Gradient Clipping Improves Neural Network Optimization for Perceptual Sound Matching

Han Han
Nantes Université, École Centrale Nantes
CNRS, LS2N, UMR 6004
Nantes, France
han.han@ls2n.fr

Vincent Lostanlen
Nantes Université, École Centrale Nantes
CNRS, LS2N, UMR 6004
Nantes, France
vincent.lostanlen@ls2n.fr

Mathieu Lagrange
Nantes Université, École Centrale Nantes
CNRS, LS2N, UMR 6004
Nantes, France
mathieu.lagrange@ls2n.fr

***Abstract*—Perceptual Sound Matching (PSM) learns the optimal synthesizer input to replicate a target sound perceptually. To achieve so, it adopts learning objectives reflective of auditory perceptual distance to derive perceptually informed gradients via automatic differentiation. Yet, learning objectives of PSM are often ill-conditioned, since not all synthesizer and perceptual representation are invertible. To address this challenge, state of the art methods adopt multi-stage training to foster convergence, rendering the training objective non-stationary. In this paper, we show that autoregressive optimization methods like Adam is unsuited to readily reflect the discrepancy in gradient conditions caused by nonstationary objectives, as well as updating weights informed by large gradients from ill-conditioned objectives. We demonstrate empirically how, with a simple formulation of weight decay and gradient clipping, one can optimize PSM with more probable convergence and better generalization. We provide possible reasoning by comparing evolutions of summarized gradient norm and gradient roughness under different optimization setups.**

***Index Terms*—Perceptual Sound Matching, Optimizer, Stochastic Gradient Descent**

I. Introduction

Given a parametric synthesizer $\boldsymbol{g}$, perceptual sound matching (PSM) aims to find a vector $\tilde{\boldsymbol{\theta}}$ such that $\boldsymbol{g}(\tilde{\boldsymbol{\theta}})$ matches some target sound $\boldsymbol{x}$. Early work on sound matching often employ genetic algorithms and spectral matching to optimize for $\boldsymbol{\theta}$. For instance, Yeeking et al developed a non-gradient-based system that iteratively updates parameter combination to approximate the Mel-Frequency Cepstral Coefficients (MFCC) of a given target sound [1]. For a neural network $\boldsymbol{f}_{\boldsymbol{w}}$ with weights $\boldsymbol{w}$, this task may be formulated as multidimensional regression via training set synthesis over a finite set $\boldsymbol{\Theta}$. With $\boldsymbol{\theta}$ and $\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{\theta})$ as dependent and independent variables, we define a "generalized nonlinear least squares" [2] objective of the form:

$$\mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{w}) = \big\langle (\boldsymbol{f}_{\boldsymbol{w}} \circ \boldsymbol{g})(\boldsymbol{\theta}) - \boldsymbol{\theta} \,\big|\, \mathrm{M}(\boldsymbol{\theta}) \,\big|\, (\boldsymbol{f}_{\boldsymbol{w}} \circ \boldsymbol{g})(\boldsymbol{\theta}) - \boldsymbol{\theta} \big\rangle, \qquad (1)$$

where $\tilde{\boldsymbol{\theta}} = (\boldsymbol{f}_{\boldsymbol{w}} \circ \boldsymbol{g})(\boldsymbol{\theta})$ and the bracket notation $\langle \boldsymbol{u} | \mathrm{M} | \boldsymbol{u} \rangle = \boldsymbol{u}^{\top} \mathbf{M} \boldsymbol{u}$ is a quadratic form in $\boldsymbol{u}$. The symmetric matrix $\mathrm{M}(\boldsymbol{\theta})$ contains domain knowledge about the perceptual significance of each linear direction in parameter space in the vicinity of $\boldsymbol{\theta}$.

The earliest methods for PSM relied on the ad hoc assumption $\mathrm{M}(\boldsymbol{\theta}) = \mathbf{I}$ for every $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, hence a nonlinear least squares objective $\mathcal{L}^{\mathrm{P}}_{\boldsymbol{\theta}}(\boldsymbol{w}) = \|(\boldsymbol{f}_{\boldsymbol{w}} \circ \boldsymbol{g})(\boldsymbol{\theta}) - \boldsymbol{\theta}\|_2^2$ which we will later denote as parameter loss or *P-loss* for short [3]. A more recent method, known as *perceptual–neural–physical (PNP)* [4], has shown the value of incorporating off-diagonal coefficients in $\mathrm{M}(\boldsymbol{\theta})$. PNP harnesses automatic differentiation of $\boldsymbol{g}$ and of a perceptual similarity function to compute an adapted matrix $\mathrm{M}(\boldsymbol{\theta})$ for each $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. The rationale behind PNP is that if $\boldsymbol{w}$ is such that $\mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{w})$ is small, then its gradient $\nabla \mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{w})$ approximates the gradient of perceptual loss in differentiable digital signal processing (DDSP) [5] while remaining almost as computationally efficient as the gradient of P-loss.

The main drawback of PNP is that it may yield a rank-deficient matrix $\mathrm{M}(\boldsymbol{\theta})$. This can be observed in practice by diagonalizing $\mathrm{M}(\boldsymbol{\theta})$ and computing its condition number, i.e., the ratio of its largest to its smallest eigenvalue. A high condition number, also known as ill-conditioning, causes numerical inaccuracies: depending on the position of $\tilde{\boldsymbol{\theta}}$ with respect to $\boldsymbol{\theta}$, the vector $(\mathrm{M}(\boldsymbol{\theta}) \cdot (\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}))$ may be near-zero. As a consequence of the chain rule, the PNP gradient $\nabla \mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{w})$ is also near-zero even though the perceptual loss associated to the reconstructed sound $\boldsymbol{g}(\tilde{\boldsymbol{\theta}}) = (\boldsymbol{g} \circ \boldsymbol{f}_{\boldsymbol{w}} \circ \boldsymbol{g})(\boldsymbol{\theta})$ may be significantly greater than zero.

Against this drawback, [4] have proposed a modified objective:

$$\mathcal{L}_{\boldsymbol{\theta},\lambda}(\boldsymbol{w}) = \big\langle (\boldsymbol{f}_{\boldsymbol{w}} \circ \boldsymbol{g})(\boldsymbol{\theta}) - \boldsymbol{\theta} \,\big|\, \mathrm{M}(\boldsymbol{\theta}) + \lambda \mathbf{I} \,\big|\, (\boldsymbol{f}_{\boldsymbol{w}} \circ \boldsymbol{g})(\boldsymbol{\theta}) - \boldsymbol{\theta} \big\rangle, \quad (2)$$

where $\lambda > 0$ is a hyperparameter. Intuitively, the additive term $\lambda \mathbf{I}$ shifts all eigenvalues of the the matrix $\mathrm{M}(\boldsymbol{\theta})$ by a constant positive offset λ, thus lowering the condition number of $(\mathrm{M}(\boldsymbol{\theta}) + \lambda \mathbf{I})$ in comparison with $\mathrm{M}(\boldsymbol{\theta})$. This approach resembles the Levenberg-Marquardt algorithm (LMA) [1] except that we seek to minimize $\mathcal{L}_{\boldsymbol{\theta},\lambda}$ with respect to neural network weights $\boldsymbol{w}$ and not simply retrieve the vector $\boldsymbol{\theta}$ from a randomly initalized parameter vector $\tilde{\boldsymbol{\theta}}$.

Yet, the introduction of Levenberg-Marquardt damping in Equation 2 brings, in turn, new drawbacks. The value of λ is adjusted dynamically depending on the spectra of $\mathrm{M}(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$ in the training set and depending on the performance of the model $\boldsymbol{f}_{\boldsymbol{w}}$ on the objectives $\mathcal{L}_{\boldsymbol{\theta}}$. This is problematic for an optimizer such as Adam, which remains the standard choice of deep learning until today.

We experiment with a PSM problem in which Adam fails to train the model $\boldsymbol{f}_{\boldsymbol{w}}$ on the PNP objective, even after Levenberg-Marquardt damping. We build upon previous work [6], that took inspiration from recent literature on large language models (LLM) [7], and adopted an alternative optimizer based on gradient clipping and weight decay (GCWD) that was found indispensable to achieving convergence in this problem setting. In this paper, we evaluate more systematically Adam and GCWD's optimization robustness across different model sizes and random initializations. We find that:

- the improvement of GCWD is consistent across physical and perceptual metrics and across model scales from 4M to 64M parameters;
- GCWD is a better optimizer than Adam whenever the objective varies greatly across iterations, as is the case with PNP due to the parameter λ.

Those evidences are further supported by considering metrics reflecting optimization conditions such as summarized gradient norm

[1] We refer to the NeurIPS 2017 Test-of-time award presentation by Ali Rahimi for an introduction to Levenberg-Marquardt damping and the challenges of training deep neural networks on ill-conditioned objectives.

and roughness. We conclude with tentative explanations regarding the improved rate of convergence and generalization ability of GCWD.

II. Method

A. Multi-scale spectrogram vs. joint time–frequency scattering

Denoting a multidimensional audio descriptor by $\mathbf{\Phi}$ and the target sound by $\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{\theta})$, spectral loss is defined as $\|(\mathbf{\Phi} \circ \boldsymbol{g})(\tilde{\boldsymbol{\theta}}) - \mathbf{\Phi}(\boldsymbol{x})\|_2^2$.

The multi-scale spectrogram (MSS) is based on a combination of short-term Fourier transforms (STFT) at various window lengths. If the target sound $\boldsymbol{x}$ and $\boldsymbol{g}(\tilde{\boldsymbol{\theta}})$ are sustained harmonic tones with the same fundamental frequency (f_0), MSS is appropriate as spectral loss for DDSP [5]. However, MSS falls short whenever f_0 is unknown [8] or on nonstationary inharmonic sounds such as percussion [9].

Joint time–frequency scattering (JTFS) alternates discrete wavelet transform and pointwise complex modulus to extract spectrotemporal modulations at various scales and rates in the time–frequency domain [10]. JTFS is a mathematical idealization of spectrotemporal receptive fields in the primary auditory cortex [11]. After automatic differentiation [12] thanks to the Kymatio software library [13], a recent publication has shown the potential of replacing MSS by JTFS in DDSP as soon as $\boldsymbol{g}$ produces nonstationary sounds [9].

However, the scalability of DDSP with JTFS is hampered by its computational cost: the forward pass $\mathbf{\Phi}$ through a 3-second sample waveform takes around 3 seconds, while reverse-mode automatic differentiation ($\nabla\mathbf{\Phi}$) takes around one minute.

B. Perceptual–neural–physical loss

Following [4], we perform a first-order Taylor expansion of the operator $(\mathbf{\Phi} \circ \boldsymbol{g})$ around the JTFS coefficients of the target sound $\mathbf{\Phi}(\boldsymbol{x})$:

$$(\mathbf{\Phi} \circ \boldsymbol{g})(\tilde{\boldsymbol{\theta}}) = \mathbf{\Phi}(\boldsymbol{x}) + \mathbf{J}_{(\mathbf{\Phi}\circ\boldsymbol{g})}(\boldsymbol{\theta}) \cdot (\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}) + O(\|\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}\|_2^2),$$

where $\mathbf{J}_{(\mathbf{\Phi}\circ\boldsymbol{g})}$ is the Jacobian of $(\mathbf{\Phi} \circ \boldsymbol{g})$. The associated Riemannian metric yields a square matrix $\mathbf{M}(\boldsymbol{\theta}) = \mathbf{J}_{(\mathbf{\Phi}\circ\boldsymbol{g})}(\boldsymbol{\theta})^\top \mathbf{J}_{(\mathbf{\Phi}\circ\boldsymbol{g})}(\boldsymbol{\theta})$ which we use to approximate spectral loss by a quadratic form:

$$\begin{aligned}\big\|(\mathbf{\Phi} \circ \boldsymbol{g})(\tilde{\boldsymbol{\theta}}) - \mathbf{\Phi}(\boldsymbol{x})\big\|_2^2 &= \big\langle \tilde{\boldsymbol{\theta}} - \boldsymbol{\theta} \big| \mathbf{M}(\boldsymbol{\theta}) \big| \tilde{\boldsymbol{\theta}} - \boldsymbol{\theta} \big\rangle + O\big(\|\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}\|_2^3\big) \\ &= \mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{w}) + O\big(\|(\boldsymbol{f_w} \circ \boldsymbol{g})(\boldsymbol{\theta}) - \boldsymbol{\theta}\|_2^3\big), \quad (3)\end{aligned}$$

where $\tilde{\boldsymbol{\theta}} = (\boldsymbol{f_w} \circ \boldsymbol{g})(\boldsymbol{\theta})$. Crucially, the matrices $\mathbf{M}(\boldsymbol{\theta})$ are constant in $\boldsymbol{w}$ for all $\boldsymbol{\theta}$. Thus, we may precompute them asynchronously in parallel. Furthermore, each of them contains only J^2 coefficients, where J is the dimension of the vector $\boldsymbol{\theta}$. Typically, $2 \leq J \leq 100$; in our application, $J = 5$. Thus, the memory footprint of all matrices $\mathbf{M}(\boldsymbol{\theta})$ is small enough to allow storage on disk before training.

Denoting by $\boldsymbol{w}[i]$ the i^{th} coordinate of the weight vector $\boldsymbol{w}$ of the model $\boldsymbol{f_w}$, the partial derivative of $\mathcal{L}_{\boldsymbol{\theta}}$ with respect to $\boldsymbol{w}[i]$ is

$$\nabla\mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{w})[i] = 2\Big\langle \boldsymbol{f_w}(\boldsymbol{x}) - \boldsymbol{\theta} \Big| \mathbf{M}(\boldsymbol{\theta}) \Big| \frac{\partial \boldsymbol{f_w}}{\partial \boldsymbol{w}[i]}(\boldsymbol{x}) \Big\rangle. \quad (4)$$

The gradient vector of the neural network output with respect to each weight (in orange) is projected onto the eigenvectors of $\mathbf{M}(\boldsymbol{\theta})$ and scaled by the corresponding eigenvalues. The final gradient scalar informing each weight update is the dot product between the scaled gradient (in red) and the error vector (in green). Consequently, a nonzero gradient vector risks being excessively stretched or decompressed, leading to extreme magnitudes that severely distorts the original parameter error vector. Despite its extremity, this perceptual scaling indeed remains accurate locally when parameter error is sufficiently close to zero. However, this does not hold when $\boldsymbol{f_w}$ is randomly initialized at the start of optimization. Non-negligible parameter error combined with $\mathbf{M}(\boldsymbol{\theta})$ can lead to incorrect scaling. This is further supported by the fact that the higher-order error terms in Equation 3 are only negligible if parameter error is small enough.

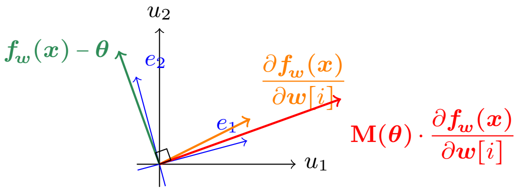
u_2
$\boldsymbol{f_w}(\boldsymbol{x}) - \boldsymbol{\theta}$
e_2
$\dfrac{\partial \boldsymbol{f_w}(\boldsymbol{x})}{\partial \boldsymbol{w}[i]}$
e_1
$\mathbf{M}(\boldsymbol{\theta}) \cdot \dfrac{\partial \boldsymbol{f_w}(\boldsymbol{x})}{\partial \boldsymbol{w}[i]}$
u_1

Fig. 1. Vectors showing an extreme case where a nonzero gradient vector yields zero gradient when combined with ill-conditioned kernel matrix $\mathbf{M}(\boldsymbol{\theta})$. Orange: gradient vector of neural network output with respect to a given weight. Blue: two eigenvectors of $\mathbf{M}(\boldsymbol{\theta})$ with e_1 corresponding to a big eigenvalue and e_2 corresponding to a small eigenvalue. Red: gradient vector after multiplied with $\mathbf{M}(\boldsymbol{\theta})$. Green: parameter error vector. In this scenario, the final gradient with respect to the weight is 0 as the red and the green vectors are perpendicular to each other, despite that neither the parameter error vector nor the gradient vector was zero.

By inspiration from the Levenberg-Marquardt algorithm, we add a corrective term $\lambda\mathbf{I}$ to shift all eigenvalues of $\mathbf{M}(\boldsymbol{\theta})$ by a constant, thereby improving the condition number of the quadratic form:

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\theta},\lambda}^{\text{PNP}}(\boldsymbol{w}) &= \big\langle \tilde{\boldsymbol{\theta}} - \boldsymbol{\theta} \big| \mathbf{M}(\boldsymbol{\theta}) + \lambda\mathbf{I} \big| \tilde{\boldsymbol{\theta}} - \boldsymbol{\theta} \big\rangle \\ &= \big\langle \tilde{\boldsymbol{\theta}} - \boldsymbol{\theta} \big| \mathbf{M}(\boldsymbol{\theta}) \big| \tilde{\boldsymbol{\theta}} - \boldsymbol{\theta} \big\rangle + \lambda\|\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}\|_2^2. \quad (5)\end{aligned}$$

The formula above can be interpreted as a linear combination between linearized perceptual loss and P-loss, with λ as multiplicative factor.

C. Adam optimizer

We generalize the PNP loss (Equation 5) to a minibatch of B samples:

$$\mathcal{L}_{\mathbf{\Theta},\lambda}(\boldsymbol{w}) = \frac{1}{B}\sum_{\boldsymbol{\theta}\in\mathbf{\Theta}} \big\langle (\boldsymbol{f_w} \circ \boldsymbol{g})(\boldsymbol{\theta}) - \boldsymbol{\theta} \big| \mathbf{M}(\boldsymbol{\theta}) + \lambda\mathbf{I} \,\big|\, (\boldsymbol{f_w} \circ \boldsymbol{g})(\boldsymbol{\theta}) - \boldsymbol{\theta} \big\rangle. \quad (6)$$

Let us denote by $\mathbf{\Theta}_t$ and λ_t the minibatch and damping value at each iteration $t \geq 0$ training the neural network $\boldsymbol{f_w}$. The Adam optimizer [14] performs an exponential moving average (EMA) of the gradient vector $\mathcal{L}_{\mathbf{\Theta}_t,\lambda_t}(\boldsymbol{w}_t)$ and its coordinate-wise square:

$$\boldsymbol{m}_t[i] = \beta_1 \boldsymbol{m}_{t-1}[i] + (1 - \beta_1)\nabla\mathcal{L}_{\mathbf{\Theta}_t,\lambda_t}(\boldsymbol{w}_t)[i], \quad (7)$$

$$\boldsymbol{v}_t[i] = \beta_2 \boldsymbol{v}_{t-1}[i] + (1 - \beta_2)\nabla\mathcal{L}_{\mathbf{\Theta}_t,\lambda_t}(\boldsymbol{w}_t)[i]^2, \quad (8)$$

where $\beta_1, \beta_2 \in [0, 1)$ are hyperparameters. Assuming that the gradients $\nabla\mathcal{L}_{\mathbf{\Theta}_t,\lambda_t}(\boldsymbol{w}_t)$, $t \geq 0$ are i.i.d. samples from a random variable, $\boldsymbol{m}_t$ and $\boldsymbol{v}_t$ estimate its first- and second-order moments. However, because $\boldsymbol{m}_0$ and $\boldsymbol{v}_0$ are initialized at zero, these estimators are biased. Adam debiases them multiplicatively: $\hat{\boldsymbol{m}}_t = \boldsymbol{m}_t/(1 - \beta_1^t)$, $\hat{\boldsymbol{v}}_t = \boldsymbol{v}_t/(1 - \beta_2^t)$. Given a sequence of learning rates $(\alpha_t)_t$ and a small constant ε, the Adam weight update rule is:

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \alpha_t \frac{\hat{\boldsymbol{m}}_t}{\varepsilon + \sqrt{\hat{\boldsymbol{v}}_t}}. \quad (9)$$

Intuitively, the learning rate α_t sets an upper bound on the step size per coordinate, while the EMA in Adam assigns comparatively smaller step sizes to weight coordinates i for which the gradient tends to have higher variance and/or more frequent sign fluctuations.

D. Limitations of Adam for PNP

For any $\boldsymbol{w}$ and $\mathbf{\Theta}$, the PNP gradient $\nabla\mathcal{L}_{\mathbf{\Theta},\lambda}(\boldsymbol{w})$ is an affine function of λ. Therefore, the second-order moment estimate $\boldsymbol{v}_t$ grows in proportion to λ^2. For large values of λ, $\hat{\boldsymbol{v}}_t$ is exposed to a risk of overflow. This risk is amplified for coordinates i such that the partial derivative of $\boldsymbol{f_w}$ with respect to $\boldsymbol{w}[i]$ at $\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{\theta})$ has a large magnitude and is collinear with $(\boldsymbol{f_w}(\boldsymbol{x}) - \boldsymbol{\theta})$ on average over $\theta \in \boldsymbol{\Theta}$.

At the initialization of neural network training ($t = 0$), we set λ_0 equal to the maximum principal eigenvalue of $\mathbf{M}(\boldsymbol{\theta})$ over all training examples $\boldsymbol{\theta}$ from the training set. In this way, the condition number of $(\mathbf{M}(\boldsymbol{\theta}) + \lambda\mathbf{I})$ is guaranteed to range between 1 and 2 for every $\boldsymbol{\theta}$.

On the PSM problem which we will present in the next section, we observe a maximum principal eigenvalue of $\lambda_0 \sim 10^{41}$, leading to $\nabla\mathcal{L}_{\boldsymbol{\Theta}_0,\lambda_0}(\boldsymbol{w}_0)[i] \sim 10^{27}$ and ultimately $\boldsymbol{v}_0[i] \sim 10^{54}$. Yet, single-precision floating-point arithmetic, the one commonly used in GPU computing, has an overflow level of the order of 10^{38}. In practice, we have observed that gradient squaring in Adam does cause overflow for a non-negligible proportion of weights $\boldsymbol{w}_t[i]$, thus canceling the term $\alpha_t\hat{\boldsymbol{m}}_t$ in Equation 9.

At first glance, this issue could be easily circumvented by rescaling gradients $\nabla\mathcal{L}_{\boldsymbol{\Theta_t},\lambda_t}(\boldsymbol{w}_t)$ by $c = 1/\lambda_0 \sim 10^{-41}$. However, this provokes a second issue, namely, a risk of underflow in EMA. Underflow arises for coordinates i such that $\nabla\mathcal{L}_{\boldsymbol{\Theta_t},\lambda_t}(\boldsymbol{w}_t)$ is small; i.e., in proportion with the smallest eigenvalue of $\boldsymbol{w}(\boldsymbol{\theta})$. For these coordinates, $\hat{\boldsymbol{v}}_t \ll \varepsilon$ and thus Adam essentially falls back to a kind of stochastic gradient with momentum β_1 and learning rate $(\alpha_t c/\varepsilon)$.

The above observations suggest that under the premise of leaving the large gradients as they are, it is best to omit the inverse second moment estimates and seek alternative ways to stabilize step size magnitudes.

E. Gradient clipping as an alternative to Adam

Gradient clipping mitigates arbitrarily large gradients by thresholding the update step size with hyperparameter ρ. In extreme cases with large gradients $\nabla\mathcal{L}_{\boldsymbol{\theta},\lambda}(\boldsymbol{w})$, it becomes equivalent to SignGD, where only the sign of gradients are utilized to update weights at a fixed step size $\alpha_t\rho$. Additionally, we adopt weight decay with damping hyperparameter $\gamma \in [0, 1)$ as it has been used in conjunction with gradient clipping in [7].

$$\boldsymbol{w}_t = (1 - \alpha_t\gamma)\boldsymbol{w}_{t-1} - \alpha_t \cdot \min\big(\max(\boldsymbol{m}_{t-1}/\varepsilon, \rho), -\rho\big) \qquad (10)$$

Weight decay originates from Tikhonov (ℓ^2) regularization in ordinary least squares regression. By constraining weight magnitudes, it has the potential to reduce overfitting [15].

F. Adaptive Levenberg-Marquardt damping

By inspiration from the Levenberg-Marquardt Algorithm (LMA), we initialize the corrective term λ_t as the largest empirical eigenvalue of $\mathbf{M}(\boldsymbol{\theta})$, around $\lambda_0 \sim 10^{41}$ in practice. Then, we reduce it by a factor of $r = 0.02$ when epoch validation loss decreases. Denoting the number of steps per epoch by T and the validation set by $\boldsymbol{\Theta}_{\text{val}}$, we set $\lambda_t = r\lambda_{t-1}$ if the step t is divisible by T and if

$$\mathcal{L}_{\boldsymbol{\Theta}_{\text{val}},\lambda_{t-1}}(\boldsymbol{w}_{t-1}) < \mathcal{L}_{\boldsymbol{\Theta}_{\text{val}},\lambda_{t-T-1}}(\boldsymbol{w}_{t-T-1}), \qquad (11)$$

and $\lambda_t = \lambda_{t-1}$ otherwise.

This design of λ evolution allows for easier convergence at the beginning and automatically increases proximity to the ill-conditioned perceptual loss in a self-regulating manner. Yet, this poses problem in EMA. As λ is damped by an accelerator driven by model weights in the previous step, gradient magnitudes undergo the same reduction. Commonly adopted EMA hyperparameters β_1, β_2 favor historical gradients and would not reflect the drastic drop in magnitude until many steps later.

III. Experimental Validation

A. Experiments

We conduct PSM experiments to compare the optimization behaviors of Adam and GCWD. All optimizers are used in conjunction with adaptive learning rate decay monitored by the validation loss. EfficientNets [16] are series of neural net architectures designed to balance scalings of the depth, width and input resolution of the consecutive convolutional blocks. It has achieved state of the art performances on image and audio classification tasks. Prior work on PSM has only been evaluated on EfficientNet-b0, the most lightweight version. We take it further in this work to perform PSM on three increasingly large EfficientNet's: b0, b4 and b7, corresponding to 4, 17 and 63.8 million weights. All models are trained five times against PNP loss in Equation 5.

As in [6], we adopt functional transformation method (FTM) drum physical model as synthesizer and L2 JTFS distance [12] as perceptual distance measure. The dataset comprises 100k synthetic drum sounds from a rectangular drum physical model solved via the functional transformation method. The five synthesis parameters are pitch, duration, roundness, inharmonicity and squareness. They are uniformly sampled from their respective ranges. We refer to [4] for more details.

We train each model with one of the two optimizers for 70 epochs with a batch size of 256 samples. We compare the convergence trend and generalization ability through the training curves and test losses.

B. Evaluation Metrics

We adopt mean squared error of the predicted parameter, and multiscale spectrogram loss (MSS) [5] as evaluation metrics. The former reflects closeness in parameter space whereas the latter calculates a summarized spectral difference localized in differing time-frequency resolution.

Additionally, we monitor gradient norm and gradient roughness during training. Gradient norm $\|\nabla\mathcal{L}_{\boldsymbol{\Theta}_t,\lambda_t}(\boldsymbol{w}_t)\|_2$ measures the mean squared error of gradients with respect to all weights in a given batch of B samples.

A huge gradient norm indicates extreme sensitivity of the loss objective to small changes in model weights. This could result from either a loss function of large magnitude, an ill-conditioned loss function, or else large weights.

We choose gradient roughness (originally termed smoothness) proposed in [17] to measure fluctuations of the optimization landscape. A slow-varying, smooth loss landscape implies small changes in gradient vectors even when one step of gradient descent results in largely disparate weights. Conversely, a spiky landscape with lots of local minimas likely has conflicting gradients even when the weights have barely changed after one update. Following the implementation in [17], we express gradient roughness s_t as

$$s_t = 2\frac{\left\|\nabla\mathcal{L}_{\boldsymbol{\Theta}_t,\lambda_t}\left(\frac{1}{2}\boldsymbol{w}_t + \frac{1}{2}\boldsymbol{w}_{t+1}\right) - \nabla\mathcal{L}_{\boldsymbol{\Theta}_t,\lambda_t}\left(\boldsymbol{w}_t\right)\right\|_2}{\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|_2}. \qquad (12)$$

A small s_t implies smooth optimization condition and vice versa.

C. Discussion

As PNP loss is a nonstationary objective with decreasing magnitude whenever epoch validation loss breaks the lowest record, drastic drops in training loss are present at the start of certain epochs. As shown in Fig. 2, models trained with GCWD (in blue) exceed their Adam counterparts (in red) by a large margin across all model sizes.

All models under Adam optimization quickly plateau and do not meet adaptive PNP loss damping criteria for the remaining training

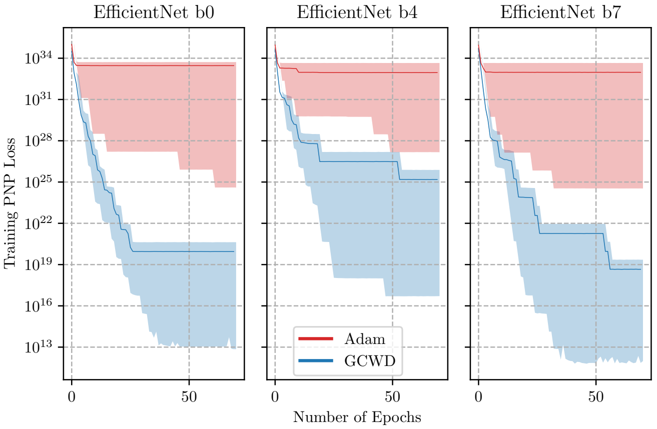

Fig. 2. Training curves of PNP loss using Adam versus GCWD. Each model is run for 5 trials. The shaded areas cover the training loss across trials whereas the solid line outlines the mean.

epochs. This is largely caused by the near-zero step sizes from scaling with inverse second moment gradient estimates. On the contrary, GCWD stably decays the λ throughout the seventy epochs with stochastically clipped nonzero step sizes. Reflected also through test losses in Table I, only the models trained with GCWD are able to converge and generalize to unseen test sounds. We observe that model sizes do not have much impact on the convergence with neither optimizer. This evidence firmly rules out the possibility of Adam overfitting smaller models and confirms that Adam fails to efficiently update the weights.

	Model	P-loss ↓	MSS ↓
Adam	b0	0.36 ± 0.0010	2.2 ± 0.008
	b4	0.36 ± 0.0020	2.0 ± 0.020
	b7	0.36 ± 0.0008	2.1 ± 0.008
GCWD	b0	0.021 ± 0.005	0.86 ± 0.10
	b4	0.020 ± 0.002	0.88 ± 0.03
	b7	0.022 ± 0.006	0.88 ± 0.10

TABLE I
MEAN AND STANDARD DEVIATION OF P-LOSS AND MSS METRICS EVALUATED AT THE FINAL EPOCH OF EACH MODEL.

To examine the optimization condition with each optimizer, we train 100 steps an EfficientNet-b0 network with PNP loss to investigate how objective change impacts gradient norm and roughness. Simulating training with a damped PNP loss, we initialize $\lambda = 10^{41}$ and manually reduce the objective by a factor of 5 at iteration 50. The gradient norm and roughness are evaluated based on sum of 10% of the total number of weights, which according to [17] closely approximates the entire weight distribution. We expect a drastic drop in gradient norm during the objective change, due to the positive correlation between gradient norm and loss magnitude.

As observed in Fig. 3, both optimizers has overall decreasing gradient norm and an extra dip at epoch 50. However at the point of objective transition, the roughness condition has worsened for Adam model and improved for GCWD. This suggests that in the consecutive updates after objective change, gradient fluctuations becomes more apparent for Adam optimizer than for GCWD. This could be explained by two postulations. First, the unregulated inverse second moment estimates in Adam step size calculation reduces the weight update to a small number, resulting in a large roughness coefficient. Second, the unregulated EMA mechanism delays reflecting the most up to date gradient conditions, thereby yielding bigger step sizes that noisify the optimization condition.

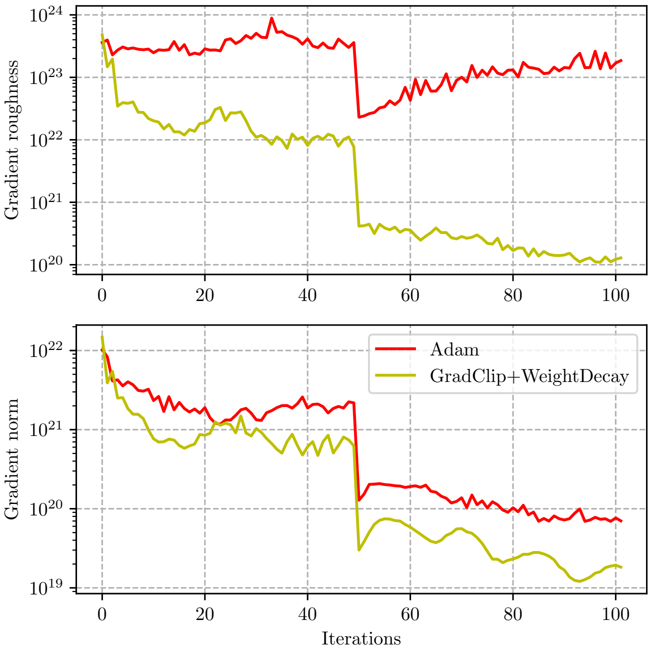

Fig. 3. Evolution of gradient norm and gradient smoothness of PNP loss. Gradient roughness evaluates consistency of gradients with respect to weights from adjacent updates. The smaller the better the optimization condition. During objective change, GCWD results in smoother optimization condition whereas Adam encounters less smooth gradient fluctuations. See Section III-C for more details.

IV. Conclusion

In this work, we presented the properties of PNP loss for perceptual sound matching, and highlighted how its loss formulation is exposed to numerical instability and nonstationarity. We demonstrated Adam's limitations in addressing such issues due to second moment estimates exceeding numerical precision bounds and EMA estimates failing to adapt to rapid changes in gradient conditions. To overcome these challenges, we proposed GCWD, which combines stochastically clipped step sizes and weight decay to stabilize training and achieve convergence that is otherwise impossible with Adam. Results reported in this paper considered a single task, namely sound matching and a single synthesizer, but ill-conditioned objective is pervasive across a wide range of learning tasks. We believe the insights presented in this work can be advantageous in tackling other challenging optimization conditions where nonsmoothness and numerical issues are faced. Future work shall verify GCWD's effectiveness in optimizing ill-conditioned objectives in other applications. Additionally, conducting experiments with isolated techniques, such as using only gradient clipping/weight decay, or removing Adam's second moment estimate, could provide deeper insights into the dynamics and comparative performance of these optimizers.

References

[1] Matthew John Yee-King and Martin S. Roth, "Synthbot: an unsupervised software synthesizer programmer," in *International Conference on Mathematics and Computing*, 2009.

[2] Albert Tarantola and Bernard Valette, "Generalized nonlinear inverse problems solved using the least squares criterion," *Reviews of Geophysics*, vol. 20, no. 2, pp. 219–232, 1982.

[3] Han Han and Vincent Lostanlen, “wav2shape: Hearing the Shape of a Drum Machine,” in *Proceedings of Forum Acusticum*, 2020.

[4] Han Han, Vincent Lostanlen, and Mathieu Lagrange, “Perceptual–Neural–Physical Sound Matching,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, June 2023.

[5] Jesse Engel, Lamtharn (Hanoi) Hantrakul, Chenjie Gu, and Adam Roberts, “DDSP: Differentiable Digital Signal Processing,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[6] Han Han, Vincent Lostanlen, and Mathieu Lagrange, “Learning to Solve Inverse Problems for Perceptual Sound Matching,” *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 32, pp. 2605–2615, Apr. 2024.

[7] Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma, “Sophia: A scalable stochastic second-order optimizer for language model pre-training,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[8] Joseph Turian and Max Henry, “I’m sorry for your loss: Spectrally-based audio distances are bad at pitch,” *Proceedings of the “I Can’t Believe It’s Not Better” Workshop (ICBINB)*, 2020.

[9] Cyrus Vahidi, Han Han, Changhong Wang, Mathieu Lagrange, György Fazekas, and Vincent Lostanlen, “Mesostructures: Beyond spectrogram loss in differentiable time-frequency analysis,” *arXiv preprint arXiv:2301.10183*, 2023.

[10] Joakim Andén, Vincent Lostanlen, and Stéphane Mallat, “Joint time–frequency scattering,” *IEEE Transactions on Signal Processing*, vol. PP, May 2019.

[11] Etienne Thoret, Baptiste Caramiaux, Philippe Depalle, and Stephen Mcadams, “Learning metrics on spectrotemporal modulations reveals the perception of musical instrument timbre,” *Nature Human Behaviour*, vol. 5, Mar 2021.

[12] John Muradeli, Cyrus Vahidi, Changhong Wang, Han Han, Vincent Lostanlen, Mathieu Lagrange, and George Fazekas, “Differentiable time-frequency scattering in kymatio,” in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2022.

[13] Mathieu Andreux, Tomás Angles, Georgios Exarchakis, Roberto Leonarduzzi, Gaspar Rochette, Louis Thiry, John Zarka, Stéphane Mallat, Joakim Andén, Eugene Belilovsky, Joan Bruna, Vincent Lostanlen, Muawiz Chaudhary, Matthew J. Hirn, Edouard Oyallon, Sixin Zhang, Carmine Cella, and Michael Eickenberg, “Kymatio: Scattering transforms in Python.,” *Journal of Machine Learning Research*, vol. 21, no. 60, pp. 1–6, Jan 2020.

[14] Diederik P. Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[15] Anders Krogh and John Hertz, “A simple weight decay can improve generalization,” in *Advances in Neural Information Processing Systems*, J. Moody, S. Hanson, and R.P. Lippmann, Eds. 1991, vol. 4, Morgan-Kaufmann.

[16] Mingxing Tan and Quoc Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in *Proceedings of the International conference on Machine Learning (ICML)*, 2019.

[17] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie, “Why gradient clipping accelerates training: A theoretical justification for adaptivity,” 2020.