BBOmix: A Tabular Benchmark for Hyperparameter Optimization of Unsupervised Biological Representation Learning

Luca Thale-Bombien[1] **Jan Ewald**[1] **Ralf König**[1] **Aaron Klein**[1,2]

[1]Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden/Leipzig, Leipzig University
[2]ELLIS Institute Tübingen

Abstract The rapid advancement of high-throughput sequencing has led to large, high-dimensional omics datasets. Deep unsupervised learning architectures, particularly Autoencoders (AEs), are increasingly used for dimensionality reduction and representation learning in this domain. However, AEs are highly sensitive to architectural choices and hyperparameters, and unsupervised optimization typically relies on reconstruction loss, which may be a poor proxy for downstream utility. Exhaustive hyperparameter optimization (HPO) is computationally expensive, leading researchers to frequently rely on suboptimal default configurations. To democratize access to large-scale unsupervised HPO research, we introduce **BBOmix**, the first open-source tabular benchmark for unsupervised representation learning on real-world biological data. Our benchmark includes 105,000 evaluations across four AE architectures and seven multi-omics modalities from the TCGA and SCHC datasets. We quantify the correlation between reconstruction loss and downstream task performance and provide an extensive evaluation of state-of-the-art single-fidelity, multi-fidelity, and transfer learning HPO methods, establishing a rigorous baseline for future research in unsupervised biological representation learning.[1]

1 Introduction

The rapid advancements in high-throughput sequencing technologies have caused a paradigm shift in the life sciences, leading to large, high-dimensional datasets (Mardis, 2008; Reuter et al., 2015). This data, often referred to as *omics* data, refers to a family of large-scale biological disciplines, like genomics or transcriptomics, each comprehensively characterizing a specific class of biological molecules and different layers of cellular processes. Each modality reflects the molecular state of biological systems at multiple levels of resolution e.g. tissue or single-cell (see Figure 1A).

Given the inherent noise and sparsity of this *omics* data, deep unsupervised learning architectures, especially Autoencoders (AE) have shown strong capabilities for dimensionality reduction, denoising and representation learning, which is a cornerstone for functional analysis in cell biology and identification of biomarkers, enabling modern personalized medicine (Kopf and Claassen, 2021; Selby et al., 2025). Despite their use for handling *omics* data, they are notoriously fragile and highly sensitive to architectural choices (e.g. depth, latent space dimensionality) and hyperparameter configurations (e.g. learning rate, weight decay), hindering the establishment of principled and standardized approaches (Joas et al., 2025).

Traditionally, hyperparameter optimization (HPO) is set in supervised learning settings, where the configuration choice is reflected directly in an objective metric, such as validation accuracy. In contrast, unsupervised learning typically optimizes the reconstruction loss, which can be a poor

[1]Code and Data available on GitHub and Huggingface: `https://github.com/Kavlahkaff/BBOmix`

 © 2026 the authors, released under CC BY 4.0

proxy for the actual usefulness and quality of a learned latent space for further downstream applications such as cell-type classification or disease prognosis. Exhaustive HPO on large-scale omics data is computationally expensive and thus researchers frequently rely on default hyperparameter choices or manual tuning based on previous experiences, resulting in suboptimal representations and potentially skewed downstream analysis.

In recent years, the machine learning community has developed numerous benchmarks for HPO and NAS on supervised classification tasks, including NAS-Bench-201 (Dong and Yang, 2020), FCNet (Klein and Hutter, 2019), and TabRepo (Salinas and Erickson, 2023), yet no equivalent exists for unsupervised representation learning and downstream task evaluation on biological data. While recent efforts such as AUTOENCODIX (Joas et al., 2025) and scSSL-Bench (Ovcharenko et al., 2025) have established standardized frameworks for evaluating AE architectures and self-supervised methods on omics data, they do not provide the exhaustive, precomputed configuration evaluations needed to benchmark HPO algorithms themselves. Beyond this, the field has historically relied on synthetic data or computer vision datasets, such as MNIST and CIFAR-10 that fail to capture the high-dimensional noise, non-random sparsity, and complex regulatory patterns inherent in biological systems (Whalen et al., 2022).

We address both limitations with **BBOmix** (Black-Box Optimization for Omics Data), the first open-source tabular HPO benchmark for unsupervised representation learning on omics data, comprising exhaustive evaluations across multiple AE architectures and omics modalities to democratize access to large-scale unsupervised HPO research in biology. Specifically our contributions are threefold:

- As our main contribution, we publicly release the first large-scale open-source dataset for HPO in an unsupervised omics setting, consisting of 105.000 evaluations across 35 tasks from 2 datasets.
- We systematically quantify the correlation between reconstruction loss and downstream task performance, providing insights into trade-offs that can function as a guideline for further research.
- Finally, we provide an extensive evaluation of state-of-the-art (SOTA) optimization methods, including single-fidelity, multi-fidelity and transfer learning approaches, thereby establishing a baseline for HPO performance in an unsupervised setting. Transfer Learning especially allows us to investigate how effective meta-learning works between the different architectures.

2 Related Work

Hyperparameter Optimization. Hyperparameter optimization (HPO) (Franceschi et al., 2025) seeks the hyperparameter configuration $\mathbf{x}_\star \in \arg\min_{\mathbf{x}\in\mathbb{X}} f(\mathbf{x})$ of a machine learning algorithm that optimizes the validation performance $f(\mathbf{x})$. In practice (Bischl et al., 2023), the objective function f is typically expensive to evaluate, has unknown closed-form solution, and can only be observed with noise, i.e., $y \sim f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma)$.

A widely used approach for HPO is Bayesian optimization (BO) (Garnett, 2023), which fits a probabilistic surrogate model, such as a Gaussian process (Snoek et al., 2012) or a tree-structured Parzen estimator (Bergstra et al., 2011), to approximate the objective function $f(\mathbf{x})$ and uses an acquisition function to balance exploration and exploitation. In many applications, inexpensive approximations of f are available, e.g., by training for fewer epochs or on subsets of the data. Multi-fidelity methods exploit these approximations to discard poor configurations early and reduce computational cost. For example, successive halving (Jamieson and Talwalkar, 2016) iteratively allocates increasing budgets to the better-performing half of configurations. Methods such as Hyperband (Li et al., 2018) and BOHB (Falkner et al., 2018) combine random or BO-guided sampling with successive halving and often outperform standard BO under limited compute budgets. This is especially important in deep learning, where a single full evaluation may require hours of GPU time.

Our benchmark records per-epoch reconstruction losses for all 105,000 runs, making it directly compatible with multi-fidelity optimization methods.

Transfer Learning. To reduce the cold-start cost of HPO, which typically begins from random or default hyperparameter configurations, transfer learning approaches initialize the search using knowledge from prior runs on related tasks (Feurer et al., 2015). For example, Wistuba et al. (2015a) employ a warm-starting strategy that reuses the most successful configurations from previous tasks as initial candidates. Another approach is to constrain the search space to a smaller region containing high-performing configurations identified during earlier runs (Perrone et al., 2019). In supervised learning settings, transfer HPO has demonstrated consistent improvements on tabular benchmarks such as TabRepo (Salinas and Erickson, 2023). However, to the best of our knowledge, there has been no systematic evaluation of whether similar gains can be achieved in unsupervised settings using omics data, where task similarity is more difficult to define and downstream metrics are not observed during training.

HPO Benchmarks. Reproducible benchmarking is central to AutoML research. To reduce the computational cost of HPO evaluation, tabular benchmarks decouple optimization from model training by exhaustively evaluating configurations offline and using table lookups during benchmarking. For continuous or high-dimensional search spaces, surrogate models can instead predict the performance of unseen configurations. Representative benchmarks include NAS-Bench-201 (Dong and Yang, 2020), FCNet (Klein and Hutter, 2019), LCBench (Zimmer et al., 2021), and TabRepo (Salinas and Erickson, 2023). A key limitation of existing HPO benchmarks is their supervised setting, where the optimization objective directly matches the downstream task. In unsupervised representation learning however, models are typically optimized for reconstruction loss, while downstream utility is not observed during training. As shown in Section 4.1, reconstruction loss does not necessarily correlate with downstream performance, and optimal configurations may differ substantially. Existing unsupervised benchmarks do not address this mismatch. Prior work has largely focused on synthetic or computer vision datasets (Locatello et al., 2019), which do not capture key properties of biological omics data, such as extreme sparsity, modality-specific noise, high feature-to-sample ratios, and biologically structured covariance. To our knowledge, **BBOmix** is the first benchmark for exhaustive HPO evaluation on real-world biological datasets in the unsupervised setting. Although HPO is performed using reconstruction loss, we additionally record downstream supervised metrics for every configuration, enabling post-hoc analysis of the relationship between unsupervised objectives and biological utility.

Autoencoder for biological representation learning. Advancements of AE based representation learning were developed simultaneous to the advent of high-throughput methods where individual layers, including DNA (genomics), RNA (transcriptomics) or proteins (proteomics), are measured. Additionally, techniques enabling single-cell resolution vastly increased the necessity of new methods for representation learning (Mamoshina et al., 2016). In the past, numerous AE architectures have been developed. Firstly, variational AE (VAE) enable generative modelling for synthetic data. More complex architectures, e.g stacked VAEs (or hierachical) emerged for multi-omics integration (Simidjievski et al., 2019) or cross-modal VAE for modality translation (Yang et al., 2021). Recent work incorporates prior-knowledge into the decoder structure to enforce latent space explainability and improve regularization. These biological-informed neural AEs, such as VEGA (Seninge et al., 2021), expiMAP (Lotfollahi et al., 2023b) or OntoVAE (Doncevic and Herrmann, 2023) rely on an hierarchical ontology describing the feature-to-latent relationship representing gene-to-function knowledge. Other applications are data imputation or denoising with AE which is increasingly important as single-cell resolution data contain large number of drop-out events (Eraslan et al., 2019). Furthermore, recent AEs where covariates are disentangled via parallel training of individual discriminator networks show new directions (Lotfollahi et al., 2023a). Despite this, the field

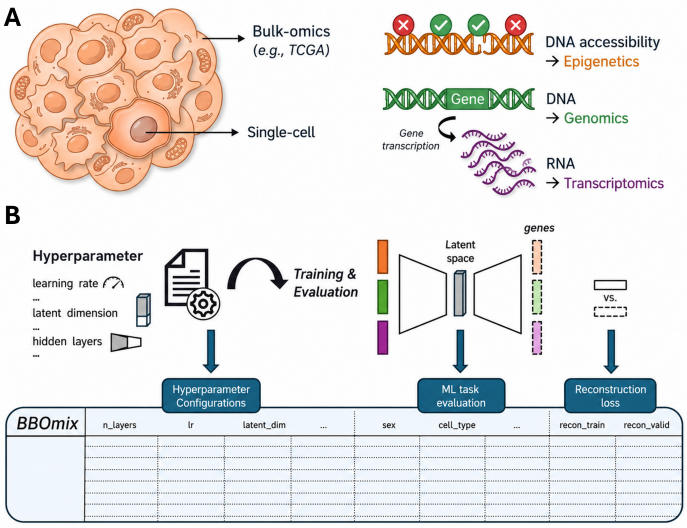


Figure 1: Overview of **BBOmix**. (**A**) Multi-omics data modalities considered, spanning bulk (e.g., TCGA) and single-cell measurements, and covering genomic, epigenetic, and transcriptomic layers. (**B**) Construction pipeline: sampled hyperparameter configurations are used to train and evaluate each autoencoder architecture; the resulting reconstruction loss, downstream ML task performance, and hyperparameter configuration are recorded for every run and aggregated into the tabular **BBOmix** benchmark.

lacks standardization. Architectural comparisons are typically conducted on a single dataset with manually selected hyperparameters, making it unclear whether performance differences reflect architectural merit or hyperparameter sensitivity. Hu and Greene (2018) established evidence that hyperparameter choices dominate outcomes in VAEs for single-cell transcriptomics stressing the necessity of rigorous HPO. More recently, the standardization issue was addressed by the development of the framework AUTOENCODIX for common AE architectures enabling HPO benchmarking (Joas et al., 2025).

3 Methodology

Leveraging AUTOENCODIX, we collected **BBOmix** as a large-scale dataset of hyperparameter configurations for four AE architectures trained on real-world multi-omics data. Figure 1 gives an overview of the data modalities and the benchmark construction pipeline. In the following, we describe the used datasets, selected AEs and outline the procedure to construct **BBOmix**.

3.1 Training Datasets

We used two established multi-omics datasets, the Cancer Genome Atlas (TCGA) (Weinstein et al., 2013) and the Single-Cell Human Cortex (SCHC) dataset (Zhu et al., 2023), which we will briefly describe in this section. Both datasets were retrieved as pre-processed including mutation score calculation as described by Joas et al. (2025); Ewald (2025). All modalities were processed in an standardized way using YAML-files determining feature selection, data scaling and random splitting into train, test and validation splits. We train and evaluate our models on each of the available modalities separately, as well as a combination of all modalities for each dataset.

The Cancer Genome Atlas The TCGA pan-cancer multi-omics dataset comprises over 10,000 human patient samples and clinical annotations for 32 cancer types including survival. The dataset contains three data modalities: transcriptomic data (RSEM normalized bulk-cell mRNA-seq), epigenetic data (DNA methylation), genomic data (a per-gene mutation score). The clinical annotations were used

as labels in downstream classification tasks (see Appendix A).
Single-Cell Human Cortex The SCHC dataset contains single-cell multi-omics measurements from the developing human cerebral cortex. The dataset has two data modalities and clinical annotations: single-cell RNA sequencing (scRNA-seq), chromatin accessibility profiling (scATAC-seq) and associated cell-level annotations. In total, the dataset contains over 45.000 rows (cells), with the annotations: cell-type, age group and sex.

3.2 Different Architectures

We consider five AE architectures spanning a range from standard AEs to biologically informed encoders. All architectures share a common backbone of fully connected layers with batch normalization, dropout and ReLU activation, differing in their latent space and decoder structure. Encoder and decoder are symmetric in depth, with hidden layer widths determined by an encoding factor applied successively from the input dimension down to the latent bottleneck. Our selection is based on the consideration to include in BBOmix in a first step matured and single network architectures to reduce complexity and enable direct cross comparison of HPO results. Beyond the current study, HPO insights from BBOmix for single architectures will serve as baseline for investigations of composed AEs like cross-modal VAEs and stacked VAEs which are formed by standard VAEs. We consider the following architectures:
Vanillix: Standard fully-connected feed-forward AE with batch normalization, dropout, and ReLU. Encoder and decoder are symmetric and we minimize MSE reconstruction loss for training.
Varix: A standard $\beta-$Variational Auto-Encoder (VAE) (Higgins et al., 2017) which follows the same architecture as Vanillix. We minimize the standard ELBO loss formulation: $Total_loss = reconstruction_loss + \beta \times KL_divergence$, where β is an additional hyperparameter.
Disentanglix: Extends VAE for higher latent space disentanglement by a decomposition of the KL-divergence into three terms to enable a higher penalty of the total-correlation. Implementation in AUTOENCODIX is based on the β-TCVAE of (Chen et al., 2018) with three different weights for the loss: total correlation β_{tc}, mutual information β_{mi}, and per-dimension KL weight β_{DimKL}.
Ontix: A biologically-informed VAE where the decoder is constrained by an ontology (pathways or chromosome-based). The decoder represents known gene-to-pathway (Reactome (Milacic et al., 2024)) or gene-positional (Chromosome) relationships. Both ontologies differ in decoder sparseness, since genes act in multiple pathways, but gene location is unique. We analyze both independently and treat them as two different architectures.

3.3 Downstream Evaluation

Beyond reconstruction quality, we assess the utility of the learned latent spaces by evaluating on a suite of downstream tasks applied to the frozen embeddings of the trained encoders. Following the protocol of Joas et al. (2025), our downstream classification tasks are derived from available clinical annotations (see Appendix A). We fit a logistic regression for classification on the embeddings and report the area under the receiver operating curve (AUC-ROC) in a one-versus-one scheme. At the end, the final downstream performance is aggregated as an average over all downstream tasks and reported alongside the individual scores.

3.4 Data Collection

Table 1 shows the distinct search spaces for each architecture, reflecting the differences in model structure and inductive bias. The search spaces contain diverse parameters, ranging from low to high-dimensional, discrete, categorical, and numerical values. For each architecture, we sampled 1000 configurations uniformly at random over the respective search space, yielding 5,000 unique architecture-configuration pairs in total. We trained each configuration for 300 epochs on each of the seven dataset-modality tasks, and record reconstruction loss at every epoch, to produce full

learning curves. To account for stochastic variation stemming from randomness in the AUTOENCODIX pipeline, we ran each configuration with three independent random seeds. The complete experimental design therefore yields:
#architectures × #configurations × #seeds × #modalities = 105,000 individual training runs.
We trained each network on Nvidia H100 GPUs, amounting to ≈10k GPU hours in total. Per-epoch loss curves are retained for all runs, enabling the benchmark to support both final-performance evaluations and learning-curve-based HPO methods, such as multi-fidelity and early-stopping approaches. The resulting dataset contains for each configuration the reconstruction loss, training runtime and accuracy on downstream tasks. We used the tabular blackbox format of the Syne Tune library (Salinas et al., 2022) to serialize the dataset. Each architecture-modality pair defines a separate blackbox, containing the full configuration to performance mapping as a queryable lookup table with dimensions for hyperparameter configurations, training fidelities and random seeds yielding 35 blackboxes in total. This format allows any HPO optimizer supported by Syne Tune to be evaluated against the benchmark, facilitating the use of surrogates to predict objective values of any configurations in the search space (Eggensperger et al., 2015). Similar to Salinas et al. (2022) we use a simple k-nearest neighbor regressor with $k = 1$ for our benchmarks.

Table 1: Hyperparameter search spaces for each AE architecture. $\mathcal{U}$ denotes a uniform distribution, $\mathcal{LU}$ a log-uniform distribution, and $\mathcal{C}$ a categorical choice.

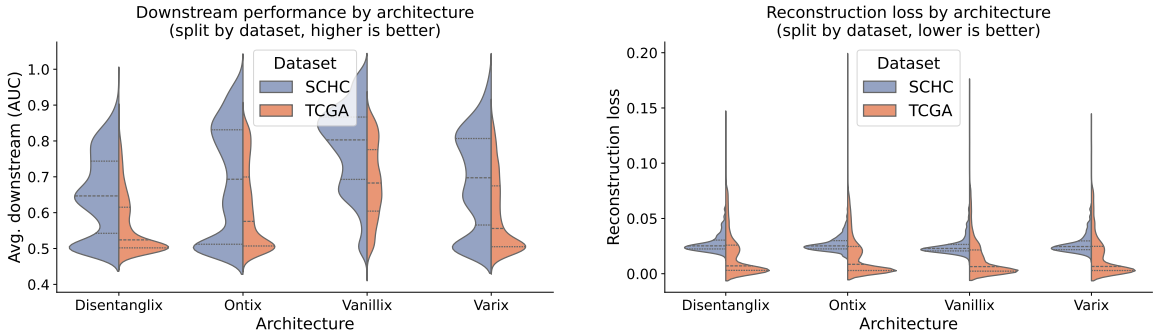
Parameter	**Symbol**	**Type**	**Range or Choices**			
			VANILLIX	VARIX	DISENTANGLIX	ONTIX
Shared architectural hyperparameters						
Input filter size	D	$\mathcal{C}$		$\{128, 256, 512, 1024, 2048, 4096\}$		
No. encoder layers	n_{layers}	$\mathcal{C}$		$\{2, 3, 4\}$		
Encoding factor	r	$\mathcal{U}$		$(1, 4)$		
Latent dimensionality	d_z	$\mathcal{C}$		$\{2, 4, 8, 16, 32, 64\}$		–
Shared training hyperparameters						
Learning rate	η	$\mathcal{LU}$		$(10^{-5}, 10^{-1})$		
Batch size	b_s	$\mathcal{C}$		$\{32, 64, 128, 256\}$		
Dropout probability	p_{drop}	$\mathcal{U}$		$(0, 0.9)$		
Weight decay	λ	$\mathcal{LU}$		$(10^{-5}, 10^{-1})$		
Architecture-specific hyperparameters						
KL divergence weight	β	$\mathcal{LU}$	–	$(10^{-4}, 10^{0})$	–	$(10^{-4}, 10^{0})$
Disentanglement	β_{tc}	$\mathcal{LU}$	–	–	$(10^{-2}, 10^{3})$	–
Disentanglement	β_{mi}	$\mathcal{LU}$	–	–	$(10^{-4}, 10^{0})$	–
Disentanglement	β_{DimKL}	$\mathcal{LU}$	–	–	$(10^{-2}, 10^{0})$	–

4 Experiment Section

We structure our experimental analysis around two questions: (1) how hyperparameter choices and their interactions shape model performance across architectures and tasks, and (2) how well different HPO optimizers exploit the benchmark's structure to find good configurations efficiently.

4.1 Reconstruction Loss as a Proxy for Downstream Performance

A practical motivation for using reconstruction loss as a cheap HPO criterion is that it is available during training, while downstream evaluation requires a separate supervised pipeline evaluating



(a) Downstream performance

(b) Reconstruction Loss

Figure 2: Violin plots of architecture performance with respect to dataset and performance metrics.

multiple tasks after training. If reconstruction loss were a reliable proxy, one could safely optimize it instead of additionally computing the true objective, drastically reducing the cost of HPO.

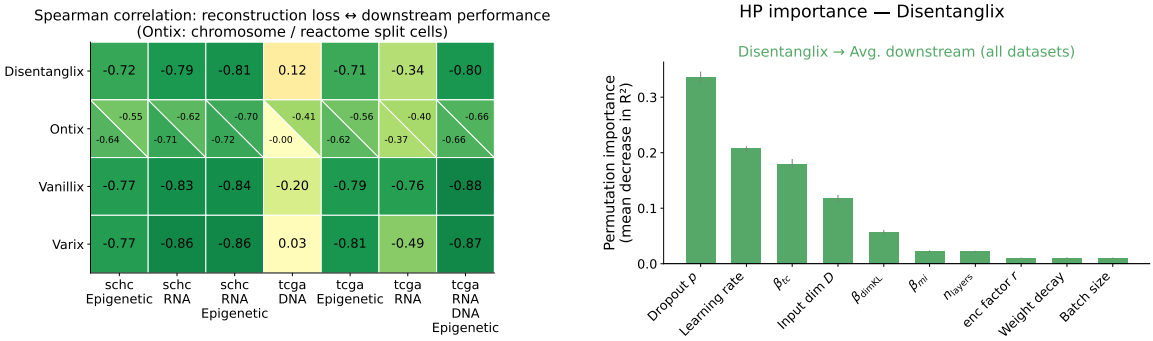
To assess whether reconstruction loss is a reliable proxy for downstream performance, we first examine the distributions of reconstruction loss and downstream performance across hyperparameter configurations. Figure 2a and Figure 2b show violin plots of downstream performance and reconstruction loss, respectively, for each architecture on the SCHC and TCGA dataset, illustrating the spread and variability induced by hyperparameter choices within each architecture. For a breakdown per downstream task see Appendix D.

We compute the Spearman rank correlation (ρ) between the reconstruction loss (to be minimized), and downstream task performance (to be maximized) for each (architecture, task) pair in Figure 3a, with Ontix cells split diagonally for its two ontology variants (chromosome / reactome). A strong negative correlation ($\rho \approx -1$) indicates that lower reconstruction loss consistently predicts better downstream performance, validating its use as a proxy during HPO.

As shown in Figure 3b, reconstruction loss is a strong, but not perfect, predictor for downstream task performance across most architectures and modalities, except DNA (genomics). Disentanglix, Vanillix, and Varix exhibit consistently strongest correlations (ρ between -0.72 and -0.88), with the sole exception of the *tcga_DNA* modality, where correlations are near zero or weakly positive. Ontix shows a similar pattern but with somewhat attenuated magnitudes (ρ ranging from -0.37 to -0.72), consistent across both the chromosome and reactome ontology splits, again with *tcga_DNA* as the primary outlier ($\rho \approx -0.00$ / -0.41). Lower reconstruction loss consistently ranks hyperparameter configurations closer to those achieving higher downstream performance, supporting its use as a low-cost HPO criterion. The notable exception is the `tcga_DNA` setting, where the proxy relationship breaks down across all architectures, suggesting that DNA methylation combined with clinical covariates in the TCGA cohort poses a qualitatively different optimisation landscape. Practitioners should therefore apply reconstruction-loss-based HPO with caution in this specific setting, while treating it as a broadly valid selection criterion elsewhere.

4.2 Hyperparameter Importance

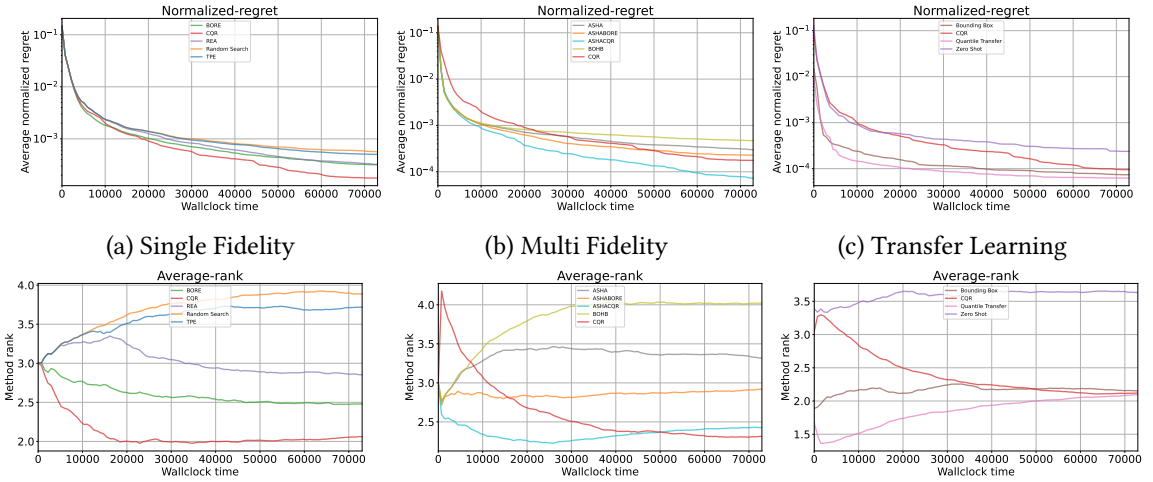
To identify which hyperparameters most strongly influence final performance, we train a random forest regressor on all hyperparameter configurations to predict each target and compute permutation importance scores, i.e. the mean decrease in out-of-sample R^2 when a single HP column is randomly shuffled (Fisher et al., 2019). We report results separately for (i) aggregate downstream performance, (ii) reconstruction loss, and (iii) each individual downstream target, yielding a comprehensive picture of which HPs matter and for which objective. Figure 3b shows the permutation importance scores exemplary for Disentanglix on average downstream task performance (see Appendix B for more results).



(a) Spearman rank correlation of reconstruction loss and downstream performance.

(b) HP importance for Disentanglix downstream task performance.

Figure 3: Hyperparameter importance and proxy metric correlation.



(a) Single Fidelity

(b) Multi Fidelity

(c) Transfer Learning

(d) Single Fidelity

(e) Multi Fidelity

(f) Transfer Learning

Figure 4: Normalized regret (a-c) and average rank (d-f) of single-fidelity, multi-fidelity and transfer-learning optimization algorithms, averaged over all architectures and tasks.

Across all architectures, **dropout** p is the single most influential hyperparameter for downstream predictive performance: it ranks first for Vanillix, Varix, and Disentanglix, and second only to learning rate for Ontix. **Learning rate** is the dominant HP for Ontix on both SCHC and TCGA, while playing a consistently strong secondary role elsewhere. **Input dimensionality** D ranks among the top three for reconstruction loss across all architectures, reflecting its direct effect on the AE's compression bottleneck, but is less decisive for downstream classification and survival tasks. Architecture-specific regularisation terms, like β for Varix and Ontix, and β_{tc} for Disentanglix, carry moderate importance for downstream performance but are largely negligible for reconstruction loss, suggesting they shape the latent geometry without strongly affecting reconstruction fidelity. Structural HPs such as n_{layers} and encoder factor r or training HPs batch size, and weight decay contribute only minor importance across objectives and architectures.

4.3 Comparison of Optimizers

We benchmark methods across three paradigms and compare them on all 35 different blackboxes (see Figure 4) minimizing reconstruction loss. See Appendix C for full descriptions of all methods. All methods use the default hyperparameters suggested by Syne Tune (Salinas et al., 2022), with

the exception of REA, where the population size was reduced to avoid sampling too many random configurations. Each method had a simulated wall-clock time budget of 72000 seconds. We repeated each optimization run 30 times with a different random seed.

To make performance comparable across tasks with heterogeneous loss scales, we report results in terms of normalized regret. For task τ, let $y^{\tau}_{m,s,t}$ denote the objective value attained by method m under seed s at wall-clock time t, and let $y^{\tau}_{\min}$ and $y^{\tau}_{\max}$ denote the smallest and largest objective values observed across all methods, seeds, and time steps on task τ. The normalized regret is then defined as

$$r^{\tau}_{m,s,t} = \frac{y^{\tau}_{m,s,t} - y^{\tau}_{\min}}{y^{\tau}_{\max} - y^{\tau}_{\min}} \in [0, 1]. \tag{1}$$

Single-fidelity methods exhaust the full budget for each configuration. We include Random Search (Bergstra and Bengio, 2012) as an unbiased baseline, TPE (Bergstra et al., 2011), BORE (Tiao et al., 2020), REA (Real et al., 2019), and CQR (Salinas et al., 2023). Among these (Figure 4a and 4d), CQR achieves the lowest normalized regret and ranks most favorably across the budget range, achieving best performance across the board. BORE and REA perform slightly worse than CQR, but both clearly outperform TPE and Random Search throughout the optimization trajectory. Random Search, while informative as an unbiased baseline, falls behind as budget increases and structured methods are able to exploit the response surface more effectively. TPE occupies an intermediate position, offering modest but inconsistent gains over random search. The rank plots (Figure 4d) confirm the ordering observed in the regret curves. The optimization trajectory plots for individual tasks in Appendix C show that all methods converge to similar final performance values given sufficient budget, but differ substantially in their early-budget behavior.

Multi-fidelity methods exploit the full per-epoch learning curves available in the benchmark to enable early stopping. We evaluate ASHA (Li et al., 2020), ASHABORE, ASHACQR, and BOHB (Falkner et al., 2018). These methods (Figure 4b and 4e) discard unpromising configurations early, achieving comparable final performance to single-fidelity methods with substantially less wall-clock time. ASHA and ASHABORE reach low normalized regret levels with one to two orders of magnitude less budget consumed relative to their single-fidelity counterparts. BOHB performs competitively at early budgets but shows higher variance across tasks compared to ASHA-based methods. ASHACQR combines ASHA's aggressive early stopping with CQR's acquisition and achieves strong anytime performance, mirroring the advantage of CQR observed in the single-fidelity setting. The rank plots (Figure 4e) show ASHABORE and ASHACQR in the top rank across the budget range, with ASHA performing well at very low budgets where its successive halving strategy most aggressively filters configurations. CQR without multi-fidelity scheduling serves as a reference and is consistently outperformed by all multi-fidelity methods at equivalent wall-clock budgets, confirming the value of exploiting the learning curve structure available in the benchmark. Still, at the end, CQR is able to catch up to the other methods and in some cases even achieves higher end-performance.

Transfer learning methods warm-start search from related tasks. We use BoundingBox (Perrone et al., 2019), ZeroShot (Wistuba et al., 2015b), and Quantile Transfer (Salinas et al., 2020). These methods (Figure 4c and 4f) initialize the search using observations from related source tasks, reducing the cold-start cost compared to methods that start from scratch. For each tested target task, all remaining tasks spanning every architecture, modality, and dataset in our benchmark serve as source tasks, and the plot shows the aggregated results across all such target tasks. BoundingBox and Quantile Transfer achieve substantially lower regret compared to CQR, though performance converges around the same end-performance. We can observe gains from transfer learning across architectures, suggesting that the learned HP importance structure is partially portable: the ranking of hyperparameters is stable enough across different biological data types and model families for source task experience to translate meaningfully to new target settings. This is consistent with the permutation importance analysis (Section 4.2), which revealed consistent dominance of learning rate across architectures and datasets, providing a stable signal for transfer. ZeroShot evaluates the

single best-predicted configuration from a historical portfolio without performing any target-task evaluations. While it provides a strong immediate starting point at virtually zero cost, it lacks the ability to adapt dynamically to the target task's specific loss landscape over time. Consequently, it is quickly outperformed by methods like BoundingBox and Quantile Transfer, which actively update their surrogates based on online evaluations.

5 Conclusions

We have presented **BBOmix**, a large-scale tabular benchmark for unsupervised HPO of AEs on multi-omics data. The benchmark comprises 105,000 training runs organized into 35 black-box tasks, covering four AE architectures across two datasets and seven modalities. Our analysis demonstrated that reconstruction loss serves as a mostly strong, yet not ideal, proxy for downstream performance across most biological domains, validating its use as a cheap HPO criterion. However, notable exceptions (such as TCGA-DNA reflecting a gene mutation score) exist, underscoring the need for careful evaluation. Furthermore, our hyperparameter importance analysis revealed that parameters such as learning rate and dropout rate consistently dictate generalization performance, regardless of the specific biological modality or architecture.

Finally, our evaluation of HPO optimizers highlighted the substantial effectiveness of transfer learning and multi-fidelity methods in this domain. These methods allow researchers to rapidly identify high-performing configurations and exploit learned structural similarities across biological datasets at a fraction of the computational cost of standard single-fidelity methods. In sum, **BBOmix** provides a foundation for the community to develop and evaluate HPO methods in the underexplored unsupervised setting, and that the biological application domain motivates further HPO research.

6 Limitations and Future Work

Our benchmark, while extensive, is subject to certain limitations. First, our evaluation focuses on simple AE variants implemented in the new, but maturing framework AUTOENCODIX. Exploring convolutional, cross-modal, or graph-based architectures for spatially resolved or networked omics data remains an important next step. Second, while reconstruction loss serves as a generally robust proxy, it is fundamentally agnostic to downstream utility, and the proxy breakdown observed in the TCGA-DNA task highlights the necessity of developing more domain-aware unsupervised training metrics. While our current benchmark focuses on two comprehensive real-world multi-omics datasets (TCGA and SCHC), future work should expand this evaluation to encompass a broader variety of biological data modalities and clinical applications.

Moreover, our results are primarily reported as trends aggregated across the full task suite (e.g., Fig. 4). While useful for establishing overall algorithm rankings, such summaries may obscure heterogeneity between individual tasks. A more fine-grained, task-level analysis, grouping tasks by shared properties such as search-space dimensionality and conditionality, architecture family, omics modality, or dataset size, would help identify which characteristics drive HPO dynamics and disentangle genuine algorithmic effects from artifacts of the task mix.

Furthermore, our findings regarding the transferability of hyperparameter importance across architectures suggest a promising direction for meta-learning. Notably, our experiments exclusively consider cross-architecture transfer, and future work should systematically compare this against same-architecture, cross-modality and cross-dataset transfer to establish which axis of task relatedness most reliably predicts the portability of hyperparameter importance structure. We envision the development of a generalized, domain-specific HPO recommendation framework that leverages historical evaluations across diverse omics tasks to automatically suggest optimal architectures and hyperparameters for novel biological datasets, significantly reducing computational cost and lowering the barrier of entry for practitioners.

Acknowledgements. The authors gratefully acknowledge the computing time made available to them on the high-performance computer at the NHR Center of TU Dresden. This center is jointly supported by the Federal Ministry of Research, Technology and Space of Germany and the state governments participating in the NHR. Luca Thale-Bombien was further supported by the BMFTR through a scholarship of DAAD project 57616814 (SECAI, School of Embedded and Composite AI) as part of the program Konrad Zuse Schools of Excellence in Artificial Intelligence. Aaron Klein acknowledges support by the EC under the grant No. 101195233 (OpenEuroLLM).

References

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., et al. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2):e1484.

Chen, R. T., Li, X., Grosse, R. B., and Duvenaud, D. K. (2018). Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31.

Doncevic, D. and Herrmann, C. (2023). Biologically informed variational autoencoders allow predictive modeling of genetic and drug-induced perturbations. *Bioinformatics*, 39(6):btad387.

Dong, X. and Yang, Y. (2020). Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*.

Eggensperger, K., Hutter, F., Hoos, H., and Leyton-Brown, K. (2015). Efficient benchmarking of hyperparameter optimizers via surrogates. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Eraslan, G., Simon, L. M., Mircea, M., Mueller, N. S., and Theis, F. J. (2019). Single-cell rna-seq denoising using a deep count autoencoder. *Nature communications*, 10(1):390.

Ewald, J. (2025). Autoencodix raw data for reproducibility. `https://doi.org/10.5281/zenodo.15518831`.

Falkner, S., Klein, A., and Hutter, F. (2018). Bohb: Robust and efficient hyperparameter optimization at scale. In *International conference on machine learning*, pages 1437–1446. PMLR.

Feurer, M., Springenberg, J., and Hutter, F. (2015). Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Fisher, A., Rudin, C., and Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81.

Franceschi, L., Donini, M., Perrone, V., Klein, A., Archambeau, C., Seeger, M., Pontil, M., and Frasconi, P. (2025). Hyperparameter optimization in machine learning. *arXiv:2410.22854 [stat.ML]*.

Garnett, R. (2023). *Bayesian Optimization*. Cambridge University Press.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR'17)*.

Hu, Q. and Greene, C. S. (2018). Parameter tuning is a key part of dimensionality reduction via deep variational autoencoders for single cell rna transcriptomics. In *BIOCOMPUTING 2019: proceedings of the Pacific symposium*, pages 362–373. World Scientific.

Jamieson, K. and Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS'16)*.

Joas, M. J., Jurenaite, N., Praščević, D., Scherf, N., and Ewald, J. (2025). Autoencodix: a generalized and versatile framework to train and evaluate autoencoders for biological representation learning and beyond. *Nature Computational Science*, pages 1–13.

Klein, A. and Hutter, F. (2019). Tabular benchmarks for joint architecture and hyperparameter optimization. *arXiv preprint arXiv:1905.04970*.

Kopf, A. and Claassen, M. (2021). Latent representation learning in biology and translational medicine. *Patterns*, 2(3).

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of machine learning research*, 18(185):1–52.

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Ben-Tzur, J., Hardt, M., Recht, B., and Talwalkar, A. (2020). A system for massively parallel hyperparameter tuning. *Proceedings of machine learning and systems*, 2:230–246.

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O. (2019). Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR.

Lotfollahi, M., Klimovskaia Susmelj, A., De Donno, C., Hetzel, L., Ji, Y., Ibarra, I. L., Srivatsan, S. R., Naghipourfar, M., Daza, R. M., Martin, B., et al. (2023a). Predicting cellular responses to complex perturbations in high-throughput screens. *Molecular systems biology*, 19(6):MSB202211517.

Lotfollahi, M., Rybakov, S., Hrovatin, K., Hediyeh-Zadeh, S., Talavera-López, C., Misharin, A. V., and Theis, F. J. (2023b). Biologically informed deep learning to query gene programs in single-cell atlases. *Nature Cell Biology*, 25(2):337–350.

Mamoshina, P., Vieira, A., Putin, E., and Zhavoronkov, A. (2016). Applications of deep learning in biomedicine. *Molecular pharmaceutics*, 13(5):1445–1454.

Mardis, E. R. (2008). The impact of next-generation sequencing technology on genetics. *Trends in genetics*, 24(3):133–141.

Milacic, M., Beavers, D., Conley, P., Gong, C., Gillespie, M., Griss, J., Haw, R., Jassal, B., Matthews, L., May, B., et al. (2024). The reactome pathway knowledgebase 2024. *Nucleic acids research*, 52(D1):D672–D678.

Ovcharenko, O., Barkmann, F., Toma, P., Daunhawer, I., Vogt, J., Schelter, S., and Boeva, V. (2025). Scssl-bench: Benchmarking self-supervised learning for single-cell data. *arXiv preprint arXiv:2506.10031*.

Perrone, V., Shen, H., Seeger, M. W., Archambeau, C., and Jenatton, R. (2019). Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning. *Advances in neural information processing systems*, 32.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2019). Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789.

Reuter, J. A., Spacek, D. V., and Snyder, M. P. (2015). High-throughput sequencing technologies. *Molecular cell*, 58(4):586–597.

Salinas, D. and Erickson, N. (2023). Tabrepo: A large scale repository of tabular model evaluations and its automl applications. *arXiv preprint arXiv:2311.02971*.

Salinas, D., Golebiowski, J., Klein, A., Seeger, M., and Archambeau, C. (2023). Optimizing hyperparameters with conformal quantile regression. In *International Conference on Machine Learning*, pages 29876–29893. PMLR.

Salinas, D., Seeger, M., Klein, A., Perrone, V., Wistuba, M., and Archambeau, C. (2022). Syne tune: A library for large scale hyperparameter tuning and reproducible research. In *International Conference on Automated Machine Learning*, pages 16–1. PMLR.

Salinas, D., Shen, H., and Perrone, V. (2020). A quantile-based approach for hyperparameter transfer learning. In *International conference on machine learning*, pages 8438–8448. PMLR.

Selby, D. A., Jakhmola, R., Sprang, M., Großmann, G., Raki, H., Maani, N., Pavliuk, D., Ewald, J., and Vollmer, S. (2025). Visible neural networks for multi-omics integration: a critical review. *Frontiers in Artificial Intelligence*, 8:1595291.

Seninge, L., Anastopoulos, I., Ding, H., and Stuart, J. (2021). Vega is an interpretable generative model for inferring biological network activity in single-cell transcriptomics. *Nature communications*, 12(1):5684.

Simidjievski, N., Bodnar, C., Tariq, I., Scherer, P., Andres Terre, H., Shams, Z., Jamnik, M., and Liò, P. (2019). Variational autoencoders for cancer data integration: design principles and computational practice. *Frontiers in genetics*, 10:1205.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Advances in Neural Information Processing Systems (NeurIPS'12)*.

Tiao, L., Klein, A., Seeger, M., Archambeau, C., Bonilla, E., and Ramos, F. (2020). Bayesian optimization by density ratio estimation.

Weinstein, J. N., Collisson, E. A., Mills, G. B., Shaw, K. R., Ozenberger, B. A., Ellrott, K., Shmulevich, I., Sander, C., and Stuart, J. M. (2013). The cancer genome atlas pan-cancer analysis project. *Nature genetics*, 45(10):1113–1120.

Whalen, S., Schreiber, J., Noble, W. S., and Pollard, K. S. (2022). Navigating the pitfalls of applying machine learning in genomics. *Nature Reviews Genetics*, 23(3):169–181.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2015a). Learning hyperparameter optimization initializations. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–10. IEEE.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2015b). Sequential model-free hyperparameter tuning. In *2015 IEEE international conference on data mining*, pages 1033–1038. IEEE.

Yang, K. D., Belyaeva, A., Venkatachalapathy, S., Damodaran, K., Katcoff, A., Radhakrishnan, A., Shivashankar, G., and Uhler, C. (2021). Multi-domain translation between single-cell imaging and sequencing data using autoencoders. *Nature communications*, 12(1):31.

Zhu, K., Bendl, J., Rahman, S., Vicari, J. M., Coleman, C., Clarence, T., Latouche, O., Tsankova, N. M., Li, A., Brennand, K. J., et al. (2023). Multi-omic profiling of the developing human cerebral cortex at the single-cell level. *Science advances*, 9(41):eadg3754.

Zimmer, L., Lindauer, M., and Hutter, F. (2021). Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. *IEEE transactions on pattern analysis and machine intelligence*, 43(9):3079–3090.

Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

   (b) Did you describe the limitations of your work? [Yes]

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Does not apply to this research.

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? (see `https://2022.automl.cc/ethics-accessibility/`) [Yes]

2. If you ran experiments...

   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources, etc.)? [Yes]

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning details and results, etc.)? [Yes]

   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [Yes]

   (d) Did you report the uncertainty of your results (e.g., the standard error across random seeds or splits)? [Yes]

   (e) Did you report the statistical significance of your results? [N/A]

   (f) Did you use enough repetitions, datasets, and/or benchmarks to support your claims? [Yes]

   (g) Did you compare performance over time and describe how you selected the maximum runtime? [Yes] We selected a maximum wallclock runtime of 72,000 simulated seconds (20 hours) to reflect a realistic compute budget for a practitioner training a single deep learning model overnight, allowing us to observe both early-budget and asymptotic optimization behaviors.

   (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

   (i) Did you run ablation studies to assess the impact of different components of your approach? [N/A] Does not apply.

3. With respect to the code used to obtain your results...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all dependencies (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation instructions, and execution commands (either in the supplemental material or as a URL)? [Yes] The Code is publicly accessible on GitHub: `https://github.com/Kavlahkaff/BBOmix`. The raw HPO data and blackboxes are available on Huggingface: `https://huggingface.co/datasets/synetune/blackbox-repository`

   (b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [Yes]

   (c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes]

(d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes]

(e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes]

4. If you used existing assets (e.g., code, data, models)…

(a) Did you cite the creators of used assets? [Yes]

(b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [Yes] Our benchmark is derived entirely from publicly available, fully de-identified biological datasets (TCGA and SCHC). No further consent is required for this secondary analysis, and the data contains no personally identifiable information (PII) or offensive content."

(c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] As described above, the used and preprocessed omics data does not contain sensitive raw data like genomic sequences but leverages publicly available and non-senstive molecular data like gene expression counts or gene mutation counts of cells and patients.

5. If you created/released new assets (e.g., code, data, models)…

(a) Did you mention the license of the new assets (e.g., as part of your code submission)? [Yes] The code for reproducibility and the benchmark data is made available via the Apache 2.0 license.

(b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [Yes]

6. If you used crowdsourcing or conducted research with human subjects…

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to institutional review board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

7. If you included theoretical results…

(a) Did you state the full set of assumptions of all theoretical results? [N/A]

(b) Did you include complete proofs of all theoretical results? [N/A]

A Clinical Annotations for Downstream Tasks

To evaluate the downstream utility of the learned latent spaces, we employed classification tasks derived from the clinical and biological metadata accompanying each dataset. Performance was measured via the Area Under the Receiver Operating Characteristic curve (AUC-ROC).

A.1 The Cancer Genome Atlas (TCGA)

The TCGA dataset includes extensive patient-level clinical data. We utilized the following targets:

- **Sex**: Binary classification of the patient's biological sex.
- **Cancer type**: Multi-class classification of the primary tumor type based on tissue origin (e.g., BRCA, LUAD).
- **Subtype**: Multi-class classification of the molecular or histological subtype.
- **Oncotree code**: Highly granular classification of the cancer lineage based on the OncoTree ontology.
- **AJCC stage**: Multi-class classification of the overall tumor stage (I, II, III, IV).
- **Path N stage**: Classification of regional lymph node involvement indicating tumor spread.
- **Grade**: Classification of the histological grade representing cellular abnormality.
- **OS status**: Binary classification of Overall Survival (living vs. deceased).
- **DSS status**: Binary classification of Disease-Specific Survival (death specifically attributed to the cancer).

A.2 Single-Cell Human Cortex (SCHC)

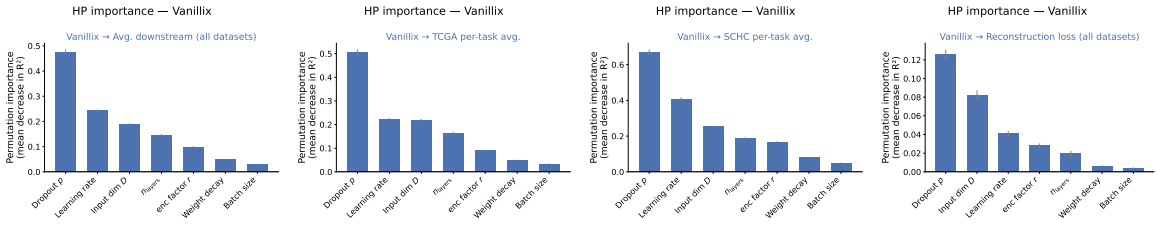
The SCHC dataset provides cell-level metadata. The downstream targets were:

- **Cell type**: Multi-class classification of the specific neural or glial cell type (e.g., excitatory neurons, astrocytes) assigned by the original authors.
- **Age group**: Classification of the donor's developmental stage (e.g., fetal, adult).
- **Sex**: Binary classification of the donor's biological sex.

B Hyperparameter Importance per Architecture

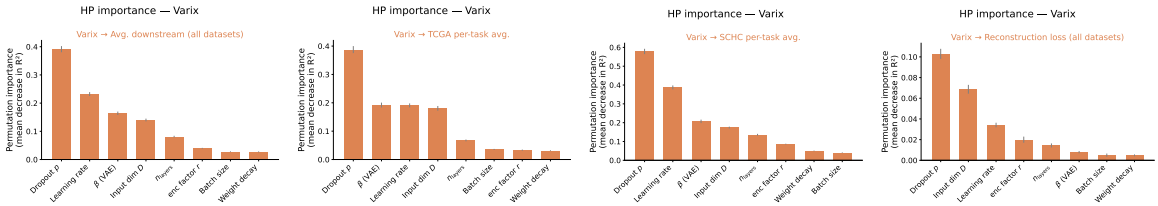
We report permutation importance scores for each architecture across three objectives: aggregate downstream performance (averaged over all datasets), TCGA per-task average, SCHC per-task average, and reconstruction loss. Figures 5–12 show the results.

We also show hyperparameter importance for each downstream metric, task and architecture in figures 9 - 12.



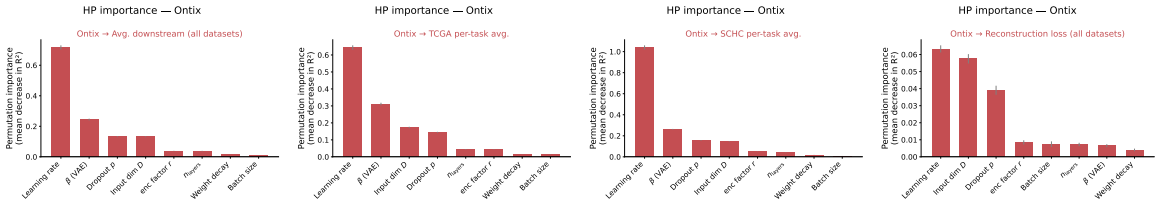
(a) Downstream (b) TCGA per-task avg. (c) SCHC per-task avg. (d) Reconstruction loss

Figure 5: HP importance Vanillix. Dropout p dominates downstream performance across both datasets, followed by learning rate. Only for Reconstruction loss, the Input Dimensionality D is ranked second.



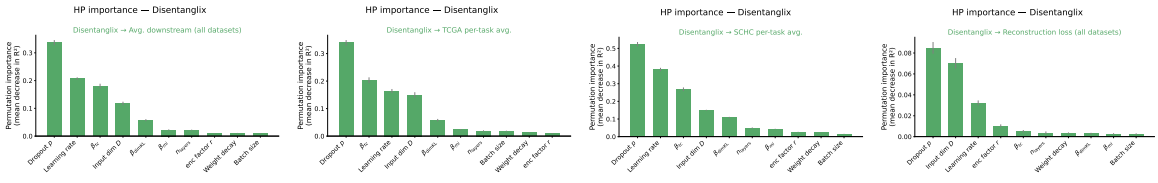
(a) Downstream (b) TCGA per-task avg. (c) SCHC per-task avg. (d) Reconstruction loss

Figure 6: HP importance Varix. Dropout p is dominant for downstream performance, followed by the learning rate and the architecture specific parameter β. For reconstruction loss, input dim D rises to second place, and the VAE-specific β term contributes only marginally.



(a) Downstream (b) TCGA per-task avg. (c) SCHC per-task avg. (d) Reconstruction loss

Figure 7: HP importance Ontix. Learning rate is exceptionally dominant, accounting for the vast majority of explained variance in downstream performance on both SCHC and TCGA. β (VAE) is the second most important HP, while dropout p ranks third. For reconstruction loss, learning rate and input dim D are nearly equally important.



(a) Downstream (b) TCGA per-task avg. (c) SCHC per-task avg. (d) Reconstruction loss

Figure 8: HP importance Disentanglix. Dropout p leads for downstream performance, followed by learning rate and β_{tc}, indicating that the total-correlation penalty meaningfully affects generalisation. For reconstruction loss, dropout p and input dim D dominate, while β_{tc} is negligible.

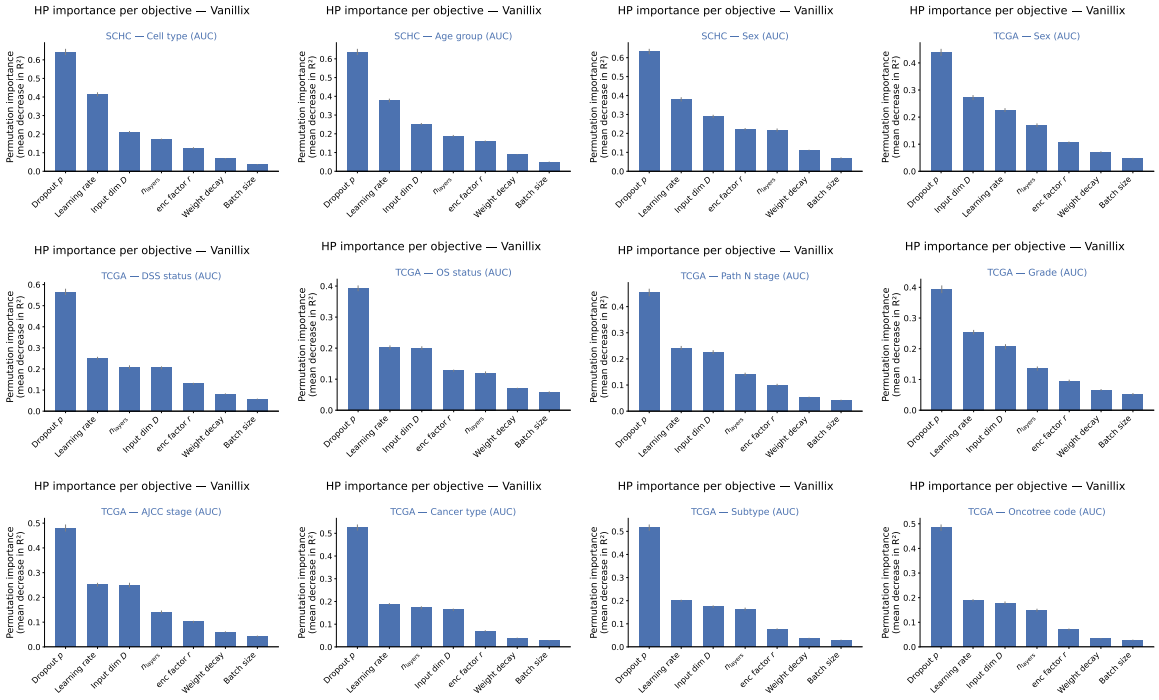


Figure 9: Per-task HP importance Vanillix. Dropout rate dominates consistently regardless of task, with β (VAE) as a stable second contributor.

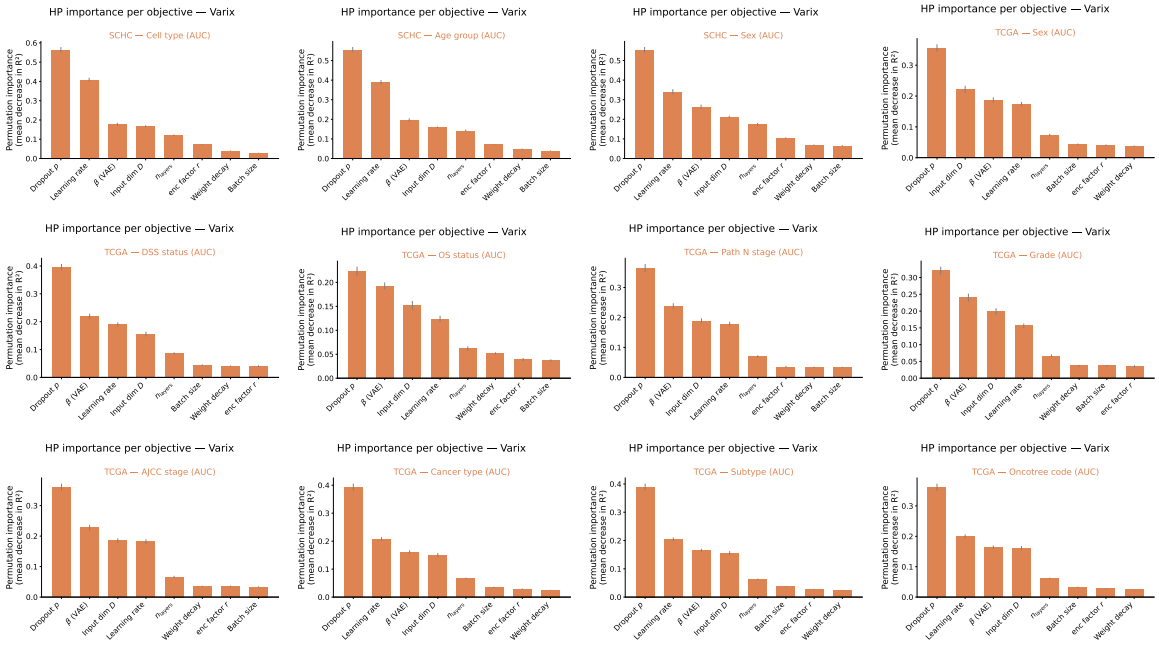


Figure 10: Per-task HP importance Varix. Dropout rate dominates consistently regardless of task, with β (VAE) as a stable second contributor.

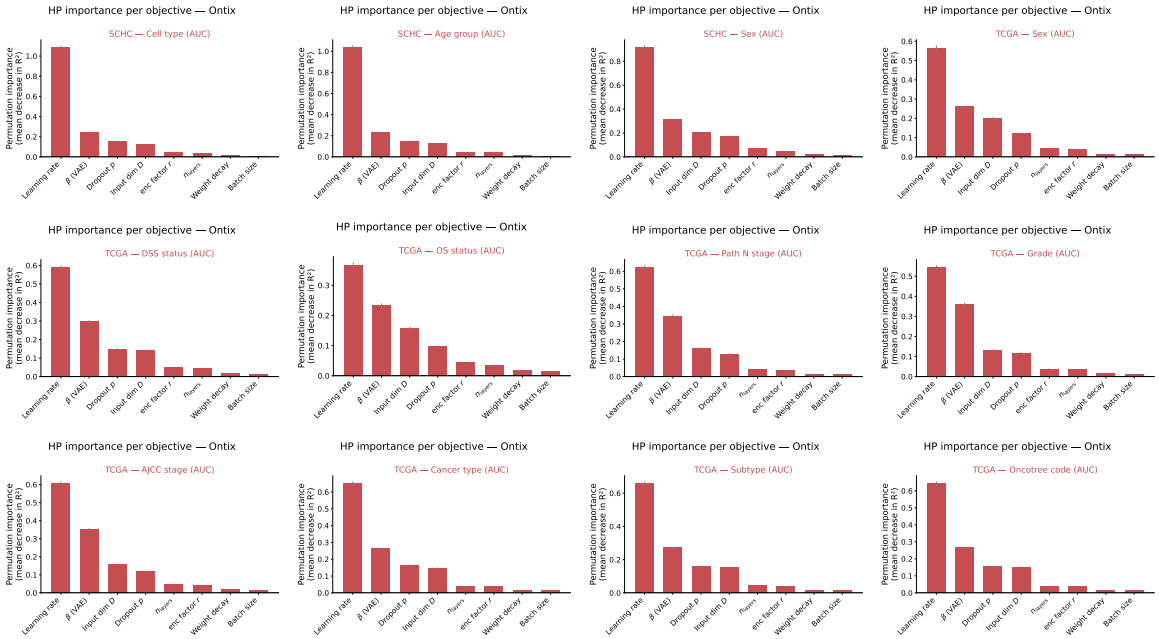


Figure 11: Per-task HP importance Ontix. Learning rate dominates consistently regardless of task, with β (VAE) as a stable second contributor.

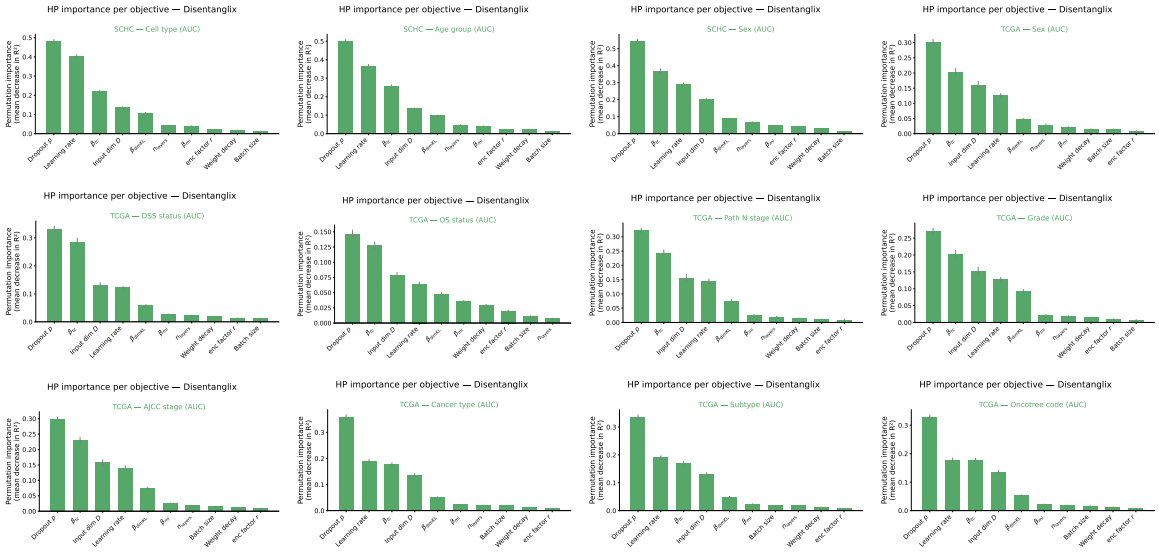


Figure 12: Per-task HP importance Disentanglix. Dropout p and β_{tc} both contribute across tasks, with a more distributed importance profile than Ontix.

C Description of Black-box Optimizers

We consider the following black-box optimization algorithms to construct our dataset:

- **BORE** (Tiao et al., 2020): Bayesian Optimization by Density-Ratio Estimation recasts hyperparameter optimization as a binary classification task. A probabilistic classifier is trained to assign high scores to configurations likely to exceed a performance threshold. We used the Syne Tune default configuration.
- **CQR** (Salinas et al., 2023): Conformalized Quantile Regression is a surrogate model for Bayesian optimization that handles heteroskedastic objective noise by estimating conditional quantiles rather than assuming fixed Gaussian observation noise. Split conformal prediction is applied to calibrate these quantile intervals with an empirical offset γ_j, providing finite-sample coverage guarantees. The calibrated quantiles enable efficient Thompson sampling, selecting the next configuration as the candidate with the lowest randomly drawn quantile value. We used the Syne Tune default configuration.
- **REA** (Real et al., 2019): Regularized Evolution is a population-based search method that modifies tournament selection via *aging*: rather than eliminating the worst-performing individuals, the oldest configurations are removed from the population, encouraging broader exploration and reducing premature convergence. We used a *population size* of 25 and *sample size* of 5 instead of the Syne Tune defaults.
- **TPE** (Bergstra et al., 2011): Tree-structured Parzen Estimator is a Bayesian optimization method that models the hyperparameter–performance relationship through density estimation rather than direct regression. Two kernel density estimators are fit: $l(x) = p(x \mid y < y^*)$ over high-performing configurations and $g(x) = p(x \mid y \geq y^*)$ over the remainder, separated by a quantile threshold y^*. Acquisition proceeds by maximizing $l(x)/g(x)$, which is proportional to the Expected Improvement. We used the Syne Tune default configuration.
- **ASHA** (Li et al., 2020): Asynchronous Successive Halving is a multi-fidelity hyperparameter optimization algorithm that allocates resources adaptively. Configurations are initially evaluated on small budgets, and only the most promising ones are promoted to higher fidelity levels, enabling efficient parallel search without requiring synchronization across workers. We used the Syne Tune default configuration.
- **ASHABORE**: A hybrid method that uses ASHA for multi-fidelity scheduling while replacing the default random promotion policy with BORE's density-ratio classifier to guide the selection of configurations at each rung level. We used the Syne Tune default configuration.
- **ASHACQR**: A hybrid method combining ASHA's successive halving schedule with CQR's conformalized quantile surrogate to select and promote configurations, inheriting both the budget efficiency of ASHA and the uncertainty calibration of CQR. We used the Syne Tune default configuration.
- **BOHB** (Falkner et al., 2018): Bayesian Optimization with Hyperband combines the strong anytime performance of Hyperband with the sample efficiency of Bayesian optimization. It uses kernel density estimators (following TPE) to model the distribution of good and bad configurations, and couples this with successive halving to allocate compute budgets adaptively. We used a *min_bandwidth* of $1e^{-1}$.
- **Quantile Transfer** (Salinas et al., 2023): Extends CQR to the transfer learning setting by leveraging performance data from related tasks. Quantile regressors trained on source tasks are used to warm-start the surrogate on a new target task, reducing the number of evaluations required to identify well-performing configurations. We used the Syne Tune default configuration.

- **Zero Shot** (Wistuba et al., 2015b): Constructs a portfolio of configurations offline from historical data across tasks, without performing any online evaluations on the target task. At deployment time, the single best-predicted configuration from this portfolio is returned, making it entirely free of target-task function evaluations. We used the Syne Tune default configuration.
- **Bounding Box** (Perrone et al., 2019): Restricts the hyperparameter search space to a tighter *bounding box* inferred from well-performing configurations observed on related tasks. Standard Bayesian optimization is then run within this reduced space, concentrating the search budget in more promising regions. We used the Syne Tune default configuration.
- **RS** (Bergstra and Bengio, 2012): Random Search draws configurations independently and uniformly from the search space, serving as a simple but competitive baseline. We used the Syne Tune default configuration.

D Per-Task Performance

We show violin plots of average downstream performance for both, SCHC and TCGA and their corresponding downstream tasks.

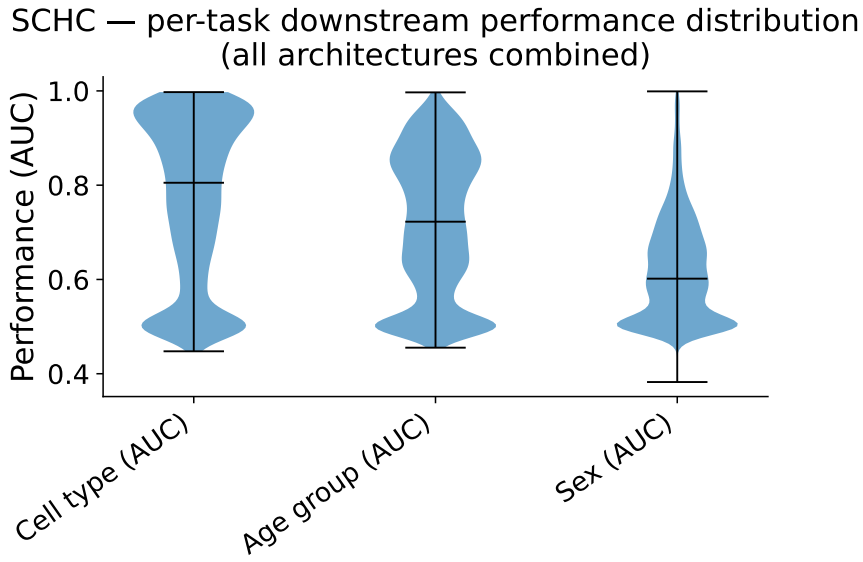


Figure 13: Per-task performance averaged over all architectures on the SCHC dataset.

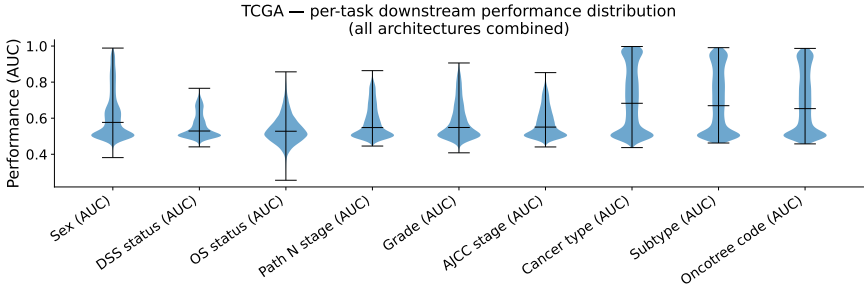


Figure 14: Per-task performance averaged over all architectures on the TCGA dataset.

E Optimization Trajectories for Reconstruction Loss

We show optimization trajectories for all methods used and all 35 blackbox tasks. Each optimizer was run with 30 different seeds and for 72000 seconds simulated wallclock time.

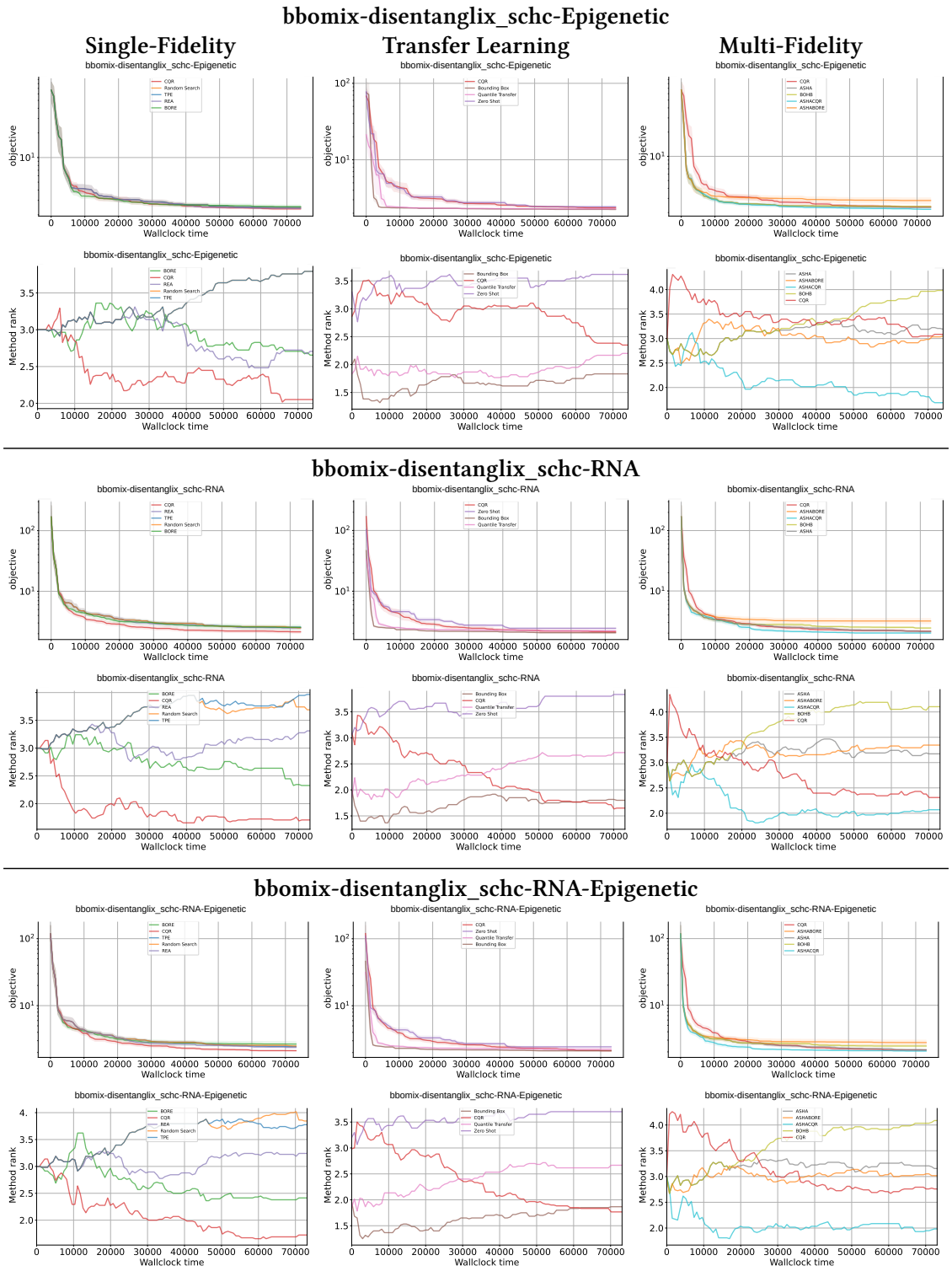


Figure 15: Results for Disentanglix tasks (Part 1).

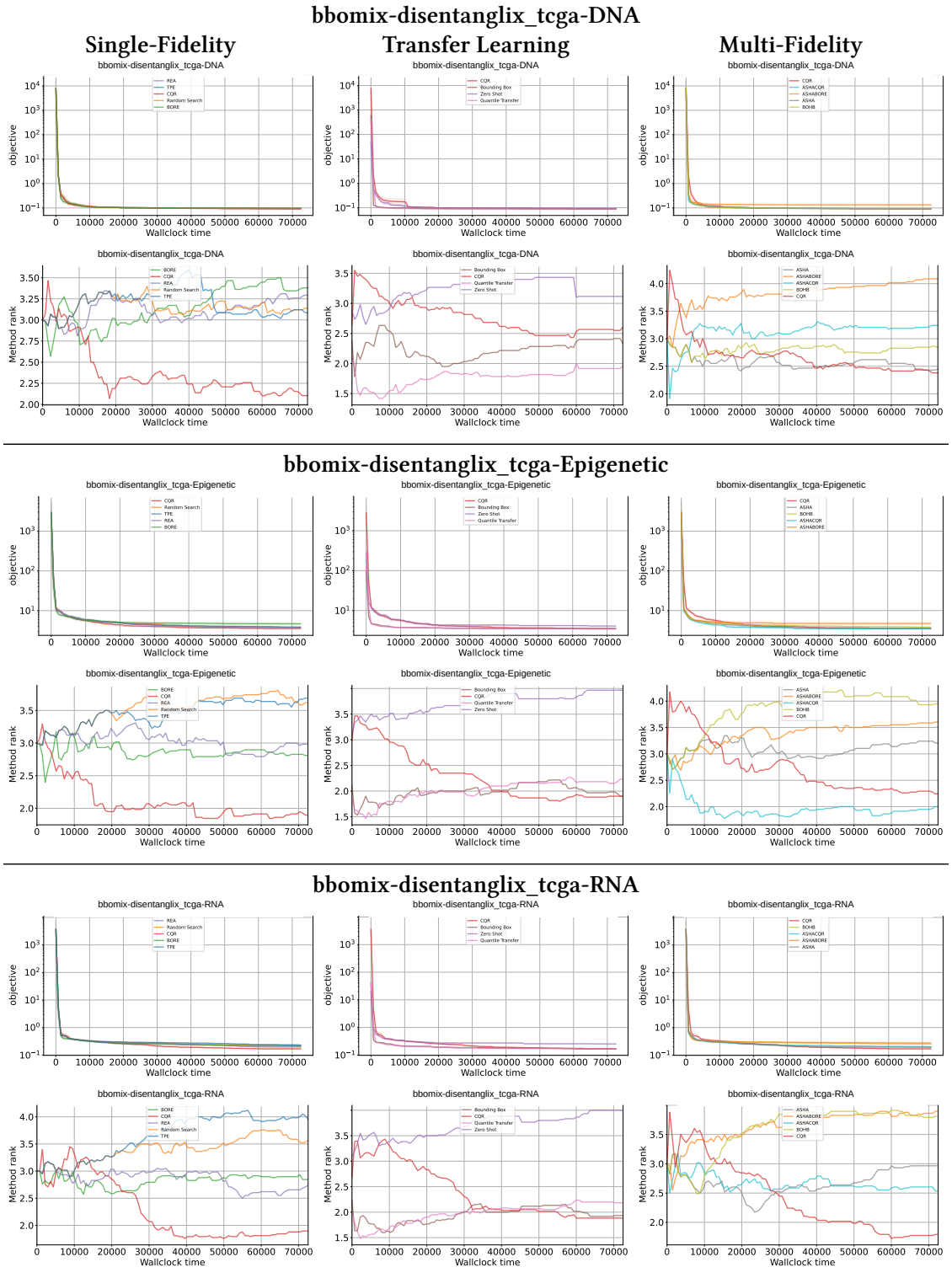


Figure 16: Results for Disentanglix tasks (Part 2).

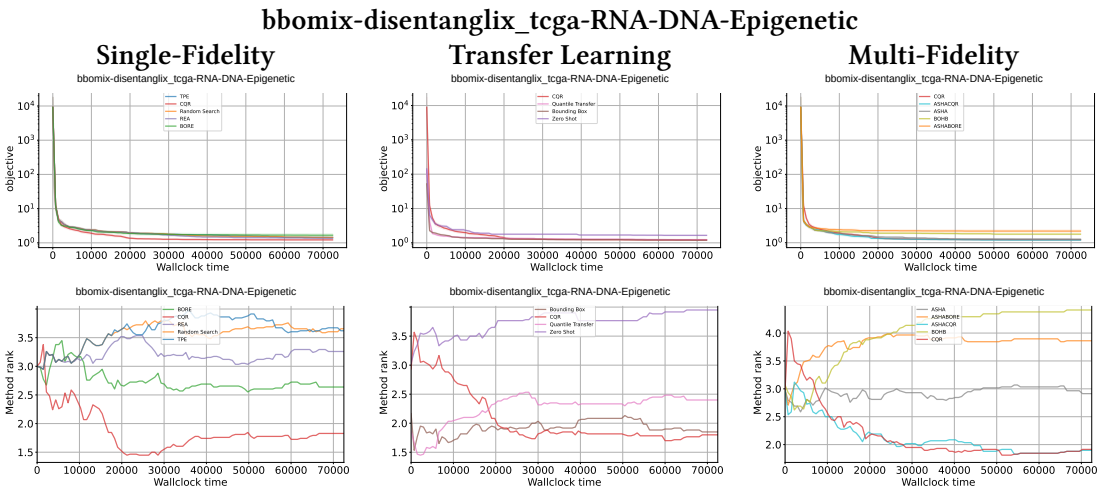


Figure 17: Results for Disentanglix (bbomix-disentanglix_tcga-RNA-DNA-Epigenetic).

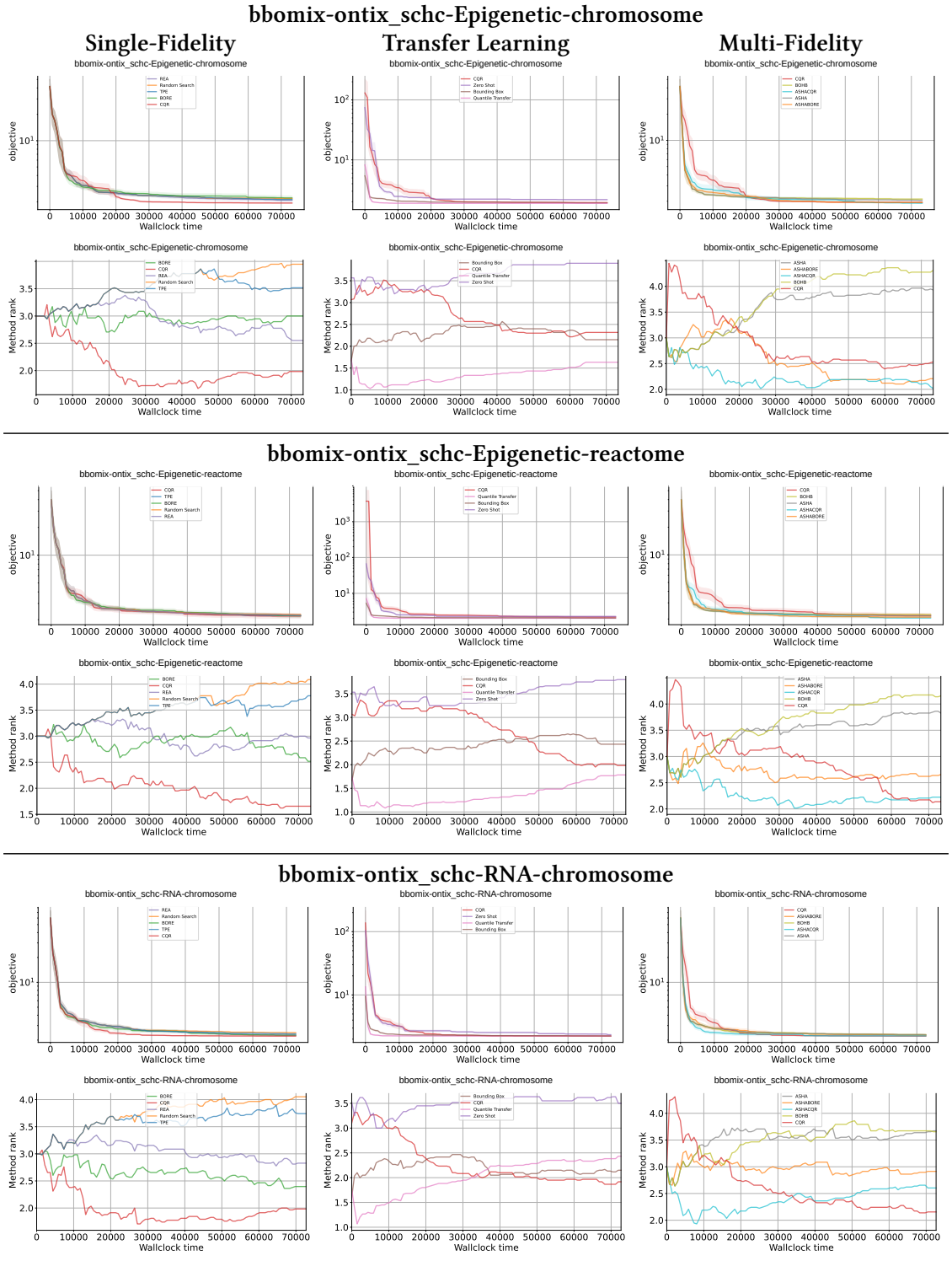


Figure 18: Results for Ontix tasks (Part 1).

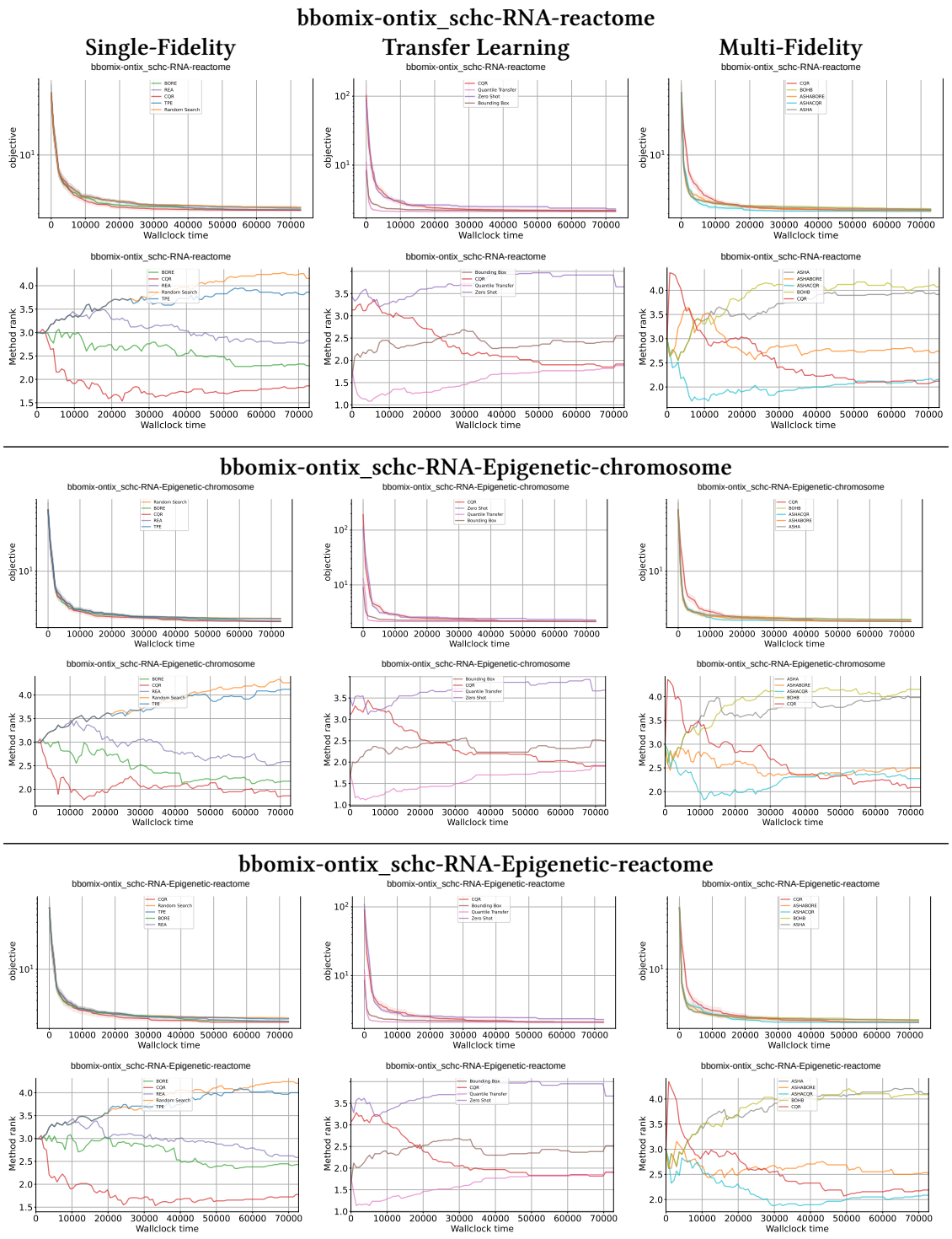


Figure 19: Results for Ontix tasks (Part 2).

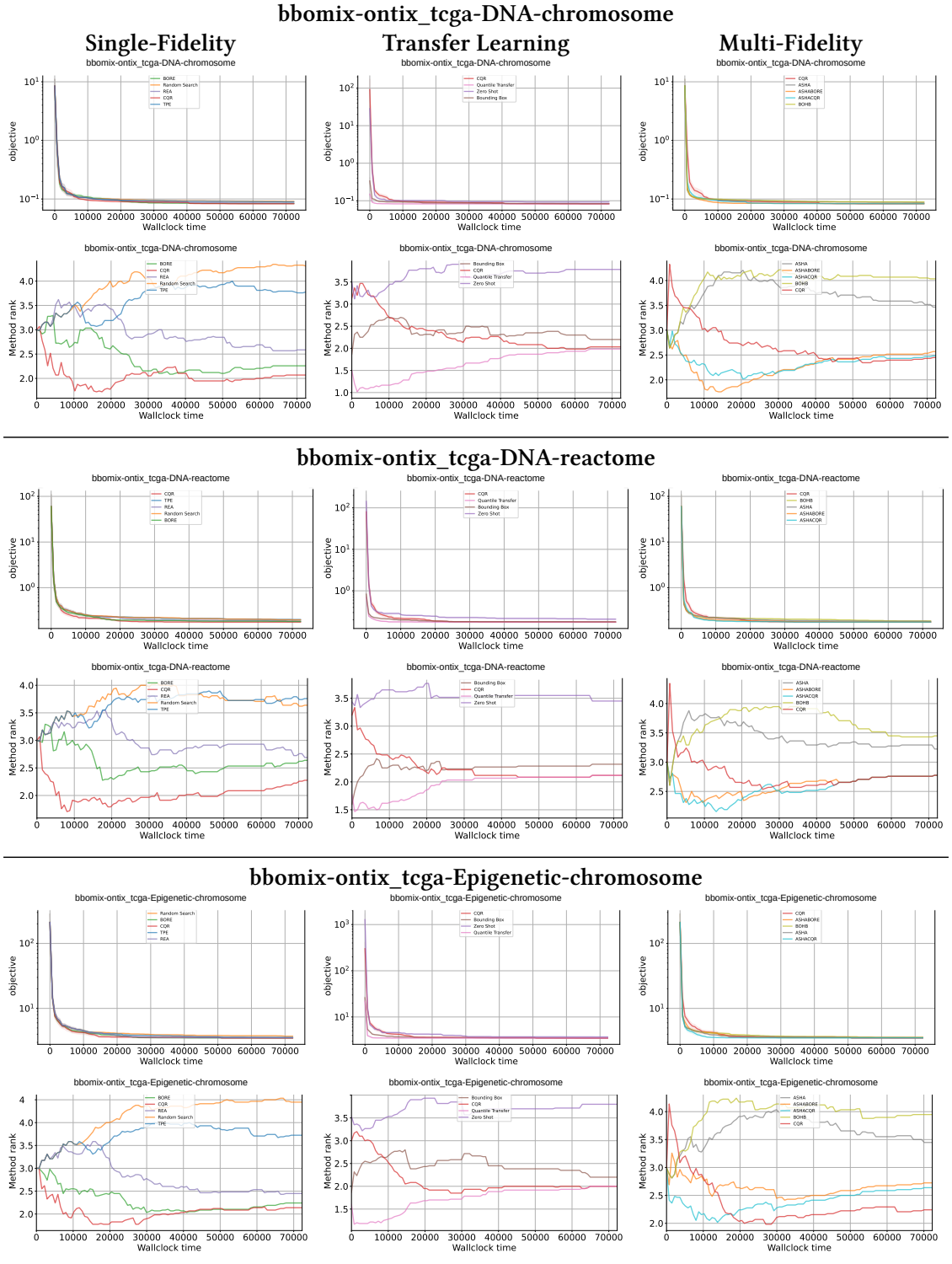


Figure 20: Results for Ontix tasks (Part 3).

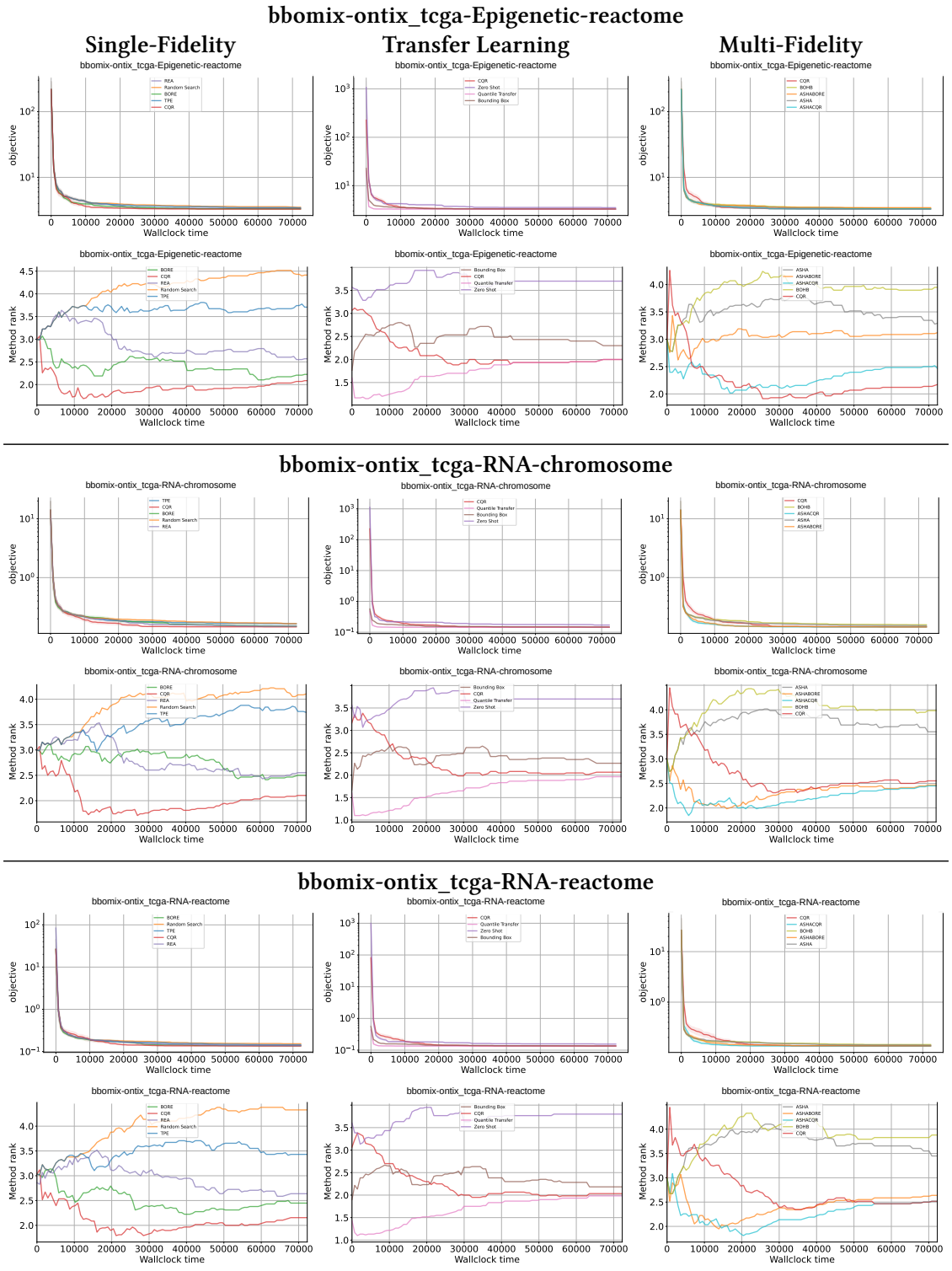


Figure 21: Results for Ontix tasks (Part 4).

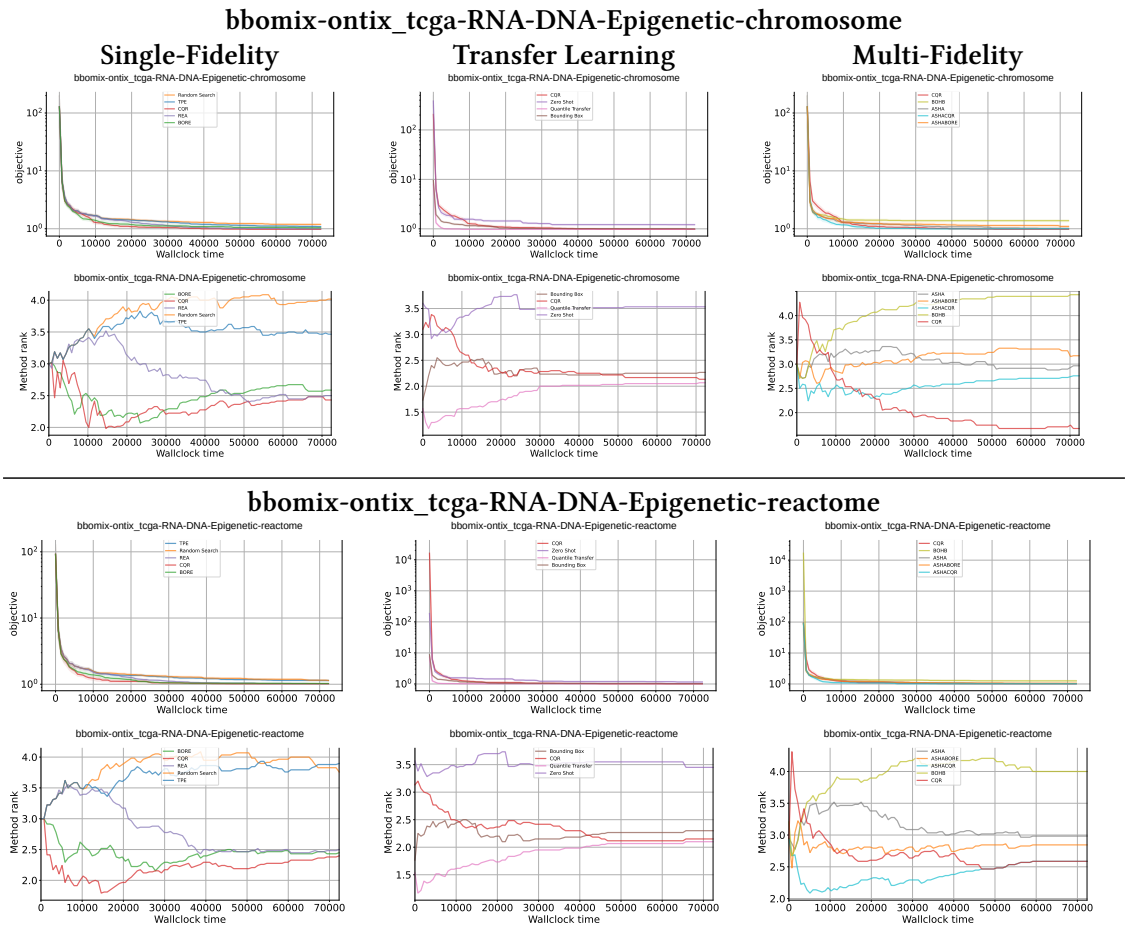


Figure 22: Results for Ontix tasks (Part 5).

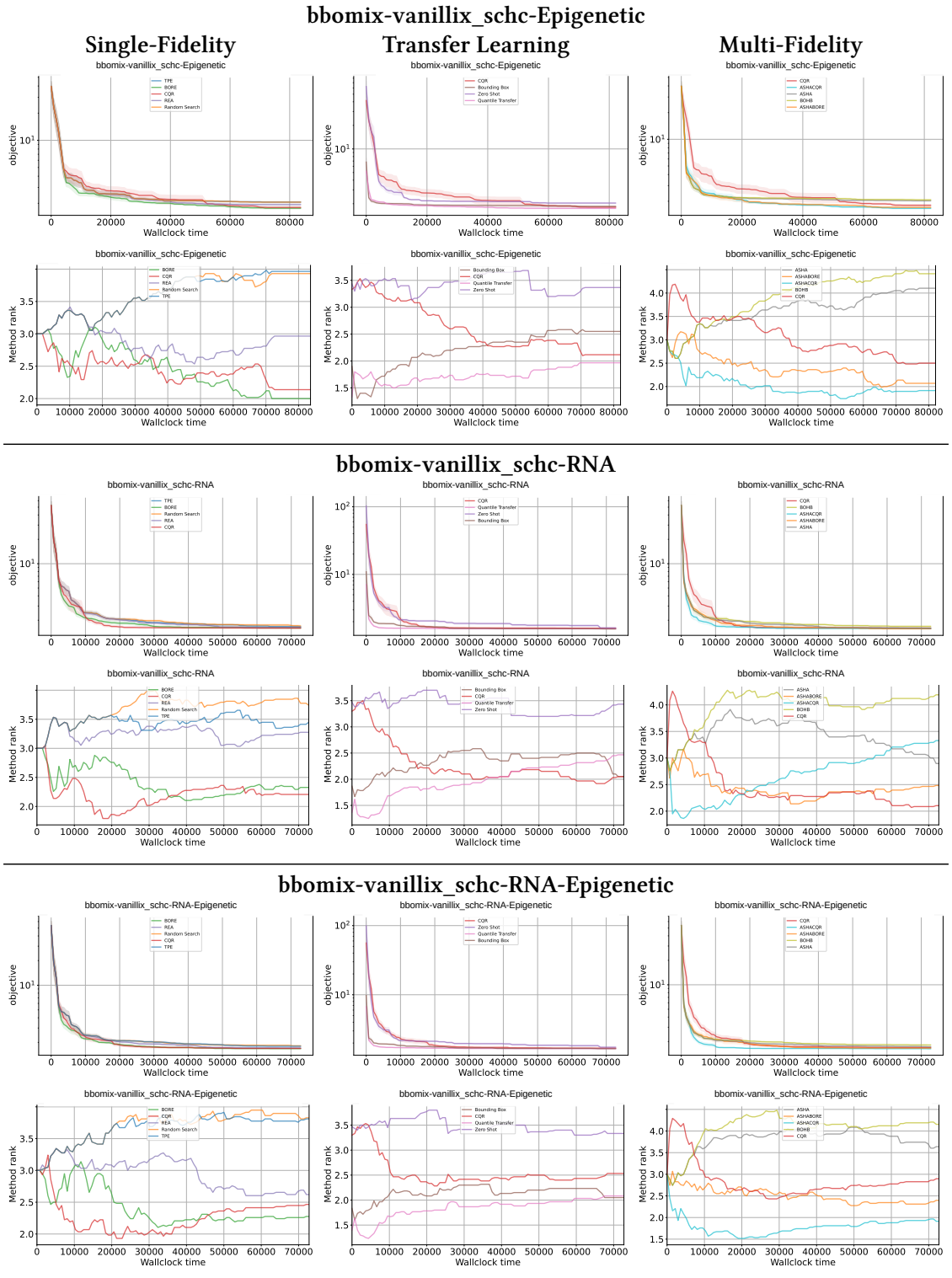


Figure 23: Results for Vanillix tasks (Part 1).

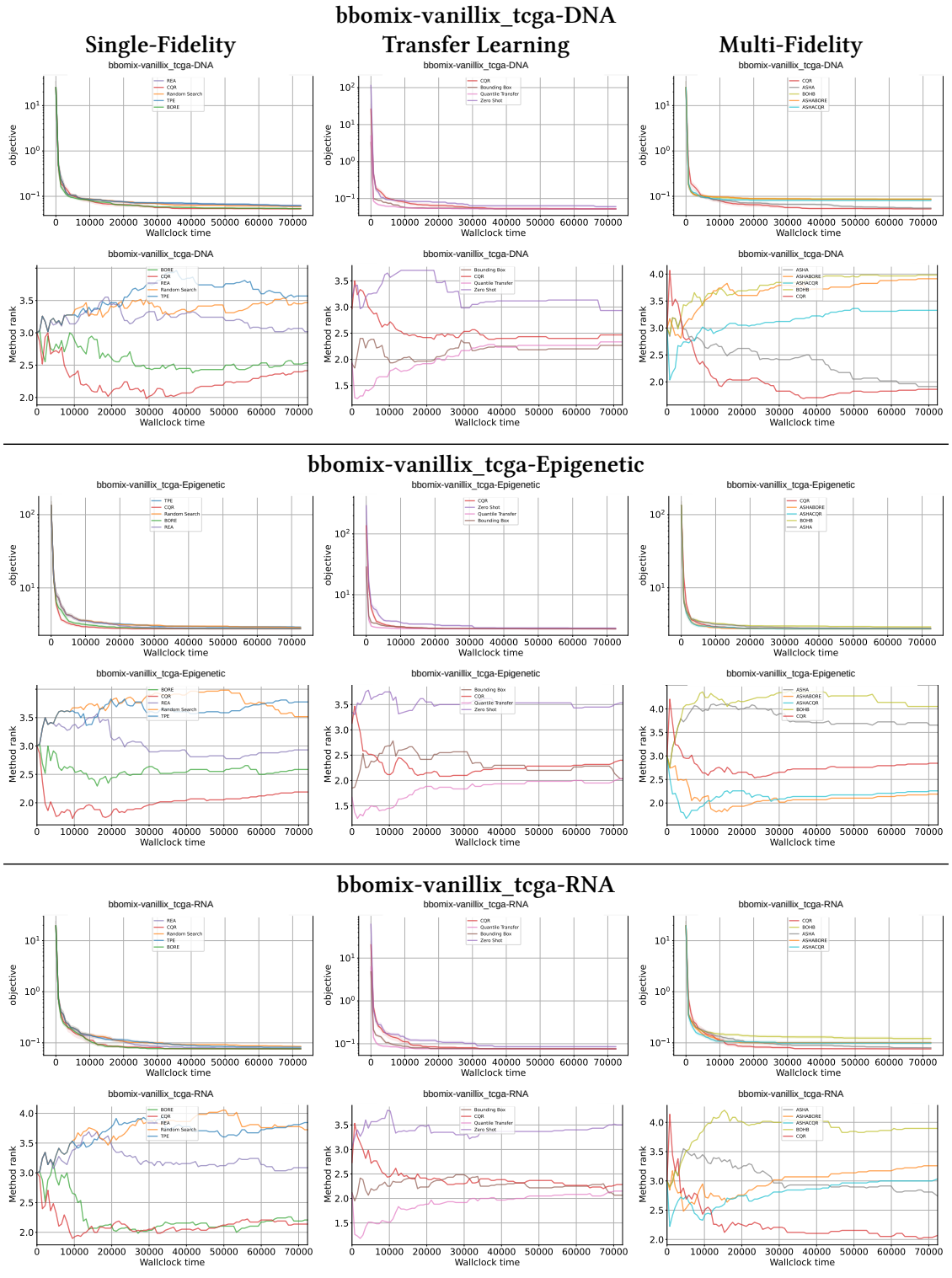


Figure 24: Results for Vanillix tasks (Part 2).

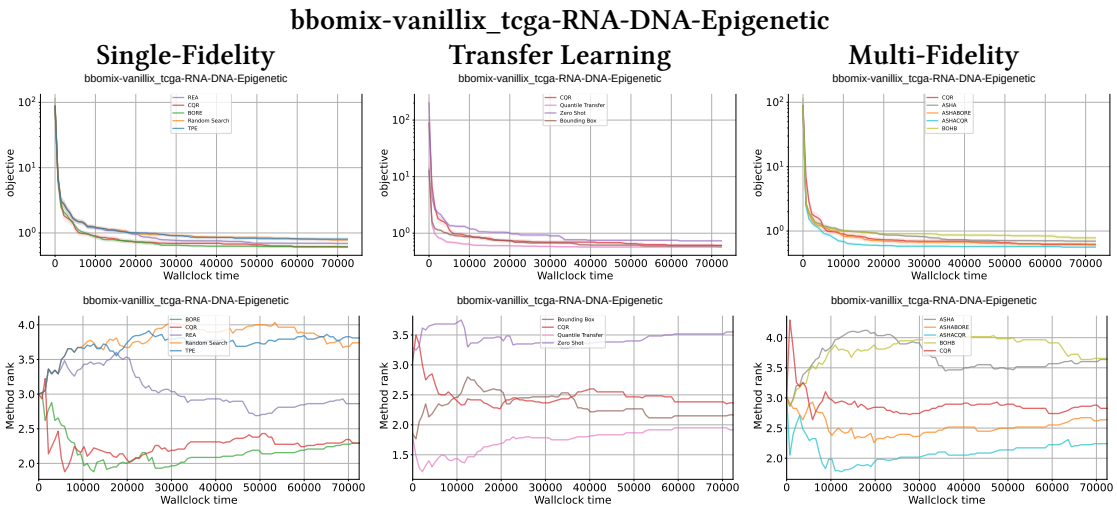


Figure 25: Results for Vanillix (bbomix-vanillix_tcga-RNA-DNA-Epigenetic).

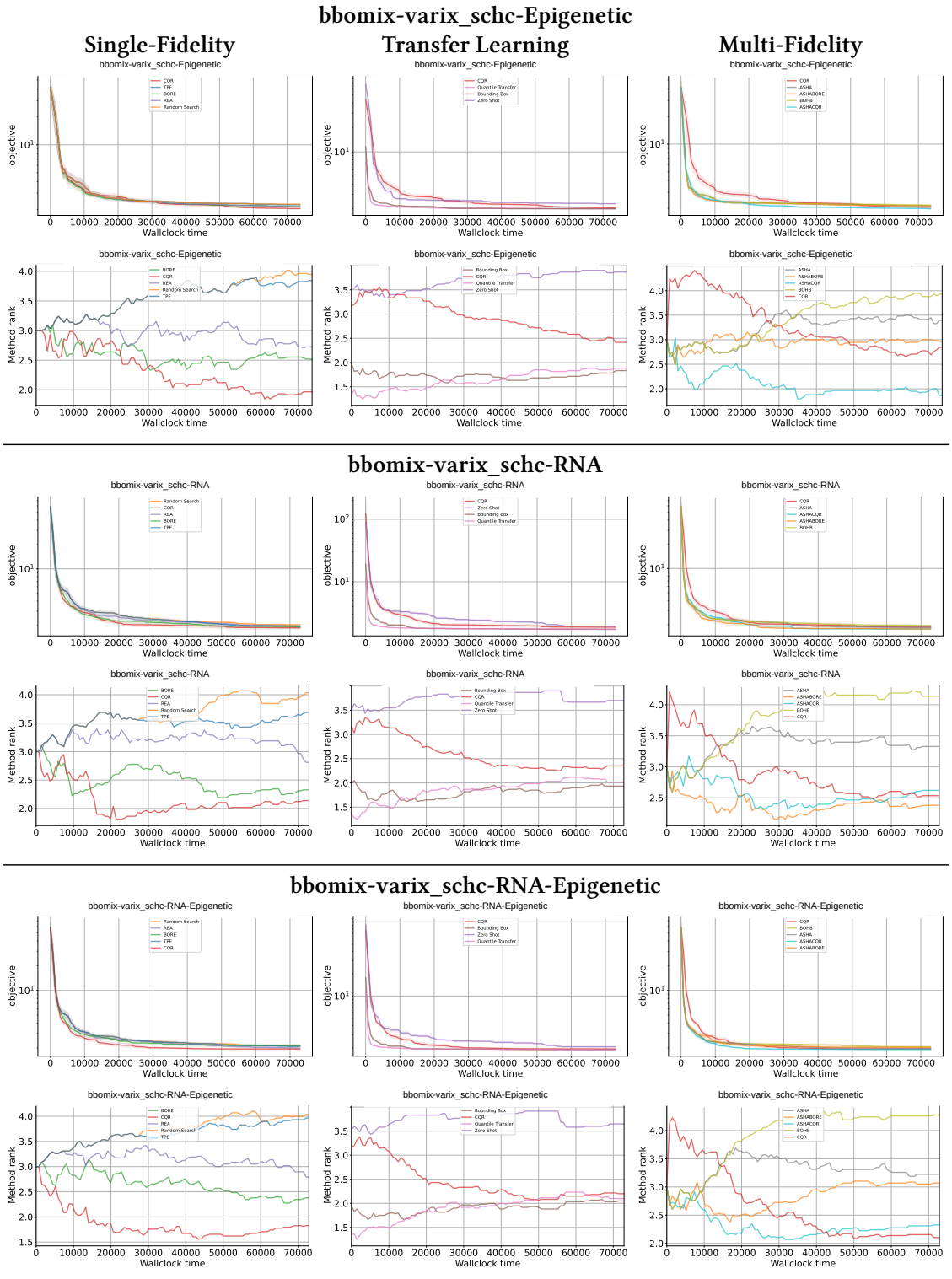


Figure 26: Results for Varix tasks (Part 1).

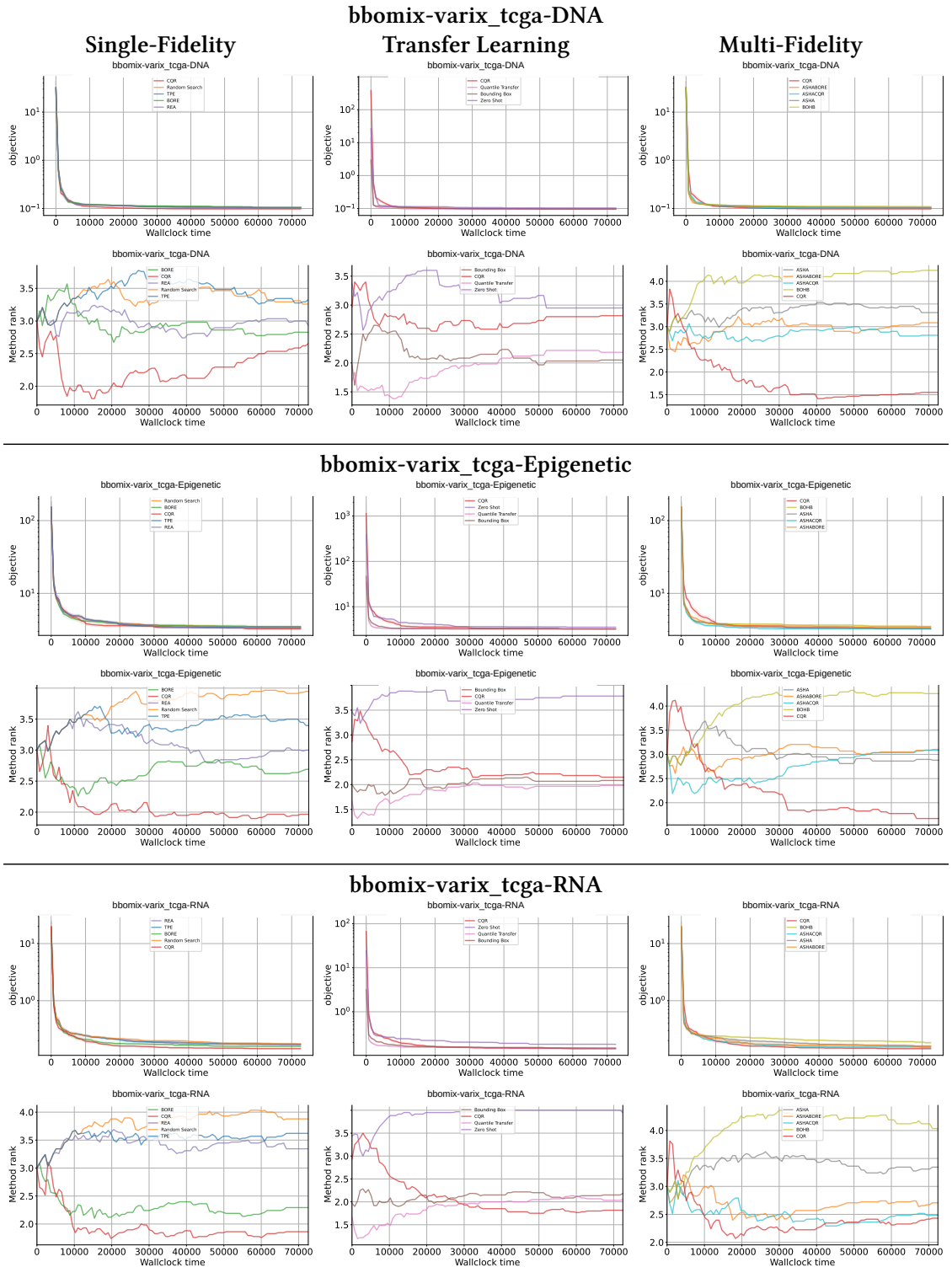


Figure 27: Results for Varix tasks (Part 2).

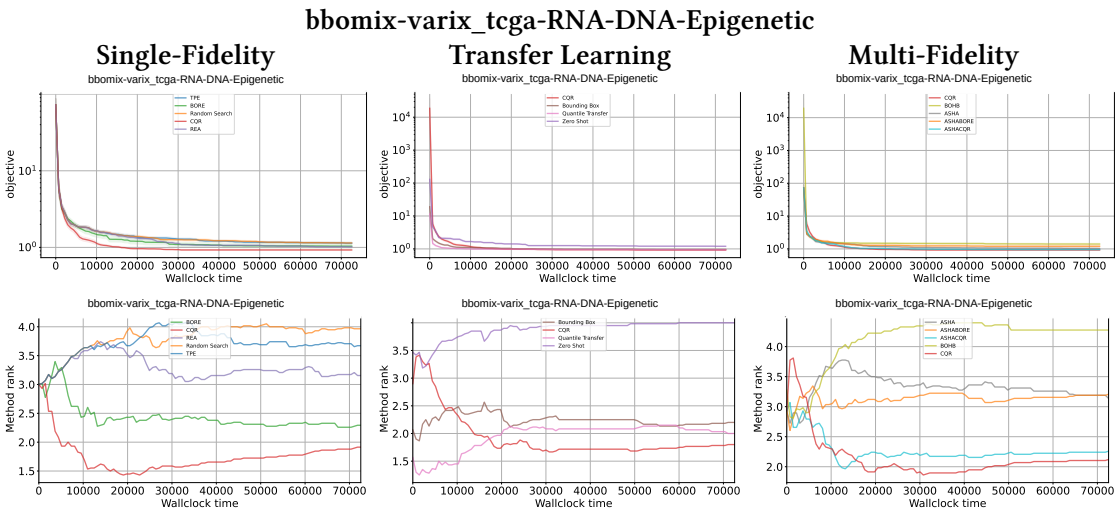


Figure 28: Results for Varix (bbomix-varix_tcga-RNA-DNA-Epigenetic).

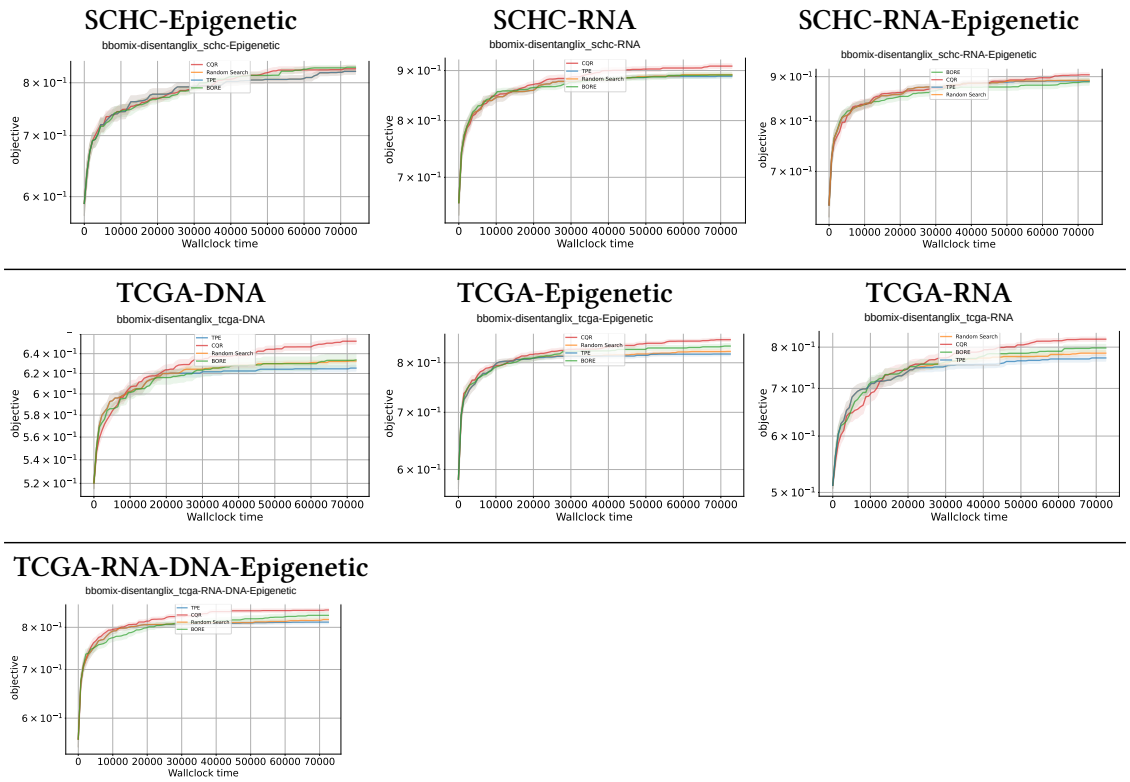


Figure 29: Disentanglix: optimization trajectories maximizing downstream performance

E.1 Optimization Trajectories of Downstream Performance

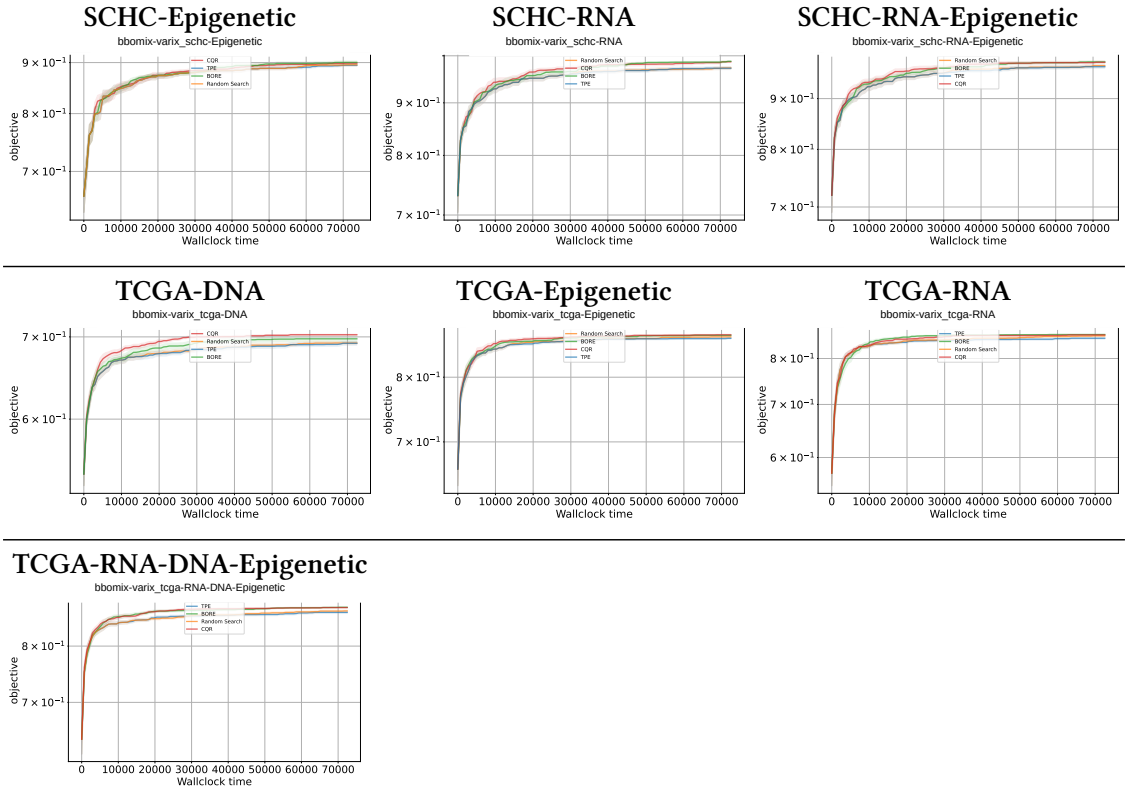


Figure 30: Varix: optimization trajectories maximizing downstream performance

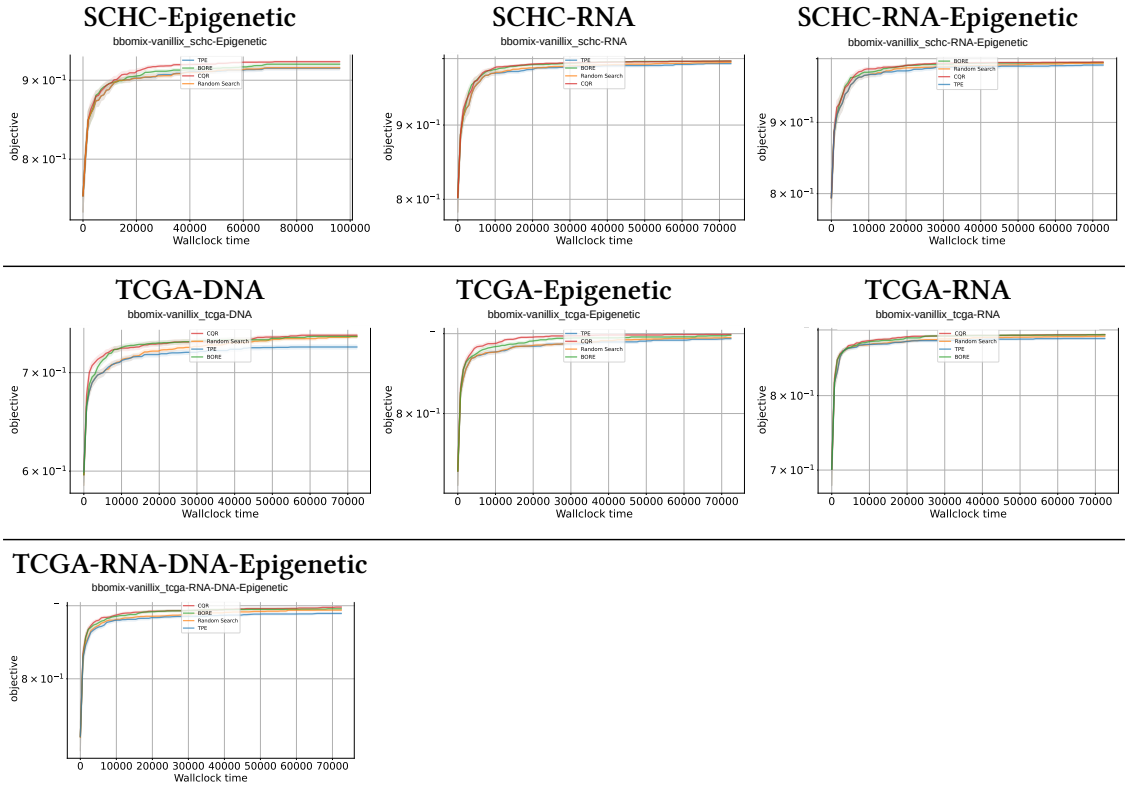


Figure 31: Vanillix: optimization trajectories maximizing downstream performance

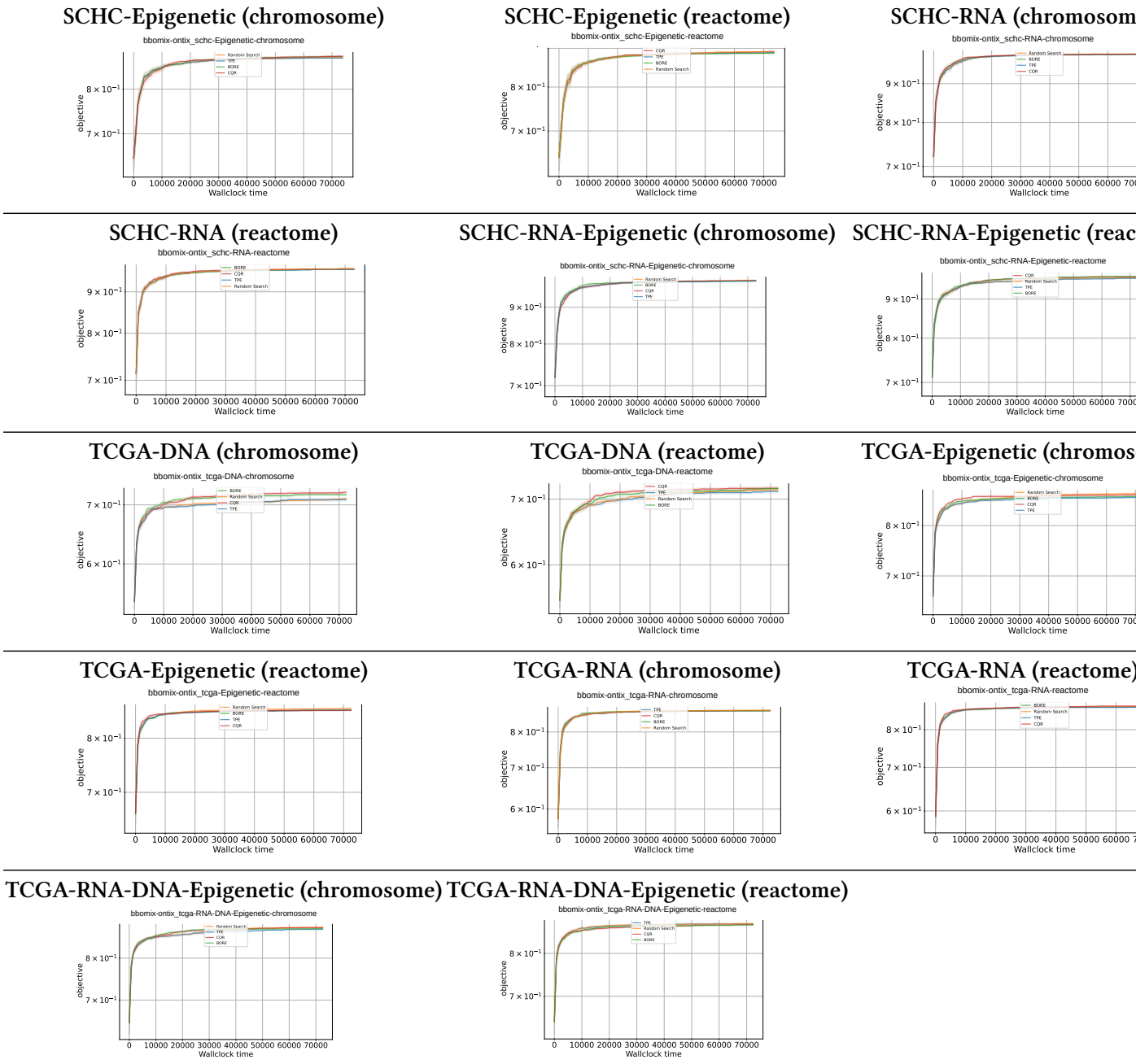


Figure 32: Ontix: optimization trajectories maximizing downstream performance