Audio Personalization through Human-in-the-loop Optimization

Abstract

We consider the problem of personalizing audio to maximize user experience. Briefly, we aim to find a filter h^*, which applied to any music or speech, will maximize the user's satisfaction. This is a black-box optimization problem since the user's satisfaction function is unknown. The key idea is to play audio samples to the user, each shaped by a different filter h_i, and query the user for their satisfaction scores $f(h_i)$. A family of "surrogate" functions is then designed to fit these scores and the optimization method gradually refines these functions to arrive at the filter $\hat{h}^*$ that maximizes satisfaction.

In this paper, we observe that a second type of querying is possible where users can tell us the individual elements $h^*[j]$ of the optimal filter h^*. Given a budget of B queries, where a query can be of either type, our goal is to find the filter that will maximize this user's satisfaction.

Our proposal builds on Sparse Gaussian Process Regression (GPR) and shows how a hybrid approach can outperform any one type of querying. Our results are validated through simulations and real world experiments, where volunteers gave feedback on music/speech audio and were able to achieve high satisfaction levels. We believe this idea of hybrid querying opens new problems in black-box optimization, and solutions can benefit other applications beyond audio personalization.

1 Introduction

Consider the problem of personalizing content to a user's taste. Content could be audio signals in a hearing aid, a salad cooked for the user, etc. Given the content c, we intend to adjust it with a linear filter h. Our goal is to find the optimal filter h^* that will maximize the user's personal satisfaction $f(h)$. Finding h^* is difficult because the function $f(h)$ is unknown; it is embedded somewhere inside the perceptual regions of the brain. Black box optimization (BBO) has been proposed for such settings, where one queries user-satisfaction scores for carefully sampled filters h_i. Using a budget of B such queries, BBO estimates $\hat{h}^*$ that is close to the true h^*.

The above problem can be called "*filter querying*" because the user is queried using different filters $h_i \in \mathbf{R}^N$. In this paper, we discuss an extension to this problem where a second type of querying is possible, called "*dimension querying*". With dimension querying, the user can be queried for each dimension of the optimal filter, namely $h^*[1], h^*[2], \ldots, h^*[N]$. In audio personalization, for example, the optimal h^* is the hearing profile of a user in the frequency domain; if we accurately estimate the hearing profile, we can maximize their satisfaction. With dimension querying, a user can listen to sound at each individual frequency $j \in \{1, N\}$ and tell us the best score $h^*[j]$. The only problem is that N can be very large, say 8000 Hz, hence it is prohibitive to query the user thousands of times.

Our solution builds on past work that uses Gaussian Process Regression (GPR). Conventional GPR models a surrogate $\hat{f}(h)$ for the user satisfaction function $f(h)$. The personalization filter is thus

Preprint. Under review.

obtained as $\hat{h}^* = \text{argmax}\, \hat{f}(h)$. Our contribution lies in encoding dimension queries in GPR's mathematical framework – we first sample a batch of q filters from GPR's posterior (as candidates for the next query), but a single winner is selected based on which has the strongest similarity to the dimension-scores. Finally, these operations are performed after GPR has been transformed into a sparse space, otherwise the query budget B becomes too large.

Results show that spending some query budget on dimension queries as opposed to spending all the budget on filter queries offers consistent benefits. We show empirical results from extensive simulations and real-world experiments. With real volunteers who were asked to rate audio quality on a scale of $[0-10]$, our proposed method, ORACLEBO, achieves an average of 3.3 points higher satisfaction, within a budget of $B = 30$ queries. Through simulations with various satisfaction functions, we find that the break-down between the two types of queries exhibits a sweet spot.

2 Problem Formulation

Consider an *unknown* real-valued function $f : \mathcal{H} \to \mathbf{R}$ where $\mathcal{H} \subseteq \mathbf{R}^N$, $N \geq 500$. Let h^* be the minimizer of $f(h)$. We want to estimate h^* using a budget of B queries, where a query can be one of two types:

1. f can be sampled at a given h_i. This query yields $f(h_i)$. We call these filter queries, Q_f.
2. An Oracle is assumed to know information about h^*. The Oracle, when queried, can give us one dimension of h^*, i.e., $h^*[j]$, for any given $j \in [1, 2, \ldots, N]$. We call these dimension queries, Q_d.

Thus, the optimization problem is,

$$\begin{aligned} \underset{\hat{h} \in \mathcal{H}}{\text{argmin}} \quad & ||f(\hat{h}) - f(h^*)||_2 \\ \text{s.t.} \quad & Q_f + Q_d \leq B \end{aligned} \tag{1}$$

where Q_f, Q_d are the number of filter and dimension queries, and the query budget $B \ll N$. f may be non-convex, may not have a closed-form expression, and its gradient is unavailable. We assume f is sparse i.e., there is a low-dimensional space that compactly describes f, so f has "low effective dimensions". We employ Sparse Bayesian Optimization that builds on Gaussian Process Regression (GPR). We review the relevant background on Bayesian optimization in the Appendix.

3 ORACLEBO

SparseBO methods like ALEBO [14] exploit the sparsity of f to create a low-dimensional embedding space $\mathcal{Y} \subseteq \mathbf{R}^d, d << N$ corresponding to $\mathcal{H} \subseteq \mathbf{R}^N$ through random projections ($\mathbf{B}^\dagger$) using only Q_f filter queries. ORACLEBO's main contribution is in modifying ALEBO's acquisition function to incorporate queries of type Q_d, namely *dimension queries*. Figure 1(a) illustrates the design of ORACLEBO – the modules in gray are the proposed extensions over literature. The two key modules are (1) **Batch Acquisition Function (BAF)**, and (2) **Dimension Matched Sampler (DMS)**.

3.1 Batch Acquisition Function (BAF)

Instead of picking one sample h', **BAF** picks q samples $\{h'_1, h'_2, \ldots, h'_q\} = \mathbf{B}^\dagger\{y'_1, y'_2, \ldots, y'_q\}$ that *jointly* maximize the acquisition function ($\mathbf{B}^\dagger$ denotes the sparse transformation). We assign a joint metric, *q-ExpectedImprovement* (qEI), to a set of q candidate points $\mathcal{Q}' = \{y'_1, y'_2, \ldots, y'_q\} \in \mathcal{Y}$.

$$\begin{aligned} \text{qEI}(y_i|\mathcal{D}_n, \mathcal{Q}) &= \mathbf{E}[[f^* - f(y_i)]^+|\mathcal{D}_n, \mathcal{Q}] \\ \text{qEI}(\mathcal{Q}) &\triangleq \{\text{qEI}(y_i|\mathcal{D}_n, \mathcal{Q})\} \end{aligned} \tag{2}$$

where $\mathcal{D}_n$ denotes current set of observations. We select the candidate set $\mathcal{Q}'$ of highest expected improvement as follows:

$$\begin{aligned} \mathcal{Q}' = \{y'_1, y'_2, \ldots, y'_q\} = \underset{\mathcal{Q}=\{y_1, y_2, \ldots, y_q\}}{\text{argmax}} \quad & max(\text{qEI}(\mathcal{Q})) \\ \text{s.t.} \; -1 \leq \mathbf{B}^\dagger y'_i \leq 1 \quad & \forall y'_i \in \mathcal{Q}' \end{aligned} \tag{3}$$

Note that the box constraint ensures **BAF** operates within the bounds of $\mathcal{H}$ in the sparse space. These points and their corresponding qEI metric values $(\mathcal{Q}', \text{qEI}(\mathcal{Q}'))$ are then passed as input to **DMS**.

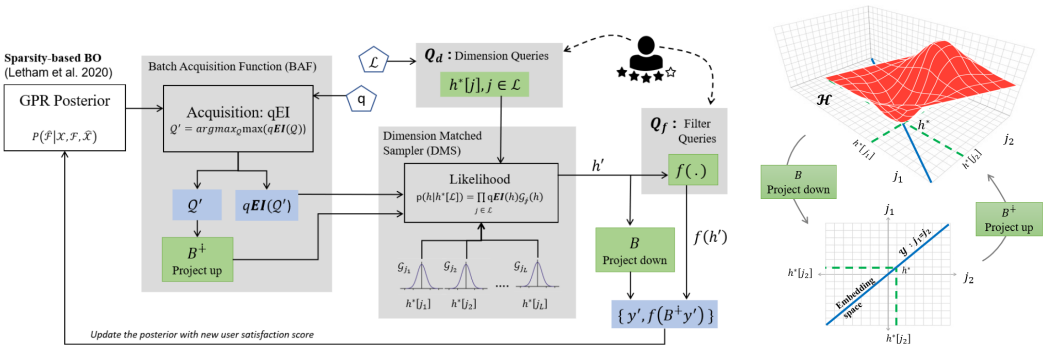

Figure 1: System flow: ORACLEBO consists of three modules: **BAF**, **DMS**, and GPR posterior. Green boxes denote the system inputs and hyper-parameters, and blue marks the module outputs. The right figure shows the transformation between the high and low dimensional spaces, made feasible by the random embedding matrix in ALEBO.

3.2 Dimension Matched Sampler (DMS)

DMS selects one sample $h' = \mathbf{B}^\dagger y'$ that best matches the information of $h^*[j], j \in \mathcal{L}$ from Q_d queries. A Q_f query is made at this h' to update the GPR posterior. Clearly, the **DMS** algorithm must operate in high-dimensional space $\mathcal{H}$ as both h' and $h^* \in \mathcal{H}$. In contrast, the **BAF** module operates in the $d-$dimensional embedding space $\mathcal{Y} \subseteq \mathbf{R}^d$. To remedy this, the **BAF**'s outputs in the embedding space are projected up to $\mathcal{H}$, i.e., $\mathcal{Q}' = \{y'_1, y'_2, \ldots, y'_q\} \rightarrow \mathcal{T}' = \{h'_1, h'_2, \ldots, h'_q\}$ as shown in Figure 1(a) (the green box labeled $\mathbf{B}^\dagger$). Figure 1(b) illustrates the translation of any point from high-dimensional space $\mathcal{H}$ to the low-dimensional embedding space $\mathcal{Y}$ and vice versa.

DMS uses a joint *likelihood* measure to preferentially order the q samples based on their degree of similarity to the L dimension queries $h^*[j], j \in \mathcal{L}$

$$P(h|h^*[\mathcal{L}]) = \prod_{j \in \mathcal{L}} \text{qEI}(h)\mathcal{G}_j(h)$$
$$h' = \underset{h \in \mathcal{T}'}{\text{argmax}}\, P(h|h^*[\mathcal{L}]) \tag{4}$$

where, $\mathcal{G}_j = \mathcal{N}(\mu = h^*[j], \sigma)$ is a Gaussian with mean $h^*[j]$ and variance σ for each Q_d $j \in \mathcal{L}$.

4 Experiment: Synthetic BlackBox Functions

We first present experiments on synthetic functions such as **Staircase Satisfaction Functions** (see Figure 5 (in Appendix)) that roughly mimic how humans rate their experiences in discrete steps [2].

Baseline and Metrics: We consider a baseline that extends ALEBO with L dimension queries. Thus ALEBO's search space is reduced from $\mathcal{R}^N$ to $\mathcal{R}^{N-L}$. Our evaluation metric is *Regret* $= (f(\hat{h}^*) - f(h^*))$. In the following figures, X-axis label "function evaluations" indicates the number of filter queries (Q_f), L denotes the number of dimension queries (Q_d). For comparison, we mark points on the graph that use the same query budget, $B = Q_f + Q_d$[1]. More details on the objective functions and evaluation parameters are included in the Appendix.

Comparison to ALEBO(L): Figure 2(a) shows the performance of ORACLEBO against ALEBO(L). ALEBO($L = 0$) performs the weakest because it does not use any Q_d. ALEBO($L = 5$) shows immediate gain since it searches only $\mathbf{R}^{N-5}$. ORACLEBO shows further improvement, implying that the combination of Q_f and Q_d queries are beneficial, even though the search space is $\mathcal{R}^N$. Observe that points marked with stars all have the same query budget $B = 90$, thus, ORACLEBO achieves high satisfaction (low regret) for a given B. When L=15, the regret is even lower.

Effect of Varying L: Figure 2(b) reports the impact of increasing L, on regret. For a fixed $B = 90$, increasing L is beneficial but only up to $L = 15$. Increasing L further offers more information about the optimal h^* but at the expense of the number of Q_f queries. Evidently, for the staircase function, the optimal L is around 15.

[1]For readability, we abuse the notation Q_d, which is equal to the number of dimension queries, L.

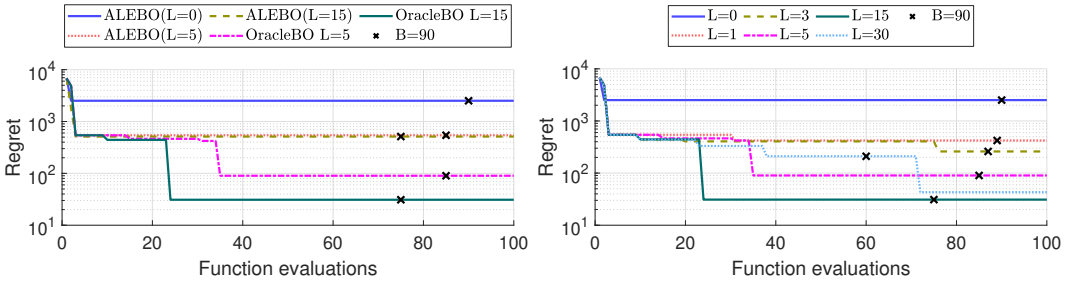

Figure 2: (a) Performance on ALEBO(L) and ORACLEBO. (b) Different number of Q_d queries on ORACLEBO.

We include further analyses such as Hyperparameter selection and results for commonly used benchmark functions such as `BRANIN`, `HARTMANN6`, and `ROSENBROCK` [25] in the Appendix.

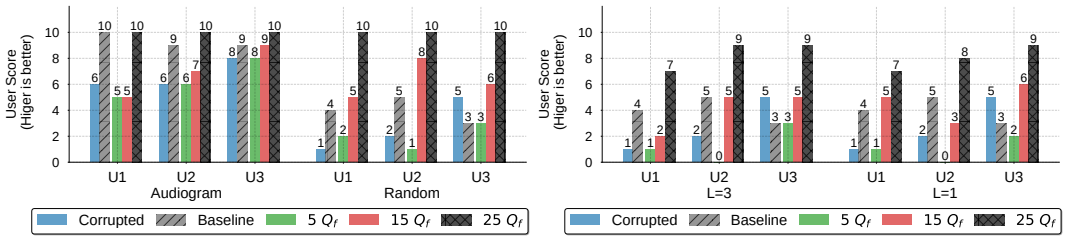

Figure 3: User score comparison on (a) $L = 5$ on hearing-loss profile and random profile. (b) $L = [1, 3]$ on random profile.

5 Experiments: Audio Personalization

We report experiments with volunteers in the context of personalizing hearing aids. Hearing aids filter the audio with h so that the user's hearing loss is compensated, and their satisfaction $f(h)$ is maximized. Hearing aid prescriptions exactly perform dimension querying where different frequency tones j are played, and their audibility is recorded as $h^*[j]$. Audio clinics play around L=7 tones and interpolate through them to generate the user's personalized filter called the *audiogram*. Interpolation is a coarse approximation of the user's true personal filter, h^*. We expect to improve the user's satisfaction over their *audiogram*, using a modest number of Q_f queries prescribed by ORACLEBO.

We invited 3 volunteers with no hearing loss. To emulate hearing loss, we deliberately corrupted the audio ("`Corrupted`") with hearing loss profiles from the public hearing-loss database, NHANES [22]. We compute the coarse-grained audiogram and compare ORACLEBO against this "`Baseline`".

Figure 3(a) plots the satisfaction score from the 3 volunteers (U1, U2, U3), The `Corrupted` signal obviously receives a low score, but the interpolated audiogram, `Baseline`, considerably improves the score. ORACLEBO matches/improves user satisfaction with $Q_d = 5$ and $Q_f = 25$ queries. With fewer Q_f of 5 and 15, ORACLEBO could not outperform `Baseline` as the search space $\mathcal{R}^{4000}$ had not been sufficiently sampled.

The audiogram `Baseline` performs quite well because human hearing is reasonably flat within octaves, hence, interpolation is adequate. We thus explore another application that injects more complex audio distortions, e.g., a cheap music speaker. We again emulate this distortion by deliberately corrupting the audio with a random filter h. Fig 3(a)-Random plots the results for the same 3 users. ORACLEBO improves the satisfaction scores even with $Q_f = 15$ queries, and achieves the maximum with $Q_f = 20$. The audio demos at various stages of the optimization are made available at [1].

6 Follow-up Work and Conclusion

This paper formulates a new problem in BBO where an Oracle can reveal information about the minimizer, h^*. This problem maps to real-world human-centric applications. After its empirical treatment in this paper, we believe there is promising follow-up work such as, (1) an analytical treatment on convergence for the hybrid $Q_f + Q_d$ querying. (2) other applications that are fit for hybrid querying. We hope ORACLEBO serves as a starter in solving problems along these directions.

References

[1] ORACLEBO demo (https://oraclebo.github.io/), 2023.

[2] A. R. Al-Roomi. Unconstrained Single-Objective Benchmark Functions Repository, 2015.

[3] B. C. Antoine Lorenzi. Human frequency discrimination, 2003.

[4] M. Binois, D. Ginsbourger, and O. Roustant. On the choice of the low-dimensional domain for global optimization via random embeddings. *Journal of global optimization*, 76:69–90, 2020.

[5] CDC. Centers for disease control and prevention (cdc). national center for health statistics (nchs). national health and nutrition examination survey data, hyattsville, md, 2011.

[6] D. Eriksson and M. Jankowiak. High-dimensional bayesian optimization with sparse axis-aligned subspaces. In *Uncertainty in Artificial Intelligence*, pages 493–503. PMLR, 2021.

[7] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

[8] P. I. Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[9] J. Gardner, C. Guo, K. Weinberger, R. Garnett, and R. Grosse. Discovering and exploiting additive structure for bayesian optimization. In *Artificial Intelligence and Statistics*, pages 1311–1319. PMLR, 2017.

[10] R. Garnett, M. A. Osborne, and P. Hennig. Active learning of linear embeddings for gaussian processes. *arXiv preprint arXiv:1310.6740*, 2013.

[11] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[12] K. Kandasamy, J. Schneider, and B. Póczos. High dimensional bayesian optimisation and bandits via additive models. In *International conference on machine learning*, pages 295–304. PMLR, 2015.

[13] J. Kirschner, M. Mutny, N. Hiller, R. Ischebeck, and A. Krause. Adaptive and safe bayesian optimization in high dimensions via one-dimensional subspaces. In *International Conference on Machine Learning*, pages 3429–3438. PMLR, 2019.

[14] B. Letham, R. Calandra, A. Rai, and E. Bakshy. Re-examining linear embeddings for high-dimensional bayesian optimization. *Advances in neural information processing systems*, 33:1546–1558, 2020.

[15] X. Lu, J. Gonzalez, Z. Dai, and N. D. Lawrence. Structured variationally auto-encoded optimization. In *International conference on machine learning*, pages 3267–3275. PMLR, 2018.

[16] R. Moriconi, M. P. Deisenroth, and K. Sesh Kumar. High-dimensional bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109:1925–1943, 2020.

[17] M. Mutny and A. Krause. Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. *Advances in Neural Information Processing Systems*, 31, 2018.

[18] D. M. Negoescu, P. I. Frazier, and W. B. Powell. The knowledge-gradient algorithm for sequencing experiments in drug discovery. *INFORMS Journal on Computing*, 23(3):346–363, 2011.

[19] C. Oh, E. Gavves, and M. Welling. Bock: Bayesian optimization with cylindrical kernels. In *International Conference on Machine Learning*, pages 3868–3877. PMLR, 2018.

[20] C. Oh, J. Tomczak, E. Gavves, and M. Welling. Combinatorial bayesian optimization using the graph cartesian product. *Advances in Neural Information Processing Systems*, 32, 2019.

[21] H. Qian, Y.-Q. Hu, and Y. Yu. Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings. In *IJCAI*, pages 1946–1952, 2016.

[22] M. K. Salmon, J. Brant, M. H. Hohman, and D. Leibowitz. Audiogram interpretation. 2022.

[23] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[24] B. Solnik, D. Golovin, G. Kochanski, J. E. Karro, S. Moitra, and D. Sculley. Bayesian optimization for a better dessert. 2017.

[25] D. B. Sonja Surjanovic. Virtual library of optimization functions, 2013.

[26] T. Ueno, T. D. Rhone, Z. Hou, T. Mizoguchi, and K. Tsuda. Combo: An efficient bayesian optimization library for materials science. *Materials discovery*, 4:18–21, 2016.

[27] J. Wang. An intuitive tutorial to gaussian processes regression. *arXiv preprint arXiv:2009.10862*, 2020.

[28] Z. Wang, C. Gehring, P. Kohli, and S. Jegelka. Batched large-scale bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 745–754. PMLR, 2018.

[29] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Feitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

A Appendix

A.1 Background Material

A.1.1 Bayesian Optimization

Bayesian optimization [8] broadly consists of the following two modules:

(1) **Surrogate model**: A family of functions that serve as candidates for the unknown objective function. The functions are commonly drawn from a Gaussian process generated by **Gaussian Process Regression (GPR)**. This essentially means that GPR generates a Gaussian posterior distribution of the likely values function f can take at any point of interest h.

(2) **Acquisition function**: A sampling strategy that prescribes the point at which f should be observed next. The GPR posterior model is used to evaluate the function at new points h', and one is picked that maximizes a desired metric. This new point h' when observed will maximally improve the GPR posterior.

We review GPR next, followed by a popular acquisition function called "Expected Improvement".

A.1.2 Gaussian Process Regression (GPR)

Non-parametric model: Gaussian processes [27] are helpful for black-box optimization because they provide a non-parametric mechanism to generate a surrogate for the unknown function f. Given a set of samples $\mathcal{X} = \{h_1, h_2, \ldots, h_K\}$ at which the function f has been observed, i.e., we know $\mathcal{F} = \{f(h_1), f(h_2), \ldots, f(h_K)\}$, we can identify an infinite number of candidate functions that match the observed function values. Figure 4 shows an example function f in 1-dimensional space.

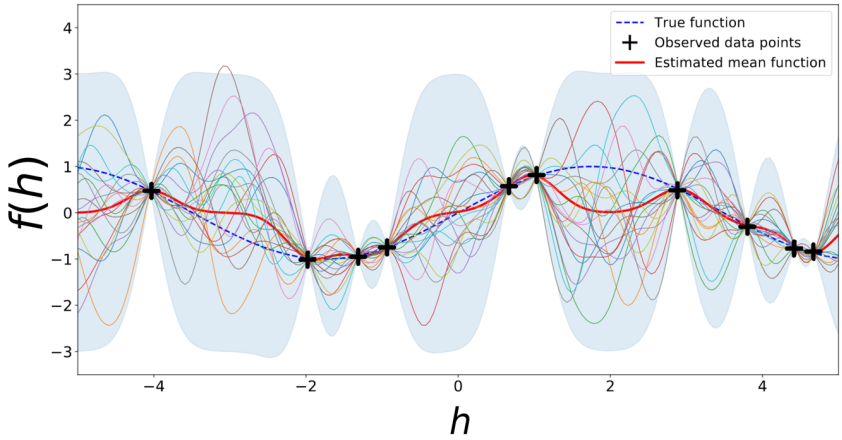

Figure 4: A GPR Posterior: The black "plus" symbols mark all the observed points. The dashed-blue line is the true f, and the red line is the estimated $\hat{f}$ (the mean of the posterior). The light blue shaded area marks the variance.

Function distribution & Kernel: GPR generates the surrogate model by defining a Gaussian distribution over these infinite candidate functions. Given the set of observations $(\mathcal{X}, \mathcal{F})$, the mean $\boldsymbol{\mu}$ of the distribution is the most likely surrogate of the function f. The covariance $\mathbf{K}$ is a kernel that dictates the smoothness (or shape) of the candidate functions and must be chosen based on domain knowledge of f. One commonly used kernel is the ARD (Automatic Relevance Determination) *power exponential kernel*:

$$k(h, h') = a_0 \exp -\frac{1}{2}(h - h')^T \Sigma^{-1} (h - h') \tag{5}$$

where, a_0 and $\Sigma = \text{diag}_i(\sigma_i)$ are the kernel parameters.

Prior & Posterior: Before any observations, the distribution defined by $\mathbf{K}$ and $\boldsymbol{\mu} = \mathbf{0}$ forms the prior distribution. Given a set of observations $(\mathcal{X}, \mathcal{F})$, the prior is updated to form the posterior

distribution over the candidate functions. Figure 4 shows the distribution of candidates and the mean surrogate model of an example f. With more observations, the current posterior serves as the prior, and the new posterior updates from the new observations. Eqn. 6 models the function f with the posterior generated by GPR.

$$P(\mathcal{F}|\mathcal{X}) \sim \mathcal{N}(\mathcal{F}|\boldsymbol{\mu}, \mathbf{K}) \tag{6}$$

where, $\boldsymbol{\mu} = \{\mu(h_1), \mu(h_2), \ldots, \mu(h_K)\}$ and $\mathbf{K}_{ij}$=$k(h_i, h_j)$, k represents a kernel function.

Predictions: To make predictions $\hat{\mathcal{F}} = f(\hat{\mathcal{X}})$ at new points $\hat{\mathcal{X}}$, GPR uses the current posterior $P(\mathcal{F}|\mathcal{X})$ to define the joint distribution of $\mathcal{F}$ and $\hat{\mathcal{F}}$, $P(\mathcal{F}, \hat{\mathcal{F}}|\mathcal{X}, \hat{\mathcal{X}})$ in Eqn. 7.

$$\begin{bmatrix} \mathcal{F} \\ \hat{\mathcal{F}} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\mathcal{X}) \\ \mu(\hat{\mathcal{X}}) \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \hat{\mathbf{K}} \\ \hat{\mathbf{K}}^T & \hat{\hat{\mathbf{K}}} \end{bmatrix} \right) \tag{7}$$

where, $\mathbf{K} = k(\mathcal{X}, \mathcal{X})$, $\hat{\mathbf{K}} = k(\mathcal{X}, \hat{\mathcal{X}})$, $\hat{\hat{\mathbf{K}}} = k(\hat{\mathcal{X}}, \hat{\mathcal{X}})$ and $(\mu(\mathcal{X}), \mu(\hat{\mathcal{X}})) = \mathbf{0}$.

The conditional distribution and hence prediction of $\hat{\mathcal{F}}$ is derived from the joint distribution shown in Eqn. 8. The proof and explanations of all the above are clearly presented in [27]).

$$P(\hat{\mathcal{F}}|\mathcal{F}, \mathcal{X}, \hat{\mathcal{X}}) \sim \mathcal{N}(\hat{\mathbf{K}}^T\mathbf{K}^{-1}\mathcal{F}, \hat{\hat{\mathbf{K}}} - \hat{\mathbf{K}}^T\mathbf{K}^{-1}\hat{\mathbf{K}}) \tag{8}$$

A.1.3 Acquisition function

In each GPR iteration, a new h must be acquired such that the observed $f(h)$ maximally improves the posterior from the previous iteration. This requires a **judicious sampling** strategy that optimizes an improvement metric. "Expected Improvement" (EI) is one such popular metric.

Expected Improvement: Given previous observations $\mathcal{D} = (\mathcal{X}, \mathcal{F})$, let $f^* = \min_{x \in \mathcal{X}} f(x)$ be the current function minimum (i.e., the minimum observed till now). If a new observation $f(h)$ is made at h, then the minimum now will be one of these:

- $f(h)$ if $f(h) \leq f^*$
- f^* if $f(h) \geq f^*$

Hence, the improvement from observing f at h is
$[f^* - f(h)]^+$, where, $a^+ = \max(a, 0)$.

We want to choose h that maximizes this improvement. However, $f(h)$ is unknown until the observation is made, so we choose h that maximizes the expectation of this improvement. *Expected Improvement* is thus defined as:

$$\mathbf{EI}(h|\mathcal{X}, \mathcal{F}) = \mathbf{E}[[f^* - f(h)]^+|\mathcal{X}, \mathcal{F}] \tag{9}$$

where, $\mathbf{E}[\cdot|\mathcal{X}, \mathcal{F}]$ is the expectation taken on the GPR posterior distribution given observations $(\mathcal{X}, \mathcal{F})$. This posterior is as specified in Eqn. 6. Thus, the next sample to make an observation at is:

$$h = \underset{h_i \in \mathbf{R}^N}{\operatorname{argmax}} \, \mathbf{EI}(h_i|\mathcal{X}, \mathcal{F}) \tag{10}$$

A.1.4 High dimensional Bayesian Optimization

Curse of dimensionality: The objective function in Eqn. 1 typically lies in a high dimensional space (i.e., $h \in \mathcal{H} \subseteq \mathbf{R}^N, N \geq 500$). Bayesian optimization works well for functions of < 20 dimensions [8]; with more dimensions, the search space $\mathcal{H}$ increases exponentially, and finding the minimum with *few* evaluations becomes untenable. One approach to reducing the number of queries is to exploit the sparsity inherent in most real-world functions.

We assume our function in Eqn. 1 is sparse, i.e., there is a low-dimensional space that compactly describes f, so f has "low effective dimensions". We review ALEBO [14], a class of methods

that exploit sparsity to create a low-dimensional embedding space using random projections. Our proposed idea builds on top of ALEBO, but we are actually agnostic of any specific sparsity method.

A.1.5 Linear Embedding using Random Projections

Random Projections: Given a function $f : \mathbf{R}^N \to \mathbf{R}$ with effective dimension d_f, ALEBO's linear embedding algorithm uses random projections to transform f to a lower dimensional embedding space. This transformation must guarantee that the minimum h^* from high dimensional space $\mathcal{H}$ gets transformed to its corresponding minimum y^* in low dimensional embedding space. The right side of Figure 1 aims to visualize this transformation. Without satisfying this property, optimization in low-dimensions is not possible.

The random embedding is defined by an embedding matrix $\mathbf{B} \in \mathbf{R}^{d \times N}$ that transforms f into its lower dimensional equivalent $f_B(y) = f(h) = f(\mathbf{B}^\dagger y)$, where $\mathbf{B}^\dagger$ is the pseudo-inverse of $\mathbf{B}$. Bayesian optimization of $f_B(y)$ is performed in the lower dimensional space $\mathbf{R}^d$.

Clipping to $\mathcal{H}$: When f is optimized over a compact subset $\mathcal{H} \subseteq \mathbf{R}^N$, We cannot evaluate f outside $\mathcal{H}$. One approach to prevent any embedding point y from being projected outside of $\mathcal{H}$ is to "clip" such points to $\mathcal{H}$. This is done by projecting the points back into $\mathcal{H}$, i.e., $f_B(y) = f(p_{\mathcal{H}}(\mathbf{B}^\dagger y))$ where, $p_{\mathcal{H}} : \mathbf{R}^N \to \mathbf{R}^N$ is the clipping projection. However, this clipping to $\mathcal{H}$ causes nonlinear distortions.

Instead, constraining the optimization to only points in $\mathcal{Y}$ that do not project outside $\mathcal{H}$, i.e., $\mathbf{B}^\dagger y \in \mathcal{H}$, prevents distortions; however, it also reduces the probability of the embedding containing the optimum h^*. ALEBO remedies this by choosing $d > d_f$ (an embedding space larger than f's effective dimensions). Then, the acquisition function evaluated in the constrained embedding space is given as:

$$\begin{aligned} &\underset{y \in \mathbf{R}^d}{\operatorname{argmax}} \, \mathbf{EI}(y) \\ \text{s.t.} \quad &-1 \leq \mathbf{B}^\dagger y \leq 1 \end{aligned} \tag{11}$$

where the constraint $-1 \leq \mathbf{B}^\dagger y \leq 1$ are linear and form a polytope.

Modifications to the Kernel: ARD kernels in $\mathcal{H}$ (shown in Eqn. 5) do not translate to a product kernel in embedding $\mathcal{Y}$, since each dimension in $\mathcal{H}$ is independent (diagonal matrix Σ in Eqn. 5). However, moving along one dimension in embedding is similar to moving across all dimensions of $\mathcal{H}$. To combat this, a *Mahalanobis Kernel* is used in the embedding. Any two points in the embedding are projected up to $\mathbf{R}^N$ ($\mathbf{B}^\dagger$) and then projected down to $\mathcal{H}$ ($\mathbf{A}$), $f_B(y) = f(\mathbf{B}^\dagger y) = f(\mathbf{A}\mathbf{B}^\dagger y)$ and $\text{Cov}[f_B(y), f_B(y')] = \exp\{-(y-y')^T \mathbf{\Gamma}(y-y')\}$ where $\mathbf{\Gamma} = (\mathbf{A}^T\mathbf{B}^\dagger)^T \Sigma (\mathbf{A}^T\mathbf{B}^\dagger)$ is a symmetric positive definite matrix. This finally ensures correctness in sparsity-based Bayesian Optimization (BO).

A.2 Synthetic Functions Experiment Details

In this section, we discuss the experiment setup for ORACLEBO's application to synthetic satisfaction functions and other BO benchmark functions. Since we have knowledge of the minimizer h^*, we simulate dimension querying Q_d at queried dimensions $\mathcal{L}$ i.e., $h^*[j], j \in \mathcal{L}$.

The two sets of objective functions to be optimized are:

(1) **Satisfaction Functions**: We generated functions [2] that have a discontinuous staircase structure to mimic the user satisfaction scoring function. The staircase structure is due to the fact that a user's audio perception might not change for a range of filters so the score remains the same and might change with sudden jumps for some filter choices, thus leading to a flat shape in some regions and steep curve in other regions. The staircase structure results in the functions having infinite local minima, infinite global minima, and zero gradient regions. These functions are naturally not suited for gradient-based optimization techniques. In our work, we use three such perception functions denoted as $P1, P2$, and $P3$. The functions are defined in Eqns 12,13,14.

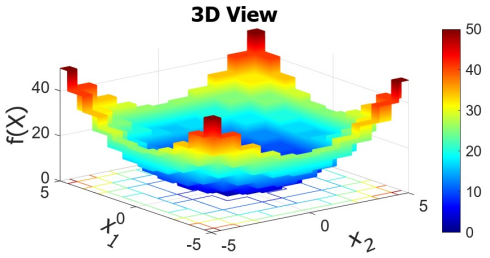

Figure 5: Satisfaction Function $P1$: Discontinuous staircase structure, containing infinite zero gradient regions.

(2) **Benchmark Functions**: We test ORACLEBO on commonly used benchmark functions in Bayesian optimization research: `BRANIN`, `HARTMANN6`, and `ROSENBROCK` [25]. These functions are denoted as B, H, and R. The functions are defined in Eqns 15,16,17.

$$f_{P1}(\mathbf{h}) = \sum_{i}^{N} (\lfloor |h_i + 0.5| \rfloor)^2 \tag{12}$$

where, $-100 \leq h_i \leq 100, i = 1, 2, \ldots, N$, h_i is filter h along dimension i. Infinite global minima at $f_{min}(\mathbf{h}^*) = 0$, and the minimizers are $-0.5 \leq h_i^* < 0.5$ (i.e.,) $h_i^* \in [-0.5, 0.5), i = 1, 2, \ldots, N$

$$f_{P2}(\mathbf{h}) = \sum_{i}^{N} (\lfloor |h_i| \rfloor) \tag{13}$$

where, $-100 \leq h_i \leq 100, i = 1, 2, \ldots, N$. Infinite global minima at $f_{min}(\mathbf{h}^*) = 0$, and the minimizers are $-1 < h_i^* < 1$ (i.e.,) $h_i^* \in (-1, 1), i = 1, 2, \ldots, N$

$$f_{P3}(\mathbf{h}) = \sum_{i}^{N} (\lfloor (h_i)^2 \rfloor) \tag{14}$$

where, $-100 \leq h_i \leq 100, i = 1, 2, \ldots, N$. Infinite global minima at $f_{min}(\mathbf{h}^*) = 0$, and the minimizers are $-1 < h_i^* < 1$ (i.e.,) $h_i^* \in (-1, 1), i = 1, 2, \ldots, N$

$$f_B(\mathbf{h}) = a(h_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)cos(x_1) + s \tag{15}$$

where, $h_i \in [-5, 10], h_2 \in [0, 15], a = 1, b = 5.1/(4\pi^2), c = 5/\pi, r = 6, s = 10, t = 1/(8\pi)$.. Three global minima at $f_{min}(\mathbf{h}^*) = 0.397887$, and the minimizers are $\mathbf{h}^* = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)$

$$f_H(\mathbf{h}) = \sum_{i}^{4} \alpha_i \exp - \sum_{j}^{6} (A_{ij}(h_j - P_{ij})^2 \tag{16}$$

where, $h_i \in (0, 1), i = 1, 2 \ldots, 6$, $\alpha = (1, 1.2, 3, 3.2)^T$, $A = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$, $P = \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$. One global minimum at $f_{min}(\mathbf{h}^*) = -3.32237$, and the minimizer is $\mathbf{h}^* = (0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.6573)$.

$$f_R(\mathbf{h}) = \sum_{i}^{N-1} (100(h_{i+1} - h_i^2)^2 + (h_i - 1)^2) \tag{17}$$

where, $h_i \in [-5, 10], i = 1, 2 \ldots, N$. One global minimum at $f_{min}(\mathbf{h}^*) = 0$, and the minimizer is $h_i^* = 1, i = 1, 2 \ldots, N$.

A.2.1 Evaluation Parameters

In our experiments, we optimize the functions for:

- f_evals = 100 function evaluations
- r_init = 5 initial random samples after which the acquisition sampling begins
- $N = 2000, \mathcal{H} \subseteq \mathbf{R}^N$ is the high-dimensional space
- $\mathbf{R}^d, d = 4$ is embedding space for $P1, P2, P3$, Branin and Rosenbrock and is $\mathbf{R}^d, d = 6$ for Hartmann6 . Branin, Rosenbrock, and Hartmann6 have effective dimensionality $d_e = 2, 4, 6$, respectively.
- Different numbers of Q_d queries $L = 0, 1, 3, 5, 15, 30$ are used and $L = 0$ implies no Q_d queries are available and ORACLEBO functions as just ALEBO
- $\mathcal{L}$ Q_d Top and Random dimensions for Q_d queries are considered.
- $q = 5$ acquisition samples are used in Batch Acquisition Function in Eqns 2,3
- In the Dimension Matched Sampler, we use variance $\sigma = 1$: $\mathcal{G}_j = \mathcal{N}(\mu = h^*[j], \sigma = 1)$ for each dimension $j \in \mathcal{L}$ in Eqn 4
- For Branin, we use the minimizer $\mathbf{h}^* = (\pi, 2.275)$ to generate dimension Q_d queries and for perception functions $P1, P2, P3$ we use the minimizer $h_i^* = 0, i = 1, 2, \ldots, N$. For Hartmann6 and Rosenbrock we use their unique minimizers
- We run 10 random runs of each experiment

A.2.2 Synthetic Experiment Additional Results

In this section, we continue our discussion of results for synthetic functions like staircase satisfaction functions and benchmark functions.

Which L out of N queries? Given $L = 15$ queries, say, different subsets of N dimensions can be chosen. Let us denote this subset as $\mathcal{L}$. If $f(h)$ hardly varies along the dimensions included in $\mathcal{L}$, then $\mathcal{L}$ contributes little to estimating the satisfaction function. Figure 6 shows ORACLEBO's regret on two different $\mathcal{L}$. Note that because the objective function f is synthesized, the variation of f against any dimension y is known. In $\mathcal{L}_{Top}$, we select the L dimensions of largest variances; $\mathcal{L}_{Rand}$ denotes the randomly selected dimensions from $\{1, N\}$. Results show that $\mathcal{L}_{Top}$ achieves lower regret (median and variance) compared to $\mathcal{L}_{Rand}$. Thus, in real applications, it helps to choose L dimensions that are likely to influence the user's satisfaction.

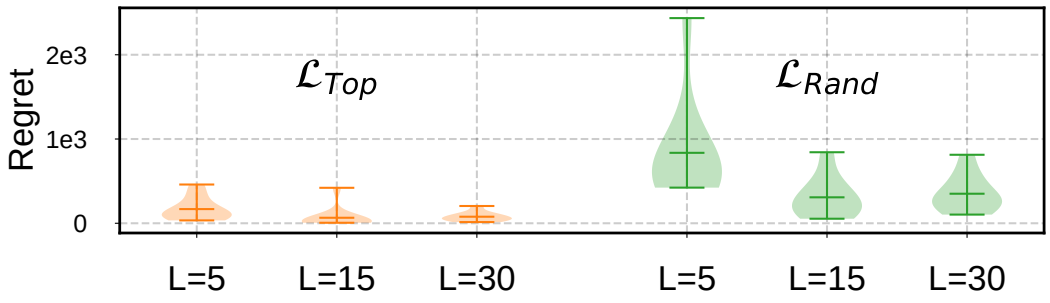

Figure 6: Distribution of ORACLEBO's regret on different Q_d subsets $\mathcal{L}_{Top}$ and $\mathcal{L}_{Rand}$.

Various objective functions: Figure 7 summarizes regret across a range of objective functions, including three different staircase functions (denoted $P1$ to $P3$) and 3 standard benchmark functions. Results confirm that ORACLEBO's performance improves until $L = 15$ dimension queries.

Hyperparameter Selection: Parameters in **BAF** and **DMS** modules include: N, dimension of the filter; d, dimension of the embedding; q, number of candidates BAF outputs, and σ, the dimensional variance in **DMS**.

Table 1 tabulates our analysis of ORACLEBO's performance over these parameters. $N = \{500, 2000\}, d = 4, q = 5, \sigma = 1$ yields the minimum regret for satisfaction function $P1$ with

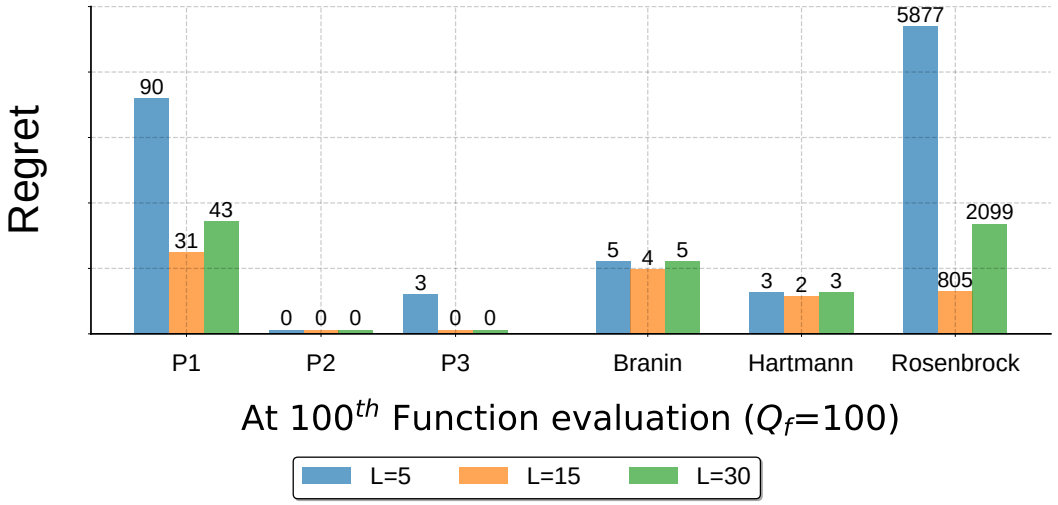

Figure 7: Different functions $f(h)$ with varying Q_d queries.

$Q_d = 5, Q_f = 100$. We pick $N = 2000$ to highlight ORACLEBO's performance in high-dimensional spaces. Thus, our choice of using these parameters for our experiments is informed by the analysis.

N	d	(q, σ)				
		(5,1)	(2,0.2)	(2,10)	(7,0.2)	(7,10)
500	4	83	445	316	1459	1459
	10	148	166	237	760	883
	20	477	551	609	1201	1255
2000	4	90	242	514	543	628
	10	2166	2331	2753	2331	2753
	20	2677	2764	3125	2764	3125
4000	4	513	606	1268	1268	1268
	10	1532	1627	2154	4006	5278
	20	1749	1993	3332	10213	10213

Table 1: Hyperparameter analysis on regret for $P1$.

A.3 Audio Personalization Experiment Details

Queries: The detailed definitions of queries on audio personalization are as follows:

1. **Query1 (Q_d): A clinical audiogram test**: Conventional hearing aids tuning involves playing pure-tone frequencies to the patient and capturing their hearing response. For each frequency j played with increasing amplitude (-10 dB to 120 dB), the user presses a button when they can hear the frequency. Doing this measurement for several frequencies gives the user's hearing loss profile $h^*[j] j = 0, 1, \ldots, N$.

   In practice, this is measured for $N = 7$ frequencies $(500Hz, 1KHz, 2KHz, 3KHz, 4KHz, 6KHz, 8KHz)$. This is too coarse a resolution to capture the exact hearing loss frequency response of a user as typically $N \geq 500$.

   So we cannot exploit these measurements (Q_d) alone to determine the personalizing filter h^*. However, these measurements do provide some information about the minimizer h^*. The audiogram test thus acts as the dimension query in ORACLEBO.

2. **Query2 (Q_f): Satisfaction function sampling**: We can choose filters h_j from the space of all filters $\mathcal{H} \subseteq \mathbf{R}^N$, apply it to any audio played to the user, and get their score $f(h_j)$. The user satisfaction function also has underlying sparsity because human hearing and hence audio perception is not the same across all frequencies [3]. Under the perception function constraints, the audio personalization problem is well suited for sparsity-based Bayesian Optimization techniques like ORACLEBO.

Baselines:

In Audio clinics, a user's audiogram is obtained through a coarse interpolation of these Q_d pure tone measurements. We use this coarse audiogram as our `Baseline`

Data Collection:

We recruit 3 volunteers of 1 male(s) and 2 female(s) without hearing loss. To simulate hearing loss or distortion due to cheap speakers, We use two different corrupting filters $b_{1:2}$.

In the case of hearing loss, we use the publicly available hearing loss profiles in the NHANES [5] database as the corrupting filter $b1$. For random distortions (cheap speakers), we generate a random corrupting filter $b2$.

The audiogram and random distorting filter measurements are available at finite frequencies $(500Hz, 1KHz, 2KHz, 3KHz, 4KHz, 6KHz, 8KHz)$ and thus act as the Q_d. We have at most 7 Q_d queries we can utilize.

A sample speech clip a is filtered with the distorting filter $b1$ or $b2$ to obtain the `corrupted` clip This audio is what is heard by a person with hearing loss or as heard from a cheap speaker, $r = b_{1:2} * a$.

The goal of the audio personalization task is to apply different filter queries Q_f, h_j to the clip r to construct the user satisfaction function and optimize it to find $\hat{h}^*$, the personal filter. This filter when applied to the audio clip should make the resulting audio sound similar to the original uncorrupted speech clip (i.e.) $r * \hat{h}^* = \hat{a} \approx a$.

The personalization filter should counteract the distortion caused by $b1$ or $b2$.

A.4 Related Work

To the best of our knowledge, ORACLEBO is the first work that combines two different types of queries, Q_f and Q_d, for Bayesian Optimization. We also believe such hybrid querying has not been applied in audio personalization. The closest work in the application context is [24] where authors used conventional BO to search for the best-rated cookie recipe at Google. We believe more applications can benefit and the burden of querying users can become practical with dimension querying (Q_d). Of course, BO has been extensively used to solve complex problems that do not have humans in the loop. These include material science [26], medicine [18], hyperparameter tuning in neural networks [23], etc.

ORACLEBO inherits sparsity-based BO frameworks based on low-dimensional embeddings. These papers use linear random projections to map from high to low-dimensional spaces [21][29][4][14]. Authors of [10][15] use a Gaussian Process to simultaneously learn the model and the embedding. Non-linear embeddings are learnt using Variation Autoencoders in [11][16][15]. ORACLEBO is agnostic to these algorithms and we expect their advantages to reflect in our performance as well. On a similar note, various papers modify the kernel [12][9][17][28] to restrict the candidate function choices [19], reflecting additional structure in the objective function. LineBO [13] optimizes the acquisition function along one-dimensional lines. TuRBO [7] employs trust regions around the current minimizer. [20], [6] use sparsity creating prior. The performance of ORACLEBO can be boosted with such kernel manipulations.