FAST BLACK-BOX OPTIMIZERS FOR LOW DELAY AUDIO SOURCE SEPARATION

Gerald Schuller

Ilmenau University of Technology,
Institute for Media Technology,
Ilmenau, Germany

ABSTRACT

The goal of this paper is to show that black box optimizers (also referred to as derivative free optimizers) recently made significant progress, such that they can also be used for signal processing applications, even for cases where only a limited time budged is available, as in online and real time applications, as shown in the presented audio source separation example. Another advantage is that they can be used, e.g. for recurrent neural networks, which have the problem of vanishing gradients. These applications are used to compare a set of black box optimizers, all also suitable for higher dimensional problems. The results show that a suitable selection and adaptation of an optimizer for an application is crucial. For the presented applications, the presented optimizer of "Random Directions" consistently performs among the best for finding a good minimum of the objective- or loss-functions, and also for the shortest processing time.

Index Terms— black box optimization, real time audio processing, vanishing gradients, blind audio source separation

1. INTRODUCTION

The goal of this paper is to draw attention to the fact that black box optimizers recently made significant progress , such that they can also be used for signal processing applications, even for cases where only a limited time budged is available, as in online and real time applications, as shown in the presented audio source separation example.

Arguably from the beginning, digital signal processing is connected with optimization. In the beginning it was the design of filters with desired properties for their passbands and stopbands, for instance with the remez algorithm or window design methods. Later it was for adaptive filters, using the LMS (Least Mean Squares) and RLS (Recursive Least Squares) algorithms. These are mostly algorithms based on gradient descent, which works best on convex objective or loss functions.

Then came filter banks with perfect reconstruction and possibly critical sampling. They either need the (near) perfect reconstruction property embodied in the optimization constraints (which then leads to only near perfect reconstruction) [1, 2, 3], or lead to very nonconvex objective or loss functions, which are difficult to optimize [4, 3].

Multichannel blind source separation is another example with possibly very non-convex objective of loss functions. The approach here is usually to solve it in the STFT domain, which makes it easier to optimize. The signals there are more narrow band and in parallel, which makes each subband signal change more slowly, hence having possible local minima appear at a lower rate. But this also adds significant delay to the processing [5]. Processing in the time domain avoids most of the delay, but also has an objective function in which local minima appear at a higher rate, which makes it more difficult to optimize. Possible applications for this low delay processing would be teleconferencing, hearing aids, or musicans playing together remotely. Particularly the latter two need delays of only a few ms. Possibly one of the first application of probabilistic of black box optimization to real time blind source separation was shown in [6, 7].

For Recurrent Neural Networks (RNN), possible long term dependencies make the problem of vanishing gradients appear, which makes the numerical computation of the gradient vector too imprecise for gradient descent methods. LSTM (Long Short-Term Memory) networks where designed with this problem in mind, but they have a more complex structure.

For neural networks, black-box optimizers may enable structures which before were too difficult to optimize, or enable training, learning, or adaptation on embedded devices, where an autograd function for gradient-based optimizers would be too complex to implement (e.g. from PyTorch or Tensorflow). For instance, on embedded devices, this could enable adaptation of speech recognition on device, to a speaker or environment.

Further applications are non-differentiable loss functions, like in adversarial machine learning [8, 9], reinforcement learning [10], e.g. in embedded reinforcement learning for small robots, or in learning audio effects [11].

These different application areas and their used optimizers also show that it is probably worthwhile to choose and adapt optimizers for the respective application area.

2. BLACK BOX OPTIMIZATION METHODS

Black-Box optimization, also referred to as derivative-free optimization, treats the objective or loss function to optimize (usually minimize) as a "black box", where only the function value for a given input is available, but no derivatives are necessary [12]. Their general principle is to replace a gradient update with a random search across a carefully designed probability distribution over a neighborhood of the argument of the objective function.

Since there are significant differences in the performance of different methods for different applications, the best way to find the best optimization algorithm for an application is to compare different optimizers. "pypop7" [13] (https://pypi.org/project/pypop7/) is a helpful Python module that combines many black-box optimizers with a common interface for easy comparability. Further examples are DEAP (Distributed Evolutionary Algorithms in Python) [14], and PySOT (Surrogate Optimization Toolbox) [15].

An important property of different black-box optimizers is their suitability for high dimensions, for instance, for optimizing large neural networks. In the following, only methods suitable for high dimensions were employed.

There are different classes of black-box optimizers. A broad class is Natural Evolution Strategies (NES), [16], which is inspired by natural evolution, operating on a parameterized distribution of candidate solutions, usually means and variances of Gaussian distributions of the search.

2.1. The Method of Random Directions

This is the proposed method, which was originally designed for the optimization of low delay filter banks [17] and multichannel blind audio source separation [7]. Both have significant numbers of local minima; low delay filter banks from their nonlinear structure, and the audio source separation from modeling the delays of the relative room impulse response between microphones [7, 18]. Instead of gradient vectors, it uses normal distributed search vectors with a slowly shrinking standard deviation, similar to the method of simulated annealing. Introducing random subspaces on the updates proves beneficial for higher dimensional objective functions. Further introducing a line search of successful directions as direction estimates significantly helps, particularly with loss functions with more smooth regions. It keeps the original argument of the objective function if it is still the best, and in this way it guarantees that the optimization does not lead to a worse result. Experience shows that this can be an important property. This also made it perform well on larger, difficult to optimize neural networks, like Recurrent Neural Networks. Its pseudo code can be seen in algorthm (1).

```
Input: $f(c)$: $\mathbb{R}^n \rightarrow \mathbb{R}$: Objective function to be
       minimized
$c_0$: Starting point coefficient vector,
T: Number of iterations,
P: Number of parallel processes,
startingscale: Standard deviation at the start
endscale: Standard deviation at the end
Initialization: $c_{best} = c_0$
for $m$=0,...,$T$ do
    $scale = endscale + (startingscale - endscale) \cdot ((1.0 - m/iterations)^2)$
    Parallel Processing:
        generate P search vectors $v_r$ with std
        deviation "scale" and zero mean on random
        subsets of coefficients;
        compute $f(c_{best} + v_r)$
    find best update vector:
      $v_{best} =_{v_r} \{f(c_{best} + v) \mid v = v_r\}$
    if $f(c_{best} + v_{best}) < f(c_{best})$ then
        Find new $c_{best}$ with coarse line search along
          successful vector $v_{best}$,
        $k_{best} =_k f(c_{best} + 2^k v_{best})$,
          $k = (-8, \ldots, 8)$ (Line search)
        $c_{best} = c_{best} + 2^{k_{best}} v_{best}$
    end
end
return $c_{best}$
```
Algorithm 1: The method of Random Directions.

3. COMPARED BLACK BOX OPTIMIZERS

The compared optimizers are representatives of different types of black box optimizers. The following methods were chosen: Mixture Model-based Evolution Strategy (MMES) [19] combines different models to form a (Gaussian) mixture model, which is then used to guide the search for optimal solutions. Rank-One Evolution Strategy (R1ES) [20] simplifies the update of the search covariance matrix to a rank-one update, making it computationally more efficient. Limited-Memory Matrix Adaptation Evolution Strategy (LMMAES) [21] focuses on optimizing the storage and computational costs associated with matrix adaptation. Separable Natural Evolution Strategy (SNES) [22] is an adaptation of NES that assumes parameter separability, allowing for more efficient updates and lower computational cost. BErnoulli Smoothing (BES) [23] focuses on generalizing optimization strategies by introducing Bernoulli-distributed noise into the process. Gradientless Descent (GLD) [24] uses a slowly shrinking search diameter (variance of a Gaussian distribution) and a coarse line search as a direction estimate. In this way it is the most similar to the method of Random Directions, except that it does not keep the original point if it is still the best, and does not use random subspace updates.

As comparison to advanced gradient-based methods, the method of Conjugate Gradients [25], and Adam [26] are chosen.

Their hyperparameters all seem to be not very sensitive in this context, so they are left at their default. Usually longer processing times or higher number of iterations lead to better results, so they are chosen, where possible, such that different optimizers roughly had the same processing times.

4. APPLICATION EXPERIMENTS

The applications were chosen such that they are difficult or impossible to optimize with traditional gradient-based optimizers. Software, models, and results are in our Github repository, for reproducible research [27].

4.1. Multichannel blind audio source separation

The time domain source separation method used is described in [18], and consists of an unmixing matrix that contains attenuation coefficients and fractional sample delays implemented with allpass filters, modeling the relative room impulse response (RRIR). The optimizer must find the attenuation coefficients and fractional sample delays. The objective function used is the negative Kullback-Leibler Divergence (KLD) between the output channels obtained to maximize the statistical independence between them. Observe that the KLD is used as objective function, because it is used in the real time processing application. It does not need the original source signal, unlike SIR, which is not available during the application. Similar for the audio quality, which is often measured as SDR and SAR (linear and non-linear distortions). It is assumed here that it is dominated by the unmixing structure and the separation. Optimization of the coefficients can be done in parallel to the audio processing, in a window over past samples, starting with the last solution as the initial point. It could be run a few times to then select the best solution. The system could also be implemented with a more precise model of the RRIR for improved separation. Here, the function to minimize is called the "objective function".

4.2. Recurrent neural network

The general application could be a generative network that learns a given sound. The target signal here is a decaying sinousoid, the impulse response of a 2nd order Infinite Impulse Response (IIR) filter at Fs=8000 sampling rate, with frequency of 440 Hz, quality factor Q=0.999 and length of 0.6 seconds, with difference eq. (1).

$$y[n] = x[n] + k[1] \cdot y[n-1] + k[2] \cdot y[n-2] \qquad (1)$$

$$\omega = 2\pi \cdot 440.0/Fs; k[1] = 2Q \cdot \cos(\omega); k[2] = -Q^2$$

$Q = 0.999$ is the quality factor (1 decays forever, see also [27]). The Pytorch Recurrent Neural Network (RNN) to emulate this signal is very simple. It has 1 input feature (1 audio channel with the impulse), 1 layer with 3 memory elements or hidden states, and 1 output. The loss function for training is the mean squared error. In this machine learning context, the function to minimize is called the "loss function".

5. RESULTS

The hardware used for the experiments consisted of a processor with 8 CPU cores, each running at 1.8 GHz. To ensure the reliability of the results, each optimization was executed 10 times, and the outcomes were visualized using box plots.

5.1. Source Separation

To compare different optimizers, the method of successive elimination is used: exclude optmizers that fail for the first critical test signal, then, if there are some with similar performance, test them in the next round with a different critical signal. This minimizes the number of comparisons.

For the first round source separation tests, a sound mix of speech and noise was generated using the module "pyroomacoustics" with an RT60 reverberation time of 0.1 seconds. The simulation setup and the Random Directions optimizer can be found in the GitHub repository [28].

Two simulated scenarios were examined: the first involved a "stereo" setup with two microphones placed 20 cm apart, and the second used eight microphones arranged in a cube formation with 20 cm side length. Informal listening

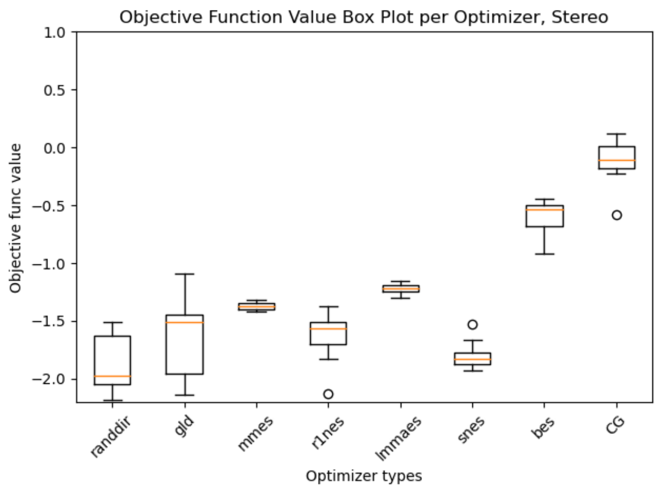

Fig. 1. The achieved minimum objective value for different optimizers as boxplot (with mean and quartiles), for the stereo microphone case. Lower is better. Observe that the method of random directions gets the best results, and CG the worst.

tests indicated that good separation was achieved when the values of the objective function were below approximately -1.8.

Fig. 1 for stereo reveals that useful separation was only achieved by the Random Directions and GLD (Gradientless Descent) optimizers. In contrast, the gradient-based method CG (Conjugate Gradient) performed poorly, resulting in no separation. Fig. 2 shows that both GLD and CG had the highest processing times, indicating that Random Directions offers a combination of good separation and efficient processing time.

Fig. 3 for the cube microphone case demonstrates that only Random Directions achieved separation (below -1.8), in the lower quartile. This suggests that running Random Directions four times is likely to yield a good solution. Fig. 4 indicates that most optimizers had similar processing times of around 20 seconds, except for GLD and CG, which required approximately 30 and over 100 seconds, respectively. Thus, Random Directions was the only effective optimizer in the cube microphone setup.

5.2. Recurrent Neural Network

In the context of Recurrent Neural Networks (RNN), Fig. 5 shows that Random Directions and MMES both achieved the best loss function values. Fig. 6 further reveals that Random Directions had a significantly shorter processing time compared to other methods. These results confirm that Random Directions is not only effective in finding the best loss function value, but is also the most efficient in terms of processing time.

6. CONCLUSIONS

Black box optimization can successfully be applied to (real time) audio signal processing, like time domain blind audio

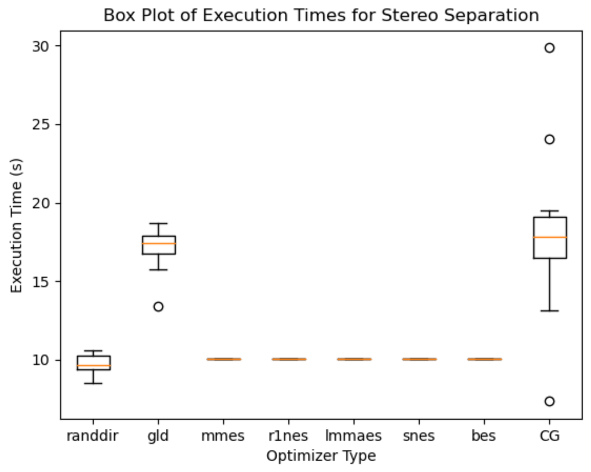

Fig. 2. The processing times for different optimizers as boxplot (with mean and quartiles), for the stereo microphone case. Lower is better. Observe the higher times for CG.

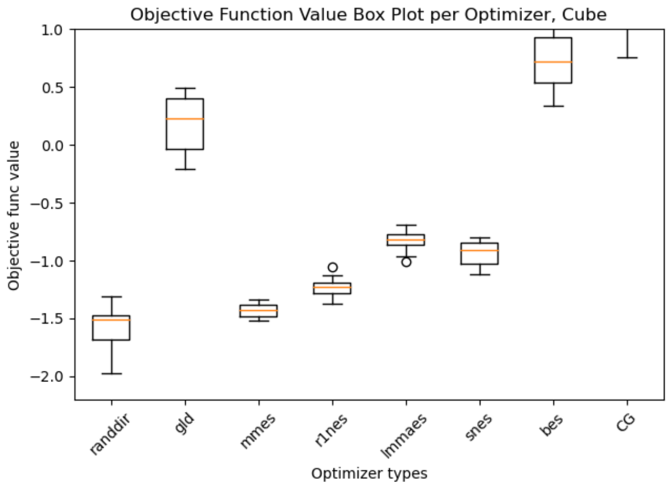

Fig. 3. The achieved minimum objective value for different optimizers as boxplot (with mean and quartiles), for the cube microphone case. Lower is better. Observe that the method of random directions gets the best results, and CG the worst.

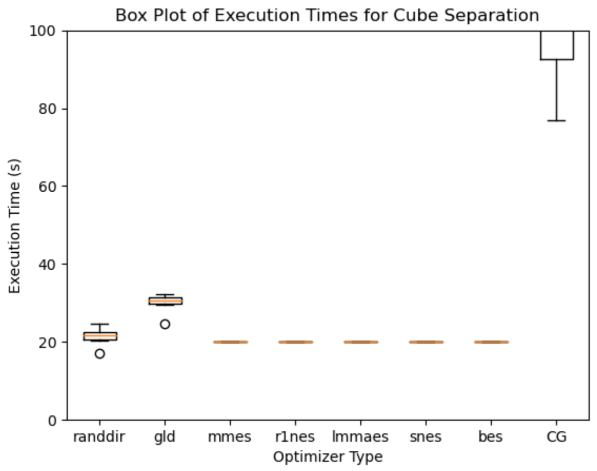

Fig. 4. The processing times for different optimizers as boxplot (with mean and quartiles), for the cube microphone case. Lower is better. Observe the higher times for CG.

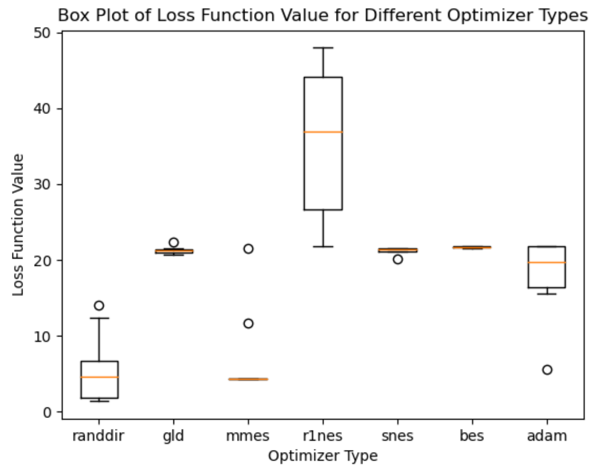

Fig. 5. The achieved minimum objective value for different optimizers as boxplot (with mean and quartiles), for the RNN. Lower is better. Observe that the method of random directions is significantly better than Adam and R1NES, and comparable to MMES.

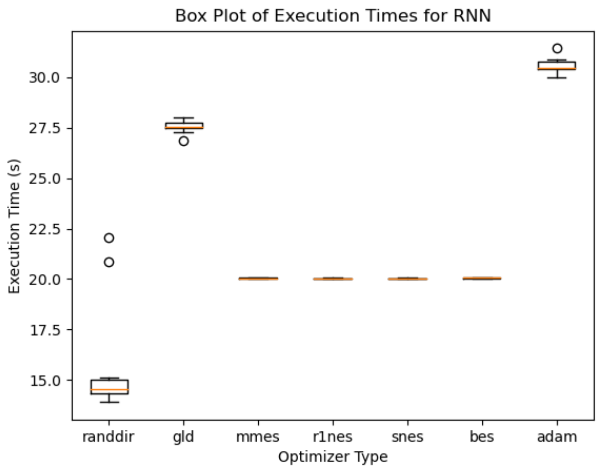

Fig. 6. The processing times for different optimizers as boxplot (with mean and quartiles), for the RNN. Lower is better. Observe the higher times for Adam and GLD.

source separation and optimizing a Recurrent Neural Network with a long impulse response, provided a suitable optimizer is used. In the presented application examples, the Random Directions optimizer was among the best for finding a minimum of the objective or loss function and also among the fastest.

7. REFERENCES

[1] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall, Aug. 2023.

[2] T.Q. Nguyen, "Near-perfect-reconstruction pseudo-QMF banks," *IEEE Transactions on Signal Processing*, vol. 42, no. 1, pp. 65–76, Jan. 1994, Conference Name: IEEE Transactions on Signal Processing.

[3] Gerald Schuller, *Filter Banks and Audio Coding: Compressing Audio Signals Using Python*, Springer International Publishing, Cham, 2020.

[4] G.D.T. Schuller and M.J.T. Smith, "New framework for modulated perfect reconstruction filter banks," *IEEE Transactions on Signal Processing*, vol. 44, no. 8, pp. 1941–1954, Aug. 1996, Conference Name: IEEE Transactions on Signal Processing.

[5] Yiteng Huang, Jacob Benesty, and Jingdong Chen, *Acoustic MIMO Signal Processing*, Springer, 2006 edition, Nov. 2006.

[6] Oleg Golokolenko and Gerald Schuller, "A fast stereo audio source separation for moving sources," in *Asilomar Conference on Signals, Systems, and Computers*, Asilomar, CA, USA, Nov 3-6 2019.

[7] Gerald Schuller and Oleg Golokolenko, "Probabilistic Optimization for Source Separation," in *2020 54th Asilomar Conference on Signals, Systems, and Computers*, Nov. 2020, pp. 534–538, ISSN: 2576-2303.

[8] Hyunjun Mun, Sunggwan Seo, Baehoon Son, and Joobeom Yun, "Black-box audio adversarial attack using particle swarm optimization," *IEEE Access*, vol. 10, pp. 23532–23544, 2022.

[9] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa, "Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding," Oct. 2018, arXiv:1808.05665 [cs, eess].

[10] Krzysztof Choromanski, Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, Deepali Jain, Yuxiang Yang, Atil Iscen, Jasmine Hsu, and Vikas Sindhwani, "Provably Robust Blackbox Optimization for Reinforcement Learning," July 2019, arXiv:1903.02993 [cs, stat].

[11] Marco Comunità, Christian J. Steinmetz, Huy Phan, and Joshua D. Reiss, "Modelling black-box audio effects with time-varying feature modulation," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2023, pp. 1–5, arXiv:2211.00497 [cs, eess].

[12] Charles Audet and Michael Kokkolaras, "Blackbox and derivative-free optimization: theory, algorithms and applications," *Optimization and Engineering*, vol. 17, no. 1, pp. 1–2, Mar. 2016.

[13] Qiqi Duan, Guochen Zhou, Chang Shao, Zhuowei Wang, Mingyang Feng, Yijun Yang, Qi Zhao, and Yuhui Shi, "PyPop7: A Pure-Python Library for Population-Based Black-Box Optimization," Mar. 2023, arXiv:2212.05652 [cs].

[14] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné, "DEAP: Evolutionary Algorithms Made Easy," *Journal of Machine Learning Research*, vol. 13, no. 70, pp. 2171–2175, 2012.

[15] David Eriksson, David Bindel, and Christine A. Shoemaker, "pySOT and POAP: An event-driven asynchronous framework for surrogate optimization," July 2019, arXiv:1908.00420 [cs, math, stat].

[16] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber, "Natural Evolution Strategies," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, Hong Kong, China, June 2008, pp. 3381–3387, IEEE.

[17] Gerald Schuller and Tanja Karp, "Modulated filter banks with arbitrary system delay: Efficient implementations and the time-varying case," *IEEE Transactions on Signal Processing*, vol. 48, no. 3, pp. 737–748, March 2000.

[18] Gerald Schuller, "Ultra Low Delay Audio Source Separation Using Zeroth-Order Optimization," in *2023 IEEE Statistical Signal Processing Workshop (SSP)*, July 2023, pp. 497–501, ISSN: 2693-3551.

[19] Xiaoyu He, Zibin Zheng, and Yuren Zhou, "MMES: Mixture Model-Based Evolution Strategy for Large-Scale Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 2, pp. 320–333, Apr. 2021, Conference Name: IEEE Transactions on Evolutionary Computation.

[20] Zhenhua Li and Qingfu Zhang, "A Simple Yet Efficient Evolution Strategy for Large-Scale Black-Box Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 637–646, Oct. 2018, Conference Name: IEEE Transactions on Evolutionary Computation.

[21] Ilya Loshchilov, Tobias Glasmachers, and Hans-Georg Beyer, "Large Scale Black-Box Optimization by Limited-Memory Matrix Adaptation," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 353–358, Apr. 2019, Conference Name: IEEE Transactions on Evolutionary Computation.

[22] Tom Schaul, Tobias Glasmachers, and Jürgen Schmidhuber, "High dimensions and heavy tails for natural evolution strategies," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, New York, NY, USA, July 2011, GECCO '11, pp. 845–852, Association for Computing Machinery.

[23] Katelyn Gao and Ozan Sener, "Generalizing Gaussian Smoothing for Random Search," in *Proceedings of the 39th International Conference on Machine Learning*. June 2022, pp. 7077–7101, PMLR, ISSN: 2640-3498.

[24] Daniel Golovin, Greg Kochanski John Karro, Chansoo Lee, Xingyou Song, and Qiuyi Zhang, "Gradientless descent: High-dimensional zeroth-order optimization," May 18 2020.

[25] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, Cambridge University Press, Cambridge, 3rd edition edition, Sept. 2007.

[26] Diederik P. Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017, arXiv:1412.6980 [cs].

[27] "TUIlmenauAMS/BlackBoxOptimizerSPcomparison: https://github.com/TUIlmenauAMS/BlackBoxOptimizer SPcomparison," .

[28] "TUIlmenauAMS Random Directions Demo: https://github.com/TUIlmenauAMS/LowDelayMultichannel SourceSeparation_Random-Directions_Demo," .