Увеличение эффективности подбора гиперпараметров

Валентин А. Абрамов
Факультет вычислительной математики и кибернетики
МГУ имени М. В. Ломоносова

Аннотация

В данной статье предложено улучшение метода дифференциальной эволюции, лежащего в основе DEHB. Алгоритм DEHB жадный – постоянно переиспользует старые значения. Предложено добавлять шум во время мутации для увеличения покрытия пространства гиперпараметров. Были представлены экспериментальные результаты, демонстрирующие эффективность этого метода в сравнении с базовым DEHB и традиционными методами выбора гиперпараметров. Результаты показывают, что изменение мутации может обеспечить более высокую точность модели при неизменном использовании вычислительных ресурсов.

Keywords Оптимизация гиперпараметров · Эволюционный алгоритм

1 Введение

В последние годы машинное обучение стало одной из самых активно развивающихся областей в информационных технологиях. Чтобы достичь высокого качества моделей машинного обучения, необходимо правильно настроить гиперпараметры алгоритмов.

Гиперпараметры являются важными параметрами моделей машинного обучения, которые не могут быть определены в процессе обучения. Традиционные методы выбора гиперпараметров, такие как сеточный поиск Larochelle et al. [2007] или случайный поиск Bergstra and Bengio [2012], могут быть неэффективными или требовать значительных вычислительных ресурсов, поэтому вместо них используют более сложные алгоритмы, например, эволюционные. Одним из таких алгоритмов является DEHB Awad et al. [2021].

2 Обзор литературы

Гиперпараметрическая оптимизация - большая и перспективная область исследований. В статье Bischl et al. [2021] приводится подробный обзор существующих методов, все алгоритмы формально описаны, поднимаются вопросы переобучения гиперпараметров на валидационную выборку и параллелизации алгоритмов. Для глубоких нейронных сетей также необходимо подбирать гиперпараметры, такие как число слоёв, learning rate и даже связи между слоями. О задачах гиперпараметрической оптимизации в нейронных сетях подробно написано в статье Talbi [2020].

Задача автоматического подбора гиперпараметров активно исследуется (Liu et al. [2023], Morales-Hernández et al. [2023]). Ранее использовались методы перебора по сетке (Larochelle et al. [2007]) или случайного поиска значений (Bergstra and Bengio [2012]), но эти методы имеют свои недостатки, такие как случайность, проклятие размерности, сложность подбора сетки, медленная скорость работы, невозможность учесть взаимосвязи между гиперпараметрами и зависимость целевой метрики от них. В 2011 году был создан подход SMBO - последовательная оптимизация, основанная на суррогатной функции (Hutter et al. [2011]). В качестве суррогатной функции изначально брали гауссовские процессы (Wistuba et al. [2018]), но они подходят не для всех задач, так как предназначены только для непрерывных гиперпараметров. Также был придуман алгоритм Succesive Halving (Jamieson and Talwalkar [2015]),

представляющий из себя жадный алгоритм, инкрементально выделяющий больше ресурсов для оценки гиперпараметров с хорошим потенциалом.

На данный момент одним из лучших методов поиска является TPE (Bergstra et al. [2011]), основанный на байесовском подходе. Также существуют методы, основанные на подходе SMBO, такие как SMAC (Hutter et al. [2011], Lindauer et al. [2022]), использующий случайный лес в качестве суррогата, или модели, использующие суррогат на основе гауссовских процессов (Schilling et al. [2016], Wistuba et al. [2018]). Также стоит упомянуть об эволюционных алгоритмах (Young et al. [2015], Alibrahim and Ludwig [2021]), так как они позволяют оптимизировать любые функции за счет интеллектуального перебора значений, основанного на таких идеях эволюции, как мутация, селекция и естественный отбор. Лучше всего себя показывают методы, основанные на алгоритме Succesive Halving – они позволяют найти лучший набор гиперпараметров в условиях ограниченности вычислительных мощностей. На основе SH созданы алгоритмы HyperBand (Li et al. [2018]), автоматически выделяющий ресурсы для нескольких запусков SH, BOHB (Falkner et al. [2018]), добавляющий в HyperBand суррогат на основе TPE, и DEHB Awad et al. [2021], добавляющий в HB идею дифференциальной эволюции.

По текущим бенчмаркам (Vanschoren et al. [2014], Zela et al. [2020]) DEHB является SOTA-алгоритмом подбора гиперпараметров, но у него есть недостатки. Из-за жадного переиспользования оптимальных значений при дифференциальной эволюции, модель застревает в локальном минимуме и целевая метрика выходит на плато. Чтобы этого избежать, необходимо перейти от эксплуатации оптимальных в смысле локального минимума значений и добавить новых кандидатов, отличающихся от эксплуатируемых.

Предлагается добавлять шумовые векторы в эволюционном шаге. Размер шума может зависеть от количества итераций, меняться от выхода на плато по целевой метрике или быть постоянным как гиперпараметр. Таким образом, благодаря шумовым векторам увеличится покрытие поверхности функции ошибки от гиперпараметров.

Для демонстрации результатов работы предложенного метода будут использованы датасеты с сайта open-ml.

3 Постановка задачи

Задача гиперпараметрической оптимизации состоит в том, чтобы имея набор данных $X \in \mathbb{R}^{N \times d}, y \in \mathbb{R}^N$, пространство гиперпараметров $\Theta \in \mathbb{R}^k$, модель $f(\theta) : \mathbb{R}^d \to \mathbb{R},\ \theta \in \Theta,\ f \in \mathbb{H}$ и целевую метрику $c(f, X, y) : \mathbb{H} \times \mathbb{R}^{N \times d} \times \mathbb{R}^N \to \mathbb{R}$, найти θ^* такую, что выполнено

$$\theta^* = \arg\min_{\theta} c(f(\theta), X, y)$$

Таким образом, зная модель f, пространство гиперпараметров Θ, набор данных X, y и целевую метрику c, необходимо найти набор гиперпараметров θ^*, при котором целевая метрика c наименьшая при обучении модели f на данных X, y.

Предполагается, что набор данных состоит из непрерывных переменных, функция c имеет бюджет вычислений. Им может быть ограничение на максимальное число итераций при обучении нейронной сети, количество деревьев в случайном лесе и градиентном бустинге и т. п.

4 Базовый метод

4.1 Differential Evolution

Алгоритм дифференциальной эволюции (Storn and Price [1997]) – эволюционный алгоритм для наборов непрерывных переменных. Поколение g состоит из N кандидатов $x_{i,g} = (x^1_{i,g}, x^2_{i,g}, ..., x^D_{i,g})$. Первое поколение инициализируется кандидатами, сэмплируемыми из равномерного распределения. Для каждого кандидата вычисляется целевая функция f. Далее для каждого кандидата $x_{i,g}$ выбираются 3 случайных кандидата $x_{r_1,g}, x_{r_2,g}, x_{r_3,g}$ и строится вектор-мутант вида $v_{i,g} = x_{r_1,g} + F(x_{r_2,g} - x_{r_3,g})$, F – число-гиперпараметр метода. Далее на основе объектов $x_{i,g}$ и $v_{i,g}$ строится новый кандидат-отпрыск $u_{i,g}$ по следующей схеме: в j-ой координате нового вектора с вероятностью p выбирается значение $x^j_{i,g}$ и с вероятностью $1 - p$ значение $v^j_{i,g}$, p – гиперпараметр. Этот шаг называется кроссовером (или кроссинговером). Далее для всех отпрысков вычисляется целевая функция и в качестве следующей популяции выбираются N лучших (в смысле значений целевой функции) кандидатов из всех $x_{i,g}$ и $u_{i,g}$.

4.2 Succesive Halving

Succesive Halving (Jamieson and Talwalkar [2015]) – алгоритм подбора гиперпараметров, подразумевающий наличие бюджета вычислений. То есть целевую функцию f можно вычислить с фиксированным бюджетом, например, можно ограничить количество итераций вычисления f. У алгоритма есть три гиперпараметра: количество начальных кандидатов N, минимальный бюджет λ и η – параметр скалирования. На первой итерации из равномерного распределения сэмплируются N кандидатов, для них вычисляется функция f с бюджетом λ. На второй итерации берутся N/η лучших кандидатов, для них вычисляется f с бюджетом $\eta\lambda$. На i-ой итерации функция f вычисляется для N/η^{i-1} лучших кандидатов с шага $i-1$ с бюджетом $\eta^{i-1}\lambda$. Таким образом, алгоритм дает больше ресурсов перспективным кандидатам и отсекает тех, кто при низком бюджете показывает себя хуже остальных.

4.3 HyperBand

HyperBand (Li et al. [2018]) расширяет идеи SH – производится несколько запусков SH с различными значениями N и λ. Для каждого нового запуска уменьшается число кандидатов и увеличивается минимальный бюджет. Таким образом, функции f, которые долго сходятся, проверяются для случайных инициализаций и с низким бюджетом, и с высоким. Главным минусов SH и HyperBand является то, что они полностью полагаются на случайное сэмплирование.

4.4 DEHB

DEHB – Differential Evolution HyperBand – развивает идеи HyperBand: сэмплирование из равномерного распределения производится один раз, далее лучшие кандидаты передаются в следующие запуски SH. На каждом шаге SH производится модифицированная дифференциальная эволюция – текущие кандидаты, полученные из предыдущего запуска SH, скрещиваются с мутантами, полученными с помощью сэмплирования лучших кандидатов с предыдущей итерации текущего запуска SH.

5 Предлагаемый метод

У алгоритма DEHB есть недостаток – он использует случайное сэмплирование всего один раз, и далее кандидаты меняются только за счет дифференциальной эволюции. Алгоритм часто застревает в локальном минимуме и выходит на плато по целевой метрике, что может быть связано с жадным переиспользованием кандидатов, лежащих на маленьком расстоянии друг от друга и имеющим близкие значения метрики.

Чтобы этого избежать, предлагается добавлять шумовые векторы к мутантам при эволюции. Благодаря шумовым векторам увеличится покрытие поверхности функции ошибки от гиперпараметров. Это приведет к тому, что некоторые наборы гиперпараметров из популяций будут находиться вне локального минимума. При скрещивании с наборами вне локального минимума DEHB сможет найти новые оптимальные решения.

Таким образом, мутант будет иметь вид $v_{i,g} = x_{r_1,g} + F(x_{r_2,g} - x_{r_3,g}) + m(t)\varepsilon$, где $m(t) : \mathbb{R} \to \mathbb{R}$ – множитель при шумовом векторе. $m(t)$ может зависеть от номера итерации и увеличиваться со временем, может зависеть от выхода на плато, и также может быть константой.

Зададим $m(t)$ следующим образом:

$$m(t) = \begin{cases} b + s, & \text{если модель находится на плато метрики качества} \\ b, & \text{иначе} \end{cases},$$

где b – базовое значение шума, s – шаг шума (параметры метода). Выход на плато определяется отсутствием улучшения качества в течение 10 итераций.

6 Вычислительные эксперименты

6.1 Исходные данные и условия эксперимента

Параметры b, s (базовый уровень шума и шаг шума соответственно) подбираются для каждого эксперимента отдельно по запускам с малым количеством прогонов.

Эксперименты проведены на реализации DEHB из библиотеки auto-ml. Для тестирования метода добавлена стратегия мутации noisy1_bin. В качестве модели для оптимизации гиперпараметров взяты RandomForestClassifier и RandomForestRegressor из библиотеки scikit-learn.

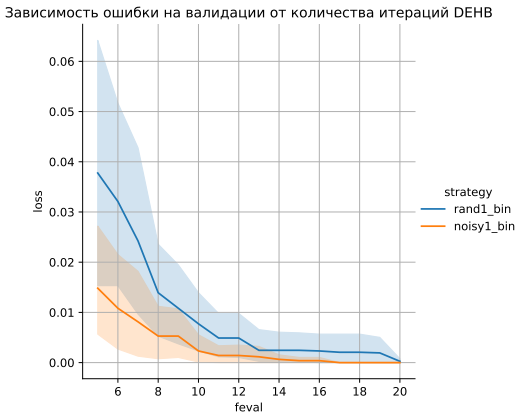
При тестировании метода с зашумлением, если не оговорено иное, шум равен 0, пока функция потерь не выйдет на плато (последние 10 итераций не будет улучшения значения функции потерь).

Метод тестируется на данных о недвижимости California Housing из библиотеки scikit-learn и наборах данных с сайта open-ml: credit-g, steel plates fault, scene.

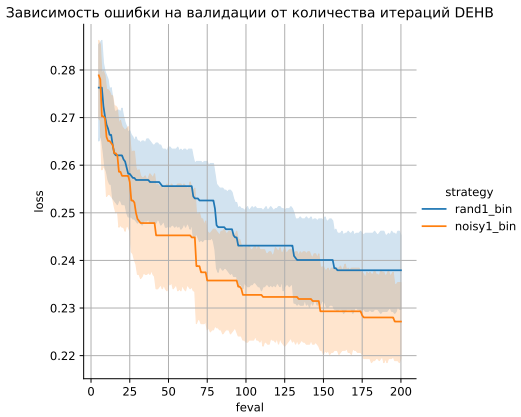
Наборы данных разбивались на обучающую, тестовую и валидационную выборку в соотношении 7:2:1. В задачах регрессии использовалась функция ошибки MSE, в задачах классификации 1 – accuracy (DEHB минимизирует).

Для каждого датасета выполнено 20 запусков DEHB со стратегией мутации rand1_bin и 20 запусков с новой стратегией. На графиках линиями изображены средние значения функции ошибки на валидационной выборке для итерации обучения на наборе гиперпараметров. Прозрачные области поверх линий - 95% доверительный интервал значений ошибки.

6.2 Эксперименты на наборах данных



Steel plates fault

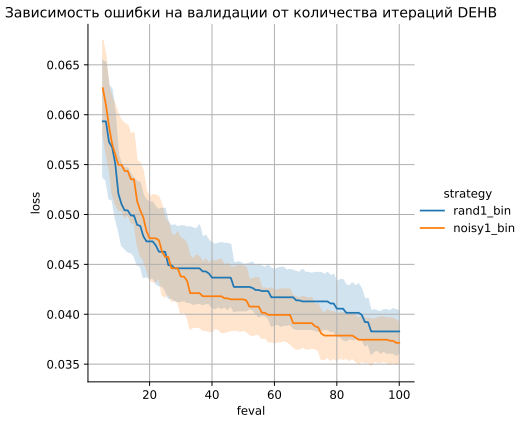


Liver patients

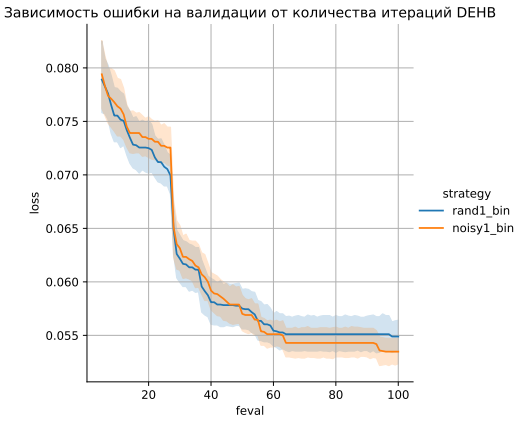
Таблица 1

В задачах детекции брака стальных пластин и в задаче детекции пациентов с заболеваниями печени предложенный метод стабильно предсказывает лучшие наборы гиперпараметров, чем базовая стратегия мутации. Кроме того, даже несмотря на усреднение ошибок по различным запускам, невооружённым взглядом видно, что базовый метод без зашумления дольше выходит с каждого плато (“ступеньки” на синем графике длиннее).

При тестировании метода на задаче предсказания городской среды по эмбеддингам картинки и задаче детекции спама метод показывает менее стабильный прирост качества, однако можно заметить, что предложенный метод действительно выходит с плато быстрее классического.



Scene detection

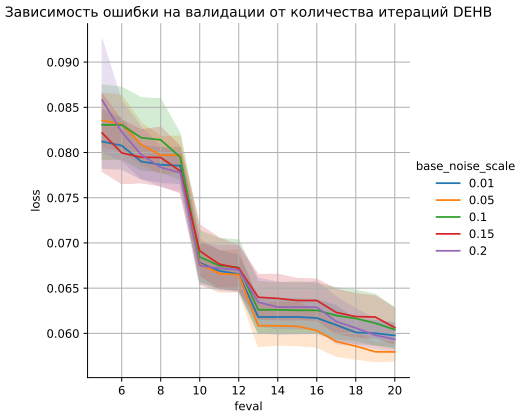


Spam detection

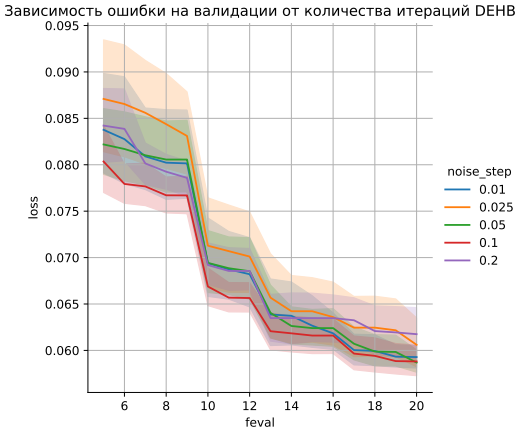
Таблица 2

6.3 Анализ ошибки

Используем датасет с данными о недвижимости для анализа зависимости ошибки метода от его гиперпараметров.



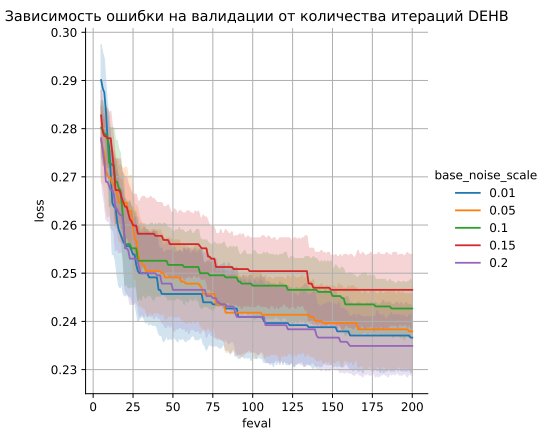
Зависимость от базового шума



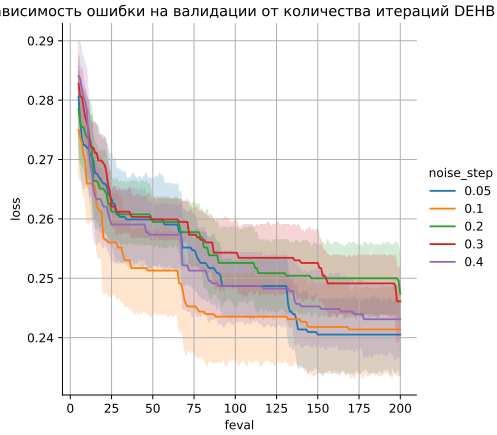
Зависимость от величины шага

Таблица 3

При рассмотрении зависимости от базового шума шаг равнялся 0. Видим, что на малом числе итераций нет значимой зависимости ошибки от параметров метода. Рассмотрим ошибку на задаче детекции пациентов с заболеваниями печени.



Зависимость от базового шума



Зависимость от величины шага

Таблица 4

Видим, что явной монотонной зависимости от базового уровня шума нет. При этом на больших уровнях базовго шума модель даже дольше находится на плато, так как новые конфигурации становятся слишком случайными и поиск идет в случайных областях вместо окрестности лучших наборов гиперпараметров.

При рассмотрении зависимости ошибки от величины шага видно, что слишком маленькие и слишком большие величины шага оказывают негативное влияние на ошибку на валидации. Однако видно, что при любых значениях метод позволяет выйти с плато и прийти к новым минимумам функции ошибок.

Также стоит заметить, что параметры метода стоит подбирать в зависимости от размерности пространства гиперпараметров, так как шум одной и той же величины с увеличением размерности будет вносить больший вклад в мутацию.

7 Выводы

Таким образом, предложенный метод показывает улучшение на некоторых датасетах. Добавление шума при выходе на плато позволяет получить новых кандидатов и продолжить подбор гиперпараметров с помощью дифференциальной эволюции.

Главной особенностью нового метода является гиперпараметр $m(t)$. Если взять его равным 0, то метод сводится к обычному DEHB. От выбора функции $m(t)$ зависит соотношение exploration/exploitation. При большим уровнях шума мутанты получаются полностью случайными, а при низких – слишком близкими к кандидатам. Это одновременно является плюсом и минусом метода – при удачно подобранном $m(t)$ алгоритм сможет быстро уходить с плато, а при неудачном – покажет себя хуже классического DEHB.

Дальнейшими направлениями исследования являются изучение возможности добавления сэмплирования кандидатов из неисследованных участков пространства гиперпараметров, как это происходит в SMBO. Также можно изменить функции мутации – например, сдвигать мутанта в направлении увеличения качества целевой метрики. Функцию кроссовера также можно сделать более сложной – не выбирать одно из двух значений, а брать промежуточное.

Список литературы

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In Proceedings of the 24th International Conference on Machine Learning, ICML '07, page 473–480, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi:10.1145/1273496.1273556. URL https://doi.org/10.1145/1273496.1273556.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(10):281–305, 2012. URL http://jmlr.org/papers/v13/bergstra12a.html.

Noor Awad, Neeratyoy Mallik, and Frank Hutter. Dehb: Evolutionary hyperband for scalable, robust and efficient hyperparameter optimization, 2021.

Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, Difan Deng, and Marius Lindauer. Hyperparameter optimization: Foundations, algorithms, best practices and open challenges, 2021.

El-Ghazali Talbi. Optimization of deep neural networks: a survey and unified taxonomy. 2020. URL https://api.semanticscholar.org/CorpusID:219427357.

Yaoyao Liu, Yingying Li, Bernt Schiele, and Qianru Sun. Online hyperparameter optimization for class-incremental learning. Proceedings of the AAAI Conference on Artificial Intelligence, 37(7):8906–8913, June 2023. ISSN 2159-5399. doi:10.1609/aaai.v37i7.26070. URL http://dx.doi.org/10.1609/aaai.v37i7.26070.

Alejandro Morales-Hernández, Inneke Van Nieuwenhuyse, and Sebastian Rojas Gonzalez. A survey on multi-objective hyperparameter optimization algorithms for machine learning. Artificial Intelligence Review, 56(8):8043–8093, 2023.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, Learning and Intelligent Optimization, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25566-3.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. Machine Learning, 107(1):43–78, 2018.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization, 2015.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. Smac3: A versatile bayesian optimization package for hyperparameter optimization. The Journal of Machine Learning Research, 23(1):2475–2483, 2022.

Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Scalable hyperparameter optimization with products of gaussian process experts. In Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16, pages 33–48. Springer, 2016.

Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In Proceedings of the workshop on machine learning in high-performance computing environments, pages 1–5, 2015.

Hussain Alibrahim and Simone A Ludwig. Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization. In 2021 IEEE Congress on Evolutionary Computation (CEC), pages 1551–1559. IEEE, 2021.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization, 2018.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale, 2018.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. ACM SIGKDD Explorations Newsletter, 15(2):49–60, 2014.

Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. arXiv preprint arXiv:2001.10422, 2020.

Rainer Storn and Kenneth Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11:341–359, 01 1997. doi:10.1023/A:1008202821328.