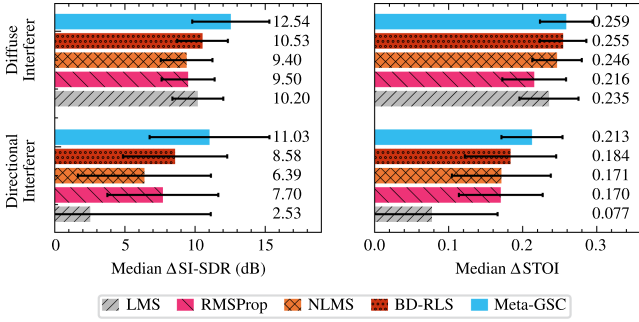
Fig. 15. Performance comparison across interferers. The directional noise is the most challenging and diffuse is the easiest.
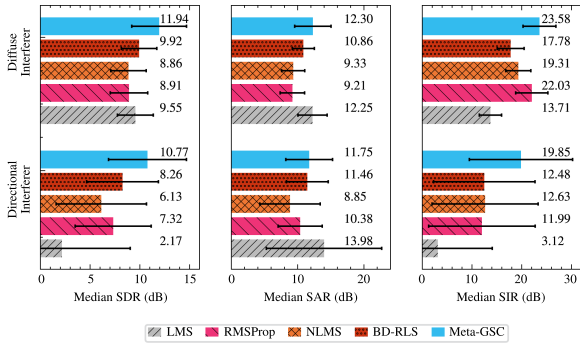

Fig. 16. BSS eval comparison across interferers. Meta-GSC provides more suppression with less distortion and artifacts.
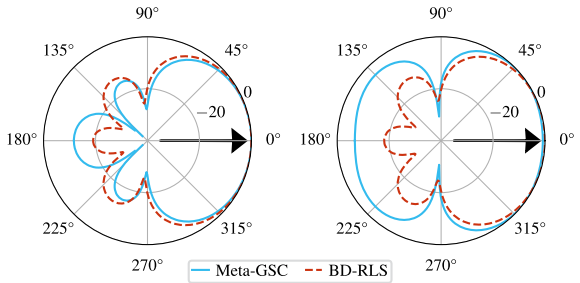

Fig. 17. Spatial response plots at ≈ 1 KHz for a directional interferer at ≈ 1 sec. (left) and ≈ 2 sec. (right).

of Fig. 15 and Fig. 16. The median input STOI was 0.675 and the median input SI-SDR was -0.67. Meta-GSC (blue, solid) outperforms BD-RLS (red, dotted) in SI-SDR performance with Meta-GSC scoring 12.54 dB improvement and BD-RLS scoring 10.53 dB improvement. In STOI, Meta-GSC outperforms BD-RLS by 0.004. The BSS Eval metrics show that Meta-GSC provides 5.8 dB more interferer suppression (SIR) while simultaneously introducing 1.4 dB fewer artifacts (SAR) and 2.02 dB less distortion (SDR) than BD-RLS. Typically, enhancement algorithms trade improved interference suppression for additional artifacts. However, the meta-training scheme produces an optimizer which simultaneously improves both.

2) Directional Interferers: The directional scenario tests the ability to suppress sources from one particular direction –

typically achieved by steering nulls in the beam pattern. We show these comparisons in the "Directional Interferer" rows of Figs. 15 and 16. The median input STOI was 0.734 and the median input SI-SDR was -0.45. Meta-GSC scores 11.03 dB on SI-SDR improvement whereas BD-RLS scores 8.58 dB. STOI performance trends similarly with Meta-GSC outperforming BD-RLS by 0.029. The BSS-Eval metrics show a similar trend with Meta-GSC providing 7.37 dB more SIR while simultaneously introducing $.29$ dB fewer artifacts (SAR) and 2.51 dB less distortion (SDR) than BD-RLS. We hypothesize Meta-GSC steers sharper nulls and learns an automatic VAD-like controller.

3) Beampattern Comparison: We compute beam plots for Meta-GSC and BD-RLS at ≈ 1 sec. and ≈ 2 sec. in a scene with a directional interferer. As expected, the models share the same look direction. However, our Meta-GSC method appears to steer more aggressive nulls as shown in Fig. 17

4) Computational Complexity: Our Meta-GSC method has a single CPU core RTF of ≈ 0.54, and 32 ms latency. The optimizer network alone has $\approx 14\ K$ complex-valued parameters and single CPU core RTF of ≈ 0.25.

IX. DISCUSSION, FUTURE WORK, AND CONCLUSION

A. Discussion

When we review the cumulative results of our approach, we note several interesting observations. First, the performance of our meta-learned AFs is strong and compares favorably to conventional optimizers across all tasks. Second, the performance difference between our meta-learned AFs and conventional AFs is larger for tasks that are traditionally more difficult to model by humans including AEC double-talk, AEC path changes, and directional interference cancellation. Third, we found that we could use a single configuration of our method for all five tasks, which significantly reduced our development time. This suggests that our learned AFs are a viable replacement of human-derived AFs for a variety of audio signal processing tasks and are most valuable for complex AF tasks that typically require more human engineering effort.

When we reflect on how our learned optimizers are able to achieve this success, we note two core reasons. First, and most obvious, our meta-learned AFs are *data-driven* and trained on a particular class of signals (e.g. speech, directional noise, etc). Thus, Meta-AF is limited by the capacity of our optimizer network and training data and not signal modeling skill. Second, by framing AF development as a meta-learning problem, we effectively distill knowledge of our meta loss into the AF loss and corresponding learned update rules, thus enabling us to learn AFs which optimize objectives that would be very difficult (e.g. our frame accumulated) or even impossible to develop manually (e.g. supervised losses, STOI, SI-SDR, etc).

B. Future Work

The field of meta-learning and meta-learned optimizers is young and has a bright future for signal processing applications. Future directions of research include improving

training methods, non-linear optimizee filtering, optimizer architecture, and the optimizer/meta loss. Particularly promising avenues for future work including identifying better optimization losses and better filter representations for Meta-AF style optimization. Overall, we are optimistic that our Meta-AF approach can benefit from both adaptive filtering advances as well as deep learning progress and will be an exciting research topic.

C. Conclusion

We present a general framework called Meta-AF to automatically develop adaptive filter update rules from data using meta-learning. Our proposed approach offers several benefits including the first general-purposes method of learning AF update rules directly from data and a self-supervised training algorithm that does not require any supervised labeled training data. To demonstrate the power of our framework, we test it on all four canonical adaptive filtering architectures and five unique tasks including system identification, acoustic echo cancellation, equalization, dereverberation, and GSC-based beamforming – all using a single configuration trained on different datasets. In all cases, we were able to train high performing AFs, which outperformed conventional optimizers as well as certain state-of-the-art methods. We are excited about the future of deep learning combined with adaptive filters and hope our complete code release will stimulate further research and rapid progress.

REFERENCES

[1] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Hoboken, NJ, USA: Prentice-Hall, 1985.

[2] V. J. Mathews, "Circuits and systems tutorials: Adaptive polynomial filters," *IEEE Signal Process. Mag.*, vol. 8, no. 3, pp. 10–26, Jul. 1991.

[3] J.-S. Soo and K. K. Pang, "Multidelay block frequency domain adaptive filter," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, no. 2, pp. 373–376, Feb. 1990.

[4] S. S. Haykin, *Adaptive Filter Theory*. New York, NY, USA: Pearson, 2008.

[5] J. A. Apolinário, J. A. Apolinário, and R. Rautmann, *QRD-RLS Adaptive Filtering*. Berlin, Germany: Springer, 2009.

[6] L. R. Rabiner, B. Gold, and C. Yuen, *Theory and Application of Digital Signal Processing*. Hoboken, NJ, USA: Prentice-Hall, 2016.

[7] N. N. Schraudolph, "Local gain adaptation in stochastic gradient descent," in *Proc. 9th Int. Conf. Artif. Neural Netw.*, 1999, pp. 569–574.

[8] S. L. Gay, "An efficient, fast converging adaptive filter for network echo cancellation," in *Proc. IEEE Asilomar Conf. Signal Sys. Comp.*, 1998, pp. 394–398.

[9] J. Benesty, T. Gänsler, D. R. Morgan, S. L. Gay, and M. M. Sondhi, *Advances in Network and Acoustic Echo Cancellation*. Berlin, Germany: Springer, 2001.

[10] E. Hänsler and G. Schmidt, *Acoustic Echo and Noise Control: A Practical Approach*. Hobonken, NJ, USA: Wiley, 2005.

[11] G. Enzner and P. Vary, "Frequency-domain adaptive Kalman filter for acoustic echo control in hands-free telephones," *Signal Process.*, vol. 86, no. 6, pp. 1140–1156, 2006.

[12] J.-M. Valin, "On adjusting the learning rate in frequency domain echo cancellation with double-talk," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 3, pp. 1030–1034, Mar. 2007.

[13] S. Malik and G. Enzner, "Model-based vs. traditional frequency-domain adaptive filtering in the presence of continuous double-talk and acoustic echo path variability," in *Proc. IWAENC*, 2008.

[14] S. Malik and G. Enzner, "Online maximum-likelihood learning of time-varying dynamical models in block-frequency-domain," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2010, pp. 3822–3825.

[15] F. Kuech, E. Mabande, and G. Enzner, "State-space architecture of the partitioned-block-based acoustic echo controller," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 1295–1299.

[16] F. Yang, G. Enzner, and J. Yang, "Frequency-domain adaptive Kalman filter with fast recovery of abrupt echo-path changes," *IEEE Signal Process. Lett.*, vol. 24, no. 12, pp. 1778–1782, Dec. 2017.

[17] M. L. Valero and E. A. Habets, "Multi-microphone acoustic echo cancellation using relative echo transfer functions," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2017, pp. 229–233.

[18] T. Haubner, A. Brendel, M. Elminshawi, and W. Kellermann, "Noise-robust adaptation control for supervised acoustic system identification exploiting a noise dictionary," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 945–949.

[19] P. A. Nelson, H. Hamada, and S. J. Elliott, "Adaptive inverse filters for stereophonic sound reproduction," *IEEE Trans. Signal Process.*, vol. 40, no. 7, pp. 1621–1632, Jul. 1992.

[20] P. A. Nelson, F. Orduna-Bustamante, and H. Hamada, "Inverse filter design and equalization zones in multichannel sound reproduction," *IEEE Trans. Signal Process.*, vol. 3, no. 3, pp. 185–192, May 1995.

[21] M. Bouchard and S. Quednau, "Multichannel recursive-least-square algorithms and fast-transversal-filter algorithms for active noise control and sound reproduction systems," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 5, pp. 606–618, Sep. 2000.

[22] N. V. George and G. Panda, "Advances in active noise control: A survey, with emphasis on recent nonlinear techniques," *Elsevier Signal Process.*, vol. 93, no. 2, pp. 363–377, 2013.

[23] L. Lu et al., "A survey on active noise control in the past decade-Part I: Linear systems," *Elsevier Signal Process.*, vol. 183, 2021, Art. no. 108039.

[24] L. Lu et al., "A survey on active noise control in the past decade–Part ii: Nonlinear systems," *Signal Process.*, vol. 181, 2021, Art. no. 107929.

[25] T. Yoshioka, H. Tachibana, T. Nakatani, and M. Miyoshi, "Adaptive dereverberation of speech signals with speaker-position change detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2009, pp. 3733–3736.

[26] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi, and B.-H. Juang, "Speech dereverberation based on variance-normalized delayed linear prediction," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 18, no. 7, pp. 1717–1731, Sep. 2010.

[27] T. Yoshioka and T. Nakatani, "Generalization of multi-channel linear prediction methods for blind MIMO impulse response shortening," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 10, pp. 2707–2720, Dec. 2012.

[28] B. Li et al., "Acoustic modeling for google home," in *Proc. Interspeech*, 2017, pp. 399–403.

[29] J. Caroselli, I. Shafran, A. Narayanan, and R. Rose, "Adaptive multichannel dereverberation for automatic speech recognition," in *Proc. Interspeech*, 2017, pp. 3877–3881.

[30] L. Drude, J. Heymann, C. Boeddeker, and R. Haeb-Umbach, "NARA-WPE: A python package for weighted prediction error dereverberation in Numpy and tensorflow for online and offline processing," in *Proc. ITG-Symp. Speech Commun.*, 2018, pp. 1–5.

[31] J. Wung et al., "Robust multichannel linear prediction for online speech dereverberation using weighted householder least squares lattice adaptive filter," *IEEE Trans. Signal Process.*, vol. 68, pp. 3559–3574, 2020.

[32] L. Griffiths and C. Jim, "An alternative approach to linearly constrained adaptive beamforming," *IEEE Trans. Antennas Propag.*, vol. 30, no. 1, pp. 27–34, Jan. 1982.

[33] O. Hoshuyama, A. Sugiyama, and A. Hirano, "A robust adaptive beamformer for microphone arrays with a blocking matrix using constrained adaptive filters," *IEEE Trans. Signal Process.*, vol. 47, no. 10, pp. 2677–2684, Oct. 1999.

[34] J. Chen, J. Benesty, and Y. Huang, "A minimum distortion noise reduction algorithm with multiple microphones," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 3, pp. 481–493, Mar. 2008.

[35] M. Souden, J. Benesty, and S. Affes, "On optimal frequency-domain multichannel linear filtering for noise reduction," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 18, no. 2, pp. 260–276, Feb. 2010.

[36] Y. A. Huang and J. Benesty, "A multi-frame approach to the frequency-domain single-channel noise reduction problem," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 4, pp. 1256–1269, May 2012.

[37] S. Markovich-Golan, S. Gannot, and I. Cohen, "A sparse blocking matrix for multiple constraints GSC beamformer," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2012, pp. 197–200.

[38] S. Gannot, E. Vincent, S. Markovich-Golan, and A. Ozerov, "A consolidated perspective on multimicrophone speech enhancement and source separation," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 4, pp. 692–730, Apr. 2017.

[39] A. N. Birkett and R. A. Goubran, "Acoustic echo cancellation using NLMS-neural network structures," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1995, pp. 3035–3038.

[40] A. B. Rabaa and R. Tourki, "Acoustic echo cancellation based on a recurrent neural network and a fast affine projection algorithm," in *Proc. IEEE 24th Annu. Conf. Ind. Electron. Soc.*, 1998, pp. 1754–1757.

[41] J. Malek and Z. Koldovský, "Hammerstein model-based nonlinear echo cancelation using a cascade of neural network and adaptive linear filter," in *Proc. IEEE Int. Workshop Acoustic Signal Enhancement*, 2016, pp. 1–5.

[42] S. Zhang and W. X. Zheng, "Recursive adaptive sparse exponential functional link neural network for nonlinear AEC in impulsive noise environment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4314–4323, Sep. 2018.

[43] M. M. Halimeh, C. Huemmer, and W. Kellermann, "A neural network-based nonlinear acoustic echo canceller," *IEEE Signal Process. Lett.*, vol. 26, no. 12, pp. 1827–1831, Dec. 2019.

[44] H. Zhang, K. Tan, and D. Wang, "Deep learning for joint acoustic echo and noise cancellation with nonlinear distortions," in *Proc. Interspeech*, 2019, pp. 4255–4259.

[45] L. Ma, H. Huang, P. Zhao, and T. Su, "Acoustic echo cancellation by combining adaptive digital filter and recurrent neural network," 2020, *arXiv:2005.09237*.

[46] A. Ivry, I. Cohen, and B. Berdugo, "Nonlinear acoustic echo cancellation with deep learning," in *Proc. Interspeech*, 2021, pp. 4773–4777.

[47] J.-M. Valin, S. Tenneti, K. Helwani, U. Isik, and A. Krishnaswamy, "Low-complexity, real-time joint neural echo control and speech enhancement based on PercepNet," in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process.*, 2021, pp. 7133–7137.

[48] Y.-L. Zhou, Q.-Z. Zhang, X.-D. Li, and W.-S. Gan, "Analysis and DSP implementation of an ANC system using a filtered-error neural network," *J. Sound Vib.*, vol. 285, no. 1–2, pp. 1–25, 2005.

[49] T. Krukowicz, "Active noise control algorithm based on a neural network and nonlinear input-output system identification model," *Arch. Acoust.*, vol. 35, no. 2, pp. 191–202, 2010.

[50] H. Zhang and D. Wang, "A deep learning approach to active noise control," in *Proc. Interspeech*, 2020, pp. 1141–1145.

[51] H. Zhang and D. Wang, "Deep ANC: A deep learning approach to active noise control," *Neural Netw.*, vol. 141, pp. 1–10, 2021.

[52] K. Kinoshita, M. Delcroix, H. Kwon, T. Mori, and T. Nakatani, "Neural network-based spectrum estimation for online WPE dereverberation," in *Proc. Interspeech*, 2017, pp. 384–388.

[53] J. Heymann, L. Drude, R. Haeb-Umbach, K. Kinoshita, and T. Nakatani, "Frame-online DNN-WPE dereverberation," in *Proc. IEEE 16th Int. Workshop Acoustic Signal Enhancement*, 2018, pp. 466–470.

[54] L. Drude et al., "Integrating neural network based beamforming and weighted prediction error dereverberation," in *Proc. Interspeech*, 2018, pp. 3043–3047.

[55] Z.-Q. Wang, G. Wichern, and J. L. Roux, "Convolutive prediction for reverberant speech separation," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2021, pp. 56–60.

[56] Z.-Q. Wang, G. Wichern, and J. L. Roux, "Convolutive prediction for monaural speech dereverberation and noisy-reverberant speaker separation," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 3476–3490, 2021.

[57] H. Erdogan, J. R. Hershey, S. Watanabe, M. I. Mandel, and J. L. Roux, "Improved MVDR beamforming using single-channel mask prediction networks," in *Proc. Interspeech*, 2016, pp. 1981–1985.

[58] J. Heymann, L. Drude, and R. Haeb-Umbach, "Neural network based spectral mask estimation for acoustic beamforming," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2016, pp. 196–200.

[59] K. Qian, Y. Zhang, S. Chang, X. Yang, D. Florencio, and M. Hasegawa-Johnson, "Deep learning based speech beamforming," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 5389–5393.

[60] Z. Zhang, Y. Xu, M. Yu, S.-X. Zhang, L. Chen, and D. Yu, "ADL-MVDR: All deep learning MVDR beamformer for target speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 6089–6093.

[61] T. Haubner, A. Brendel, and W. Kellermann, "End-to-end deep learning-based adaptation control for frequency-domain adaptive system identification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2022, pp. 766–770.

[62] H. Zhang, S. Kandadai, H. Rao, M. Kim, T. Pruthi, and T. Kristjansson, "Deep adaptive AEC: Hybrid of deep learning and adaptive acoustic echo cancellation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2022, pp. 756–760.

[63] J. Casebeer, N. J. Bryan, and P. Smaragdis, "Auto-DSP: Learning to optimize acoustic echo cancellers," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2021, pp. 291–295.

[64] J. Casebeer, J. Donley, D. Wong, B. Xu, and A. Kumar, "NICE-beam: Neural integrated covariance estimators for time-varying beamformers," 2021, *arXiv:2112.04613*.

[65] R. Scheibler and M. Togami, "Surrogate source model learning for determined source separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 176–180.

[66] R. S. Sutton, "Adapting bias by gradient descent: An incremental version of delta-bar-delta," in *Proc. AAAI*, 1992, pp. 171–176.

[67] A. R. Mahmood, R. S. Sutton, T. Degris, and P. M. Pilarski, "Tuning-free step-size adaptation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2012, pp. 2121–2124.

[68] J. Schmidhuber, "Learning to control fast-weight memories: An alternative to dynamic recurrent networks," *Neural Comput.*, vol. 4, pp. 131–139, 1992.

[69] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, "Neural optimizer search with reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 459–468.

[70] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," in *Proc. ICLR*, 2017.

[71] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[72] M. Andrychowicz et al., "Learning to learn by gradient descent by gradient descent," in *Proc. NeurIPS*, 2016, pp. 3988–3996.

[73] O. Wichrowska et al., "Learned optimizers that scale and generalize," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3751–3760.

[74] L. Metz, N. Maheswaranathan, J. Nixon, D. Freeman, and J. Sohl-Dickstein, "Understanding and correcting pathologies in the training of learned optimizers," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 4556–4565.

[75] T. Chen et al., "Training stronger baselines for learning to optimize," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7332–7343.

[76] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Hoboken, NJ, USA: Prentice-Hall, 1975.

[77] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," Cited on, 2012.

[78] S. T. Alexander and A. L. Ghimikar, "A method for recursive least squares filtering based upon an inverse QR decomposition," *IEEE Trans. Signal Process.*, vol. 41, no. 1, pp. 20–30, Jan. 1993.

[79] P. Strobach, "Low-rank adaptive filters," *IEEE Trans. Signal Process.*, vol. 44, no. 12, pp. 2932–2947, Dec. 1996.

[80] F. Yang, G. Enzner, and J. Yang, "On the convergence behavior of partitioned-block frequency-domain adaptive filters," *IEEE Trans. Signal Process.*, vol. 69, pp. 4906–4920, 2021.

[81] Z. Wang, Y. Na, Z. Liu, B. Tian, and Q. Fu, "Weighted recursive least square filter and neural network based residual ECHO suppression for the AEC-challenge," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 141–145.

[82] WebRTC acoustic Echo Cancellation v3, 2019. Accessed: Sep. 10, 2022. [Online]. Available: https://webrtc.googlesource.com/src

[83] Acoustic Echo Cancellation Challenge–ICASSP 2021, 2021. Accessed: Sep. 10, 2022. [Online]. Available: https://www.microsoft.com/en-us/research/academic-program/acoustic-echo-cancellation-challenge-icassp-2021/results

[84] V. Manohar, S.-J. Chen, Z. Wang, Y. Fujita, S. Watanabe, and S. Khudanpur, "Acoustic modeling for overlapping speech recognition: JHU CHiME-5 challenge system," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 6665–6669.

[85] J. L. Roux, N. Ono, and S. Sagayama, "Explicit consistency constraints for STFT spectrograms and their application to phase reconstruction," in *Proc. Interspeech*, 2008, pp. 23–28.

[86] J. Bradbury et al., "JAX: Composable transformations of Python NumPy programs, 2018," 2020. [Online]. Available: http://github.com/google/jax

[87] T. Hennigan, T. Cai, T. Norman, and I. Babuschkin, "Haiku: Sonnet for JAX," 2020. [Online]. Available: http://github.com/deepmind/dm-haiku

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

[89] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2014.

[90] M. Wolter and A. Yao, "Complex gated recurrent neural networks," in *Proc. NeurIPS*, 2018, pp. 10557–10567.

[91] K. Sridhar et al., "ICASSP 2021 acoustic echo cancellation challenge: Datasets, testing framework, and results," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 151–155.

[92] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 5220–5224.

[93] G. Enzner, H. Buchner, A. Favrot, and F. Kuech, "Acoustic echo control," in *Academic Press Library in Signal Processing*, vol. 4. Amsterdam, The Netherlands: Elsevier, 2014.

[94] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 7, pp. 2125–2136, Sep. 2011.

[95] G. J. Mysore, "Can we automatically transform speech recorded on common consumer devices in real-world environments into professional production quality speech?–A dataset, insights, and challenges," *IEEE Signal Process. Lett.*, vol. 22, no. 8, pp. 1006–1010, Aug. 2015.

[96] K. Kinoshita et al., "A summary of the REVERB challenge: State-of-the-art and remaining challenges in reverberant speech processing research," *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 1, 2016, Art. no. 7.

[97] A. Jukić, T. V. Waterschoot, and S. Doclo, "Adaptive speech dereverberation using constrained sparse multichannel linear prediction," *IEEE Signal Process. Lett.*, vol. 24, no. 1, pp. 101–105, Jan. 2017.

[98] J. L. Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR–half-baked or well done?" in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 626–630.

[99] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 4, pp. 1462–1469, Jul. 2006.

[100] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third CHiME speech separation and recognition challenge: Dataset, task and baselines," in *Proc. IEEE Workshop Autom. Speech Recognit. Understanding*, 2015, pp. 504–511.

[101] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third CHiME speech separation and recognition challenge: Analysis and outcomes," *Comput. Speech Lang.*, Nov. 46, pp. 605–626, 2017.