PRIVILEGED GUIDANCE FOR LEARNING FITNESS LANDSCAPE REPRESENTATIONS FROM LIMITED OBSERVATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning informative fitness landscape representations from limited observations is challenging because sparse function evaluations provide only partial and sampling-sensitive views of the underlying landscape. Although richer observations, function structures, and optimizer behaviors can provide complementary landscape information, they are costly or unavailable when representing a new problem. To address this issue, we propose Privileged Guidance for Fitness Landscape Learning (PGFL), which improves low-budget representation learning by combining consistency across different limited observations with richer information available only during pretraining. Specifically, PGFL employs a landscape encoder that combines pointwise relational encoding with dimension-aware latent aggregation to obtain fixed-length representations applicable across dimensions. Multi-view alignment promotes representation consistency across different sampling designs and transformed function instances. PGFL further exploits higher-budget observations through cross-budget distillation, while function structures and optimizer behaviors provide complementary structural and behavioral supervision for the learned representations. After pretraining, all privileged information and training-only components are removed, and the frozen encoder requires only limited observations of a new function for downstream use. Experiments on the 24 noiseless BBOB functions evaluate PGFL on high-level landscape-property prediction and automated algorithm selection, together with cross-dimensional generalization, component ablations, and representation analyses. The source code is provided in the supplementary material.

1 INTRODUCTION

Characterizing an optimization problem is important for understanding its fitness landscape and selecting suitable optimization algorithms. In continuous black-box optimization, such characterization can only be inferred from sampled solutions and their objective evaluations. Exploratory landscape analysis (ELA) summarizes these observations through numerical descriptors and has been widely used for landscape understanding and algorithm selection (Mersmann et al., 2011; Kerschke & Trautmann, 2019b). However, obtaining landscape information itself consumes function evaluations, which can be non-negligible when the evaluation budget is limited (van der Blom & Vermetten, 2026). It is therefore desirable to learn informative landscape representations from a limited number of observations, before substantial evaluations have been spent on the new problem.

Existing studies characterize sampled landscapes from different perspectives. Classical ELA relies on predefined statistical descriptors, while recent neural approaches learn landscape representations through supervised prediction, reconstruction, or self-supervised pretraining (Seiler et al., 2022; van Stein et al., 2023; Cenikj et al., 2023b; Seiler et al., 2025). Algorithm-dependent information has also been incorporated through performance prediction or short probing trajectories (Cenikj et al., 2024; Renau & Hart, 2024). Despite these advances, learning from limited observations remains challenging. In particular, different sampling designs may provide substantially different views of the same underlying landscape (Renau et al., 2020), while a small number of evaluations may reveal only partial information about its structure and its interaction with optimization algorithms. Simply learning from the available low-budget observations may therefore provide insufficient supervision for obtaining an informative and robust representation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

An important observation is that substantially richer information can be obtained during offline pretraining than when representing a new black-box problem. For generated training functions, one can acquire additional function evaluations, access their construction structures, and run optimizers to observe their search behaviors. Such information is informative for representation learning but is expensive or unavailable at deployment. This setting is closely related to learning using privileged information, where additional training-only information is exploited to improve a model that operates without it at test time (Vapnik & Vashist, 2009; Vapnik & Izmailov, 2015; Lopez-Paz et al., 2015). This motivates us to simultaneously consider two aspects of low-budget landscape representation learning: learning consistency across different limited observations and transferring richer training-only information into the representation extracted from these observations.

To this end, we propose *Privileged Guidance for Fitness Landscape Learning* (PGFL) to learn informative landscape representations from limited observations. The key idea of PGFL is to improve low-budget representation learning by combining representation consistency across different low-budget views with richer information available only during pretraining. Specifically, PGFL employs a landscape representation encoder to characterize sampled landscapes, multi-view alignment to reduce the influence of observation variations, and privileged guidance to exploit higher-budget observations, function structures, and optimizer behaviors. After pretraining, only the landscape encoder is retained to represent new problems from limited observations. The main contributions of this paper are summarized as follows:

- We propose PGFL, a privileged-guidance framework for learning informative fitness landscape representations from limited observations. Its core idea is to jointly exploit consistency across different low-budget observations and richer information available only during pretraining, making the learned encoder benefit from additional supervision without requiring privileged information at deployment.
- To this end, three complementary designs are developed for PGFL. First, pointwise relational encoding and dimension-aware latent aggregation are designed to characterize sampled landscapes across different problem dimensions and observation sizes. Second, sampling-view and instance-view alignment are introduced to reduce representation variation caused by different observations and related function instances. Finally, cross-budget distillation, structural guidance, and optimizer-behavior guidance exploit higher-budget observations, function structures, and optimizer search behaviors, respectively, to enrich the supervision available to the low-budget representation.
- Extensive experiments on the 24 noiseless BBOB functions demonstrate the effectiveness of PGFL on high-level landscape-property prediction and automated algorithm selection, together with additional analyses of its robustness and component effectiveness.

2 Related Work

Landscape characterization and learned representations. Exploratory landscape analysis characterizes continuous black-box optimization problems through numerical descriptors computed from sampled solutions and their objective values (Mersmann et al., 2011). Representative features capture objective-value distributions, local geometry, meta-model relations, and nearest-better structures (Kerschke et al., 2015), and have been widely used for landscape-property prediction and algorithm selection (Kerschke & Trautmann, 2019b). However, their estimates are sensitive to the sampling strategy and sample size (Renau et al., 2020), while additional observations directly increase the evaluation cost (van der Blom & Vermetten, 2026). Recent studies therefore learn landscape representations directly from sampled observations. Representative approaches include feature-free image- and point-based models (Seiler et al., 2022; Prager et al., 2022), transformer-based TransOpt and TransOptAS (Cenikj et al., 2023b; 2024), and the VAE-based DoE2Vec (van Stein et al., 2023). Most closely related, Deep-ELA employs self-supervised contrastive pretraining on augmented landscape views (Seiler et al., 2025). Different from existing methods that mainly learn landscape representations from sampled observations and their augmented views, PGFL further exploits richer information available only during pretraining. Specifically, it combines pointwise relational encoding and dimension-aware latent aggregation with multi-view alignment, and introduces cross-budget distillation, function-structural guidance, and optimizer-behavior guidance. Consequently, the learned encoder can benefit from richer supervision during pretraining while requiring only limited observations of a new problem at deployment.

Privileged information for representation learning. Learning using privileged information (LUPI) exploits additional information available during training but unavailable at deployment (Vap-

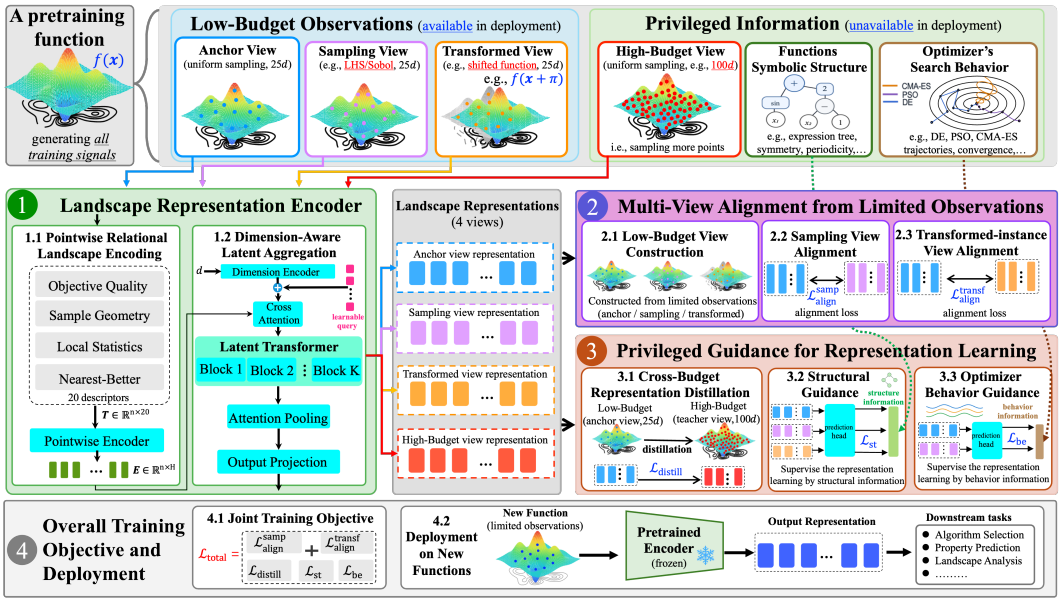


Figure 1: Overview of the proposed PGFL framework, consisting of a landscape representation encoder, multi-view alignment, and privileged guidance.

nik & Vashist, 2009; Vapnik & Izmailov, 2015), and generalized distillation further connects this setting with teacher–student knowledge transfer (Lopez-Paz et al., 2015). Recent studies continue to extend this paradigm. Yang et al. combine privileged information with self-supervised representation learning (Yang et al., 2025), while Martínez-García et al. study privileged knowledge transfer under imperfect teachers through teacher privileged distillation (Martinez-Garcia et al., 2025). Provodin et al. further revisit when privileged information can be effectively transferred to models operating without it (Provodin et al., 2024). Existing LUPI and distillation methods generally transfer training-only information to models that operate without such information at deployment. PGFL specializes this paradigm to fitness landscape representation by using higher-budget observations, function structure, and optimizer behavior as optimization-specific privileged supervision. All privileged information is removed after pretraining, so the learned encoder requires only limited observations and the problem dimension at deployment. More related work on representations of algorithm–problem interactions is provided in **Appendix A.1**.

3 Methodology

Overview. To learn fitness landscape representations from limited observations, we propose PGFL, whose overview is shown in Fig. 1. PGFL mainly consists of a landscape representation encoder, multi-view alignment, and privileged guidance. The encoder maps sampled observations into fixed-length landscape representations through pointwise relational encoding and dimension-aware latent aggregation. Multi-view alignment learns consistent representations from different low-budget observations of the same function, while privileged guidance exploits additional training information, including high-budget observations, function structures, and optimizer behaviors, to enhance low-budget representation learning. The *general procedure of PGFL* is as follows. First, for each pretraining function, three low-budget views, including an anchor view, a sampling view, and a transformed-instance view, are constructed, while a high-budget view, the function structure, and optimizer behaviors are additionally obtained as privileged training information. Second, the views are encoded into their corresponding landscape representations, where the anchor representation is aligned with the sampling and transformed representations. Third, the high-budget representation distills the anchor representation; meanwhile, the function-structural information guides all three low-budget representations, whereas optimizer-behavior information guides the anchor and sampling representations. Finally, these objectives jointly pre-train the landscape encoder. At deployment, all training-only components are removed, and the frozen encoder maps limited observations of a new function into a fixed-length representation for downstream tasks. For clarity, the notation used throughout the methodology is summarized in **Appendix B.1**.

3.1 LANDSCAPE REPRESENTATION ENCODER

Pointwise Relational Landscape Encoding. Consider a minimization objective f and a set of sampled observations $\mathcal{D}_n(f) = \{(\mathbf{x}_i, y_i)\}_{i=1}^n,\ y_i = f(\mathbf{x}_i)$, where $\mathbf{x}_i \in \mathbb{R}^d$ denotes a sampled solution and $y_i \in \mathbb{R}$ is its objective value.

For each sampled point, we construct four groups of descriptors: *(1) objective quality descriptors*, which characterize its relative objective quality within the sampled set; *(2) sample geometry descriptors*, which describe its spatial relations to representative sampled points and local neighbors; *(3) local landscape descriptors*, which capture local objective variation and the distribution of better points at multiple neighborhood scales; and *(4) nearest-better descriptors*, which characterize the spatial and objective relations between the current point and its closest sampled point with a better objective value. Each group contains *five descriptors*, yielding

$$\mathbf{t}_i = [\mathbf{t}_i^{\text{quality}}; \mathbf{t}_i^{\text{geometry}}; \mathbf{t}_i^{\text{local}}; \mathbf{t}_i^{\text{nb}}] \in \mathbb{R}^{20}, \qquad \mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_n]^\top \in \mathbb{R}^{n\times 20}. \tag{1}$$

Detailed definitions and computation of all 20 descriptors are provided in **Appendix B.2**. A shared point encoder then maps all descriptors of each point into an H-dimensional embedding $\mathbf{e}_i$:

$$\mathbf{e}_i = \text{PointEnc}(\mathbf{t}_i) \in \mathbb{R}^H, \qquad \mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_n]^\top \in \mathbb{R}^{n\times H}, \tag{2}$$

where H is the latent dimension (set to 128 in our work), and PointEnc is a two-layer MLP ($20 \to H \to H$) with layer normalization and GELU activation (Hendrycks & Gimpel, 2016).

Dimension-Aware Latent Aggregation. Given the point embeddings $\mathbf{E} \in \mathbb{R}^{n\times H}$, the problem dimension d is encoded as $\mathbf{c}_d$ by applying a learnable dimension encoder DimEnc to $\widehat{d}$:

$$\widehat{d} = \frac{\log(1+d)}{\log(1+d_{\max})},\ \mathbf{c}_d = \text{DimEnc}(\widehat{d}) \in \mathbb{R}^H,\ \text{DimEnc is a two-layer MLP } (1 \to H \to H), \tag{3}$$

where $d_{\max}$ denotes the maximum dimension used for dimension normalization during pretraining. Let $\mathbf{Q} \in \mathbb{R}^{L\times H}$ contain L learnable latent queries. The dimension context is added to latent queries:

$$\mathbf{U}^{(0)} = \mathbf{Q} + \mathbf{1}_L \mathbf{c}_d^\top \in \mathbb{R}^{L\times H}, \tag{4}$$

where $\mathbf{1}_L \in \mathbb{R}^L$ is an all-ones vector, and the obtained $\mathbf{U}^{(0)}$ denotes dimension-aware latent queries.

The dimension-conditioned latent queries $\mathbf{U}^{(0)}$ then aggregate the point embeddings $\mathbf{E}$ through one cross-attention operation to obtain $\mathbf{U}^{(1)}$, followed by K latent Transformer blocks to obtain $\mathbf{U}^{(K+1)}$:

$$\mathbf{U}^{(1)} = \text{LN}\left(\mathbf{U}^{(0)} + \text{MHA}(\mathbf{U}^{(0)}, \mathbf{E}, \mathbf{E})\right),\ \mathbf{U}^{(r+1)} = \text{Block}_r\left(\mathbf{U}^{(r)}\right),\ r = 1, \ldots, K, \tag{5}$$

where $\mathbf{U}^{(r)} \in \mathbb{R}^{L\times H}$, MHA denotes multi-head attention, and LN denotes layer normalization.

Finally, a learnable pooling query $\mathbf{q}_{\text{pool}} \in \mathbb{R}^H$ aggregates the latent representations $\mathbf{U}^{(K+1)}$ by

$$\mathbf{p} = \text{MHA}\left(\mathbf{q}_{\text{pool}}, \mathbf{U}^{(K+1)}, \mathbf{U}^{(K+1)}\right) \in \mathbb{R}^H, \tag{6}$$

and the final landscape representation $\mathbf{z} \in \mathbb{R}^R$ can be obtained by

$$\mathbf{z} = \text{GELU}\left(\mathbf{W}_o\, \text{LN}(\mathbf{p}) + \mathbf{b}_o\right) \in \mathbb{R}^R,\ R = \frac{H}{2} \tag{7}$$

where $\mathbf{W}_o \in \mathbb{R}^{R\times H}$ and $\mathbf{b}_o \in \mathbb{R}^R$ are learnable parameters. The complete landscape encoder is denoted by $\mathbf{z} = e_{\boldsymbol{\theta}}(\mathbf{T}, d)$. Further architectural details are provided in **Appendix B.3**.

3.2 MULTI-VIEW ALIGNMENT FROM LIMITED OBSERVATIONS

Low-Budget View Construction. For the pretraining function f of dimension d introduced above, let $\mathcal{T}$ denote its base expression structure and let n_l denote the number of observations used for each low-budget view. We construct three views:

$$V^A = (\mathbf{X}^A, f(\mathbf{X}^A)),\ V^S = (\mathbf{X}^S, f(\mathbf{X}^S)),\ V^T = (\mathbf{X}^A, f'(\mathbf{X}^A)), \tag{8}$$

where $\mathbf{X}^A, \mathbf{X}^S \in \mathbb{R}^{n_l\times d}$. The Anchor view V^A is obtained by Uniform sampling. The Sampling view V^S evaluates the same function f by sampling $\mathbf{X}^S$ through an independent Latin hypercube or Sobol design (Helton & Davis, 2003). The Transformed-Instance view V^T reuses the Anchor sampling locations $\mathbf{X}^A$ but evaluates a related instance f' generated from the same base expression structure $\mathcal{T}$ at the same dimension d. Thus, A–S differs in the sampled observations of the same function, whereas A–T differs in the function instance while sharing the sampling locations.

Each view is then converted into point descriptors and encoded by the shared landscape encoder:

$$\mathbf{T}^v = \phi(V^v), \qquad \mathbf{z}^v = e_{\boldsymbol{\theta}}(\mathbf{T}^v, d) \in \mathbb{R}^R, \qquad v \in \{A, S, T\}, \tag{9}$$

where $\phi(\cdot)$ denotes the pointwise relational descriptor construction introduced in Sec. 3.1. Detailed function generation and view construction are provided in **Appendix B.4**.

Sampling-View Alignment. For a minibatch containing m pretraining functions, let $g \in \{1, \dots, m\}$ index the functions. The Anchor and Sampling representations of the g-th function are denoted by $\mathbf{z}_g^A$ and $\mathbf{z}_g^S$, respectively. A sampling projector $p_s : \mathbb{R}^R \to \mathbb{R}^H$ maps them to normalized projected representations:

$$\mathbf{u}_g^{A,s} = \mathrm{Norm}(p_s(\mathbf{z}_g^A)), \qquad \mathbf{u}_g^{S,s} = \mathrm{Norm}(p_s(\mathbf{z}_g^S)), \text{ where } \mathrm{Norm}(\mathbf{a}) = \mathbf{a}/\|\mathbf{a}\|_2. \tag{10}$$

For each minibatch of pretraining functions, $(\mathbf{u}_g^{A,s}, \mathbf{u}_g^{S,s})$ forms a positive pair, whereas projected representations from the other functions in the minibatch serve as negatives. As a result, the sampling-view loss is

$$\mathcal{L}_{align}^{\mathrm{samp}} = \mathcal{L}_{\mathrm{con}}(\{\mathbf{u}_g^{A,s}, \mathbf{u}_g^{S,s}\}_{g=1}^m). \tag{11}$$

Instance-View Alignment. For the same minibatch, the Anchor and Transformed-Instance representations of the g-th function are denoted by $\mathbf{z}_g^A$ and $\mathbf{z}_g^T$. A separate instance projector $p_t : \mathbb{R}^R \to \mathbb{R}^H$ produces

$$\mathbf{u}_g^{A,t} = \mathrm{Norm}(p_t(\mathbf{z}_g^A)), \qquad \mathbf{u}_g^{T,t} = \mathrm{Norm}(p_t(\mathbf{z}_g^T)). \tag{12}$$

Similarly, $(\mathbf{u}_g^{A,t}, \mathbf{u}_g^{T,t})$ forms a positive pair, while others serve as negatives, and the corresponding loss is

$$\mathcal{L}_{align}^{\mathrm{trans}} = \mathcal{L}_{\mathrm{con}}(\{\mathbf{u}_g^{A,t}, \mathbf{u}_g^{T,t}\}_{g=1}^m). \tag{13}$$

The two alignment branches use independent projectors p_s and p_t, while both optimize the shared landscape encoder. Detailed contrastive loss and projection-head are provided in **Appendix B.5** and **B.6**.

3.3 Privileged Guidance for Low-Budget Representation Learning

Multi-view alignment learns consistency among low-budget observations. During pretraining, the generated functions additionally provide information that is unavailable for a new black-box problem. We exploit three such sources of privileged guidance: higher-budget observations, function structure, and optimizer behavior.

Cross-Budget Representation Distillation. For the function f and its Anchor observation $\mathbf{X}^A \in \mathbb{R}^{n_l \times d}$, let $n_h > n_l$ denote a higher observation budget. We append additional samples $\mathbf{X}^+ \in \mathbb{R}^{(n_h - n_l) \times d}$ to construct

$$\mathbf{X}^H = \mathbf{X}^A \,\|\, \mathbf{X}^+, \qquad V^H = (\mathbf{X}^H, f(\mathbf{X}^H)), \text{ where } \| \text{ is row-wise concatenation.} \tag{14}$$

Thus, V^A and V^H evaluate the same function instance, and the high-budget view contains Anchor observations. Then, the high-budget view V^H is encoded to obtain the high-budget landscape representation $\bar{\mathbf{z}}^H$ by an EMA teacher $e_{\bar{\boldsymbol{\theta}}}(\cdot)$, which has the same landscape-encoder as the student but holds a separate set of parameters $\bar{\boldsymbol{\theta}}$:

$$\bar{\mathbf{z}}^H = e_{\bar{\boldsymbol{\theta}}}(\phi(V^H), d), \tag{15}$$

whereas $\mathbf{z}^A$ is obtained from the online student encoder $e_{\boldsymbol{\theta}}(\cdot)$. Let p_b denote a budget projector and q_b a student-side predictor. Their normalized outputs are

$$\mathbf{u}^{A,b} = \mathrm{Norm}(q_b(p_b(\mathbf{z}^A))), \qquad \bar{\mathbf{u}}^{H,b} = \mathrm{sg}[\mathrm{Norm}(\bar{p}_b(\bar{\mathbf{z}}^H))], \tag{16}$$

where $\bar{p}_b$ is the EMA counterpart of p_b and $\mathrm{sg}(\cdot)$ denotes stop-gradient. For a minibatch containing m pretraining functions, the cross-budget distillation loss is

$$\mathcal{L}_{\mathrm{distill}} = 1 - \frac{1}{m} \sum_{g=1}^m (\mathbf{u}_g^{A,b})^\top \bar{\mathbf{u}}_g^{H,b}. \tag{17}$$

Let $\boldsymbol{\Theta}_b$ collect the online student encoder and budget-projector parameters, and let $\bar{\boldsymbol{\Theta}}_b$ denote their teacher counterparts. The teacher is updated by $\bar{\boldsymbol{\Theta}}_b \leftarrow \mu_t \bar{\boldsymbol{\Theta}}_b + (1 - \mu_t)\boldsymbol{\Theta}_b$, where $\mu_t \in [0, 1)$ is the EMA momentum. The predictor q_b belongs only to the student branch. More details can be found in **Appendix B.7**.

Structural Guidance. Each generated function f is associated with the base expression structure $\mathcal{T}$ introduced in Sec. 3.2. We summarize $\mathcal{T}$ by a continuous structural target $\mathbf{s} = \mathrm{Struct}(\mathcal{T}) \in [0, 1]^{D_s}$, where D_s denotes the structural-target dimension. In this work, $D_s = 8$ structural attributes are considered to characterize complementary properties of the generated structure: *(1) expression complexity*, *(2) operator entropy*, *(3) smooth-nonlinear ratio*, *(4) nonlinear nesting*, *(5) periodic-operator ratio*, *(6) nonsmooth-operator ratio*, *(7) multiplicative/divisive-operator ratio*, and *(8) coordinate-structure ratio*. These attributes provide a compact description of how the generated function is constructed without reconstructing the complete expression tree.

A structural prediction head $h_s : \mathbb{R}^R \to [0,1]^{D_s}$ maps each low-budget representation to

$$\widehat{\mathbf{s}}^{\,v} = h_s(\mathbf{z}^v), \qquad v \in \mathcal{V}_s = \{A, S, T\}. \tag{18}$$

We implement h_s as a two-layer MLP with a GELU hidden activation and a Sigmoid output. Since the three views are derived from the same base expression structure, they share the same structural target. For a minibatch of m functions, the structural guidance loss is

$$\mathcal{L}_{\mathrm{st}} = \frac{1}{m|\mathcal{V}_s|} \sum\nolimits_{g=1}^{m} \sum\nolimits_{v \in \mathcal{V}_s} \rho(\widehat{\mathbf{s}}_g^{\,v}, \mathbf{s}_g), \tag{19}$$

where $\rho(\cdot,\cdot)$ denotes the coordinate-averaged Smooth L1 loss. The structural target is used only as privileged pretraining supervision and does not use downstream landscape-property labels. Details of the eight structural attributes and the structural-head architecture are provided in **Appendix B.8**.

Optimizer-Behavior Guidance. We further obtain privileged supervision from offline interactions between the generated function and a set of probe optimizers $\mathcal{A}_{\mathrm{probe}}$. For each optimizer $a \in \mathcal{A}_{\mathrm{probe}}$, let $\Gamma_a(f)$ denote its short search trajectory on f. The trajectory is summarized into an optimizer-specific behavioral target

$$\mathbf{b}_a = \mathrm{Beh}\left(\Gamma_a(f)\right) \in \mathbb{R}^{D_b}, \tag{20}$$

where D_b denotes the behavioral-target dimension. The resulting target characterizes how optimizer a responds to the underlying landscape during its search process.

To predict optimizer-specific behavior from a common landscape representation, each probe optimizer is assigned a learnable identity embedding $\mathbf{e}_a \in \mathbb{R}^{D_e}$, where D_e denotes the embedding width. A shared behavioral prediction head maps the concatenated landscape representation and optimizer embedding to

$$\widehat{\mathbf{b}}_a^{\,v} = h_b([\mathbf{z}^v; \mathbf{e}_a]) \in \mathbb{R}^{D_b}. \tag{21}$$

Behavioral supervision is applied to $\mathcal{V}_b = \{A, S\}$, because both views evaluate the same function instance f on which the probe trajectories are generated. For a minibatch of m functions, the behavioral guidance loss is

$$\mathcal{L}_{\mathrm{be}} = \frac{1}{m|\mathcal{V}_b||\mathcal{A}_{\mathrm{probe}}|} \sum\nolimits_{g=1}^{m} \sum\nolimits_{v \in \mathcal{V}_b} \sum\nolimits_{a \in \mathcal{A}_{\mathrm{probe}}} \rho(\widehat{\mathbf{b}}_{g,a}^{\,v}, \mathbf{b}_{g,a}), \tag{22}$$

where $\rho(\cdot,\cdot)$ denotes the coordinate-averaged Smooth L1 loss. The Transformed-Instance view is excluded because it evaluates the related instance f' rather than the function f used to generate the behavioral targets.

In this work, $\mathcal{A}_{\mathrm{probe}}$ consists of DE/rand/1/bin, global-best PSO, and CMA-ES. We set $D_b = 8$ and characterize each optimizer trajectory using *(1–4) normalized best-so-far performance at four search stages*, *(5) mean trajectory performance*, *(6) improvement frequency*, *(7) population-diversity contraction*, and *(8) stagnation ratio*. These behavioral targets serve only as privileged pretraining signals rather than labels for the downstream algorithm-selection task. Detailed definitions and trajectory summarization are provided in **Appendix B.9**.

3.4 Overall Training Objective and Deployment

The two multi-view alignment loss and three privileged-guidance loss jointly optimize the landscape encoder:

$$\mathcal{L} = \lambda_{\mathrm{s}}\mathcal{L}_{\mathrm{align}}^{\mathrm{samp}} + \lambda_{\mathrm{t}}\mathcal{L}_{\mathrm{align}}^{\mathrm{trans}} + \alpha_t\left(\lambda_{\mathrm{d}}\mathcal{L}_{\mathrm{distill}} + \lambda_{\mathrm{g}}(\mathcal{L}_{\mathrm{st}} + \mathcal{L}_{\mathrm{be}})\right),\ \alpha_t = \min(1, (t+1)/T_w), \tag{23}$$

where λ_{s}, λ_{t}, λ_{d}, and λ_{g} control the corresponding objectives, and T_w denotes the warm-up horizon. The two alignment objectives are active throughout pretraining, whereas α_t gradually introduces the three privileged-guidance objectives. The student encoder and online training heads are optimized by backpropagation, while the EMA teacher is updated only through the EMA rule. The concrete loss weights and training settings are reported in the experimental settings. Detailed loss and training configuration are in **Appendix B.10**.

After pretraining, only the point-descriptor construction ϕ and the student landscape encoder $e_{\boldsymbol{\theta}}$ are retained. Given a new black-box function with n_l observations, $\mathbf{z} = e_{\boldsymbol{\theta}}(\phi(\mathbf{X}, \mathbf{y}), d) \in \mathbb{R}^R$. The high-budget observations, EMA teacher, base expression structure, optimizer trajectories, projection heads, predictors, and auxiliary prediction heads are discarded. The frozen representation $\mathbf{z}$ can then be directly used by downstream landscape-analysis models. More details are in **Appendix B.10**.

4 Experiments

This section will answer the following research questions: **RQ1**: How effective are the representations learned by PGFL compared with existing landscape representations? **RQ2**: How well does PGFL generalize to unseen problem dimensions? **RQ3**: How do the major components of PGFL affect performance? **RQ4**: How does the representation learned by PGFL differ from conventional ELA features in terms of redundancy and cross-instance stability? **RQ5**: How does the sampling budget affect PGFL?

Table 1: HLP prediction results on BBOB (macro-F1 ↑; – means the result is unavailable).

Property	D	ELA	Deep-ELA		VAE-24	VAE-32	PCT	PGFL
		$n = 25d$	$n = 25d$	$n = 50d$	$n = 256$	$n = 256$	$n = 100$	$n = 25d$
Multi-modality	2	$0.91_{\pm 0.02}$	$0.89_{\pm 0.02}$	$0.94_{\pm 0.01}$	$0.82_{\pm 0.03}$	$0.82_{\pm 0.02}$	$\mathbf{0.99}_{\pm 0.01}$	$0.98_{\pm 0.02}$
	3	$0.92_{\pm 0.01}$	$0.92_{\pm 0.02}$	$0.93_{\pm 0.01}$	–	–	$0.96_{\pm 0.01}$	$\mathbf{0.97}_{\pm 0.01}$
	5	$0.92_{\pm 0.01}$	$0.92_{\pm 0.01}$	$0.93_{\pm 0.01}$	$0.84_{\pm 0.01}$	$0.85_{\pm 0.01}$	$0.93_{\pm 0.00}$	$\mathbf{0.94}_{\pm 0.01}$
	10	$0.91_{\pm 0.01}$	$0.93_{\pm 0.01}$	$0.92_{\pm 0.01}$	$0.73_{\pm 0.02}$	$0.72_{\pm 0.02}$	$0.83_{\pm 0.04}$	$\mathbf{0.94}_{\pm 0.02}$
	overall	$0.92_{\pm 0.02}$	$0.92_{\pm 0.02}$	$0.93_{\pm 0.01}$	–	–	$0.93_{\pm 0.01}$	$\mathbf{0.95}_{\pm 0.01}$
Global structure	2	$0.93_{\pm 0.01}$	$0.92_{\pm 0.02}$	$0.96_{\pm 0.01}$	$0.85_{\pm 0.02}$	$0.86_{\pm 0.02}$	$\mathbf{0.98}_{\pm 0.01}$	$0.94_{\pm 0.01}$
	3	$0.92_{\pm 0.01}$	$0.93_{\pm 0.01}$	$0.90_{\pm 0.01}$	–	–	$0.95_{\pm 0.01}$	$\mathbf{0.96}_{\pm 0.01}$
	5	$0.92_{\pm 0.01}$	$0.94_{\pm 0.01}$	$\mathbf{0.95}_{\pm 0.01}$	$0.71_{\pm 0.05}$	$0.71_{\pm 0.06}$	$0.90_{\pm 0.01}$	$0.94_{\pm 0.01}$
	10	$0.91_{\pm 0.02}$	$0.91_{\pm 0.01}$	$\mathbf{0.94}_{\pm 0.01}$	$0.63_{\pm 0.04}$	$0.61_{\pm 0.04}$	$0.82_{\pm 0.01}$	$\mathbf{0.94}_{\pm 0.01}$
	overall	$0.92_{\pm 0.01}$	$0.93_{\pm 0.02}$	$0.94_{\pm 0.02}$	–	–	$0.91_{\pm 0.01}$	$\mathbf{0.95}_{\pm 0.01}$
Funnel structure	2	$0.98_{\pm 0.01}$	$0.97_{\pm 0.01}$	$\mathbf{0.99}_{\pm 0.01}$	$0.95_{\pm 0.02}$	$0.96_{\pm 0.02}$	$\mathbf{0.99}_{\pm 0.01}$	$\mathbf{0.99}_{\pm 0.01}$
	3	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	–	–	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$
	5	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$0.95_{\pm 0.03}$	$0.96_{\pm 0.03}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$
	10	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$0.91_{\pm 0.02}$	$0.91_{\pm 0.02}$	$0.98_{\pm 0.02}$	$\mathbf{1.00}_{\pm 0.00}$
	overall	$0.99_{\pm 0.01}$	$0.99_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.01}$	–	–	$0.99_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.01}$
$+/\approx/-$		9/3/0	8/4/0	5/6/1	9/0/0	9/0/0	7/3/2	–

4.1 Experimental Setup

Pretraining and implementation. We pretrain PGFL on 120,000 noiseless random expression-tree functions over $[-5, 5]^d$, with $d \in \{2, 3, 5, 10\}$ across the pretraining set. Each low-budget view uses n=$25d$ function evaluations, while the nested high-budget teacher view uses $100d$ evaluations. The landscape encoder uses hidden width H=128, L=32 latent queries, K=4 latent Transformer blocks, and representation width R=64. Training uses a minibatch of 32 pretraining functions for 3,750 updates with AdamW and a learning rate of $3{*}10^{-4}$. The objective weights $(\lambda_{\mathrm{s}}, \lambda_{\mathrm{t}}, \lambda_{\mathrm{d}}, \lambda_{\mathrm{g}})$ are set to $(1, 0.5, 1, 0.5)$, with the privileged-guidance objectives linearly warmed up during the first 10% of training. The exported encoder is selected using an independently generated validation set of 256 functions and is frozen in all downstream experiments. Detailed function-generation, architecture, privileged-target, and optimization settings are provided in **Appendix C.1**.

Benchmarks and baselines. We evaluate on the 24 BBOB functions. Experiments consider $d \in \{2, 3, 5, 10\}$ and use $25d$ evaluations to construct each representation. Results are evaluated over ten sampling seeds using Uniform, Latin hypercube, and Sobol sampling. We compare PGFL with ELA, Deep-ELA, two DoE2Vec variants (VAE-24 and VAE-32), point cloud transformer (PCT) (Seiler et al., 2022), and single best solver (SBS) that applies the portfolio member with the best aggregate performance to every problem instance (Bischl et al., 2016). ELA and Deep-ELA are evaluated with both $25d$ and $50d$ observations, whereas DoE2Vec uses its 256-evaluation setting and PCT uses $n = 100$; the latter two are therefore not treated as strictly cost-matched comparisons. Implementation and sampling details for all baselines are in **Appendix C.2**.

Downstream evaluation. The frozen representations are evaluated on high-level landscape-property (HLP) prediction and automated algorithm selection (AAS) (Kerschke & Trautmann, 2019a). HLP predicts multi-modality, global structure, and funnel structure using random forests and is evaluated by macro-F1 (Opitz & Burst, 2019). For AAS, random-forest (RF) and distance-weighted kNN selectors choose from a fixed optimizer portfolio. We use relative expected running time $\mathrm{relERT} = \frac{C_{\mathrm{feat}} + \mathrm{ERT}(a_{\mathrm{selected}})}{\mathrm{ERT}(a_{\mathrm{VBS}})}$, where C_{feat} includes the function evaluations used to construct the representation. Lower relERT is better. The data splits, solver portfolio, selector configurations, solver budgets, and failure handling are detailed in **Appendix C.3**.

4.2 **RQ1**: Overall Performance on Downstream Tasks

HLP Task. To evaluate the effectiveness of PGFL in the HLP task, we compare it with the considered landscape representations over the BBOB functions. The results are reported in Table 1, where the Wilcoxon rank-sum test (Derrac et al., 2011) is conducted at a significance level of 0.05 and $+/\approx/-$ indicates that PGFL performs significantly better than, statistically similarly to, or significantly worse than the compared method, respectively. As can be observed, PGFL achieves the best or comparable performance in most property–dimension settings and consistently outperforms the cost-matched ELA and Deep-ELA in most comparisons. Moreover, PGFL remains competitive with Deep-ELA using twice the observation budget and PCT under its available setting, while inferior results occur only in a few cases. These results demonstrate the effectiveness of the representations learned by PGFL for high-level landscape-property prediction.

AAS Task. For the AAS task, we compare PGFL with the considered methods using RF and kNN selectors, with the relERT results reported in Table 2. The same Wilcoxon rank-sum test is used for statistical comparison. As can be observed, PGFL achieves the best or competitive relERT in most dimension–selector settings and obtains strong aggregate performance with both RF and kNN. Only a few exceptions are observed, such as RF at $d = 10$ and kNN at $d = 5$, where some compared methods achieve lower relERT. These results demonstrate the effectiveness of PGFL for automated algorithm selection.

Table 2: AAS results on BBOB (relERT ↓; mean ± std.). Best means are bold.

D	SBS	ELA				Deep-ELA				VAE-24		VAE-32		PCT	PGFL	
		$25d$		$50d$		$25d$		$50d$		256		256		100	$25d$	
		RF	*k*NN	RF	*k*NN	RF	*k*NN	RF	*k*NN	RF	*k*NN	RF	*k*NN		RF	*k*NN
2	19.68	17.32 ±4.85	30.89 ±13.17	16.85 ±3.97	20.07 ±8.83	17.13 ±3.24	22.72 ±11.43	17.36 ±5.18	17.50 ±7.49	25.18 ±5.12	33.62 ±10.11	25.65 ±7.42	34.17 ±10.39	17.61 ±0.75	16.37 ±3.24	**16.00** ±9.97
3	21.09	18.70 ±5.00	17.55 ±7.90	21.66 ±3.10	21.24 ±6.24	18.55 ±3.75	17.54 ±6.16	18.27 ±4.75	18.77 ±4.78	–	–	–	–	18.34 ±5.00	**17.18** ±4.46	17.34 ±5.15
5	6.69	6.49 ±0.71	11.87 ±4.43	8.13 ±0.57	13.36 ±8.28	6.34 ±0.66	10.16 ±8.02	7.40 ±0.57	17.54 ±11.06	7.98 ±1.01	15.12 ±7.52	7.68 ±0.67	12.09 ±2.36	8.03 ±0.34	**6.03** ±0.75	11.25 ±4.33
10	**6.24**	7.41 ±1.88	11.75 ±4.85	9.15 ±1.76	13.67 ±5.31	6.82 ±1.75	12.64 ±9.84	10.22 ±1.32	16.16 ±12.52	9.17 ±1.47	17.75 ±5.33	8.73 ±1.55	15.80 ±4.16	11.54 ±0.17	7.62 ±1.69	8.95 ±2.00
overall	13.43	12.48 ±1.71	18.02 ±4.22	13.95 ±1.60	17.09 ±2.79	12.21 ±1.15	15.77 ±4.00	13.31 ±1.33	17.49 ±5.16	–	–	–	–	13.88 ±1.25	**11.80** ±1.37	13.39 ±2.36
+/ ≈ /−		3/0/1	4/0/0	4/0/0	4/0/0	3/0/1	3/0/1	4/0/0	4/0/0	3/0/0	3/0/0	3/0/0	3/0/0	4/0/0	–	–

4.3 **RQ2**: Cross-Dimensional Generalization

To evaluate the cross-dimensional generalization of PGFL, we apply the frozen encoder to the unseen dimension $d = 20$, which is twice the largest dimension used during pretraining, without updating its parameters. Table 3 reports the HLP and AAS results of PGFL and ELA under different observation budgets. As can be observed, PGFL remains competitive on HLP with only $25d$ observations, while achieving the best AAS results with both RF and kNN, outperforming ELA with either $25d$ or $50d$ observations. These results demonstrate that PGFL can effectively generalize its learned landscape representations to unseen problem dimensions without adapting the pretrained encoder.

Table 3: Cross-dimensional generalization to the unseen dimension $d = 20$ (HLP: macro-F1 ↑; AAS: relERT ↓; mean ± std.). Best means are bold.

Method	Budget	HLP ↑				AAS ↓	
		Overall	Multi.	Global	Funnel	RF	*k*NN
SBS	–	–				7.82	
ELA	$25d$	$0.94_{\pm 0.01}$	$0.92_{\pm 0.01}$	$0.90_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.00}$	$3.96_{\pm 0.29}$	$7.88_{\pm 5.21}$
	$50d$	$\mathbf{0.95}_{\pm 0.01}$	$\mathbf{0.93}_{\pm 0.01}$	$\mathbf{0.92}_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.00}$	$5.18_{\pm 0.10}$	$6.67_{\pm 1.10}$
PGFL	$25d$	$0.94_{\pm 0.01}$	$\mathbf{0.93}_{\pm 0.01}$	$0.91_{\pm 0.02}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{3.91}_{\pm 0.19}$	$\mathbf{6.39}_{\pm 3.40}$

4.4 **RQ3**: Component Effectiveness

Table 4: Ablation results of PGFL (HLP: macro-F1 ↑; AAS: relERT ↓; mean ± std.). Best means are bold.

Variant	HLP ↑				AAS ↓	
	Overall	Multi.	Global	Funnel	RF	*k*NN
w/o multi-view	$0.86_{\pm 0.01}$	$0.80_{\pm 0.02}$	$0.78_{\pm 0.02}$	$0.99_{\pm 0.00}$	$12.79_{\pm 1.25}$	$16.23_{\pm 4.04}$
w/o budget distillation	$0.93_{\pm 0.01}$	$0.90_{\pm 0.01}$	$0.91_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.00}$	$12.57_{\pm 1.24}$	$13.97_{\pm 2.66}$
w/o structural guidance	$0.93_{\pm 0.01}$	$0.91_{\pm 0.01}$	$0.91_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.00}$	$12.52_{\pm 1.55}$	$14.20_{\pm 1.76}$
w/o behavior guidance	$0.92_{\pm 0.01}$	$0.89_{\pm 0.01}$	$0.9_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.00}$	$12.61_{\pm 1.43}$	$14.35_{\pm 1.62}$
PGFL	$\mathbf{0.97}_{\pm 0.01}$	$\mathbf{0.95}_{\pm 0.01}$	$\mathbf{0.95}_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.01}$	$\mathbf{11.80}_{\pm 1.37}$	$\mathbf{13.39}_{\pm 2.36}$

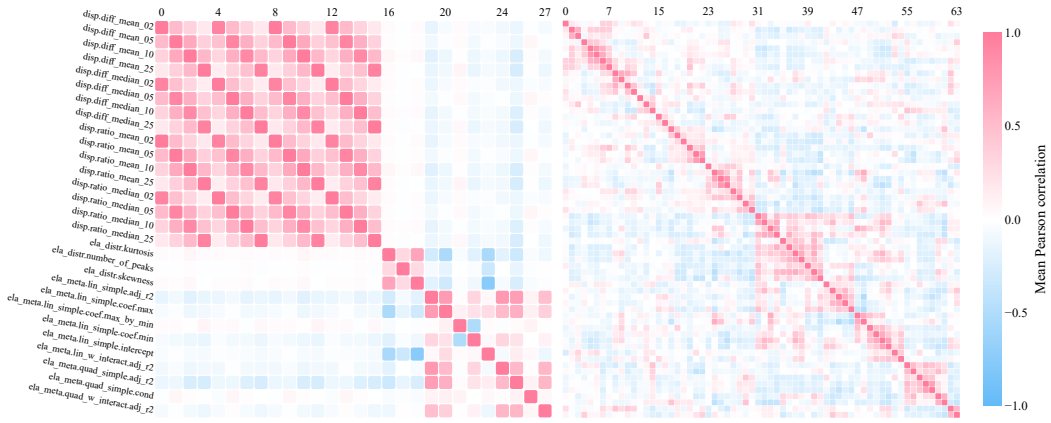
To evaluate the contribution of the major pretraining components, we construct four variants by removing multi-view alignment, cross-budget distillation, structural guidance, and behavior guidance, respectively, while keeping the remaining settings unchanged. The corresponding HLP and AAS results are reported in Table 4. As can be observed, the complete PGFL achieves the best overall performance on both tasks, while removing any component leads to performance degradation; among them, removing multi-view alignment causes the most evident decrease, whereas cross-budget distillation and the two privileged-guidance objectives also provide consistent improvements. These results demonstrate that the proposed components are all effective and provide complementary information for learning informative landscape representations.

4.5 **RQ4**: Understanding the Learned Landscape Representation

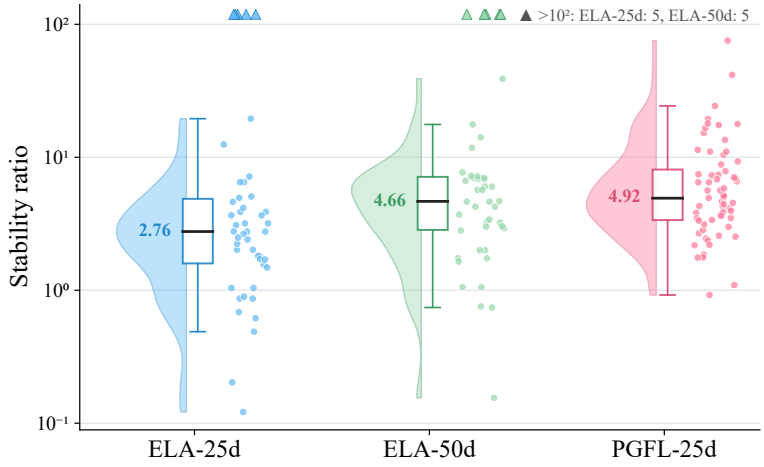
To further investigate the characteristics of the learned representation, we compare PGFL with ELA in terms of coordinate redundancy and cross-instance stability on the 24 BBOB functions. Figure 2 presents the feature-correlation matrices and the centered landscape-stability distributions under the corresponding analysis settings. As can be observed, PGFL exhibits substantially weaker off-diagonal correlations than ELA, while its median stability ratio reaches 4.92, compared with 2.76 for the cost-matched ELA-$25d$ and 4.66 for ELA-$50d$. These results indicate that PGFL learns landscape representations with less redundant coordinates and stronger cross-instance stability, providing additional evidence for the quality of its learned representations.

4.6 **RQ5**: Effect of the Sampling Budget

To investigate the effect of the sampling budget on PGFL, we evaluate it with $10d$, $25d$, and $100d$ observations while keeping the downstream protocols unchanged. Table 5 reports the corresponding HLP and AAS results, where relERT additionally accounts for the representation cost incurred by different sampling budgets. As can be observed, $25d$ provides the best overall trade-off: it substantially improves HLP over $10d$ and achieves the best kNN result, whereas increasing the budget to $100d$ brings no further HLP gain and clearly worsens AAS;



(a) Feature-correlation analysis using the 28-feature ELA subset and all 64 PGFL coordinates. Weaker off-diagonal Pearson correlations indicate lower coordinate redundancy.



(b) Cross-instance stability analysis using 46 non-runtime ELA features and all 64 PGFL coordinates. Higher between-function/within-function variance ratios indicate stronger cross-instance stability.

Figure 2: Representation redundancy and cross-instance stability of ELA and PGFL on BBOB. Both analyses use all 24 functions, instances 1–20, and $d \in \{2, 3, 5, 10\}$ with Uniform sampling.

Table 5: Effect of the sampling budget on PGFL (HLP: macro-F1 ↑; AAS: relERT ↓; mean ± std.).

Budget	HLP ↑				AAS ↓	
	Overall	Multi.	Global	Funnel	RF	*k*NN
$10d$	$0.84_{\pm 0.01}$	$0.79_{\pm 0.01}$	$0.76_{\pm 0.02}$	$0.98_{\pm 0.01}$	$\mathbf{11.39}_{\pm 0.65}$	$14.77_{\pm 4.92}$
$100d$	$0.94_{\pm 0.01}$	$0.91_{\pm 0.01}$	$0.92_{\pm 0.02}$	$\mathbf{1.00}_{\pm 0.00}$	$16.60_{\pm 1.43}$	$15.79_{\pm 1.10}$
$\mathbf{25d}$	$\mathbf{0.97}_{\pm 0.01}$	$\mathbf{0.95}_{\pm 0.01}$	$\mathbf{0.95}_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.01}$	$11.80_{\pm 1.37}$	$\mathbf{13.39}_{\pm 2.36}$

the $10d$ setting remains slightly better with RF due to its lower representation cost. These results support $25d$ as the default sampling budget of PGFL. ***Additional results*** of the sensitivity to the training loss weights is further analyzed in **Appendix D.1.**

5 Conclusion

In this paper, we proposed PGFL for learning informative fitness landscape representations from limited observations. PGFL combines multi-view alignment with privileged guidance from higher-budget observations, function structures, and optimizer behaviors, while requiring only low-budget observations at deployment. Experiments on the BBOB suite demonstrate the effectiveness of the learned representations for landscape-property prediction and automated algorithm selection, together with promising generalization to an unseen problem dimension and consistent benefits from the proposed components. Future work will extend PGFL to more diverse and higher-dimensional black-box optimization problems.

AI USE STATEMENT

During the preparation of this manuscript, the authors used GPT-6 solely as an auxiliary tool for language polishing, grammar and spelling checking, and improving the consistency of terminology and descriptions. The LLM was not used to generate research ideas, design methods, conduct data analysis, interpret results, or draw conclusions. All LLM-assisted outputs were reviewed, verified, and revised by the authors, who take full responsibility for the content and claims of this paper.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or sensitive data. The experiments are conducted using synthetically generated functions and the standard BBOB benchmark. PGFL is a general-purpose methodology for black-box optimization; any deployment in safety-critical or socially consequential applications would require separate domain-specific assessment of potential risks and impacts.

REPRODUCIBILITY STATEMENT

We provide the methodological and experimental details needed to reproduce the results throughout the paper and supplementary material. The methodology section specifies the landscape encoder, multi-view alignment, the three privileged-guidance objectives, and the overall training objective. The experimental setup describes the pretraining-function generation, observation budgets, model and optimization configurations, BBOB evaluation protocols, downstream HLP and AAS settings, baseline configurations, and random-seed conventions. The appendix further provides detailed definitions of the pointwise relational descriptors, low- and high-budget view construction, contrastive alignment, cross-budget distillation, structural and optimizer-behavior targets, training and deployment procedures, as well as additional sensitivity analyses. The source code and implementation materials required to reproduce the experiments are included in the supplementary material.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL https://arxiv.org/abs/1607.06450.

Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, et al. Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016.

Peter A. N. Bosman, Jörn Grahl, and Dirk Thierens. AMaLGaM IDEAs in noiseless black-box optimization benchmarking. In *Genetic and Evolutionary Computation Conference Companion*, 2009. URL https://sci2s.ugr.es/sites/default/files/files/TematicWebSites/EAMHCO/contributionsGECCO09/p2247-bosman.pdf.

Gjorgjina Cenikj, Gašper Petelin, Carola Doerr, Peter Korošec, and Tome Eftimov. DynamoRep: Trajectory-based population dynamics for classification of black-box optimization problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023a. doi: 10.1145/3583131.3590401.

Gjorgjina Cenikj, Gašper Petelin, and Tome Eftimov. TransOpt: Transformer-based representation learning for optimization problem classification. *arXiv preprint arXiv:2311.18035*, 2023b. URL https://arxiv.org/abs/2311.18035.

Gjorgjina Cenikj, Gašper Petelin, and Tome Eftimov. TransOptAS: Transformer-based algorithm selection for single-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 403–406, 2024. doi: 10.1145/3638530.3654191.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607, 2020. URL https://proceedings.mlr.press/v119/chen20j.html.

Maurice Clerc and James Kennedy. The particle swarm—explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002. doi: 10.1109/4235.985692.

Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and evolutionary computation*, 1(1):3–18, 2011.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent—a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html.

Nikolaus Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *Genetic and Evolutionary Computation Conference Companion*, 2009. URL https://sci2s.ugr.es/sites/default/files/files/TematicWebSites/EAMHCO/contributionsGECCO09/p2389-hansen.pdf.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016. URL https://arxiv.org/abs/1604.00772.

Jon C Helton and Freddie Joe Davis. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliability Engineering & System Safety*, 81(1):23–69, 2003.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016. URL https://arxiv.org/abs/1606.08415.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/abs/1503.02531.

Pascal Kerschke and Heike Trautmann. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolutionary computation*, 27(1):99–127, 2019a.

Pascal Kerschke and Heike Trautmann. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolutionary Computation*, 27(1):99–127, 2019b. doi: 10.1162/evco_a_00236.

Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. Detecting funnel structures by means of exploratory landscape analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 265–272, 2015. doi: 10.1145/2739480.2754642.

Peter Korošec and Tome Eftimov. Opt2Vec – a continuous optimization problem representation based on the algorithm's behavior: A case study on problem classification. *Information Sciences*, 680:121134, 2024. doi: 10.1016/j.ins.2024.121134.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. BI-population CMA-ES algorithms with surrogate models and line searches. In *Genetic and Evolutionary Computation Conference Companion*, 2013. URL https://numbbo.github.io/gforge/data/media/pdf2013/w0306-loshchilov.pdf.

Zeyuan Ma, Jiacheng Chen, Hongshu Guo, and Yue-Jiao Gong. Neural exploratory landscape analysis for meta-black-box-optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=EEI5R89Cmv.

Mario Martinez-Garcia, Inaki Inza, and Jose A Lozano. Teacher privileged distillation: How to deal with imperfect teachers? *Knowledge-Based Systems*, 316:113338, 2025.

Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 829–836, 2011. doi: 10.1145/2001576.2001690.

Michał Okulewicz and Mateusz Zaborski. Benchmarking SHADE algorithm enhanced with model based optimization on the BBOB noiseless testbed. In *Genetic and Evolutionary Computation Conference Companion*, 2021. doi: 10.1145/3449726.3463290.

Juri Opitz and Sebastian Burst. Macro f1 and macro f1. *arXiv preprint arXiv:1911.03347*, 2019.

M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964. doi: 10.1093/comjnl/7.2.155.

M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in Optimization and Numerical Analysis*, pp. 51–67. Kluwer Academic Publishers, 1994. doi: 10.1007/978-94-015-8330-5_4.

Raphael Patrick Prager, Moritz Vinzent Seiler, Heike Trautmann, and Pascal Kerschke. Automated algorithm selection in single-objective continuous optimization: A comparative study of deep learning and landscape analysis methods. In *Parallel Problem Solving from Nature – PPSN XVII*, pp. 3–17, 2022. doi: 10.1007/978-3-031-14714-2_1.

Danil Provodin, Bram van den Akker, Christina Katsimerou, Maurits Kaptein, and Mykola Pechenizkiy. Rethinking knowledge transfer in learning using privileged information. *arXiv preprint arXiv:2408.14319*, 2024.

Quentin Renau and Emma Hart. On the utility of probing trajectories for algorithm-selection. In *Applications of Evolutionary Computation*, pp. 98–114, 2024. doi: 10.1007/978-3-031-56852-7_7.

Quentin Renau, Carola Doerr, Johann Dréo, and Benjamin Doerr. Exploratory landscape analysis is strongly sensitive to the sampling strategy. In *Parallel Problem Solving from Nature – PPSN XVI*, 2020. doi: 10.1007/978-3-030-58115-2_10.

Luis Miguel Rios and Nikolaos V. Sahinidis. Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013. doi: 10.1007/s10898-012-9951-y.

Moritz Vinzent Seiler, Raphael Patrick Prager, Pascal Kerschke, and Heike Trautmann. A collection of deep learning-based feature-free approaches for characterizing single-objective continuous fitness landscapes. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 657–665, 2022. doi: 10.1145/3512290.3528834.

Moritz Vinzent Seiler, Pascal Kerschke, and Heike Trautmann. Deep-ELA: Deep exploratory landscape analysis with self-supervised pretrained transformers for single- and multi-objective continuous optimization problems. *Evolutionary Computation*, 2025. doi: 10.1162/evco_a_00367.

I. M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967. doi: 10.1016/0041-5553(67)90144-9.

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, 1997. doi: 10.1023/A:1008202821328.

Ye Tian, Shichen Peng, Xingyi Zhang, Tobias Rodemann, Kay Chen Tan, and Yaochu Jin. A recommender system for metaheuristic algorithms for continuous optimization based on deep recurrent neural networks. *IEEE Transactions on Artificial Intelligence*, 1(1):5–18, 2020. doi: 10.1109/TAI.2020.3022339.

Koen van der Blom and Diederick Vermetten. On the influence of the feature computation budget on per-instance algorithm selection for black-box optimization. *arXiv preprint arXiv:2605.04954*, 2026. URL https://arxiv.org/abs/2605.04954.

Bas van Stein, Fu Xing Long, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. DoE2Vec: Deep-learning based features for exploratory landscape analysis. *arXiv preprint arXiv:2304.01219*, 2023. URL https://arxiv.org/abs/2304.01219.

Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *The Journal of Machine Learning Research*, 16(1):2023–2049, 2015.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Costas Voglis, Grigoris S. Piperagkas, Konstantinos E. Parsopoulos, Dimitris G. Papageorgiou, and Isaac E. Lagaris. MEMPSODE: An empirical assessment of local search algorithm impact on a memetic algorithm using noiseless testbed. In *Genetic and Evolutionary Computation Conference Companion*, 2012. URL https://www.cmap.polytechnique.fr/~nikolaus.hansen/proceedings/2012/GECCO/companion/p245.pdf.

Haoyu Yang, Michael Steinbach, Genevieve Melton, Vipin Kumar, and Gyorgy Simon. Combining self-supervision and privileged information for representation learning from tabular data: H. yang et al. *Knowledge and Information Systems*, 67(8):6907–6935, 2025.

A Additional Related Work

A.1 Representations of Algorithm–Problem Interactions

A complementary line of research characterizes optimization problems through the behavior of algorithms interacting with them, rather than solely through static observations sampled before optimization. DynamoRep summarizes optimization trajectories using descriptive statistics of population dynamics, thereby capturing information from the interaction between an optimizer and the underlying problem (Cenikj et al., 2023a). The resulting features have been investigated for tasks including problem classification, algorithm classification, and algorithm selection. Opt2Vec instead learns representations directly from populations visited during an optimization run using an autoencoder, enabling dynamic problem characterization from algorithm–problem interactions (Korošec & Eftimov, 2024). Another line adopts short optimization runs as probing information for algorithm selection. In particular, probing-trajectory methods characterize a problem from sequences of objective performance generated by briefly running candidate solvers, providing an algorithm-centric alternative to static landscape features (Renau & Hart, 2024). More recently, NeurELA learns dynamic landscape features from observations collected during optimization using an attention-based neural encoder, and pretrains the feature extractor for use across MetaBBO tasks (Ma et al., 2025). Together, these studies demonstrate that search trajectories and population dynamics provide complementary information about how optimization algorithms interact with a landscape.

PGFL exploits optimizer behavior from a different perspective. The trajectories generated by multiple probe optimizers are neither used as the final problem representation nor required as inputs to downstream tasks. Instead, they are summarized into optimizer-behavior targets that serve only as privileged supervision during pretraining, encouraging the low-budget landscape encoder to retain information related to how different optimizers respond to the problem. Behavioral supervision is applied to the anchor and sampling views of the original function instance, but not to the transformed-instance view, since transformations of the objective function may substantially alter optimizer trajectories. Moreover, optimizer-behavior guidance is only one component of PGFL and is jointly used with higher-budget representation distillation and function-structure guidance. After pretraining, all optimizer trajectories, behavioral targets, and auxiliary heads are removed, so representing a new problem does not require running any probe optimizer. Therefore, the main distinction is the training–deployment separation: PGFL transfers optimizer-response information into a representation that can subsequently be extracted solely from limited observations, rather than introducing trajectory information as an additional deployment-time requirement.

B Additional Method Details

B.1 Background and Notation

Sampled landscape observations. We consider a noiseless continuous black-box minimization objective

$$f : \mathcal{X}_d \to \mathbb{R}, \qquad \mathcal{X}_d \subseteq \mathbb{R}^d, \tag{24}$$

where $d \in \mathbb{N}_+$ denotes the problem dimension. Given n sampled points $\mathbf{x}_i \in \mathcal{X}_d$ and their evaluations $y_i = f(\mathbf{x}_i)$, we write

$$\mathcal{D}_n(f) = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \qquad \mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top \in \mathbb{R}^{n\times d}, \qquad \mathbf{y} = (y_1, \dots, y_n)^\top \in \mathbb{R}^n. \tag{25}$$

Throughout the paper, smaller objective values indicate better solutions.

Representation interface. Let ϕ denote the pointwise relational descriptor construction and $e_{\boldsymbol{\theta}}$ the landscape encoder. Their interface is

$$\mathbf{T} = \phi(\mathbf{X}, \mathbf{y}) \in \mathbb{R}^{n\times D_t}, \qquad \mathbf{z} = e_{\boldsymbol{\theta}}(\mathbf{T}, d) \in \mathbb{R}^R, \qquad R = \frac{H}{2}, \tag{26}$$

where D_t is the point-descriptor width, H is the encoder hidden width, and R is the landscape-representation width.

Pretraining functions and instances. During pretraining, a generated objective f is associated with a base expression structure $\mathcal{T}$ and instance parameters $\boldsymbol{\vartheta}$:

$$f = \mathrm{Inst}(\mathcal{T}, d, \boldsymbol{\vartheta}). \tag{27}$$

A related instance generated from the same base expression structure and problem dimension is written as

$$f' = \mathrm{Inst}(\mathcal{T}, d, \boldsymbol{\vartheta}'). \tag{28}$$

The notation f is retained throughout the paper because its problem dimension is already explicit through $f : \mathcal{X}_d \to \mathbb{R}$.

Views and budgets. Let n_l and n_h denote the low- and high-budget observation counts, respectively, with $n_l < n_h$. We use

$$v \in \mathcal{V} = \{A, S, T, H\} \tag{29}$$

to denote the Anchor, Sampling, Transformed-Instance, and High-Budget views. For a view v,

$$V^v = (\mathbf{X}^v, \mathbf{y}^v), \qquad \mathbf{T}^v = \phi(\mathbf{X}^v, \mathbf{y}^v), \qquad \mathbf{z}^v = e_{\boldsymbol{\theta}}(\mathbf{T}^v, d). \tag{30}$$

Minibatch notation. When multiple pretraining functions are considered jointly, $g \in \{1, \ldots, m\}$ indexes the functions in a minibatch. The g-th function has dimension d_g, and its view representation is denoted by $\mathbf{z}_g^v$. Hence, i indexes sampled points, v indexes views of one function, and g indexes different functions.

The online encoder parameters are denoted by $\boldsymbol{\theta}$, while the EMA teacher parameters are denoted by $\bar{\boldsymbol{\theta}}$.

Table 6: Main notation used in the methodology.

Symbol	Meaning
$f, \mathcal{X}_d, d$	Objective, search domain, and dimension
n, i	Observation count and sampled-point index
$\mathbf{X}, \mathbf{y}, \mathcal{D}_n(f)$	Sample locations, values, and observations
$\phi, \mathbf{T}, D_t$	Descriptor construction, matrix, and width
$e_{\boldsymbol{\theta}}, \mathbf{z}, H, R$	Encoder, representation, hidden widths, output representation widths
$\mathcal{T}, \boldsymbol{\vartheta}$	Base expression structure and instance parameters
f'	Related instance generated from the same $\mathcal{T}$
n_l, n_h	Low- and high-budget observation counts
A, S, T, H	Anchor, Sampling, Transformed, High-Budget views
m, g, d_g	Minibatch size, function index, and its dimension
$\boldsymbol{\theta}, \bar{\boldsymbol{\theta}}$	Online and EMA-teacher encoder parameters

B.2 Details of Pointwise Relational Landscape Encoding

For each landscape view, point descriptors are constructed independently from its complete observation set. We first introduce the notation and basic quantities shared by the four descriptor groups.

Notation and basic operators. Consider a noiseless minimization objective $f : \mathcal{X}_d \to \mathbb{R}$ with

$$\mathcal{X}_d = [\boldsymbol{\ell}, \mathbf{u}] \subset \mathbb{R}^d, \tag{31}$$

where $\boldsymbol{\ell}, \mathbf{u} \in \mathbb{R}^d$ denote the lower and upper search bounds. Given n sampled observations,

$$\mathcal{D}_n(f) = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \qquad y_i = f(\mathbf{x}_i), \tag{32}$$

we use

$$\mathcal{I}_n = \{1, \ldots, n\} \tag{33}$$

to denote the sampled-point index set.

For a scalar a, define

$$\mathrm{clip}(a, L, U) = \min\{U, \max\{L, a\}\}. \tag{34}$$

We use $\log(\cdot)$ for the natural logarithm, $\|\cdot\|_2$ for the Euclidean norm, $\oslash$ for elementwise division, and $\mathbb{I}[\mathcal{C}]$ for the indicator of condition $\mathcal{C}$.

For

$$\mathbf{y} = (y_1, \ldots, y_n)^\top, \tag{35}$$

let

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i \tag{36}$$

denote the sample mean. Let $Q_p(\mathbf{y})$ denote the empirical p-quantile. Accordingly,

$$\mathrm{median}(\mathbf{y}) = Q_{0.5}(\mathbf{y}), \qquad \mathrm{IQR}(\mathbf{y}) = Q_{0.75}(\mathbf{y}) - Q_{0.25}(\mathbf{y}). \tag{37}$$

We further use $\mathrm{std}(\cdot)$ to denote the standard deviation of a finite set of values.

Coordinate and objective preprocessing. Each sampled coordinate is first normalized to the unit hypercube:

$$\widetilde{\mathbf{x}}_i = (\mathbf{x}_i - \boldsymbol{\ell}) \oslash (\mathbf{u} - \boldsymbol{\ell}) \in [0,1]^d. \tag{38}$$

For any two sampled points $i, j \in \mathcal{I}_n$, their dimension-normalized distance is

$$\delta_{ij} = \frac{\|\widetilde{\mathbf{x}}_i - \widetilde{\mathbf{x}}_j\|_2}{\sqrt{d}} \in [0,1]. \tag{39}$$

The division by $\sqrt{d}$ compensates for the principal growth of Euclidean distance with the problem dimension.

Let

$$s_y = \mathrm{std}\left(\{y_i\}_{i=1}^n\right), \qquad q_y = \mathrm{IQR}(\mathbf{y}). \tag{40}$$

We compute two normalized objective values:

$$\eta_i = \mathrm{clip}\left(\frac{y_i - \bar{y}}{s_y}, -10, 10\right), \tag{41}$$

and

$$\eta_i^{\mathrm{rob}} = \mathrm{clip}\left(\frac{y_i - Q_{0.5}(\mathbf{y})}{q_y}, -10, 10\right). \tag{42}$$

When s_y or q_y is numerically close to zero, the corresponding normalized quantity is set to zero.

Let

$$r_i \in [0,1] \tag{43}$$

denote the normalized ascending average rank of y_i. Thus, smaller objective values receive smaller ranks, while ties share the corresponding average rank.

We further define

$$y_{\min} = \min_{j \in \mathcal{I}_n} y_j, \qquad y_{\max} = \max_{j \in \mathcal{I}_n} y_j, \tag{44}$$

and, for $y_{\max} > y_{\min}$,

$$g_i = \frac{y_i - y_{\min}}{y_{\max} - y_{\min}} \in [0,1]. \tag{45}$$

The set of sampled best points is

$$\mathcal{B} = \{j \in \mathcal{I}_n : y_j = y_{\min}\}. \tag{46}$$

Finally, the center of the normalized sampled point cloud is

$$\mathbf{c} = \frac{1}{n}\sum_{i=1}^n \widetilde{\mathbf{x}}_i \in [0,1]^d. \tag{47}$$

Multi-scale neighborhoods. For each point i, local neighborhoods are constructed from the distances in Eq. 39. We use

$$\begin{aligned} k_s &= \min\left(n-1, \max(3, \lceil 0.02n \rceil)\right), \\ k_m &= \min\left(n-1, \max(5, \lceil 0.05n \rceil)\right), \\ k_l &= \min\left(n-1, \max(10, \lceil 0.10n \rceil)\right). \end{aligned} \tag{48}$$

For $q \in \{s, m, l\}$, let

$$\mathcal{N}_i^{(q)} \subset \mathcal{I}_n \setminus \{i\}, \qquad |\mathcal{N}_i^{(q)}| = k_q, \tag{49}$$

denote the k_q nearest sampled points to point i. The relative neighborhood sizes allow the local descriptors to adapt to different observation counts.

Group I: Objective-quality descriptors (1–5). The first group characterizes the objective quality of point i relative to the complete sampled set:

$$\mathbf{t}_i^{\mathrm{quality}} = [\eta_i;\ \eta_i^{\mathrm{rob}};\ r_i;\ g_i;\ \log(1+g_i)] \in \mathbb{R}^5. \tag{50}$$

The first two descriptors preserve relative objective-scale information, whereas the remaining descriptors characterize the position of the point in the sampled objective-value ordering.

Table 7: Objective-quality descriptors.

Idx.	Descriptor	Symbol	Range
1	Standardized objective	η_i	$[-10, 10]$
2	Robust standardized objective	η_i^{rob}	$[-10, 10]$
3	Relative rank	r_i	$[0, 1]$
4	Gap to sampled best	g_i	$[0, 1]$
5	Compressed gap	$\log(1 + g_i)$	$[0, \log 2]$

Group II: Sample-geometry descriptors (6–10). The second group characterizes the spatial relation between point i and the sampled point cloud. We define

$$d_i^{\mathrm{best}} = \min_{j \in \mathcal{B}} \delta_{ij}, \tag{51}$$

$$d_i^{\mathrm{center}} = \frac{\|\widetilde{\mathbf{x}}_i - \mathbf{c}\|_2}{\sqrt{d}}, \tag{52}$$

and

$$d_i^{\mathrm{nn}} = \min_{j \in \mathcal{I}_n \setminus \{i\}} \delta_{ij}. \tag{53}$$

For the medium-scale neighborhood,

$$\bar{d}_i^{(m)} = \frac{1}{k_m} \sum_{j \in \mathcal{N}_i^{(m)}} \delta_{ij}, \tag{54}$$

and

$$s_{d,i}^{(m)} = \mathrm{std}\left(\{\delta_{ij} : j \in \mathcal{N}_i^{(m)}\}\right). \tag{55}$$

Thus,

$$\mathbf{t}_i^{\mathrm{geometry}} = [d_i^{\mathrm{best}};\, d_i^{\mathrm{center}};\, d_i^{\mathrm{nn}};\, \bar{d}_i^{(m)};\, s_{d,i}^{(m)}] \in \mathbb{R}^5. \tag{56}$$

Table 8: Sample-geometry descriptors.

Idx.	Descriptor	Symbol	Range
6	Distance to sampled best	d_i^{best}	$[0, 1]$
7	Distance to sample center	d_i^{center}	$[0, 1]$
8	Nearest-neighbor distance	d_i^{nn}	$[0, 1]$
9	Mean neighborhood distance	$\bar{d}_i^{(m)}$	$[0, 1]$
10	Neighborhood distance deviation	$s_{d,i}^{(m)}$	≥ 0

These descriptors replace raw coordinates with relative geometric quantities, so their width does not grow with d.

Group III: Local-landscape descriptors (11–15). The third group characterizes local objective variation and the prevalence of better sampled points at multiple spatial scales. We first define

$$\Delta_i^{\mathrm{loc}} = \eta_i - \frac{1}{k_m} \sum_{j \in \mathcal{N}_i^{(m)}} \eta_j \in [-20, 20], \tag{57}$$

and

$$s_{\eta,i}^{(m)} = \mathrm{std}\left(\{\eta_j : j \in \mathcal{N}_i^{(m)}\}\right). \tag{58}$$

For $q \in \{s, m, l\}$, the better-neighbor fraction is

$$\beta_i^{(q)} = \frac{1}{k_q} \sum_{j \in \mathcal{N}_i^{(q)}} \mathbb{I}[y_j < y_i]. \tag{59}$$

The complete local descriptor is

$$\mathbf{t}_i^{\mathrm{local}} = [\Delta_i^{\mathrm{loc}};\, s_{\eta,i}^{(m)};\, \beta_i^{(s)};\, \beta_i^{(m)};\, \beta_i^{(l)}] \in \mathbb{R}^5. \tag{60}$$

The three better-neighbor fractions distinguish whether better observations occur only in a very local region or remain prevalent at broader spatial scales.

Table 9: Local-landscape descriptors.

Idx.	Descriptor	Symbol	Range
11	Local relative objective	Δ_i^{loc}	$[-20, 20]$
12	Local objective deviation	$s_{\eta,i}^{(m)}$	≥ 0
13	Small-scale better-neighbor fraction	$\beta_i^{(s)}$	$[0, 1]$
14	Medium-scale better-neighbor fraction	$\beta_i^{(m)}$	$[0, 1]$
15	Large-scale better-neighbor fraction	$\beta_i^{(l)}$	$[0, 1]$

Group IV: Nearest-better descriptors (16–20). For each point i, define its strictly better sampled set

$$\mathcal{B}_i^+ = \{j \in \mathcal{I}_n : y_j < y_i\}. \tag{61}$$

If $\mathcal{B}_i^+ \neq \emptyset$, let

$$j_i^+ \in \arg\min_{j \in \mathcal{B}_i^+} \delta_{ij} \tag{62}$$

denote its nearest strictly better sampled point.

We define

$$d_i^+ = \delta_{i,j_i^+}, \tag{63}$$

$$\Delta_i^y = \max\left(\eta_i - \eta_{j_i^+}, 0\right), \tag{64}$$

and

$$\Delta_i^r = \max\left(r_i - r_{j_i^+}, 0\right). \tag{65}$$

Let $\epsilon > 0$ be a small numerical constant. Two ratio-based descriptors are

$$\gamma_i = \log\left[1 + \min\left(\frac{\Delta_i^y}{\max(d_i^+, \epsilon)}, 10^6\right)\right], \tag{66}$$

and

$$\kappa_i = \log\left[1 + \min\left(\frac{d_i^+}{\max(d_i^{\mathrm{nn}}, \epsilon)}, 10^6\right)\right]. \tag{67}$$

The complete nearest-better descriptor is

$$\mathbf{t}_i^{\mathrm{nb}} = [d_i^+;\, \Delta_i^y;\, \Delta_i^r;\, \gamma_i;\, \kappa_i] \in \mathbb{R}^5. \tag{68}$$

Table 10: Nearest-better descriptors.

Idx.	Descriptor	Symbol	Range
16	Nearest-better distance	d_i^+	$[0, 1]$
17	Nearest-better improvement	Δ_i^y	$[0, 20]$
18	Nearest-better rank gap	Δ_i^r	$[0, 1]$
19	Compressed improvement rate	γ_i	$[0, \log(1 + 10^6)]$
20	Compressed distance ratio	κ_i	$[0, \log(1 + 10^6)]$

If $\mathcal{B}_i^+ = \emptyset$, all five nearest-better descriptors are set to zero:

$$\mathbf{t}_i^{\mathrm{nb}} = \mathbf{0}_5. \tag{69}$$

No additional nearest-better existence indicator is appended.

Final pointwise descriptor. The four groups are concatenated as

$$\mathbf{t}_i = [\mathbf{t}_i^{\mathrm{quality}}; \mathbf{t}_i^{\mathrm{geometry}}; \mathbf{t}_i^{\mathrm{local}}; \mathbf{t}_i^{\mathrm{nb}}] \in \mathbb{R}^{D_t}, \qquad D_t = 20. \tag{70}$$

Applying the same construction to all sampled points yields

$$\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_n]^\top \in \mathbb{R}^{n \times D_t}. \tag{71}$$

Although $\mathbf{t}_i$ is associated with one sampled point, most of its components are relational quantities computed with respect to the complete observation set. Therefore, every landscape view independently recomputes its descriptor matrix from its own sampled coordinates and objective values.

Shared point encoder. Each point descriptor is mapped into an H-dimensional embedding by the same point encoder, using GELU (Hendrycks & Gimpel, 2016) and layer normalization (LN) (Ba et al., 2016):

$$\mathbf{h}_i = \mathbf{W}_{p,1}\mathbf{t}_i + \mathbf{b}_{p,1}, \tag{72}$$

$$\mathbf{e}_i = \mathbf{W}_{p,2}\,\mathrm{GELU}\left(\mathrm{LN}(\mathbf{h}_i)\right) + \mathbf{b}_{p,2} \in \mathbb{R}^H, \tag{73}$$

where

$$\mathbf{W}_{p,1} \in \mathbb{R}^{H\times D_t}, \qquad \mathbf{W}_{p,2} \in \mathbb{R}^{H\times H}. \tag{74}$$

All points share the same encoder parameters, producing

$$\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_n]^\top \in \mathbb{R}^{n\times H}. \tag{75}$$

The construction does not concatenate the raw d-dimensional coordinates with the point descriptors. Hence, D_t remains fixed as the problem dimension changes, while geometric information is represented through normalized distances, local neighborhoods, and nearest-better relations.

Computational implementation. Several descriptor groups require pairwise distances between the sampled points. The dominant descriptor-construction cost is therefore

$$O(n^2 d) \tag{76}$$

in time, with

$$O(n^2) \tag{77}$$

storage for the distance matrix. Pairwise squared distances can be computed using

$$\|\widetilde{\mathbf{x}}_i - \widetilde{\mathbf{x}}_j\|_2^2 = \|\widetilde{\mathbf{x}}_i\|_2^2 + \|\widetilde{\mathbf{x}}_j\|_2^2 - 2\widetilde{\mathbf{x}}_i^\top \widetilde{\mathbf{x}}_j, \tag{78}$$

avoiding the explicit construction of an $n \times n \times d$ difference tensor.

B.3 Details of Dimension-Aware Latent Aggregation

The latent aggregation module receives the point-embedding matrix

$$\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_n]^\top \in \mathbb{R}^{n\times H}, \tag{79}$$

where n denotes the number of sampled points and H the hidden width. The module contains a dimension encoder, L learnable latent queries, one point-to-latent cross-attention layer, K latent Transformer blocks, and an attention-pooling layer. We use M attention heads, an intermediate feed-forward width F, and

$$R = \frac{H}{2} \tag{80}$$

as the final representation width. Their concrete settings are given in the experimental parameter settings.

Dimension encoding. For $d \in \mathbb{N}_+$, the problem dimension is normalized as

$$\widehat{d} = \frac{\log(1+d)}{\log(1+d_{\max})}, \tag{81}$$

where $d_{\max}$ is the reference maximum dimension used during pretraining.

The scalar $\widehat{d}$ is mapped to an H-dimensional context vector:

$$\mathbf{c}_d = \mathbf{W}_{d,2}\,\mathrm{GELU}\left(\mathbf{W}_{d,1}\widehat{d} + \mathbf{b}_{d,1}\right) + \mathbf{b}_{d,2} \in \mathbb{R}^H, \tag{82}$$

where

$$\mathbf{W}_{d,1} \in \mathbb{R}^{H\times 1}, \qquad \mathbf{W}_{d,2} \in \mathbb{R}^{H\times H}. \tag{83}$$

Let

$$\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_L]^\top \in \mathbb{R}^{L\times H} \tag{84}$$

contain L learnable latent queries. The dimension context is added to every latent query:

$$\mathbf{U}^{(0)} = \mathbf{Q} + \mathbf{1}_L \mathbf{c}_d^\top \in \mathbb{R}^{L\times H}. \tag{85}$$

The dimension context is therefore introduced at the latent aggregation stage rather than appended to every point descriptor. This keeps the point-descriptor schema unchanged across dimensions while allowing the function-level aggregation to depend explicitly on d.

No clipping is applied to $\widehat{d}$. Consequently, for a dimension larger than the pretraining reference $d_{\max}$, Eq. 81 yields $\widehat{d} > 1$, and the dimension encoder performs continuous extrapolation without changing the network architecture.

Multi-head attention. For query, key, and value matrices

$$\mathbf{Q}' \in \mathbb{R}^{N_q \times H}, \qquad \mathbf{K}', \mathbf{V}' \in \mathbb{R}^{N_k \times H}, \tag{86}$$

the M-head attention operation (Vaswani et al., 2017) is

$$\mathrm{MHA}(\mathbf{Q}', \mathbf{K}', \mathbf{V}') = \mathrm{Concat}(\mathbf{O}_1, \ldots, \mathbf{O}_M)\mathbf{W}^O, \tag{87}$$

where the width of each attention head is

$$d_h = \frac{H}{M}. \tag{88}$$

For $h \in \{1, \ldots, M\}$,

$$\mathbf{O}_h = \mathrm{softmax}\left(\frac{(\mathbf{Q}'\mathbf{W}_h^Q)(\mathbf{K}'\mathbf{W}_h^K)^\top}{\sqrt{d_h}} + \mathbf{M}_{\mathrm{mask}}\right)(\mathbf{V}'\mathbf{W}_h^V), \tag{89}$$

where

$$\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V \in \mathbb{R}^{H \times d_h}, \qquad \mathbf{W}^O \in \mathbb{R}^{H \times H}. \tag{90}$$

The softmax is applied row-wise. $\mathbf{M}_{\mathrm{mask}}$ is an optional additive padding mask.

Point-to-latent cross-attention. For the input cross-attention,

$$N_q = L, \qquad N_k = n. \tag{91}$$

The dimension-aware latent matrix provides the queries, while the point embeddings provide the keys and values:

$$\mathbf{U}^{(1)} = \mathrm{LN}\left(\mathbf{U}^{(0)} + \mathrm{MHA}(\mathbf{U}^{(0)}, \mathbf{E}, \mathbf{E})\right) \in \mathbb{R}^{L \times H}. \tag{92}$$

Thus, the variable-length point set is converted into a latent sequence whose length L is independent of the number of sampled observations.

Latent Transformer blocks. The cross-attention output is processed by K latent Transformer blocks:

$$\mathbf{U}^{(r+1)} = \mathrm{Block}_r\left(\mathbf{U}^{(r)}\right), \qquad r = 1, \ldots, K. \tag{93}$$

All latent matrices satisfy

$$\mathbf{U}^{(r)} \in \mathbb{R}^{L \times H}. \tag{94}$$

Each block contains latent self-attention and a feed-forward network. Using pre-normalization notation, one block can be written as

$$\widetilde{\mathbf{U}}^{(r)} = \mathbf{U}^{(r)} + \mathrm{MHA}\left(\mathrm{LN}(\mathbf{U}^{(r)}), \mathrm{LN}(\mathbf{U}^{(r)}), \mathrm{LN}(\mathbf{U}^{(r)})\right), \tag{95}$$

followed by

$$\mathbf{U}^{(r+1)} = \widetilde{\mathbf{U}}^{(r)} + \mathrm{FFN}\left(\mathrm{LN}(\widetilde{\mathbf{U}}^{(r)})\right), \tag{96}$$

where

$$\mathrm{FFN}(\mathbf{u}) = \mathbf{W}_{f,2}\,\mathrm{GELU}\left(\mathbf{W}_{f,1}\mathbf{u} + \mathbf{b}_{f,1}\right) + \mathbf{b}_{f,2}, \tag{97}$$

with widths

$$H \to F \to H. \tag{98}$$

Dropout operations are omitted from the equations for clarity and are reported with the architecture settings.

Attention pooling and output projection. After the K latent Transformer blocks,

$$\mathbf{U}^{(K+1)} \in \mathbb{R}^{L \times H}. \tag{99}$$

A learnable pooling query

$$\mathbf{q}_{\mathrm{pool}} \in \mathbb{R}^H \tag{100}$$

aggregates the latent set:

$$\mathbf{p} = \mathrm{MHA}\left(\mathbf{q}_{\mathrm{pool}}, \mathbf{U}^{(K+1)}, \mathbf{U}^{(K+1)}\right) \in \mathbb{R}^H. \tag{101}$$

The singleton query dimension is omitted from the notation.

The final landscape representation is

$$\mathbf{z} = \mathrm{GELU}\left(\mathbf{W}_o\,\mathrm{LN}(\mathbf{p}) + \mathbf{b}_o\right) \in \mathbb{R}^R, \qquad R = \frac{H}{2}, \tag{102}$$

where

$$\mathbf{W}_o \in \mathbb{R}^{R \times H}, \qquad \mathbf{b}_o \in \mathbb{R}^R. \tag{103}$$

No ℓ_2 normalization is applied to $\mathbf{z}$ itself. Normalization is applied only by the corresponding training-time projection heads when required by the alignment or distillation objectives.

Padding and point ordering. Observation sets with different numbers of sampled points can be padded to a common point-axis length within a minibatch. Padded positions are excluded by $\mathbf{M}_{\mathrm{mask}}$ in Eq. 89.

No positional encoding is added to the point embeddings. Because the point encoder is shared and attention aggregates keys and values as a set, synchronously permuting the sampled points and the associated mask does not change the encoded landscape representation. The neural aggregation is therefore permutation-invariant with respect to the input ordering of sampled observations.

Computational complexity. For n point embeddings, L latent queries, and hidden width H, the point-to-latent cross-attention requires

$$O(nLH) \tag{104}$$

attention computation. Each latent self-attention block requires

$$O(L^2H), \tag{105}$$

leading to

$$O(nLH + KL^2H) \tag{106}$$

for the dominant attention operations of the latent aggregation module.

This complexity refers only to the neural aggregation stage. As detailed in Appendix B.2, construction of the relational point descriptors still requires pairwise operations over the sampled observations and therefore retains an $O(n^2d)$ component.

B.4 Details of Low-Budget View Construction

Base expression generation. For problem dimension d, the pretraining generator first constructs a base expression structure $\mathcal{T}$ using the tree-based generation approach of Tian et al. (2020). Generation starts from

$$F_0(\mathbf{x}) = \mathrm{mean}(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^d, \tag{107}$$

and repeatedly expands randomly selected leaves. A unary expansion wraps the selected leaf with a unary operator, whereas a binary expansion combines it with an additional operand.

Table 11: Components used to generate base expression structures.

Component	Candidate structures
Operands	constants, full decision vector $\mathbf{x}$, selected coordinates, shifted-coordinate vectors, dense coordinate transforms, and coordinate-index vectors
Binary operators	addition, subtraction, elementwise multiplication, and protected division
Unary operators	sign/scale transformations, power/absolute-value transformations, periodic transformations, rounding, and protected logarithmic or exponential transformations
Cross-coordinate operators	summation, mean, cumulative operations, product aggregation, and maximum aggregation
Additional structures	flat, multimodal, strongly multimodal, and coordinate-coupling transformations

The generator uses the component families summarized in Table 11.

Redundant compositions, constant subtrees, and repeated aggregations are simplified before the expression is compiled into a deterministic mapping

$$F_{\mathcal{T}} : \mathbb{R}^d \to \mathbb{R}. \tag{108}$$

Random constants and matrices sampled during generation are fixed after compilation, and numerically sensitive operators use protected evaluations.

Function instantiation. The base structure is converted into a concrete function instance through

$$f(\mathbf{x}) = aF_{\mathcal{T}}\left(\left[(\mathbf{x} - \mathbf{o})\mathbf{Q}^\top\right] \odot \mathbf{s}_\kappa\right) + b, \tag{109}$$

where $\mathbf{o}$ controls the input translation, $\mathbf{Q}$ specifies the coordinate orientation, $\mathbf{s}_\kappa$ controls dimension-wise conditioning, $a > 0$ is the positive objective scaling, and b is the objective translation.

The conditioning vector is constructed as

$$s_{\kappa,j} = \kappa^{\frac{j-1}{2(d-1)}}. \tag{110}$$

The Transformed-Instance view uses a related instance

$$f'(\mathbf{x}) = a' F_{\mathcal{T}}\left(G_{\boldsymbol{\vartheta}',d}(\mathbf{x})\right) + b', \tag{111}$$

which retains the same base expression structure and problem dimension. The implemented transformation retains the conditioning level while changing the location, signed coordinate arrangement, and positive affine objective transformation.

Anchor and Sampling views. Let $\mathbf{X}^A \in \mathbb{R}^{n_l \times d}$ contain Uniform samples from the search domain. The Anchor view is

$$V^A = (\mathbf{X}^A, f(\mathbf{X}^A)). \tag{112}$$

The Sampling view uses an independent sampling matrix $\mathbf{X}^S \in \mathbb{R}^{n_l \times d}$:

$$V^S = (\mathbf{X}^S, f(\mathbf{X}^S)). \tag{113}$$

$\mathbf{X}^S$ is generated by either Latin hypercube (Helton & Davis, 2003) or Sobol sampling (Sobol, 1967).

For a randomized Latin hypercube design in the unit hypercube,

$$q_{ij} = \frac{\pi_j(i) - 1 + \xi_{ij}}{n_l}, \tag{114}$$

where π_j is a permutation of $\{1, \ldots, n_l\}$ and $\xi_{ij} \sim \mathcal{U}(0,1)$. The resulting unit-box points are mapped to the original search domain.

Transformed-Instance view. The Transformed-Instance view evaluates f' at the unchanged Anchor locations:

$$V^T = (\mathbf{X}^A, f'(\mathbf{X}^A)). \tag{115}$$

Consequently, the Anchor and Transformed-Instance views share sampling locations but differ in the evaluated function instance.

Table 12: Relations among the three low-budget views.

View	Function	Sample locations	Primary variation
A	f	$\mathbf{X}^A$	Reference
S	f	$\mathbf{X}^S$	Sampling
T	f'	$\mathbf{X}^A$	Function instance

Independent descriptor construction. Every view reconstructs its descriptors independently:

$$\mathbf{T}^v = \phi(\mathbf{X}^v, \mathbf{y}^v), \qquad v \in \{A, S, T\}. \tag{116}$$

Thus, objective normalization, ranks, neighborhoods, and nearest-better relations are always determined from the complete observation set of the current view. In particular, coordinate reuse between A and T does not imply descriptor reuse.

B.5 Details of Sampling-View Alignment

Let $R = H/2$ denote the landscape-representation width and let P denote the contrastive projection width. The sampling projection head is

$$\widetilde{p}_s(\mathbf{z}) = \mathbf{W}_{s,2}\,\mathrm{GELU}(\mathbf{W}_{s,1}\mathbf{z} + \mathbf{b}_{s,1}) + \mathbf{b}_{s,2}, \tag{117}$$

followed by

$$p_s(\mathbf{z}) = \frac{\widetilde{p}_s(\mathbf{z})}{\|\widetilde{p}_s(\mathbf{z})\|_2} \in \mathbb{R}^P. \tag{118}$$

The hidden and projection widths are reported in the experimental settings.

For a minibatch of m pretraining functions,

$$\mathbf{u}_g^{A,s} = p_s(\mathbf{z}_g^A), \qquad \mathbf{u}_g^{S,s} = p_s(\mathbf{z}_g^S). \tag{119}$$

Define

$$\mathcal{I}_s = \{(g, A), (g, S) : g = 1, \ldots, m\}. \tag{120}$$

The positive mapping is

$$\pi_s(g, A) = (g, S), \qquad \pi_s(g, S) = (g, A). \tag{121}$$

For normalized projected vectors, the symmetric contrastive objective follows the normalized temperature-scaled cross-entropy formulation (Chen et al., 2020):

$$\mathcal{L}_{\mathrm{con}}(\mathcal{I}, \pi) = -\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \log \frac{\exp(\mathbf{u}_i^\top \mathbf{u}_{\pi(i)}/\tau)}{\sum_{j \in \mathcal{I} \setminus \{i\}} \exp(\mathbf{u}_i^\top \mathbf{u}_j/\tau)}, \tag{122}$$

where $\tau > 0$ is the contrastive temperature.

Accordingly,

$$\mathcal{L}_{\mathrm{align}}^{\mathrm{samp}} = \mathcal{L}_{\mathrm{con}}(\mathcal{I}_s, \pi_s). \tag{123}$$

For each function g, only its paired Anchor/Sampling representation is treated as positive; all projected views belonging to other functions in the same branch are negatives.

B.6 Details of Instance-View Alignment

The instance-alignment branch uses another projection head $p_t : \mathbb{R}^R \to \mathbb{R}^P$. Its architecture is identical to the sampling projector but all parameters are independent:

$$p_t(\mathbf{z}) = \mathrm{Norm}\left[\mathbf{W}_{t,2}\,\mathrm{GELU}(\mathbf{W}_{t,1}\mathbf{z} + \mathbf{b}_{t,1}) + \mathbf{b}_{t,2}\right]. \tag{124}$$

For each function,

$$\mathbf{u}_g^{A,t} = p_t(\mathbf{z}_g^A), \qquad \mathbf{u}_g^{T,t} = p_t(\mathbf{z}_g^T). \tag{125}$$

Define

$$\mathcal{I}_t = \{(g, A), (g, T) : g = 1, \ldots, m\}, \tag{126}$$

with

$$\pi_t(g, A) = (g, T), \qquad \pi_t(g, T) = (g, A). \tag{127}$$

The instance-alignment loss is

$$\mathcal{L}_{\mathrm{align}}^{\mathrm{trans}} = \mathcal{L}_{\mathrm{con}}(\mathcal{I}_t, \pi_t), \tag{128}$$

where $\mathcal{L}_{\mathrm{con}}$ is defined in Eq. 122.

The same Anchor representation participates in the sampling and instance branches, but the independent projectors p_s and p_t allow the two training relations to be represented in separate projection spaces. Both projectors are discarded after pretraining.

B.7 Details of Cross-Budget Representation Distillation

This branch applies teacher–student knowledge transfer (Hinton et al., 2015) to representations obtained at different observation budgets. Its online predictor and EMA target follow the representation-prediction mechanism of BYOL (Grill et al., 2020).

Nested high-budget observation. Let $\mathbf{X}^+ \in \mathbb{R}^{(n_h - n_l) \times d}$ contain additional Uniform samples. The High-Budget observation is

$$\mathbf{X}^H = \mathbf{X}^A \parallel \mathbf{X}^+, \qquad V^H = (\mathbf{X}^H, f(\mathbf{X}^H)). \tag{129}$$

Hence,

$$\mathcal{D}_{n_l}^A(f) \subset \mathcal{D}_{n_h}^H(f). \tag{130}$$

The high-budget observation is independently tokenized:

$$\mathbf{T}^H = \phi(V^H), \tag{131}$$

so the descriptors of the Anchor points can differ from those computed in the low-budget Anchor view because their neighborhood and normalization statistics are recomputed using the complete high-budget observation set.

Student and teacher branches. The low-budget Anchor representation is

$$\mathbf{z}^A = e_{\boldsymbol{\theta}}(\mathbf{T}^A, d), \tag{132}$$

whereas the high-budget representation is produced by an EMA copy of the same landscape-encoder architecture:

$$\bar{\mathbf{z}}^H = e_{\bar{\boldsymbol{\theta}}}(\mathbf{T}^H, d). \tag{133}$$

The online and teacher encoders have identical architectures but maintain separate parameter copies.

Budget projection and prediction. The student budget projector is

$$p_b : \mathbb{R}^R \to \mathbb{R}^P, \tag{134}$$

and uses the same projector architecture as the alignment branches but independent parameters. A student-only predictor

$$q_b : \mathbb{R}^P \to \mathbb{R}^P \tag{135}$$

is applied after p_b.

The student output is

$$\mathbf{u}^{A,b} = \mathrm{Norm}\left[q_b(p_b(\mathbf{z}^A))\right]. \tag{136}$$

The teacher uses an EMA budget projector $\bar{p}_b$ and no predictor:

$$\bar{\mathbf{u}}^{H,b} = \mathrm{sg}\left[\mathrm{Norm}(\bar{p}_b(\bar{\mathbf{z}}^H))\right]. \tag{137}$$

For a minibatch,

$$\mathcal{L}_{\mathrm{distill}} = 1 - \frac{1}{m}\sum_{g=1}^{m}(\mathbf{u}_g^{A,b})^\top \bar{\mathbf{u}}_g^{H,b}. \tag{138}$$

EMA update. Let $\mathbf{\Theta}_b$ collect the online encoder and budget-projector parameters, and let $\bar{\mathbf{\Theta}}_b$ contain their teacher counterparts. After an online update,

$$\bar{\mathbf{\Theta}}_b \leftarrow \mu_t \bar{\mathbf{\Theta}}_b + (1-\mu_t)\mathbf{\Theta}_b. \tag{139}$$

The teacher receives no gradient. The predictor q_b belongs only to the student branch.

When an EMA schedule is used, it can be written as

$$\mu_t = \mu_{\min} + \frac{\mu_{\max} - \mu_{\min}}{2}\left[1 - \cos(\pi r_t)\right], \qquad r_t \in [0,1], \tag{140}$$

where the concrete momentum endpoints are reported in the experimental settings.

B.8 Details of Structural Guidance

Each base expression structure $\mathcal{T}$ is summarized by

$$\mathbf{s} = [s_1, \ldots, s_{D_s}]^\top \in [0,1]^{D_s}, \qquad D_s = 8. \tag{141}$$

The eight attributes are summarized in Table 13.

Table 13: Structural attributes used as privileged targets.

Idx.	Attribute	Description
1	Complexity	Log-normalized number of operators
2	Operator entropy	Normalized entropy of the operator-category distribution
3	Smooth-nonlinear ratio	Fraction of operators belonging to smooth nonlinear categories
4	Nonlinear nesting	Fraction of nonlinear nodes nested under another nonlinear operator
5	Periodic ratio	Fraction of periodic operators
6	Nonsmooth ratio	Fraction of nonsmooth operators
7	Multiplicative ratio	Fraction of multiplicative/divisive structures
8	Coordinate-structure ratio	Fraction of variable operands involving structured coordinate operations

Complexity and operator entropy. Let n_{op} denote the number of operators in $\mathcal{T}$. The normalized complexity takes the form

$$s_1 = \min\left(\frac{\log(1+n_{\mathrm{op}})}{\log(1+N_{\mathrm{ref}})}, 1\right), \tag{142}$$

where N_{ref} is the reference operator count used by the generator.

Suppose the generator partitions operators into C_{op} categories and p_k is the proportion in category k. The normalized entropy is

$$s_2 = -\frac{\sum_{k=1}^{C_{\mathrm{op}}} p_k \log p_k}{\log C_{\mathrm{op}}}, \qquad 0\log 0 = 0. \tag{143}$$

Ratio-based structural attributes. The remaining components are normalized category counts. Smooth nonlinear operators include power-like smooth transformations, protected square-root/logarithmic/exponential transformations, while periodic operators include sinusoidal transformations. Nonsmooth structures include absolute-value-, rounding-, and maximum-type operations. Multiplicative structures include multiplication, protected division, reciprocal-type operations, and product aggregation. Coordinate structures include explicit coordinate selection, coordinate offset/coupling, dense transformations, and dimension-index operations.

The nonlinear-nesting target measures the fraction of nonlinear operators that occur within the subtree of another nonlinear operator.

Structural prediction head. The structural head is

$$h_s : \mathbb{R}^R \to [0,1]^{D_s}, \tag{144}$$

implemented as

$$h_s(\mathbf{z}) = \sigma\left[\mathbf{W}_{s,2}\,\mathrm{GELU}(\mathbf{W}_{s,1}\mathbf{z} + \mathbf{b}_{s,1}) + \mathbf{b}_{s,2}\right], \tag{145}$$

where $\sigma(\cdot)$ is the elementwise sigmoid.

The same target $\mathbf{s}$ supervises A, S, and T because the three low-budget views share the same base expression structure.

Smooth L1 objective. For vectors $\widehat{\mathbf{r}}, \mathbf{r} \in \mathbb{R}^D$, the coordinate-averaged Smooth L1 loss is

$$\rho(\widehat{\mathbf{r}}, \mathbf{r}) = \frac{1}{D}\sum_{j=1}^{D} \rho_\beta(\widehat{r}_j - r_j), \tag{146}$$

where

$$\rho_\beta(e) = \begin{cases} e^2/(2\beta), & |e| < \beta, \\ |e| - \beta/2, & |e| \geq \beta. \end{cases} \tag{147}$$

The structural-guidance loss in the main text applies this objective to all three low-budget views.

B.9 Details of Optimizer-Behavior Guidance

Probe trajectories. For each probe optimizer $a \in \mathcal{A}_{\mathrm{probe}}$, an independent short optimization run on the main function instance f produces

$$\Gamma_a(f) = \{\mathcal{P}_{a,t}, y_{a,t}^{\mathrm{best}}\}_t, \tag{148}$$

where $\mathcal{P}_{a,t}$ denotes the population or search state at progress step t and $y_{a,t}^{\mathrm{best}}$ is the best objective value observed so far.

In this work,

$$\mathcal{A}_{\mathrm{probe}} = \{\mathrm{DE}, \mathrm{PSO}, \mathrm{CMA\text{-}ES}\}. \tag{149}$$

The probe variants are based on differential evolution (Storn & Price, 1997), particle swarm optimization (Clerc & Kennedy, 2002), and CMA-ES (Hansen, 2016); their configurations are summarized in Table 14.

Table 14: Probe optimizers used to generate behavioral supervision.

Optimizer	Probe variant
DE	DE/rand/1/bin
PSO	Global-best PSO
CMA-ES	Compact full-covariance CMA-ES

The probe runs use their own search populations and evaluations and do not reuse the Anchor or Sampling points. Their concrete optimization budgets, random-seed counts, and optimizer hyperparameters are reported in the experimental settings.

Trajectory normalization. Performance is normalized relative to the low-budget Anchor observation. At a trajectory checkpoint with best-so-far value y_t^{best}, the normalized performance is

$$v_t = \frac{1}{5}\,\mathrm{clip}\left(\frac{\mathrm{median}(\mathbf{y}^A) - y_t^{\mathrm{best}}}{\mathrm{IQR}(\mathbf{y}^A)}, -5, 5\right). \tag{150}$$

Table 15: Behavioral attributes used as privileged targets.

Idx.	Attribute	Definition
1	Early-stage performance	Normalized best-so-far performance at one-quarter search progress
2	Mid-stage performance	Normalized best-so-far performance at one-half search progress
3	Late-stage performance	Normalized best-so-far performance at three-quarter search progress
4	Final performance	Normalized best-so-far performance at full search progress
5	Mean trajectory performance	Mean normalized performance over the trajectory windows
6	Improvement frequency	Fraction of trajectory windows containing a valid improvement
7	Diversity contraction	Change in population diversity from the initial to the final search stage
8	Stagnation ratio	Longest consecutive stagnation period normalized by trajectory length

Behavioral target. Each optimizer trajectory is summarized by

$$\mathbf{b}_a = [b_{a,1}, \ldots, b_{a,8}]^\top \in \mathbb{R}^8. \tag{151}$$

The components are given in Table 15.

The behavior target therefore contains both objective-progress information (Attributes 1–6) and population/search-dynamics information (Attributes 7–8).

Optimizer-conditioned prediction. Each probe optimizer has a learnable identity embedding

$$\mathbf{e}_a \in \mathbb{R}^{D_e}. \tag{152}$$

The behavior head takes the concatenation

$$[\mathbf{z}^v; \mathbf{e}_a] \in \mathbb{R}^{R+D_e} \tag{153}$$

and predicts

$$\widehat{\mathbf{b}}_a^v = \mathbf{W}_{b,2}\,\mathrm{GELU}\,[\mathbf{W}_{b,1}[\mathbf{z}^v; \mathbf{e}_a] + \mathbf{b}_{b,1}] + \mathbf{b}_{b,2}. \tag{154}$$

No sigmoid is applied to the output because several behavioral attributes may take signed values.

The same behavior target is used for the Anchor and Sampling representations:

$$v \in \mathcal{V}_b = \{A, S\}. \tag{155}$$

The Transformed-Instance representation is excluded because its function instance differs from the one on which the probe trajectory was generated.

If multiple independent probe runs are used for one optimizer, their seed-level profiles must be aggregated according to the implementation before forming the final $\mathbf{b}_a$.

B.10 Overall Training and Deployment Details

Information routing. Table 16 summarizes how the four views participate in the five objectives.

Table 16: View–objective routing during PGFL pretraining.

View	Sampling	Instance	Distill.	Structure	Behavior
A	✓	✓	Student	✓	✓
S	✓	–	–	✓	✓
T	–	✓	–	✓	–
H	–	–	Teacher	–	–

Overall objective. The complete pretraining objective is

$$\begin{aligned}\mathcal{L} = \lambda_{\mathrm{s}}\mathcal{L}_{\mathrm{align}}^{\mathrm{samp}} + \lambda_{\mathrm{t}}\mathcal{L}_{\mathrm{align}}^{\mathrm{trans}} \\ + \alpha_t\left(\lambda_{\mathrm{d}}\mathcal{L}_{\mathrm{distill}} + \lambda_{\mathrm{st}}\mathcal{L}_{\mathrm{st}} + \lambda_{\mathrm{b}}\mathcal{L}_{\mathrm{be}}\right),\end{aligned} \tag{156}$$

where

$$\alpha_t = \min\left(1, \frac{t+1}{T_w}\right). \tag{157}$$

The alignment objectives are active from the beginning, whereas the privileged-guidance objectives are gradually introduced during the warm-up period.

Pretraining procedure. A complete update proceeds as follows:

1. Sample a minibatch of pretraining functions and construct V^A, V^S, and V^T for each function.
2. Construct the nested High-Budget view V^H and obtain the structural and optimizer-behavior targets.
3. Encode A, S, and T using the online landscape encoder and encode H using the EMA teacher.
4. Compute $\mathcal{L}_{\mathrm{align}}^{\mathrm{samp}}$, $\mathcal{L}_{\mathrm{align}}^{\mathrm{trans}}$, $\mathcal{L}_{\mathrm{distill}}$, $\mathcal{L}_{\mathrm{st}}$, and $\mathcal{L}_{\mathrm{be}}$ according to Table 16.
5. Update the online encoder and training heads by minimizing Eq. 156.
6. Update the teacher encoder and teacher budget projector through the EMA rule in Appendix B.7.

The concrete loss weights, warm-up length, optimizer, learning rate, minibatch size, training duration, and EMA momentum settings are reported in the experimental settings.

Deployment. After pretraining, the following training-only components are removed: the Sampling-, Instance-, and Budget-projection heads, the student predictor, the structural and behavioral prediction heads, the EMA teacher, the base expression structure, the high-budget observations, and all probe-optimizer trajectories.

For a new black-box function, only a low-budget observation set is required:

$$\mathbf{z} = e_{\boldsymbol{\theta}}\left(\phi(\mathbf{X}, \mathbf{y}), d\right) \in \mathbb{R}^R. \tag{158}$$

Thus, the deployment path is simply

$$(\mathbf{X}, \mathbf{y}) \xrightarrow{\phi} \mathbf{T} \xrightarrow{e_{\boldsymbol{\theta}}} \mathbf{z}. \tag{159}$$

The resulting frozen representation can then be supplied to downstream landscape-analysis models without requiring any privileged information.

C Additional Experimental Settings

C.1 Pretraining Functions and Model Configuration

This section provides the complete experimental instantiation of the pretraining functions, multi-view construction, network architecture, privileged targets, and optimization procedure used for PGFL.

C.1.1 Pretraining Function Distribution

Pretraining set. We generate 120,000 noiseless continuous pretraining functions. The problem dimension is balanced over

$$d \in \{2, 3, 5, 10\}, \tag{160}$$

and each generated function is evaluated over

$$\mathcal{X}_d = [-5, 5]^d. \tag{161}$$

Hence, each pretraining dimension contributes the same number of underlying functions. The expression structure is used only to generate objective functions and privileged structural targets; the landscape encoder itself receives only sampled coordinates and their objective values.

Base expression-tree generation. For $\mathbf{x} = (x_1, \ldots, x_d)$, each random expression tree starts from

$$F_0(\mathbf{x}) = \mathrm{mean}(\mathbf{x}). \tag{162}$$

The tree is then expanded

$$K_{\mathrm{tree}} \sim \mathrm{Unif}\{4, 5, 6, 7, 8\} \tag{163}$$

Table 17: Components used to construct the random pretraining expression trees.

Component	Definition
Operands	Random constants $c \sim \mathcal{U}(1, 10)$, the complete decision vector $\mathbf{x}$, the first coordinate x_1, shifted coordinates $(x_2, \dots, x_d, 0)$, a fixed dense transform $\mathbf{xA}$, and the coordinate-index vector $(1, \dots, d)$.
Binary operators	Addition, subtraction, elementwise multiplication, and protected division.
Unary operators	Sign/scale transformations, power- and absolute-value-based transformations, periodic transformations, rounding, and protected logarithmic/exponential transformations.
Cross-coordinate operators	Summation, mean, cumulative summation, product aggregation, and maximum aggregation.

times. At every expansion, a leaf is reached through a randomly selected branch. A unary operator directly wraps the selected leaf, whereas a binary operator combines it with an additionally sampled operand. Operand and operator sampling is non-uniform so that basic arithmetic and standard variable operands are more frequent than strong variable-mixing or aggregation structures.

The operand and operator families used by the generator are summarized in Table 17.

The generator additionally injects at most one global structural pattern after the base tree is generated. Table 18 reports the implemented probabilities.

Table 18: Additional structural patterns injected into the pretraining expression trees.

Structure	Transformation	Probability
None	$F \leftarrow F$	0.60
Flat structure	$F \leftarrow \mathrm{round}(F)$	0.05
Multimodal structure	$F \leftarrow F + \sin(2\pi F)$	0.10
Strong multimodal structure	$F \leftarrow F + 10\sin(2\pi F)$	0.05
Coordinate coupling I	Replace $\mathbf{x}$ by $\mathbf{x} - x_1\mathbf{1}$	0.05
Coordinate coupling II	Replace $\mathbf{x}$ by $\mathbf{x} - (x_2, \dots, x_d, 0)$	0.05
Coordinate coupling III	Replace $\mathbf{x}$ by $\mathbf{xA}$	0.05
Coordinate coupling IV	Replace $\mathbf{x}$ by $(1, \dots, d) \odot \mathbf{x}$	0.05

Redundant constructions, including repeated negations, repeated reciprocal operations, constant subtrees, and repeated aggregations, are simplified before the tree is compiled into a deterministic function

$$F_{\mathcal{T}} : \mathbb{R}^d \to \mathbb{R}. \tag{164}$$

Random constants and random transformation matrices are fixed when the expression is compiled so that repeated evaluations of the same generated function are deterministic. Protected numerical implementations are used for division, square-root, logarithmic, and exponential operations to avoid invalid values and numerical overflow.

Function instantiation. A generated base structure is instantiated as

$$f(\mathbf{x}) = aF_{\mathcal{T}}\left(\left[(\mathbf{x} - \mathbf{o})\mathbf{Q}^\top\right] \odot \mathbf{s}_\kappa\right) + b, \tag{165}$$

where

$$\mathbf{o} \sim \mathcal{U}([-4, 4]^d) \tag{166}$$

controls the optimum translation. The orientation matrix $\mathbf{Q}$ is a random orthogonal matrix with probability 0.5 and the identity matrix otherwise. The conditioning level is uniformly sampled from

$$\kappa \in \{1, 10, 10^2, 10^3, 10^4\}, \tag{167}$$

and the dimension-wise scale is

$$s_{\kappa,j} = \kappa^{\frac{j-1}{2(d-1)}}. \tag{168}$$

Finally,

$$a \sim \mathrm{LogUniform}(0.5, 2), \qquad b \sim \mathcal{U}(-10, 10), \tag{169}$$

control positive objective scaling and objective translation, respectively.

C.1.2 Pretraining Views and Sampling Budgets

Each underlying pretraining function produces three low-budget student views and one high-budget teacher view. Their configurations are summarized in Table 19.

Table 19: Views constructed for each pretraining function.

View	Function	Sampling design	Budget	Role
Anchor (A)	Main instance f	Uniform	$25d$	Low-budget reference
Sampling (S)	Same f	Independent LHS or Sobol	$25d$	Sampling alignment
Transformed (T)	Related instance f'	Reuses Anchor locations	$25d$	Instance alignment
High-budget (H)	Same f	Anchor points plus additional Uniform points	$100d$	EMA teacher

The Sampling view alternates between Latin hypercube and Sobol sampling across pretraining functions; only one Sampling view is constructed for each underlying function.

The Transformed-Instance view uses the same base expression structure and conditioning level as the main function while changing the optimum location, signed coordinate arrangement/orientation, and positive affine transformation of the objective. It reuses the Anchor locations rather than drawing another sample.

The High-Budget view is nested with the Anchor view:

$$\mathbf{X}^H = \mathbf{X}^A \parallel \mathbf{X}^+, \qquad |\mathbf{X}^A| = 25d, \qquad |\mathbf{X}^+| = 75d, \qquad |\mathbf{X}^H| = 100d, \tag{170}$$

where $\mathbf{X}^+$ contains new Uniform points. Therefore,

$$\mathcal{D}^A_{25d}(f) \subset \mathcal{D}^H_{100d}(f). \tag{171}$$

Importantly, every view independently reconstructs its 20-dimensional pointwise descriptors. Thus, even when two views share coordinates, their normalized objectives, ranks, neighborhoods, and nearest-better relations are recomputed using the complete observation set of the corresponding view.

Observation counts. The exact numbers of sampled points at each pretraining dimension are listed in Table 20.

Table 20: Numbers of function evaluations used by the low- and high-budget views during pretraining.

d	**Low budget** ($25d$)	**High budget** ($100d$)
2	50	200
3	75	300
5	125	500
10	250	1000

C.1.3 Landscape Encoder Configuration

Table 21 reports the complete configuration of the landscape encoder used in all experiments.

The encoder therefore follows the dimensional flow

$$[n, 20] \to [n, 128] \to [32, 128] \to [32, 128] \to [128] \to [64]. \tag{172}$$

No positional encoding is applied to sampled points. L2 normalization is applied only inside the training-time contrastive and distillation projection branches; the exported 64-dimensional representation itself is not normalized by the encoder.

Table 21: Landscape encoder configuration used in the reported experiments.

Component / parameter	Setting
Point-descriptor width	20
Point encoder	$20 \to 128 \to 128$
Hidden width H	128
Dimension encoder	$1 \to 128 \to 128$
Dimension-normalization reference $d_{\max}$	10
Number of latent queries L	32
Attention heads M	4
Dimension per attention head	32
Point-to-latent cross-attention layers	1
Latent Transformer blocks K	4
Transformer FFN width	512
Transformer normalization	Pre-norm
Dropout	0.1
Learnable pooling queries	1
Representation width R	64
L2 normalization of final representation	No

C.1.4 Training-Time Projection and Auxiliary Heads

The training-only modules are configured as summarized in Table 22.

Table 22: Training-only projection, distillation, and auxiliary heads.

Module	Architecture
Sampling projector	$64 \to 64 \to 128$, GELU, followed by L2 normalization
Instance projector	$64 \to 64 \to 128$, GELU, followed by L2 normalization
Budget projector	$64 \to 64 \to 128$, GELU, followed by L2 normalization
Budget predictor	$128 \to 128 \to 128$, followed by L2 normalization
Structural head	$64 \to 32 \to 8$, GELU, Sigmoid output
Optimizer-ID embedding	8 dimensions per probe optimizer
Behavioral head	$(64 + 8) \to 64 \to 8$, GELU, linear output

The Sampling, Instance, and Budget projectors are parameterized independently. The EMA teacher contains an EMA copy of the landscape encoder and the Budget projector. It does not contain the student-side Budget predictor.

C.1.5 Structural and Behavioral Privileged Targets

Structural targets. The structural target has width

$$D_s = 8. \tag{173}$$

The eight components are expression complexity, operator entropy, smooth-nonlinear ratio, nonlinear nesting, periodic ratio, nonsmooth ratio, multiplicative/divisive ratio, and coordinate-structure ratio. Their exact definitions are provided in Appendix B.8. The target is generated once for each base expression structure and shared by the Anchor, Sampling, and Transformed-Instance views. No BBOB HLP labels are used to construct these targets.

Behavioral targets. The behavioral target is constructed using three probe optimizers:

$$\mathcal{A}_{\text{probe}} = \{\text{DE}, \text{PSO}, \text{CMA-ES}\}. \tag{174}$$

Their configurations are summarized in Table 23.

Table 23: Probe-optimizer settings used to generate behavioral privileged targets.

Probe	Configuration
DE	DE/rand/1/bin with $F = 0.5$ and $CR = 0.9$
PSO	Global-best PSO with inertia 0.7298 and $c_1 = c_2 = 1.49618$
CMA-ES	Compact full-covariance implementation
Evaluation budget	$50d$ function evaluations per optimizer and seed
Independent seeds	3 per optimizer
Trajectory windows	8
Behavioral width	$D_b = 8$ per optimizer
Optimizer-ID embedding	$D_e = 8$

The probe optimizers generate their own populations and objective evaluations; they do not reuse Anchor, LHS, or Sobol points. The Anchor objective values are used only as the reference for trajectory normalization.

At a trajectory checkpoint with best-so-far value y_t^{best}, the normalized performance is

$$v_t = \frac{1}{5}\,\text{clip}\left(\frac{\text{median}(\mathbf{y}^A) - y_t^{\text{best}}}{\text{IQR}(\mathbf{y}^A)}, -5, 5\right). \tag{175}$$

For every probe optimizer, the eight target components are:

1. normalized best-so-far performance at 25% search progress;
2. normalized best-so-far performance at 50% search progress;
3. normalized best-so-far performance at 75% search progress;
4. normalized best-so-far performance at 100% search progress;
5. mean normalized performance over all eight trajectory windows;
6. fraction of windows containing a valid improvement;
7. initial-to-final population-diversity contraction, clipped to $[-1, 1]$;
8. longest consecutive stagnation period divided by the eight trajectory windows.

For seed $r \in \{1, 2, 3\}$, let $\mathbf{b}_{g,a}^{(r)} \in \mathbb{R}^8$ denote the profile of probe optimizer a. The target used for training is the arithmetic mean over the three seeds:

$$\mathbf{b}_{g,a} = \frac{1}{3}\sum_{r=1}^{3} \mathbf{b}_{g,a}^{(r)}. \tag{176}$$

The complete behavioral target is therefore

$$\mathbf{B}_g = [\mathbf{b}_{g,\text{DE}}, \mathbf{b}_{g,\text{PSO}}, \mathbf{b}_{g,\text{CMA-ES}}]^\top \in \mathbb{R}^{3\times 8}. \tag{177}$$

The same behavioral target supervises the Anchor and Sampling views. The Transformed-Instance view is excluded because the probe trajectories are generated on the main function instance. Detailed target definitions are provided in Appendix B.9.

The three probe optimizers above are used only for privileged pretraining. They are different from the 12-solver portfolio used in the downstream AAS experiments described in Appendix C.3.

C.1.6 Optimization and Loss Configuration

The five pretraining objectives use the weights

$$(\lambda_{\text{s}}, \lambda_{\text{t}}, \lambda_{\text{d}}, \lambda_{\text{g}}) = (1,\ 0.5,\ 1,\ 0.5), \tag{178}$$

corresponding to Sampling-View Alignment, Instance-View Alignment, Cross-Budget Distillation, Structural Guidance, and Optimizer-Behavior Guidance, respectively.

The contrastive temperature used by both alignment objectives is

$$\tau = 0.1. \tag{179}$$

Cross-budget distillation uses one minus cosine similarity, while Structural and Behavioral Guidance use coordinate-averaged Smooth L1 losses.

The three privileged-guidance losses are linearly warmed up during the first 10% of training. For the reported run,

$$T = 3750, \qquad T_w = 375, \tag{180}$$

and

$$\alpha_t = \min\left(1, \frac{t+1}{375}\right). \tag{181}$$

The two alignment objectives are active at their full weights from the first update.

EMA teacher. The EMA teacher is initialized from the online student encoder and Budget projector. The teacher contains no predictor and receives no gradient updates. Its momentum follows a cosine transition from

$$\mu_{\min} = 0.99 \tag{182}$$

to

$$\mu_{\max} = 0.999. \tag{183}$$

Using normalized training progress $r \in [0, 1]$, the schedule is written as

$$\mu(r) = \mu_{\min} + \frac{\mu_{\max} - \mu_{\min}}{2}\left(1 - \cos(\pi r)\right). \tag{184}$$

C.1.7 Training, Validation, and Checkpoint Selection

The complete optimization settings are given in Table 24.

Table 24: Optimization and model-selection settings.

Setting	Value
Training functions	120,000
Training dimensions	$\{2, 3, 5, 10\}$, balanced
Batch size	32 underlying functions
Low-budget student views per batch	96
High-budget teacher views per batch	32
Total optimization updates	3750
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Learning rate	3×10^{-4}
Weight decay	10^{-4}
Learning-rate scheduler	None
Gradient-norm clipping	1.0
Numerical precision	BF16 mixed precision
Training random seed	0
Validation functions	256
Validation-generation seed	1,000,003
Validation interval	Every 100 updates and at termination
Privileged-loss warm-up	First 375 updates
Contrastive temperature	0.1
EMA momentum	$0.99 \rightarrow 0.999$ cosine schedule

The 256 validation functions are generated independently from the 120,000 training functions using the fixed validation-generation seed. They are generated and cached before training, so every checkpoint is evaluated on exactly the same validation set. Validation is performed without gradients and uses the complete pretraining objective.

The checkpoint attaining the minimum validation objective is exported as the frozen landscape encoder. No BBOB downstream test result is used for checkpoint selection. For the reported training run, the selected checkpoint occurs at update 3300, while candidate training continues through update 3750.

Batch padding. Mixed-dimensional functions appear in the same minibatch. A balanced batch of 32 underlying functions typically contains eight functions from each pretraining dimension. The low-budget branch therefore contains

$$32 \times 3 = 96 \tag{185}$$

student views.

After point-axis padding to the largest observation count in the batch, a typical full batch has

$$\text{student tensor: } [96, 250, 20], \tag{186}$$

and

$$\text{teacher tensor: } [32, 1000, 20]. \tag{187}$$

The associated Boolean masks have shapes $[96, 250]$ and $[32, 1000]$. Padded points are excluded from point-to-latent attention through the key-padding mask.

C.1.8 Parameter Counts and Deployment Model

Table 25 reports the parameter counts of the reported implementation.

Table 25: Parameter counts of the encoder and training-only modules.

Module	Parameters
Point encoder	19,456
Dimension encoder	16,768
Learnable latent queries	4,096
Input cross-attention	66,048
Post-cross-attention LayerNorm	256
Four latent Transformer blocks	793,088
Pooling query	128
Attention pooling	66,048
Encoder output layer	8,512
Deployment encoder total	**974,400**
Sampling projector	12,480
Instance projector	12,480
Budget projector	12,480
Budget predictor	33,024
Structural head	2,344
Optimizer-behavior head	5,216
Online training model total	**1,052,424**
EMA teacher (encoder + Budget projector)	986,880

After pretraining, only the 974,400-parameter landscape encoder is exported. All projection heads, the Budget predictor, Structural and Behavioral heads, the EMA teacher, the high-budget observations, expression structures, and probe trajectories are removed.

C.2 Benchmark, Sampling, and Baseline Settings

C.2.1 BBOB Benchmark

All downstream evaluations use the 24 noiseless BBOB benchmark functions. Unless explicitly stated otherwise, the evaluated dimensions are

$$d \in \{2, 3, 5, 10\}, \tag{188}$$

and PGFL constructs a landscape representation from

$$n = 25d \tag{189}$$

function evaluations.

The landscape encoder is always frozen after pretraining. Thus, BBOB functions are never used to update the encoder parameters.

C.2.2 Sampling Protocol and Repeated Evaluations

We consider three sampling designs:

$$\{\text{Uniform}, \text{Latin hypercube}, \text{Sobol}\}. \tag{190}$$

Sampling seeds

$$0, 1, \ldots, 9 \tag{191}$$

are used in the downstream experiments.

For every seed, sampled observations are regenerated and the downstream predictor is re-fitted, while the pretrained PGFL encoder remains unchanged. Thus, no downstream result involves fine-tuning of the representation encoder.

Reported means and standard deviations summarize the ten seed-level results. The sampling-design results are aggregated within each seed before computing the reported across-seed mean and standard deviation. Consequently, the reported variability reflects sensitivity to sampled observations and downstream-model fitting rather than variability across independently pretrained PGFL encoders.

C.2.3 Compared Landscape Representations

We compare PGFL with classical ELA (Mersmann et al., 2011) and four learning-based representation configurations: Deep-ELA (Seiler et al., 2025), DoE2Vec VAE-24 and VAE-32 (van Stein et al., 2023), and PCT (Seiler et al., 2022; Prager et al., 2022). The available evaluation configurations are summarized in Table 26.

Table 26: Sampling-budget settings of the compared landscape representations.

Method	Representation type	Setting	Cost-matched to PGFL?
PGFL	Learned	$25d$	Reference
ELA	Hand-crafted	$25d$	Yes
ELA	Hand-crafted	$50d$	No; $2\times$ budget
Deep-ELA	Learned	$25d$	Yes
Deep-ELA	Learned	$50d$	No; $2\times$ budget
DoE2Vec VAE-24	Learned	256 evaluations	No
DoE2Vec VAE-32	Learned	256 evaluations	No
PCT	Learned	Available $n = 100$ protocol	Protocol-specific

PGFL, ELA-$25d$, and Deep-ELA-$25d$ therefore form the principal cost-matched comparison. ELA-$50d$ and Deep-ELA-$50d$ are included to examine whether doubling the observation budget allows these baselines to close the performance gap.

The DoE2Vec configurations use their available 256-evaluation protocol, while PCT uses its available $n = 100$ protocol. These configurations are included as relevant learned-representation comparisons but are not described as strictly budget-matched $25d$ baselines.

If a baseline result is unavailable for a particular method–dimension–task combination, the corresponding table entry is marked with a dash. Missing entries are not imputed or replaced by results from another setting.

C.2.4 Evaluation-Cost Accounting

For downstream algorithm selection, the cost of obtaining the landscape representation is explicitly counted. For PGFL under the default setting,

$$c_{\text{feat}} = 25d. \tag{192}$$

For a baseline, c_{feat} is set according to the number of function evaluations required by its corresponding representation protocol.

This convention distinguishes comparisons based solely on predictor quality from comparisons of the complete decision pipeline, where additional landscape sampling itself consumes black-box evaluations.

C.3 Downstream Evaluation Protocol

C.3.1 Frozen-Encoder Evaluation

All downstream experiments evaluate the frozen landscape representation. For every sampled black-box function, PGFL first constructs its pointwise relational descriptors and applies the exported encoder to obtain

$$\mathbf{z} \in \mathbb{R}^{64}. \tag{193}$$

The encoder remains in evaluation mode and no encoder parameter is updated.

Only the task-specific downstream predictor is trained on the corresponding training split. This protocol is used for both high-level landscape-property prediction and automated algorithm selection.

C.3.2 High-Level Landscape-Property Prediction

The HLP experiment evaluates whether the frozen representation preserves high-level landscape semantics.

Three categorical BBOB properties are predicted independently:

1. multi-modality;
2. global structure;
3. funnel structure.

For each problem dimension, BBOB instances

$$0, \ldots, 99 \tag{194}$$

form the training set, while instances

$$100, \ldots, 119 \tag{195}$$

form the disjoint test set.

A separate random-forest classifier with

$$300 \tag{196}$$

trees is fitted for each landscape property. No representation fine-tuning is performed.

Performance is measured by macro-F1:

$$\mathrm{MacroF1} = \frac{1}{C} \sum_{c=1}^{C} \mathrm{F1}_c, \tag{197}$$

where C is the number of classes of the evaluated property. Higher values indicate better balanced classification performance.

The overall HLP score is the arithmetic mean of the three property-level macro-F1 scores:

$$\mathrm{HLP}_{\mathrm{overall}} = \frac{1}{3} \left(\mathrm{F1}_{\mathrm{multi}} + \mathrm{F1}_{\mathrm{global}} + \mathrm{F1}_{\mathrm{funnel}} \right). \tag{198}$$

C.3.3 Automated Algorithm Selection

Solver portfolio. The AAS portfolio contains 12 optimizers:

1. HCMA (Loshchilov et al., 2013);
2. SHADE-LM (Okulewicz & Zaborski, 2021);
3. BIPOP-CMA-ES (Hansen, 2009);
4. AMALGAM (Bosman et al., 2009);
5. Nelder–Mead (Rios & Sahinidis, 2013);
6. DE-SIMPLEX (Voglis et al., 2012);
7. NEWUOA (Rios & Sahinidis, 2013);
8. MEMPSODE (Voglis et al., 2012);
9. Powell (Powell, 1964);
10. MCS (Rios & Sahinidis, 2013);
11. DIRECT (Rios & Sahinidis, 2013);
12. COBYLA (Powell, 1994).

These downstream candidate solvers are distinct from the three probe optimizers used to construct privileged behavioral targets during PGFL pretraining.

Instances and cross-validation. AAS uses BBOB instances

$$\{1, 2, 3, 4, 5\}. \tag{199}$$

We perform leave-one-instance-out evaluation. In every fold, one instance is held out for testing and the remaining instances are used to fit the selector.

Selectors. Two selector models are evaluated:

1. a random forest with 300 trees;
2. distance-weighted k-nearest neighbors with
$$k = 3. \tag{200}$$

The same frozen landscape representation is used as selector input.

Solver evaluation budget. Each portfolio solver receives

$$500d \tag{201}$$

function evaluations. A successful run is evaluated at target precision

$$10^{-2}. \tag{202}$$

Relative expected running time. The primary AAS metric is relative expected running time:

$$\mathrm{relERT} = \frac{c_{\mathrm{feat}} + \mathrm{ERT}(a_{\mathrm{selected}})}{\mathrm{ERT}(a_{\mathrm{VBS}})}, \tag{203}$$

where c_{feat} is the number of black-box evaluations required to construct the selector input, a_{selected} is the solver selected by the evaluated method, and a_{VBS} is the virtual best solver for the corresponding problem.

The representation-construction cost is therefore included rather than treated as free preprocessing. For PGFL under the default setting,

$$c_{\mathrm{feat}} = 25d. \tag{204}$$

Lower relERT indicates better performance.

Failure handling. If the selected solver does not reach the target precision within its $500d$-evaluation budget, its ERT contribution is replaced using a PAR10-style penalty:

$$\mathrm{ERT}(a_{\mathrm{selected}}) = 10 \times 500d. \tag{205}$$

The representation cost c_{feat} remains included in Eq. 203.

A problem is excluded from AAS evaluation if all 12 portfolio members fail on that problem.

Reference solvers. We additionally report the single best solver (SBS), i.e., the single portfolio member with the best aggregate performance under the corresponding evaluation protocol.

The virtual best solver (VBS) is used only as the oracle denominator in Eq. 203. Because the VBS has zero feature-selection cost and defines the normalization reference, its normalized relERT is

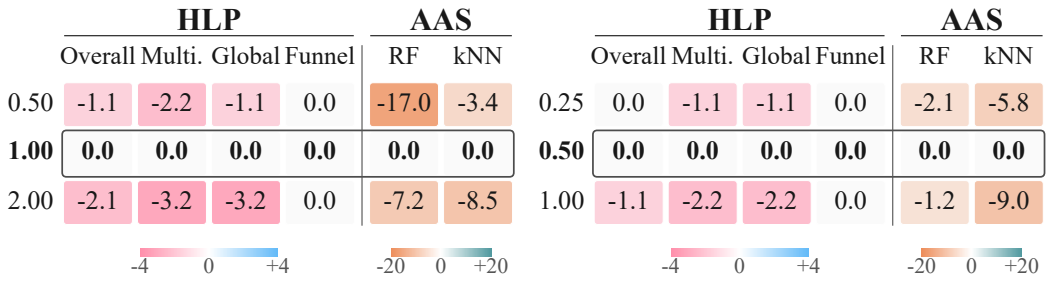
$$1, \tag{206}$$

and it is omitted from the result tables.

D Additional Experimental Analysis

D.1 Sensitivity to Auxiliary-Loss Weights

We further investigate the sensitivity of PGFL to the weights of its training objectives. Specifically, λ_{d} controls the cross-budget representation-distillation loss, while λ_{g} jointly controls the structural- and optimizer-behavior-guidance losses. Starting from the default configuration $(\lambda_{\mathrm{d}}, \lambda_{\mathrm{g}}) = (1.0, 0.5)$, we vary one weight at a time while fixing the other at its default value and keeping all remaining pretraining and downstream-evaluation settings unchanged. For λ_{d}, we consider $\{0.5, 1.0, 2.0\}$, while λ_{g} is varied over $\{0.25, 0.5, 1.0\}$. Both HLP prediction and AAS are reevaluated for each configuration.



(a) Sensitivity to λ_{d}. (b) Sensitivity to λ_{g}.

Figure 3: Sensitivity of PGFL to the auxiliary-loss weights. Each cell reports the direction-aligned relative performance change (%) with respect to the default configuration. Positive and negative values indicate improvement and degradation, respectively, and the outlined row denotes the default setting.

Table 27: Sensitivity to the auxiliary-loss weights (HLP: macro-F1 ↑; AAS: relERT ↓; mean ± std.). Best means are bold, including ties.

Loss weights		**HLP ↑**				**AAS ↓**	
λ_{d}	λ_{g}	**Overall**	**Multi.**	**Global**	**Funnel**	**RF**	***k*NN**
0.50	0.50	$0.94_{\pm 0.01}$	$0.91_{\pm 0.01}$	$0.92_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.00}$	$13.81_{\pm 3.32}$	$13.84_{\pm 2.81}$
1.00	0.25	$\mathbf{0.95}_{\pm 0.01}$	$0.92_{\pm 0.01}$	$0.92_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.00}$	$12.05_{\pm 1.88}$	$14.16_{\pm 3.19}$
1.00	1.00	$0.94_{\pm 0.01}$	$0.91_{\pm 0.01}$	$0.91_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.00}$	$11.94_{\pm 1.47}$	$14.60_{\pm 2.44}$
2.00	0.50	$0.93_{\pm 0.01}$	$0.90_{\pm 0.01}$	$0.90_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.00}$	$12.65_{\pm 1.05}$	$14.53_{\pm 4.58}$
1.00	**0.50**	$\mathbf{0.97}_{\pm 0.01}$	$\mathbf{0.95}_{\pm 0.01}$	$\mathbf{0.95}_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.01}$	$\mathbf{11.80}_{\pm 1.37}$	$\mathbf{13.39}_{\pm 2.36}$

Since a larger macro-F1 is preferred for HLP whereas a smaller relERT is preferred for AAS, we use a direction-aligned relative change to present the results in a unified manner. Given the performance m under a perturbed configuration and the performance m_0 under the default configuration, the relative changes for HLP and AAS are respectively computed as

$$\Delta_{\mathrm{HLP}} = 100\frac{m - m_0}{m_0}, \qquad \Delta_{\mathrm{AAS}} = 100\frac{m_0 - m}{m_0}. \tag{207}$$

Accordingly, a positive value indicates an improvement over the default configuration, a negative value indicates degradation, and zero corresponds to the default result. Because the relative-change ranges of HLP and AAS are different, separate color scales are used for the two tasks. The outlined row in each panel denotes the default value of the corresponding weight.

Figure 3 shows that the HLP performance is generally stable under the tested variations of both loss weights. When λ_{d} is changed from its default value of 1.0 to 0.5 or 2.0, the relative degradation of the HLP metrics remains within 3.2%, while funnel-structure prediction is unaffected. A similar pattern is observed for λ_{g}: varying it from 0.5 to 0.25 or 1.0 changes the HLP metrics by at most 2.2%. These results indicate that the learned representations are relatively stable with respect to moderate variations of the two auxiliary-loss weights on the HLP task.

The AAS results exhibit a somewhat higher sensitivity. For λ_{d}, reducing the weight to 0.5 causes relative degradations of 17.0% and 3.4% for RF and kNN, respectively, while increasing it to 2.0 results in degradations of 7.2% and 8.5%. This observation suggests that an appropriate contribution from the higher-budget teacher is important for transferring information useful to algorithm selection. In comparison, changing λ_{g} has a smaller effect on RF, with degradations of 2.1% and 1.2% at 0.25 and 1.0, respectively, although kNN is relatively more sensitive, with corresponding degradations of 5.8% and 9.0%. Therefore, AAS is more sensitive to the auxiliary-loss weights than HLP, particularly when the distillation weight is reduced substantially.

For completeness, Table 27 reports the corresponding raw means and standard deviations. The default configuration obtains the lowest relERT for both RF and kNN and achieves the best or tied performance on the HLP metrics. For example, decreasing λ_{d} from 1.0 to 0.5 increases the RF relERT from 11.80 to 13.81, whereas increasing it to 2.0 results in an RF relERT of 12.65. Similarly, changing λ_{g} to 0.25 or 1.0 preserves compet-

itive HLP performance but yields higher relERT than the default setting. These raw results are consistent with the relative-change patterns visualized in Fig. 3.

Overall, the default setting $(\lambda_{\mathrm{d}}, \lambda_{\mathrm{g}}) = (1.0, 0.5)$ provides the strongest overall configuration across the two downstream tasks. The surrounding settings generally retain similar HLP performance, while the larger variations observed on AAS indicate that the auxiliary objectives should not be assigned excessively small or large weights. Within the tested one-at-a-time ranges, however, the performance changes are gradual rather than showing a sharp isolated optimum, suggesting that PGFL does not rely on a narrowly tuned pair of auxiliary-loss weights. We note that this analysis varies one weight at a time and therefore does not characterize interactions between simultaneous changes of λ_{d} and λ_{g}.

E Limitations and Discussion

The results demonstrate that PGFL can learn useful landscape representations from limited observations, but several aspects of its current design delimit where these conclusions should be applied. Importantly, most of these limitations arise from the same design choices that make privileged pretraining and low-budget deployment possible. We discuss four issues that we consider particularly important below.

E.1 Dependence on the Privileged Pretraining Distribution

Core issue. PGFL relies on a deliberately asymmetric training–deployment setting. During pretraining, the encoder can exploit information that is unavailable for a new black-box function, including higher-budget observations, function structures, and optimizer behaviors. This additional supervision is powerful, but it also introduces a dependence on *what types of privileged information are available during pretraining*.

More formally, let $p_{\mathrm{pre}}(f)$ denote the distribution of generated pretraining functions and $p_{\mathrm{tar}}(f)$ denote the distribution of future target problems. The privileged objectives are optimized under

$$\mathbb{E}_{f\sim p_{\mathrm{pre}}}\left[\mathcal{L}_{\mathrm{distill}} + \mathcal{L}_{\mathrm{st}} + \mathcal{L}_{\mathrm{be}}\right], \tag{208}$$

whereas deployment ultimately concerns representations sampled from p_{tar}. Therefore, strong privileged supervision does not by itself guarantee transfer when

$$p_{\mathrm{pre}}(f) \not\approx p_{\mathrm{tar}}(f). \tag{209}$$

Where can the mismatch come from? In the current implementation, this issue appears at three levels.

1. *Function coverage.* Pretraining uses a large collection of automatically generated expression-tree functions. Although this produces diverse landscapes, the generator still defines which structural patterns are frequently observed and which ones are rare or absent.
2. *Structural coverage.* Structural guidance compresses each expression into eight predefined structural attributes. These targets capture useful properties such as expression complexity, operator composition, periodicity, and nonlinear nesting, but they do not constitute a complete description of the function structure.
3. *Behavioral coverage.* Optimizer-behavior guidance is obtained from three probe optimizers: DE/rand/1/bin, global-best PSO, and CMA-ES. Consequently, the behavioral supervision characterizes the landscape through the responses of this particular optimizer set rather than through all possible search dynamics.

The last point is especially relevant for algorithm selection. The behavioral target should be interpreted as a *probe of landscape behavior*, rather than a complete characterization of how arbitrary optimizers interact with the landscape. A landscape that is difficult for an optimizer family not represented during pretraining may therefore expose behavioral information that PGFL has never been explicitly encouraged to encode.

There is also a cost implication. At deployment, PGFL uses only $25d$ observations in its default configuration, but the privileged information is substantially richer during pretraining. For example, the teacher representation is constructed from a $100d$ high-budget view, and behavioral targets require running multiple probe optimizers. Thus, PGFL should be understood as *shifting expensive information acquisition from deployment to offline pretraining*, rather than eliminating that cost entirely. This distinction is unproblematic for inexpensive generated functions, but would become important if pretraining itself were performed on expensive simulators or physical experiments.

Possible direction. A natural extension is therefore not simply to generate *more* synthetic functions, but to increase the diversity of the privileged-information sources. Promising possibilities include combining multiple

function generators, incorporating historical optimization traces from real applications, and constructing probe sets that cover complementary optimizer families. An adaptive scheme could further select probe optimizers according to the current uncertainty of the representation, thereby allocating expensive behavioral supervision only where it provides new information.

E.2 Information and Computational Bottlenecks of Relational Encoding

Core issue. The pointwise relational interface is an important component of PGFL because it converts observations from different dimensions into a common format. Each sampled point is represented by only 20 descriptors,

$$\mathbf{t}_i = [\mathbf{t}_i^{\text{quality}}; \mathbf{t}_i^{\text{geometry}}; \mathbf{t}_i^{\text{local}}; \mathbf{t}_i^{\text{nb}}] \in \mathbb{R}^{20}, \tag{210}$$

with five descriptors for each of objective quality, sample geometry, local landscape structure, and nearest-better relations. This construction provides a compact and dimension-compatible interface, but it necessarily acts as an *information bottleneck*: the encoder never observes information that has already been discarded by the descriptor map. The current methodology explicitly uses these relational descriptors rather than raw coordinates as the encoder input.

For example, Euclidean distances and local-neighborhood statistics provide useful geometric summaries, but they may not fully preserve *directional information*, *anisotropic interactions*, or *higher-order dependencies* among multiple decision variables. Two sampled landscapes could therefore produce similar relational statistics despite differing in a property relevant to a particular downstream task. This is not necessarily problematic for the HLP and AAS tasks considered in this work, but it limits the claim that the representation preserves *all* information available in the original observations.

Scalability beyond the low-budget regime. The latent aggregation mechanism addresses one important computational issue: instead of applying self-attention over all n sampled points, PGFL first aggregates them into a fixed number L of latent queries. Ignoring descriptor construction, its main attention cost can be summarized approximately as

$$\mathcal{O}(nLH) + \mathcal{O}(KL^2H), \tag{211}$$

where H is the hidden width and K is the number of latent Transformer blocks. This makes the neural aggregation substantially less dependent on the number of observations than full pointwise self-attention.

However, the complete pipeline is not linear in n. Several relational descriptors require nearest-neighbor and pairwise-distance information, so the current descriptor construction still contains an approximately

$$\boxed{\mathcal{O}(n^2 d)} \tag{212}$$

geometric preprocessing term. Under the default $n = 25d$ setting, this is manageable for the dimensions studied in the paper. The same design becomes less attractive if either the dimension or the sampling budget grows substantially.

There is a second high-dimensional concern. As dimensionality increases, distance-based neighborhoods can become less discriminative due to the concentration of distances. Because several PGFL descriptors explicitly rely on nearest-neighbor, nearest-better, and multi-scale neighborhood relations, sufficiently high-dimensional problems may require a different notion of local geometry. Our current $d = 20$ experiment does not test the regime where this effect would become dominant.

Possible direction. Future versions of PGFL could relax both bottlenecks by combining *learned raw-observation features* with selected relational descriptors. For scalability, dense pairwise relations could be replaced with k-nearest-neighbor graphs, approximate nearest-neighbor search, landmark points, or sparse geometric summaries. Such designs would preserve the useful relational inductive bias of PGFL without requiring every pair of observations to interact explicitly.

E.3 Task-Agnostic Alignment May Suppress Task-Relevant Variations

Core issue. Multi-view alignment assumes that certain changes in the observed landscape should not substantially alter its representation. Specifically, PGFL aligns

$$V^A \leftrightarrow V^S \tag{213}$$

for different sampling designs of the same function, and

$$V^A \leftrightarrow V^T \tag{214}$$

for related function instances generated from the same underlying structure. The corresponding contrastive objectives therefore encourage

$$\mathbf{z}^A \approx \mathbf{z}^S, \qquad \mathbf{z}^A \approx \mathbf{z}^T, \tag{215}$$

in their respective projection spaces. This alignment mechanism is a central component of the current framework.

The ablation results in this paper show that multi-view alignment is beneficial for the considered HLP and AAS tasks. Nevertheless, an important distinction is that **consistency is not always equivalent to information preservation**. The adopted transformed instance can change properties such as optimum location, coordinate arrangement or orientation, and objective scaling. PGFL intentionally encourages the encoder to become less sensitive to such changes.

This is appropriate when they are nuisance factors. However, some downstream tasks may depend precisely on these properties. For instance, the relative performance of an optimizer can depend on whether the optimizer is rotation invariant, scale invariant, or sensitive to coordinate-wise conditioning. In such a case, aggressively aligning two transformed landscapes could remove information that is useful for predicting optimizer performance.

The underlying tension can be summarized as

$$\underbrace{\text{remove nuisance variation}}_{\text{desired invariance}} \quad \text{vs.} \quad \underbrace{\text{retain task-relevant variation}}_{\text{desired discriminability}}. \tag{216}$$

The current method resolves this trade-off globally through fixed alignment objectives, rather than deciding which properties should be invariant for a specific downstream task.

Why the current experiments do not fully resolve this. The strong ablation performance of the complete model establishes that the chosen alignment is useful *on average* for the evaluated tasks. It does not show that every transformation used in alignment is harmless for every downstream prediction problem. Nor does it identify which information is deliberately suppressed by the representation.

Possible direction. A more general design could separate the representation into invariant and transformation-sensitive components,

$$\mathbf{z} = [\mathbf{z}_{\text{inv}}; \mathbf{z}_{\text{var}}], \tag{217}$$

where only $\mathbf{z}_{\text{inv}}$ is aligned across selected views. Alternatively, transformation-conditioned or equivariant objectives could preserve predictable changes rather than forcing all related views to become similar. Such a design would allow downstream tasks to exploit instance-specific information when it is useful while retaining the robustness benefits of multi-view learning.

E.4 Cross-Dimensional Transfer Is Still an Empirical Extrapolation

Core issue. PGFL is architecturally capable of processing multiple dimensions because the point interface has a fixed width and the latent queries are conditioned on the problem dimension. Nevertheless, *architecture compatibility should not be confused with dimension-universal generalization*.

The normalized dimension input used by PGFL is

$$\widehat{d} = \frac{\log(1+d)}{\log(1+d_{\max})}. \tag{218}$$

During pretraining, the encoder observes

$$d \in \{2, 3, 5, 10\}, \qquad d_{\max} = 10. \tag{219}$$

Consequently, for the unseen dimension used in our transfer experiment,

$$\widehat{d}_{20} = \frac{\log(21)}{\log(11)} \approx 1.27, \tag{220}$$

which already lies *outside the normalized range observed during pretraining*. The dimension encoder is therefore performing genuine extrapolation rather than interpolation.

The $d = 20$ experiment provides encouraging evidence: with only $25d$ observations, PGFL obtains an HLP overall macro-F1 of 0.94, an RF relERT of 3.91, and a kNN relERT of 6.39. These results show that the learned representation remains useful at a dimension twice the largest value seen during pretraining. However, one successful extrapolation point cannot establish that the same behavior will persist at $d = 50$, 100, or beyond.

The empirical scope is broader than dimension alone. The current transfer claims are also bounded in three other respects.

1. *Problem class.* Experiments focus on noiseless, single-objective continuous BBOB functions. Noisy, constrained, discrete, mixed-variable, and multiobjective landscapes may require qualitatively different relational information.
2. *Transfer protocol.* The PGFL encoder is frozen, but the downstream HLP predictor or algorithm selector is still fitted according to the corresponding evaluation protocol. Therefore, the current experiment establishes **representation transfer**, not zero-shot transfer of an entire task predictor.
3. *Pretraining uncertainty.* The reported standard deviations mainly reflect variation in sampling and downstream evaluation. They do not quantify variability across independently pretrained PGFL encoders.

The second distinction is particularly important. At deployment, PGFL indeed discards the EMA teacher, high-budget observations, function structures, optimizer trajectories, and all training-only heads; only the descriptor construction and frozen student encoder are retained. Nevertheless, a downstream learner is still required for the target HLP or AAS task.

Possible direction. A stronger generalization study should therefore expand along multiple axes rather than adding a single additional benchmark. For example, future work could *(i)* evaluate a sequence of unseen dimensions such as $d \in \{20, 40, 80\}$, *(ii)* include noisy, constrained, and mixed-variable problems, *(iii)* repeat the complete pretraining process using multiple random seeds, and *(iv)* investigate protocols in which both the encoder and downstream predictor remain fixed. Together, these experiments would distinguish architectural dimension-handling ability from genuine out-of-distribution representation generalization.