Sample-Efficient Optimization in the Latent Space of Deep Generative Models via Weighted Retraining

Austin Tripp[*] ajt212@cam.ac.uk
University of Cambridge

Erik Daxberger[*] ead54@cam.ac.uk
University of Cambridge; MPI for Intelligent Systems, Tübingen

José Miguel Hernández-Lobato jmh233@cam.ac.uk
University of Cambridge; Alan Turing Institute; Microsoft Research
[*]equal contribution

Abstract

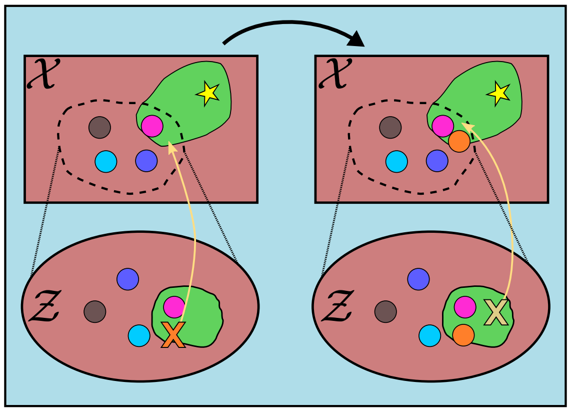
We introduce a method for efficiently optimizing an expensive black-box objective function over a high-dimensional, structured input space. To this end, we perform the optimization in the low-dimensional, continuous latent space learned by a deep generative model (DGM). We actively steer this space to be highly useful for optimization, by periodically *retraining* the DGM on the inputs queried during the optimization, and *weighting* those inputs according to their objective function value. This weighted retraining is easy to implement on top of existing methods, yet significantly improves their efficiency and performance.

1. Introduction

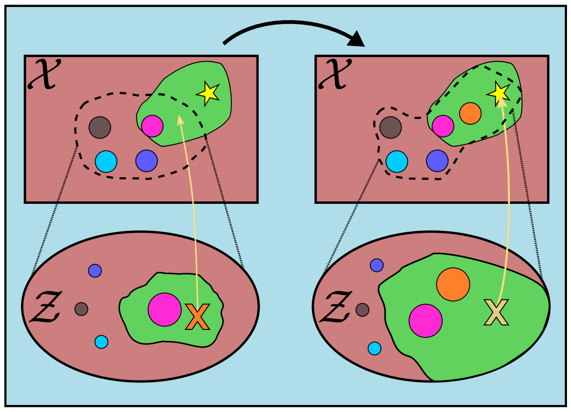
Many important problems in science and engineering can be formulated as optimizing a function $f : \mathcal{X} \mapsto \mathbb{R}$ over an input space $\mathcal{X}$ which is *high-dimensional* (i.e. 100+ effective dimensions) and/or *structured* (i.e. discrete or non-Euclidean such as graphs, sequences, and sets). Solving such problems becomes even more challenging if $f(\mathbf{x})$ is *black box* (i.e. no known analytic form or derivative information available) and is *expensive* to evaluate (e.g. in terms of time or energy cost). A notable example of such a problem is drug design, where $\mathcal{X}$ is the space of molecular graphs and f is evaluated using expensive wet-lab experiments.

A promising approach to tackle such problems is a two-stage procedure which we will refer to as *latent space optimization (LSO)*. In the first stage, a deep generative model (DGM) $g : \mathcal{Z} \mapsto \mathcal{X}$ is trained to map tensors in a low-dimensional, continuous space $\mathcal{Z}$ onto the data manifold in input space $\mathcal{X}$, effectively constructing a low-dimensional, continuous analog of the high-dimensional, structured optimization problem. In the second stage, the objective function is optimized over this learned latent space using well-studied, sample-efficient techniques such as Bayesian optimization (Shahriari et al., 2015). Despite showing promise in fields such as chemical design (Gómez-Bombarelli et al., 2018; Kusner et al., 2017), automatic machine learning (Lu et al., 2018; Luo et al., 2018), and conditional image generation (Nguyen et al., 2016), previous instances of LSO use a general-purpose DGM, rather than a DGM explicitly designed to be amenable to efficient optimization.

In this paper, we propose that the described shortcoming of current LSO methods can be addressed with a combination of *weighting of the data distribution* and *periodic retraining of the DGM*. We motivate and present this *weighted retraining* in Section 2, empirically demonstrate its effectiveness in Section 3, and discuss its wider context in Section 4.



(a) Standard Latent Space Optimization



(b) Extension with Weighted Retraining

Figure 1: Schematic of two iterations of LSO with and without our proposed extension of weighted retraining of the generative model $g : \mathcal{Z} \mapsto \mathcal{X}$. **Symbols**: Red/green regions correspond to points with low/high objective function values, respectively. The yellow star is the global optimum in $\mathcal{X}$. Coloured circles are data points; their radius in $\mathcal{Z}$ corresponds to their assigned weight. Crosses are queries made during optimization. The dashed line surrounds the region of $\mathcal{X}$ modelled by g.

2. Weighted Retraining for Latent Space Optimization

2.1 Motivation

To understand the shortcomings of LSO, it is necessary to first examine in detail the role of the DGM. State of the art DGMs such as variational autoencoders (VAEs) (Kingma and Welling, 2013) and generative adversarial networks (GANs) (Goodfellow et al., 2014) are trained with a prior $p(\mathbf{z})$ over the latent space $\mathcal{Z}$. This means that although the resulting function $g : \mathcal{Z} \mapsto \mathcal{X}$ is defined over the entire latent space $\mathcal{Z}$, it is effectively only trained on points in regions of $\mathcal{Z}$ with high probability under p (the *feasible region* of $\mathcal{Z}$).[1] Therefore, if optimization is performed outside of the feasible region, the resulting samples are usually low quality or invalid (Gómez-Bombarelli et al., 2018; Kusner et al., 2017; Nguyen et al., 2016; Griffiths and Miguel Hernández-Lobato, 2020; White, 2016).

Restricting the optimization to the feasible region solves this problem, but also creates new ones. Although the points in the feasible region of $\mathcal{Z}$ typically correspond to valid points in $\mathcal{X}$, the training objective of g encourages these points to roughly match the distribution of the training dataset $\mathcal{D}$ in aggregate. If most points $\mathbf{x} \in \mathcal{D}$ in the training dataset are low-scoring (i.e. have highly sub-optimal objective function values $f(\mathbf{x})$), one therefore expects that most of the feasible region of $\mathcal{Z}$ also contains low-scoring points. Not only does this make the optimization problem more difficult to solve (like finding the proverbial "needle in a haystack"), but we conjecture that it may actually leave insufficient space in the feasible region for a large number of novel, high-scoring points that lie outside the training distribution modelled by the DGM.

1. For example, a d-dimensional Gaussian distribution has non-zero probability density everywhere, but almost all the probability mass in in a spherical shell of radius $\sqrt{d}$. See *this blog post* for a visualization.

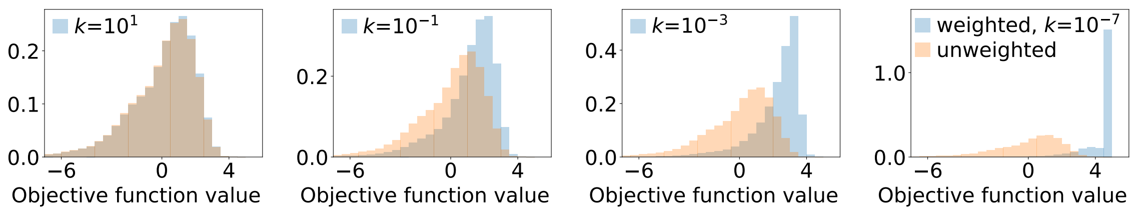


Figure 2: Distribution of objective function values of the ZINC dataset (see Section 3), weighted uniformly and with the rank weighting in equation (1). Large k approaches uniform weighting, while small k place most weight on the best (i.e. largest) function values.

The effect of this phenomenon on the optimization procedure is illustrated in Figure 1a. Because most of the training data in $\mathcal{X}$ lies in the low-scoring (red) region, the DGM learns a latent space $\mathcal{Z}$ which contains mostly low-scoring points. Although $\mathcal{Z}$ also contains high-scoring points, there are very few of them, and they are clustered around the training data; a good optimizer will find these points, as shown in Figure 1a, but will generally be unable to find the global optimum if it lies far from the training data. Furthermore, because the DGM is fixed during the optimization procedure, discovering new high-scoring points does nothing to help the DGM model high-scoring regions better, keeping the global optimum forever out of reach regardless of the success of optimization.

2.2 Weighting and Retraining Individually

Conceptually, the optimization in Figure 1a can be improved by 1) training g to model more of the high-scoring (green) region, and 2) using new points acquired during optimization to improve g's coverage of the high-scoring region. Retraining the DGM on points acquired during optimization would address point 2, and has been previously mentioned in Eismann et al. (2018). While it is unclear how to address point 1 in general, we propose that one approach would be to train the DGM on a data distribution that systematically assigns more probability mass on high-scoring points and less mass on low-scoring points, which would cause a larger fraction of $\mathcal{Z}$ to consist of high-scoring points.

Given only a limited amount of training data $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$, a simple way to achieve this effect is to *reweight* the training dataset. Recall that the training objective of common DGMs involves the expected value of a loss function $\mathcal{L}$ with respect to the data distribution,[2] which is typically estimated using the empirical distribution over the training dataset, i.e. $\mathbb{E}_{p(\mathbf{x})}[\mathcal{L}(\mathbf{x})] \approx \frac{1}{N}\sum_{i=1}^N \mathcal{L}(\mathbf{x}_i)$. This can be viewed as a special case of the weighted objective $\sum_{i=1}^N w_i \mathcal{L}(\mathbf{x}_i)$ with $w_i = \frac{1}{N}, \forall i$. By assigning an explicit weight w_i to each data point (normalized such that $\sum_{i=1}^N w_i = 1$), an arbitrary weighting of the data can be achieved.

We propose choosing w_i in a way that systematically assigns higher values to high-scoring points, and lower values to low-scoring points. While there are many reasonable

2. For a VAE, the loss $\mathcal{L}$ is the per-datapoint evidence lower bound (ELBO) (Kingma and Welling, 2013); for a GAN, the loss $\mathcal{L}$ is the discriminator score (Goodfellow et al., 2014).

Algorithm 1 Latent Space Optimization with Weighted Retraining (changes in blue)

```
1: Input: Data $\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i))\}_{i=1}^N$, evaluation budget $M$, objective function $f(\mathbf{x})$,
   generative model $g(\mathbf{z})$, inverse model $q(\mathbf{x})$, retrain budget $R$, weighting function $w(\mathbf{x})$
2: for $1, \dots, R$ do
3:   Train generative model $g$ and inverse model $q$ on data $\mathcal{D}$ weighted by $\mathcal{W} = \{w(\mathbf{x})\}_{\mathbf{x}\in\mathcal{D}}$
4:   for $1, \dots, M/R$ do
5:     Compute latent variables $\mathcal{Z} = \{\mathbf{z} = q(\mathbf{x})\}_{\mathbf{x}\in\mathcal{D}}$
6:     Fit objective model $h$ to $\mathcal{Z}$ and $\mathcal{D}$, and suggest new latent $\tilde{\mathbf{z}}$ based on $h$
7:     Obtain corresponding input $\tilde{\mathbf{x}} = g(\tilde{\mathbf{z}})$, evaluate $f(\tilde{\mathbf{x}})$ and set $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\tilde{\mathbf{x}}, f(\tilde{\mathbf{x}}))\}$
8:   end for
9: end for
10: Output: Augmented dataset $\mathcal{D}$
```

ways to do this, we decide to use a rank-based weight function,

$$w(\mathbf{x}; \mathcal{D}, k) \propto \frac{1}{kN + \operatorname{rank}_{f,\mathcal{D}}(\mathbf{x})}, \quad \operatorname{rank}_{f,\mathcal{D}}(\mathbf{x}) = \left| \{\mathbf{x}_i : f(\mathbf{x}_i) > f(\mathbf{x}),\ \mathbf{x}_i \in \mathcal{D}\} \right| \tag{1}$$

which assigns a weight roughly proportional to the reciprocal (zero-based) rank of each data point. We choose equation (1) because it yields positive weights, is resilient to outliers, and has a hyperparameter k controlling the degree of weighting. If $k = \infty$, equation (1) corresponds to uniform weighting, i.e. $w_i = \frac{1}{N}, \forall i$, and as $k \to 0$ the weights place progressively more mass on the highest-scoring points, with the limit of $k = 0$ placing *all* mass on the single point with the highest objective function value. This is illustrated in Figure 2.

2.3 Weighted Retraining Together

In addition to individually improving LSO, we also propose that data weighting and retraining have a synergistic effect when applied together because data weighting improves the effectiveness of the retraining. As a DGM usually requires at least 10^4 data points for training, augmenting this dataset with $\approx 10^1$ points is unlikely to significantly impact the DGM training.[3] However, if these new points are assigned a large weight because they are high-scoring, then they will have a disproportionate impact on the DGM during retraining, causing the DGM to change much more significantly than it would under uniform weighting.

Figure 1b illustrates the combined effect of data weighting and DGM retraining. Compared to Figure 1a, in the first iteration in Figure 1b the high-scoring pink point is given a higher weight, causing the feasible region to extend farther into the green region at the expense of the red region. This allows a better first point (orange) to be chosen. In the second iteration in Figure 1b, weighted retraining on that point reshapes the latent space again, bringing the global optimum into the feasible region, where it is ultimately reached.

We refer to the combination of these techniques as *weighted retraining*. Algorithm 1 shows pseudocode for LSO with weighted retraining, highlighting our proposed changes.

3. While one could also retrain the DGM on *only* the new data points, this might lead to the well-known phenomenon of catastrophic forgetting (McCloskey and Cohen, 1989).

3. Empirical Evaluation

3.1 Experimental Setup

We demonstrate the effectiveness of weighted retraining on three high-dimensional, structured optimization problems. On each problem, we perform an ablation study, comparing the effects of 1) neither retraining nor weighting (i.e., the default for current methods), 2) retraining but no weighting, 3) weighting but no retraining (i.e., using a fixed model initially trained on a weighted dataset), and 4) both weighting and retraining. When retraining is performed, it is done after every 50th function evaluation. We study the following problems:

2D Shape Area Maximization Task. We optimize for the image with the shape of largest area (i.e. the largest number of pixels with value 1) in the space of 64×64 binary images. **Data:** All $245,760$ images of squares from the dSprites dataset of binary shapes (Matthey et al., 2017), which have a maximum area of ≈ 400. **Model:** A standard convolutional VAE, with $\mathcal{Z} = \mathbb{R}^5$. **Optimizer:** We enumerate a grid in latent space $\mathcal{Z}$ with $\|\mathbf{z}\|_2 \leq 5$, to emulate a perfect optimizer (this is only feasible since $\mathcal{Z}$ is low-dimensional).

Arithmetic Expression Fitting Task. We optimize in the space of single-variable arithmetic expressions generated by a formal grammar, using an identical setup to Kusner et al. (2017). The objective is to find an expression with minimal mean squared error to the target expression $\mathbf{x}^* = \texttt{1/3 * v * sin(v*v)}$, computed over 1000 values of `v` evenly-spaced between -10 and $+10$. **Data:** 50,000 univariate arithmetic expressions generated by the formal grammar from (Kusner et al., 2017). **Model:** A grammar VAE (Kusner et al., 2017). **Optimizer:** Bayesian optimization with the expected improvement acquisition function (Jones et al., 1998) and a sparse Gaussian process with 500 inducing points (Titsias, 2009).

Chemical Design Task. We aim to produce drug-like molecules by optimizing the penalized logP score, following previous works (Kusner et al., 2017; Jin et al., 2019; Dai et al., 2018; You et al., 2018; Zhou et al., 2019). **Data:** The ZINC250k molecule dataset (Irwin et al., 2012). **Model:** A junction tree VAE with the same parameters as in Jin et al. (2019), which is the state of the art on this task. **Optimizer:** Same as for the expression task.

We use equation (1) with $k = 10^{-5}$ for the 2D shape task to exaggerate the effect of weighted retraining, and a more moderate value of $k = 10^{-3}$ for the other tasks.

3.2 Results

Rather than simply reporting the best score achieved after a fixed number of objective function evaluations, we instead plot the expected best score as a function of the number function evaluations in order to show the behaviour of the various algorithms over time. The expectation is calculated using three identical runs with different random seeds (except for the 2D shape task, where the optimization procedure used is deterministic). The results are shown in Figure 3. Firstly, it can be seen that weighting and retraining help individually, with the effect of weighting being individually more pronounced. However, weighted retraining together significantly outperforms all other methods on all three tasks. Notice in particular that the performance often sharply increases following retraining (gray dashed lines), while weighting without retraining tends to perform well but then plateau: this directly supports the improvement mechanism described in Section 2.3.

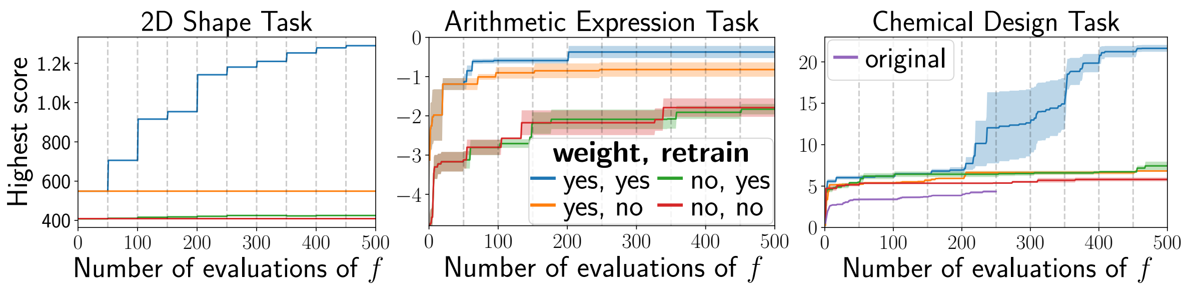


Figure 3: Mean ± one standard error (over three random seeds) of the best scores on the shape task (*left*), arithmetic expression task (*center*), and chemical design task (*right*). Weighted retraining achieves significantly better optimization performance and higher sample efficiency than the baselines on all three tasks. The purple line labelled "original" (right) shows the results from Jin et al. (2019), extracted from their `GitHub` repository. The grey dashed vertical lines indicate the iterations in which the DGM is retrained.

The performance on the chemical design task (Figure 3, right) is particularly noteworthy: the best previously achieved score was 11.84 and was obtained after ≈ 5000 objective function evaluations (Zhou et al., 2019). By contrast, our best score is 22.55, and was achieved after only 500 evaluations. The scores achieved with weighted retraining thus not only beat the results of all previous methods, but do so using far fewer samples.

4. Discussion and Conclusion

We proposed a method for efficient black-box optimization over high-dimensional, structured spaces, combining latent space optimization with weighted retraining. While being conceptually simple and easy to implement on top of previous methods, weighted retraining significantly boosts their efficiency and performance on challenging optimization problems.

In the context of optimization more generally, the main advantage of LSO with weighted retraining is its sample efficiency. Many previous works have addressed similar problems such as chemical design with reinforcement learning, which is known to be very sample inefficient (Li, 2018; You et al., 2018; Zhou et al., 2019; Guimaraes et al., 2018; Olivecrona et al., 2017; Popova et al., 2018; Simm et al., 2020), but has the advantage of being able to explore beyond the distribution of the training data to a degree that standard LSO is incapable of. Weighted retraining directly addresses this issue, which may possibly lead to LSO becoming a more widely-used general optimization method.

There are many avenues for future work. Firstly, we observed that weighted retraining was less beneficial when used with poorly-performing optimization algorithms. The latent space of DGMs can be challenging to optimize over, motivating further research to optimize more effectively and/or make the space more amenable to optimization. Another promising idea is the use of a weighting *schedule* instead of a fixed weighting, which may allow balancing exploration vs. exploitation similar to simulated annealing (Van Laarhoven and Aarts, 1987). Finally, we are eager to apply our approach to more real-world problems, further establishing machine learning as a critical tool for advancing science and engineering.

Acknowledgments

We thank Ross Clarke, Gregor Simm, David Burt, and Javier Antorán for insightful feedback and discussions. AT acknowledges funding via a C T Taylor Cambridge International Scholarship. ED acknowledges funding by the EPSRC and Qualcomm. This work has been performed using resources provided by the Cambridge Tier-2 system operated by the University of Cambridge Research Computing Service (http://www.hpc.cam.ac.uk) funded by EPSRC Tier-2 capital grant EP/P020259/1.

References

H. Dai, Y. Tian, B. Dai, S. Skiena, and L. Song. Syntax-Directed Variational Autoencoder for Structured Data. *arXiv:1802.08786 [cs]*, Feb. 2018. URL `http://arxiv.org/abs/1802.08786`. arXiv: 1802.08786.

S. Eismann, D. Levy, R. Shu, S. Bartzsch, and S. Ermon. Bayesian optimization and attribute adjustment. In *Proceedings of the Thirty-Fourth Conference (2018)*, page 11, Monterey, California, USA, Aug. 2018. Association for Uncertainty in Artificial Intelligence.

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.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

R.-R. Griffiths and J. Miguel Hernández-Lobato. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chemical Science*, 11(2): 577–586, 2020. doi: 10.1039/C9SC04026A. URL `https://pubs.rsc.org/en/content/articlelanding/2020/sc/c9sc04026a`. Publisher: Royal Society of Chemistry.

G. L. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. C. Farias, and A. Aspuru-Guzik. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv:1705.10843 [cs, stat]*, Feb. 2018. URL `http://arxiv.org/abs/1705.10843`. arXiv: 1705.10843.

J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman. ZINC: A Free Tool to Discover Chemistry for Biology. *Journal of Chemical Information and Modeling*, 52(7):1757–1768, July 2012. ISSN 1549-9596. doi: 10.1021/ci3001277. URL `https://doi.org/10.1021/ci3001277`. Publisher: American Chemical Society.

W. Jin, R. Barzilay, and T. Jaakkola. Junction Tree Variational Autoencoder for Molecular Graph Generation. *arXiv:1802.04364 [cs, stat]*, Mar. 2019. URL `http://arxiv.org/abs/1802.04364`. arXiv: 1802.04364.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1945–1954, Sydney, NSW, Australia, Aug. 2017. JMLR.org.

Y. Li. Deep Reinforcement Learning. *arXiv:1810.06339 [cs, stat]*, Oct. 2018. URL `http://arxiv.org/abs/1810.06339`. arXiv: 1810.06339.

X. Lu, J. Gonzalez, Z. Dai, and N. Lawrence. Structured variationally auto-encoded optimization. In *International Conference on Machine Learning*, pages 3267–3275, 2018.

R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu. Neural architecture optimization. In *Advances in neural information processing systems*, pages 7816–7827, 2018.

L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

M. McCloskey and N. J. Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In G. H. Bower, editor, *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press, Jan. 1989. doi: 10.1016/S0079-7421(08)60536-8. URL `http://www.sciencedirect.com/science/article/pii/S0079742108605368`.

A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3387–3395. Curran Associates, Inc., 2016.

M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9(1):48, Sept. 2017. ISSN 1758-2946. doi: 10.1186/s13321-017-0235-x. URL `https://doi.org/10.1186/s13321-017-0235-x`.

M. Popova, O. Isayev, and A. Tropsha. Deep reinforcement learning for de novo drug design. *Science Advances*, 4(7):eaap7885, July 2018. ISSN 2375-2548. doi: 10.1126/sciadv.aap7885. URL `https://advances.sciencemag.org/content/4/7/eaap7885`.

B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

G. N. C. Simm, R. Pinsler, and J. M. Hernández-Lobato. Reinforcement Learning for Molecular Design Guided by Quantum Mechanics. *arXiv:2002.07717 [cs, stat]*, Feb. 2020. URL `http://arxiv.org/abs/2002.07717`. arXiv: 2002.07717.

M. Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

P. J. Van Laarhoven and E. H. Aarts. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer, 1987.

T. White. Sampling Generative Networks. *arXiv:1609.04468 [cs, stat]*, Dec. 2016. URL `http://arxiv.org/abs/1609.04468`. arXiv: 1609.04468.

J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6410–6421. Curran Associates, Inc., 2018.

Z. Zhou, S. Kearnes, L. Li, R. N. Zare, and P. Riley. Optimization of Molecules via Deep Reinforcement Learning. *Scientific Reports*, 9(1):1–10, July 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-47148-x. URL `https://www.nature.com/articles/s41598-019-47148-x`. Number: 1 Publisher: Nature Publishing Group.