2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

Feature-Conditioned Reinforcement Learning for Generalizable Engineering Optimization: Benchmarking on Multimodal Test Functions

Varun S. Chavan[1], Mitsuru Endo[1], Zexin Shan[1], and Yukio Tsutsui[2,3]

Abstract— **Generalization is essential for accelerating optimization workflow, enhancing scalability and reducing computational cost. This paper presents a novel Feature-Conditioned Reinforcement Learning (FC-RL) method for engineering optimization, which enables generalization across diverse problem scenarios by conditioning the RL policy on explicit problem features. Unlike traditional optimization methods that require hyperparameter tuning for each new task, FC-RL leverages a meta-network to condition its behavior dynamically through Feature-wise Linear Modulation (FiLM). The experiments conducted across six multi-modal benchmark functions with varying dimensions, search-space, and global optimum shifts demonstrate that FC-RL often achieves global convergence without the need of retraining or retuning, while producing consistent and reproducible results due to its deterministic policy. Although FC-RL's performance depends on sufficient training diversity and is limited to novel scenarios within the known environments, these findings indicate that the proposed method is particularly well-suited for repetitive engineering optimization tasks involving variable problem configurations across multiple systems.**

I. Introduction

Optimization is fundamental in engineering for tasks such as design, controller tuning, and trajectory planning. In particular, optimization can drive towards using less material, energy, computation, and time while still meeting performance goals [1]. For instance, electric motor systems were responsible for 53% of global electric energy consumption in 2016 [2]. Therefore, even a slight design optimization can result in large savings. However, in such engineering settings, the objective functions are often challenging, high-dimensional, noisy and multi-modal, rendering traditional gradient-based methods inapplicable [3]. Consequently, black-box optimization techniques such as metaheuristic algorithms becomes important for real-world engineering applications.

Metaheuristic algorithms like Particle Swarm Optimization [4], Firefly Algorithm [5], and Genetic Algorithm [6] have seen wide adoption and have demonstrated strong performance in engineering optimization problems. Their population-based, stochastic search mechanisms enable simultaneous exploration of multiple regions in the search space, while their gradient-free nature allows them to handle complex objective functions, particularly when their hyperparameters are properly tuned. For instance, PSO has strong global search ability and its performance can be further improved using techniques such as adjusting the inertia weight [7] or using example-based learning [8] to reduce the chance of getting stuck in local optima. Similarly, the Firefly Algorithm works well by naturally dividing the population into groups based on attraction, allowing it to find multiple good solutions without needing special niching methods [5]. Genetic Algorithms also use strategies like structured populations and promoting diversity to avoid early convergence and explore more of the search space [6]. However, a common drawback of these methods is their reliance on fixed or manually tuned hyperparameters, which limits their adaptability and necessitates time-consuming retuning for each new optimization problem configuration.

To improve adaptability, Reinforcement Learning (RL) has emerged as a promising method and is often combined with metaheuristics. For example, Q-learning has been applied within particle swarm algorithm [9], allowing particles to dynamically choose their update strategies based on past rewards, which helps maintain diversity and avoid premature convergence in multi-modal landscapes. Also, Standalone RL-based optimizers have been developed to learn search behaviors directly from training over multiple functions. One such example is the use of deep RL agent trained across a diverse set of benchmark functions [10], enabling it to capture generic search heuristics within that distribution. Another example incorporates a two-phase structure, combining learned local descent with an RL-based escape policy that enables transitions between local optima [11]. While these RL methods improve adaptability within the scope of their training, they typically are optimized for fixed functions and often generalize poorly to different problems, necessitating retraining.

In this paper, we propose a novel Feature-Conditioned Reinforcement Learning (FC-RL) method for engineering optimization that can converge globally to optimum and generalize to diverse problems without retraining or retuning. This is achieved by exploring the use of a meta-network to condition the RL policy inspired by Feature-wise Linear Modulation (FiLM) [12]. We evaluated FC-RL on comprehensive set of challenging multi-modal benchmark functions. The experimental results demonstrate that our method consistently converges to the global minimum and generalizes well across diverse benchmark scenarios without retraining or retuning.

*This work was supported by Yaskawa Electric Corp.

[1]V. Chavan, M. Endo and Z. Shan are with the Faculty of Mechanical Engineering, Institute of Science Tokyo, Meguro, Tokyo 152-8550, Japan chavan.v.e883@m.isct.ac.jp, mitsuru.endo@ieee.org

[2]Y. Tsutsui is with the Faculty of Electrical Engineering, Institute of Science Tokyo, Meguro, Tokyo 152-8550, Japan

[3]Y. Tsutsui is with the Yaskawa Electric Corporation, 2-1 Kurosakishiroishi, Yahatanishi-ku, Kitakyushu 806-0004 Japan

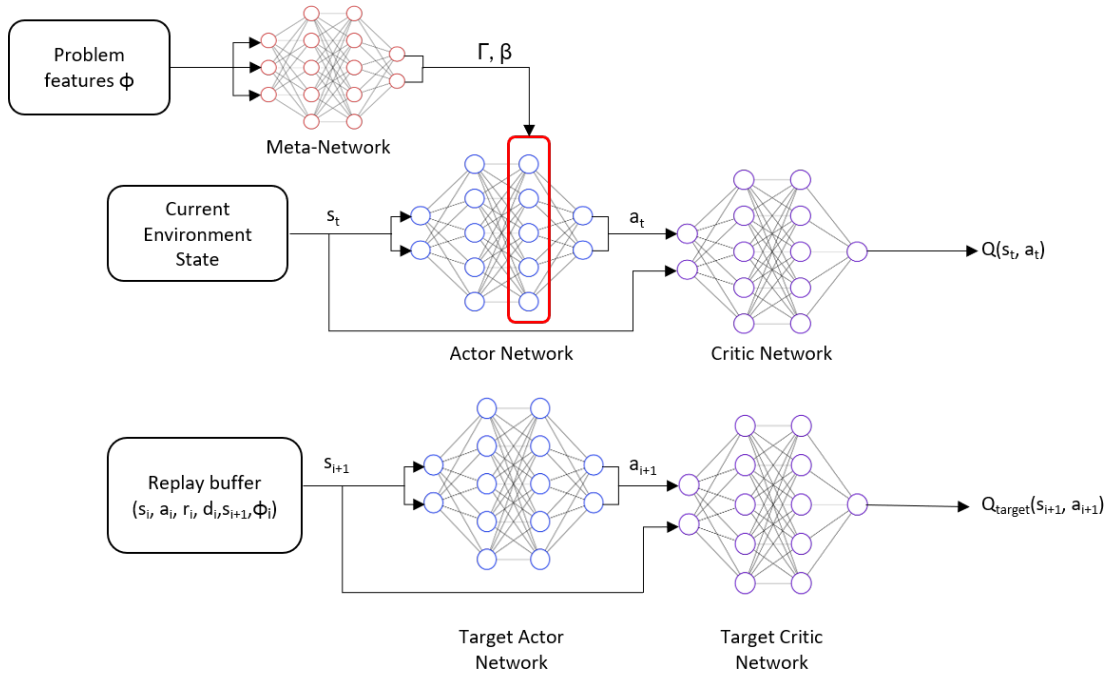


Fig. 1: Conceptual Diagram of Feature-Conditioned Reinforcement Learning

II. Proposed Method

In this section, we introduce the proposed FC-RL method by formulating the optimization process as a Markov Decision Process (MDP), which enables reinforcement learning to model optimization as a sequential decision-making problem. We then present the architecture, highlighting key design elements that enhance generalizability across problem instances and support global convergence.

A. Problem formulation

We consider the problem of optimizing continuous, multi-modal objective functions as MDP over bounded domains [9] [13]. Given a function $f : \mathbb{R}^d \to \mathbb{R}$, where d denotes the dimensionality of the input space, the objective is to find the global minimum:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathscr{D}} f(\mathbf{x}) \tag{1}$$

where $\mathscr{D}$ denotes the search-space, and $\mathbf{x} \in \mathscr{D}$ is a candidate solution vector. In our MDP formulation, the state s_t at time step t corresponds to the current candidate solution x_t. At each step, the agent observes the state $\mathbf{s}_t$, and outputs an action a_t, which represents an update direction and magnitude. The new solution $\mathbf{s}_{t+1}$ is then evaluated using the function f, and a scalar reward r_t is assigned. This reward is designed to guide the agent toward lower function values while also emphasizing proximity to minima. Specifically, we use a shaped reward function:

$$r_t = -f(\mathbf{s}_{t+1}) + \alpha \exp\left(-\frac{f(\mathbf{s}_{t+1})}{\delta}\right) \tag{2}$$

Here, α and δ are reward shaping parameters that amplify positive reinforcement in regions near the global optimum. These amplifications helps the agent recognize improvements that would otherwise be overshadowed by large negative reward earlier in the episode. As a result, meaningful progress is more effectively reinforced during the training. Additionally, a large terminal reward is provided upon reaching the global minimum which serves as a strong learning signal that encourage convergence and efficient task completion.

The policy $\pi(a_t \mid \mathbf{s}_t, \phi)$ guiding the agent's action is conditioned on a problem-specific feature vector ϕ, which encodes meta-information about the optimization problem. This conditioning allows the policy to generalize across different optimization problems by adapting to specific characteristics of the each task and its configuration.

B. Architecture overview

The proposed FC-RL method builds on the Deep Deterministic Policy Gradient (DDPG) algorithm [14] as illustrated in Fig. 1. The agent interacts with the environment represented by an objective function f, to learn an optimal policy through trial and error. This policy modeled using an actor network μ_θ, receives the current state s_t as input and outputs action a_t. The state-action pair is then passed to a second network called the critic network Q_ψ , which estimates the Q-value $Q(s_t, a_t)$, representing the expected cumulative reward from the point onward. The critic provides gradients that guide the training of the actor network to improve the expected return. The actor is updated by minimizing the following loss:

$$\mathscr{L}_{actor} = -\mathbb{E}_{(s,\phi)\sim\mathscr{R}}\left[\left(Q_\psi(s, \mu_\theta(s, \Gamma_\phi, \beta_\phi))\right)\right] \tag{3}$$

Algorithm 1 Feature-Conditioned Reinforcement Learning

Input: Total episodes N, time-steps T, batch size B, discount factor γ, soft-update rate τ, OU Noise $\mathscr{N}_t$, meta start episode E_{meta}
Initialize actor μ_θ, critic Q_ψ, meta-network $\mathscr{M}_\omega$
Initialize target networks $\mu_{\theta'} \leftarrow \mu_\theta$, $Q_{\psi'} \leftarrow Q_\psi$
Initialize replay buffer $\mathscr{R}$
for episode $= 1$ **to** N **do**
  Obtain problem feature vector ϕ
  if episode $< E_{\text{meta}}$ **then**
    Set meta-parameters $(\Gamma_\phi, \beta_\phi) \leftarrow (\mathbf{I}, \mathbf{0})$
  else
    Compute meta-outputs $(\Gamma_\phi, \beta_\phi) \leftarrow \mathscr{M}_\omega(\phi)$
  end if
  Reset environment; set fixed state $\mathbf{s}_0$
  for $t = 1$ **to** T **do**
    Select $a_t \leftarrow \mu_\theta(\mathbf{s}_t, \Gamma_\phi, \beta_\phi) + \mathscr{N}_t$
    Observe reward r_t, and next state $\mathbf{s}_{t+1}$
    Store $(\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1}, \phi)$ in $\mathscr{R}$
    Sample mini-batch $\{(\mathbf{s}_i, a_i, r_i, \mathbf{s}_{i+1}, \phi_i)\}_{i=1}^{B}$ from $\mathscr{R}$
    Compute targets:

$$y_i = r_i + \gamma Q_{\psi'}\big(\mathbf{s}_{i+1}, \mu_{\theta'}(\mathbf{s}_{i+1}, \Gamma_{\phi_i}, \beta_{\phi_i})\big)$$

    Critic update:

$$\mathscr{L}_Q = \frac{1}{B}\sum_{i=1}^{B}\big(Q_\psi(\mathbf{s}_i, a_i) - y_i\big)^2$$

    Minimize $\mathscr{L}_Q$ w.r.t. ψ
    Actor & Meta update:

$$\mathscr{L}_{\text{actor}} = -\frac{1}{B}\sum_{i=1}^{B} Q_\psi\big(\mathbf{s}_i, \mu_\theta(\mathbf{s}_i, \Gamma_{\phi_i}, \beta_{\phi_i})\big)$$

    Minimize $\mathscr{L}_{\text{actor}}$ w.r.t. θ (and ω if episode $\geq E_{\text{meta}}$)
    Soft-update targets:

$$\theta' \leftarrow \tau\,\theta + (1-\tau)\,\theta', \quad \psi' \leftarrow \tau\,\psi + (1-\tau)\,\psi'$$

  end for
end for

To ensure stability during training, target networks $\mu_{\theta'}$ and $Q_{\psi'}$ are maintained as slowly updated copies of the actor and critic networks [14], [15]. These target networks samples transitions $(s_i, a_i, r_i, s_{i+1}, \phi_i)$ from the replay buffer $\mathscr{R}$ [14], [15] to compute loss $\mathscr{L}_Q$ for updating the critic network. The critic loss is typically given by:

$$\mathscr{L}_Q = \mathbb{E}_{(s_i, a_i, r_i, s_{i+1}, \phi)\sim\mathscr{R}}\left[\big(Q_\psi(s_i, a_i) - y_i\big)^2\right] \tag{4}$$

where the target value y_i is computed using the Bellman equation as:

$$y_i = r_i + \gamma Q_{\psi'}(s_{i+1}, \mu_{\theta'}(s_{i+1}, \Gamma_{\phi_i}, \beta_{\phi_i})) \tag{5}$$

To encourage exploration Ornstein–Uhlenbeck (OU) noise $\mathscr{N}_t$ [14], [16] is applied during training.

To enable generalization across multiple tasks and configurations, the FC-RL method uses an additional meta-network $\mathscr{M}_\omega$ to condition the actor network inspired by FiLM [12]. This network receives input as a fixed-format feature vector ϕ as shown in Fig. 1 that describes properties of the problem instance. The feature vector is structured as $\phi = [\phi_{cont}, \phi_{bin}]$, where $\phi_{cont} \in \mathbb{R}^m$ contains normalized continuous features such as dimensionality and search-space bounds, and $\phi_{bin} \in [0,1]^k$ is a binary task identifier. Based on this input, the meta-network outputs two vectors, scale $\Gamma(\phi)$ and bias $\beta(\phi)$. These vectors are respectively applied multiplicatively and additively to the final hidden layer h of the actor network, effectively performing an affine transformation.

$$h' = \Gamma(\phi) \odot h + \beta(\phi) \tag{6}$$

This mechanism allows for stable conditioning of the actor network behavior based on the characteristics of the problem. In essence, the meta-network enables context-aware changes by directly integrating external meta-information, thereby facilitating more robust generalization. To ensure meaningful updates, training of the meta-network is delayed until episode E_{meta}, allowing the actor and critic networks to first stabilize and avoiding early learning from noisy gradients

To accommodate variable-sized inputs arising from different problem dimensions, we apply a standard zero-padding approach [17] that enables a single actor network to process inputs of varying lengths without requiring architectural changes. The actor network is configured to accept a fixed-size input vector corresponding to the maximum supported dimension. For problems with fewer dimensions, the remaining input entries are padded with zeros. These zero-valued components carry no functional signal and therefore do not affect the learning process.

TABLE I: Selected benchmark functions

Function	Modality	Separability
Sphere	Unimodal	Separable
Rosenbrock	Unimodal	Non-separable
Griewank	Multimodal	Non-separable
Ackley	Multimodal	Non-separable
Rastrigin	Multimodal	Separable
Levy	Multimodal	Non-separable

III. Experimental Setup

To evaluate the proposed FC-RL method, we used six well-known mathematical benchmark functions commonly employed in global optimization research due to their known global minimum [18]. As shown in Table I, these functions vary in modality, complexity, separability and landscape. For instance, the Sphere function offers a smooth convex landscape to test basic convergence, while the Levy function with its rugged, multi-modal landscape tests the ability to escape local optima.

To further increase the problem complexity and promote generalization, we apply shifted variants of each function by relocating their global optimum from the origin to randomly selected points in the domain. Given an original function

TABLE II: Agent hyperparameters and configuration

Item	Value
Actor Learning Rate	1e−4
Critic Learning Rate	1e−3
Discount Factor γ	0.99
Target Update τ	0.01
OU Noise $\mathscr{N}_t$	1
Replay Buffer $\mathscr{R}$	1e5
Batch Size B	64

TABLE III: Training configuration of benchmark functions

Function	Dim	Max. Search Bounds	Shift
Sphere	3–15	$[-5.12, 5.12]$	0-5
Rosenbrock	3–15	$[-5, 10]$	0-10
Griewank	3–15	$[-600, 600]$	0-100
Ackley	3–15	$[-32, 32]$	0-32
Rastrigin	3–15	$[-5.12, 5.12]$	0-5
Levy	3–15	$[-10, 10]$	0-10

$f(\mathbf{x})$, the shifted version is defined as:

$$f_{\text{shifted}}(\mathbf{x}) = f(\mathbf{x} - \mathbf{x}_s) \tag{7}$$

where $\mathbf{x}_s$ is the shift vector, sampled uniformly within the search-space. This encourages the agent to adapt its strategy based on the function characteristics rather than memorizing known solutions. These benchmark functions serve as the environment with which the FC-RL agent interacts.

The benchmark's dimensionality, bounds, and shift values were empirically chosen to create diverse, challenging scenarios that effectively test generalization under varying optimization conditions.

A. Implementation Details

In this study, the actor μ_θ and critic network Q_ψ were constructed as a feedforward neural network, each with two hidden layers of 256 neurons. All the hidden layers use rectified linear unit (ReLU) activation functions, with the exception of the final layer of the actor network, which uses the hyperbolic tangent (tanh) activation to constrain the output to the (-1, +1) range. These outputs were further mapped to valid action ranges defined by the function-specific search bounds in the environment. The actor network was designed to handle up to 35-D input vectors, with zero-padding applied for lower dimensions.

The meta-network $\mathscr{M}_\omega$ essential for generalization across benchmark functions was also implemented using a feed-forward neural network, with two hidden layers containing 128 and 256 neurons respectively. A 9-D feature vector ϕ encoding meta-information was used to condition the meta-network. The vector included binary task identifier ϕ_{bin} that specifies one of the six benchmark function type, as well as normalized continuous features ϕ_{cont} encoding input dimensionality, search-space bound, and global optimum shift. The agent's hyperparameters and configuration are summarized in Table II. The reward shaping parameters $\alpha = 1000$ and $\delta = 400$ were chosen because the agent struggled to learn when $f(x) < 400$, so δ marked the point where learning needed improving. α amplifies positive reward to counteract early large negative rewards and help the agent recognize progress.

B. Training Strategy

In this study, we trained the FC-RL agent for empirically chosen 15,000 episodes using a staged learning process that gradually increases problem difficulty. At each episode, the optimization landscape was defined by the selected benchmark function, its dimensionality, search-space bound and global minimum shift. An episode ends either when a solution meets a performance threshold i.e, reaching close to global minimum $f(x) < 0.01$ or after a maximum of 1000 steps. The number of episodes allocated to each training phase was empirically selected to ensure sufficient time for learning robust optimization policies under varying conditions

We organized the training into three progressive phases:

- **Phase 1: Fixed. (4500 episodes)** Each episode uses a fixed combination of benchmark function type, dimension, bounds, and shift. This allows the agent to build a foundational policy in stable environments.
- **Phase 2: Partially Randomized. (4500 episodes)** The benchmark function type and dimension follow the same order as before, but the bounds and shift values were now randomly sampled. This introduces variability in the search-space, requiring the agent to adapt to differently scaled versions of the same function.
- **Phase 3: Fully Randomized. (6000 episodes)** The environment was entirely randomized across the six benchmark function types, dimensionality, bounds, and shift. This phase encourages the agent to generalize across unseen combinations and search spaces.

In Phase 1, the agent underwent structured training, where the benchmark function and its configurations were varied in a staged manner. Specifically, six benchmark functions were each trained on for 900 episodes. Within each 900-episode span, the input dimensionality increased from 3 to 15 in increments of 3, with a change every 150 episodes. The bounds and shift remained fixed for each benchmark function. This incremental strategy allowed the agent to gradually learn to increasing problem complexity while encountering a variety of objective landscapes.

In Phase 2, variability was introduced by randomly changing the search-space bounds and shift within the maximum search bounds mentioned in Table III, while maintaining the same staged variation in benchmark functions for 900 episodes, with the dimensionality gradually increasing in increments of 3.

In Phase 3, the agent trained on randomly sampled benchmark functions, varying dimensionality, search-space bounds, and global minimum shifts without any fixed progression or schedule. This randomness prevents overfitting and drives the policy to generalize across diverse optimization tasks.

Throughout the training, dimensionality varied between 3

TABLE IV: Performance comparison of FC-RL and baseline methods (DDPG, PSO, DIRECT, GA) on In-distribution problems across benchmark functions. Results are reported as mean ± standard deviation over 10 independent runs.

Function	Dim	Shift	Bounds	FC-RL f(x)	DDPG f(x)	DIRECT f(x)	PSO f(x)	GA f(x)
Sphere	5	3	(-5, +5)	0.04 ± 0.00	0.04 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	10	2	(-5, +5)	0.09 ± 0.00	0.09 ± 0.00	2.32 ± 0.00	0.02 ± 0.00	0.05 ± 0.02
	12	4	(-5, +5)	0.00 ± 0.00	0.00 ± 0.00	3.85 ± 0.00	5.84 ± 2.05	0.09 ± 0.02
	15	3.5	(-5, +5)	0.00 ± 0.00	0.00 ± 0.00	0.41 ± 0.00	10.78 ± 4.67	0.17 ± 0.08
Rosenbrock	5	4.5	(-5, +10)	0.00 ± 0.00	0.00 ± 0.00	3.13 ± 0.00	0.04 ± 0.00	3.65 ± 1.32
	10	6	(-5, +10)	0.00 ± 0.00	0.00 ± 0.00	8.2 ± 0.00	4.41 ± 2.57	75.14 ± 31.20
	12	4	(-5, +10)	0.00 ± 0.00	0.00 ± 0.00	10.20 ± 0.00	11.02 ± 13.92	104.19 ± 41.28
	15	9	(-5, +10)	0.00 ± 0.00	0.00 ± 0.00	13.36 ± 0.00	48.99 ± 44.11	213.8 ± 92.82
Griewank	5	30	(-600, +600)	0.09 ± 0.00	0.09 ± 0.00	0.11 ± 0.00	0.02 ± 0.01	0.47 ± 0.23
	10	50	(-600, +600)	0.05 ± 0.00	0.05 ± 0.00	0.17 ± 0.00	0.06 ± 0.02	1.11 ± 0.12
	12	90	(-600, +600)	0.06 ± 0.00	0.06 ± 0.00	0.23 ± 0.00	0.06 ± 0.04	1.17 ± 0.07
	15	150	(-600, +600)	10.37 ± 0.00	10.37 ± 0.00	0.36 ± 0.00	0.06 ± 0.03	1.43 ± 0.2
Ackley	5	23	(-32, +32)	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.47 ± 0.62	1.48 ± 0.52
	10	6	(-32, +32)	0.00 ± 0.00	0.00 ± 0.00	4.32 ± 0.00	0.00 ± 0.00	2.79 ± 0.54
	12	12	(-32, +32)	0.00 ± 0.00	0.00 ± 0.00	12.1 ± 0.00	0.76 ± 0.67	3.19 ± 0.37
	15	28	(-32, +32)	0.00 ± 0.00	0.00 ± 0.00	19.26 ± 0.00	19.65 ± 0.51	3.51 ± 0.69
Rastrigin	5	4.5	(-5, +5)	0.00 ± 0.00	0.00 ± 0.00	0.77 ± 0.00	5.73 ± 2.19	0.25 ± 0.17
	10	3	(-5, +5)	0.00 ± 0.00	0.00 ± 0.00	55.41 ± 0.00	12.24 ± 7.43	2.33 ± 1.15
	12	2	(-5, +5)	0.00 ± 0.00	0.00 ± 0.00	35.48 ± 0.00	9.03 ± 5.06	3.05 ± 0.65
	15	4	(-5, +5)	0.00 ± 0.00	0.00 ± 0.00	232.8 ± 0.00	74.02 ± 9.5	6.39 ± 2.05
Levy	5	3	(-10, +10)	0.09 ± 0.00	0.09 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	10	8	(-10, +10)	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	7.92 ± 2.88	0.07 ± 0.02
	12	6	(-10, +10)	0.05 ± 0.00	0.05 ± 0.00	0.12 ± 0.00	4.5 ± 2.8	0.15 ± 0.05
	15	9	(-10, +10)	0.07 ± 0.00	0.07 ± 0.00	48.25 ± 0.00	46.11 ± 11.38	0.46 ± 0.11

and 15, while search-space bounds and shift values were constrained to remain within the maximum specified search-space bounds adopted from [18]. Table III summarizes the applied ranges.

IV. Results

After training, ability of the FC-RL agent to reach the global minimum under varying problem instances was evaluated. We categorized these test scenarios as In-distribution and Out-of-distribution, based on whether the problem was seen or unseen during the training.

A comparative evaluation was also conducted against three widely used optimization methods, Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and the DIRECT algorithm. These methods were selected to represent a broad spectrum of global optimization strategies. PSO from the swarm intelligence family, GA from evolutionary algorithms, and DIRECT as a deterministic, model-based approach. This diversity ensures that the comparison captures different strengths and trade-offs in handling optimization problems. All the baseline optimization methods were implemented using popular publicly available Python libraries. PSO was implemented using the PySwarms library [19], GA using PyGAD [20], and the DIRECT algorithm using the NLopt library [21]. To ensure fair and practical assessment, all the baseline methods were executed using fixed hyperparameter settings mentioned in Appendix B for all the test scenarios without any instance-specific tuning.

Additionally, the performance of the DDPG RL agent, trained in the similar manner as the proposed FC-RL method was analyzed to assess the significance of incorporating a meta-network.

A. In-distribution problems

In-distribution problems are the test instances whose configurations including dimensionality, search-space bounds, and global minimum shift, were sampled from the same distribution as used during training. Across all six benchmark functions, FC-RL successfully converged to the global optimum, reaching the predefined success threshold of $f(x) < 0.01$, without retraining or retuning. The agent demonstrated consistent convergence across multiple test instances as shown in Table IV, indicating a stable policy within the trained distribution. In broader evaluation of 600 randomly sampled in-distribution test instances (100 per benchmark), the agent reached global minimum i.e., $f(x) < 0.01$ in over 83% of the cases, and achieved near-optimal performance i.e., $f(x) < 5$ in all the cases. It can also be observed that the proposed method outperformed or matched the performance of all baseline methods, particularly in higher-dimensional test scenarios.

B. Out-of-distribution problems

Test instances with previously unseen configurations, including higher-dimensional inputs (15-35), extended search-space bounds beyond those seen during the training, and global minima located outside the previously explored regions, are out-of-distribution problems. Despite the increased complexity, the FC-RL agent showed strong convergence behavior under moderate deviations as seen in the Table V, often reaching near-optimal solutions without retraining or fine-tuning. In this context, Moderate deviation refers to test scenarios where the input dimensionality and search-space bounds were approximately two or three times those

TABLE V: Performance comparison of FC-RL and baseline methods (DDPG, PSO, DIRECT, GA) on Out-of-distribution problems across benchmark functions. Results are reported as mean ± standard deviation over 10 independent runs.

Function	Dim	Shift	Bounds	FC-RL f(x)	DDPG f(x)	DIRECT f(x)	PSO f(x)	GA f(x)
Sphere	20	3	(-5, +5)	0.00 ± 0.00	0.00 ± 0.00	11.11 ± 0.00	10.22 ± 3.43	4.33 ± 3.81
	25	8	(-10, +10)	0.00 ± 0.00	0.00 ± 0.00	480.0 ± 0.00	285.6 ± 57.20	16.96 ± 3.14
	30	13	(-15, +15)	0.00 ± 0.00	900.0 ± 0.00	$2.6e3 \pm 0.00$	$1.3e3 \pm 313.8$	199.5 ± 96.7
	35	40	(-50, +50)	0.00 ± 0.00	$1.1e4 \pm 0.00$	$2.6e4 \pm 0.00$	$1.7e4 \pm 3.5e3$	$2.1e3 \pm 413.2$
	35	105	(-110, +110)	874.9 ± 0.00	$4.2e4 \pm 0.00$	$1.8e5 \pm 0.00$	$1.4e5 \pm 2.4e4$	$2.4e3 \pm 4.1e3$
Rosenbrock	20	4	(-10, +10)	0.00 ± 0.00	0.00 ± 0.00	18.36 ± 0.00	55.62 ± 28.69	$3.8e3 \pm 1.3e3$
	25	12	(-15, +15)	0.00 ± 0.00	0.00 ± 0.00	23.79 ± 0.00	72.67 ± 48.88	$2.7e4 \pm 1.4e4$
	30	25	(-35, +35)	0.00 ± 0.00	$4.1e9 \pm 0.00$	28.82 ± 0.00	127.15 ± 42.71	$4.1e6 \pm 1.3e6$
	35	47	(-50, +50)	0.00 ± 0.00	$2e10 \pm 0.00$	33.77 ± 0.00	$1e3 \pm 2.4e3$	$2.2e7 \pm 5.2e6$
	35	110	(-120, +120)	0.00 ± 0.00	$3.3e11 \pm 0.00$	33.8 ± 0.00	$1.5e3 \pm 1.5e3$	$7.8e8 \pm 1.3e8$
Griewank	20	100	(-600, +600)	0.07 ± 0.00	0.07 ± 0.00	1.26 ± 0.00	0.05 ± 0.04	9.82 ± 2.46
	25	120	(-650, +650)	3.5 ± 0.00	3.5 ± 0.00	3.5 ± 0.00	0.04 ± 0.03	17.57 ± 2.72
	30	150	(-700, +700)	19.75 ± 0.00	34.75 ± 0.00	1.07 ± 0.00	0.46 ± 0.48	58.08 ± 7.29
	35	250	(-800, +800)	197.8 ± 0.00	222.8 ± 0.00	2.31 ± 0.00	17.61 ± 12.66	103.22 ± 8.29
	35	500	(-1000, +1000)	$1.4e3 \pm 0.00$	$1.5e3 \pm 0.00$	219.3 ± 0.00	253.8 ± 101.6	160.2 ± 30.26
Ackley	20	28	(-32, +32)	0.00 ± 0.00	0.00 ± 0.00	19.92 ± 0.00	19.85 ± 0.299	9.81 ± 0.85
	25	45	(-50, +50)	0.00 ± 0.00	0.00 ± 0.00	19.99 ± 0.00	20.17 ± 0.105	20.06 ± 0.00
	30	70	(-80, +80)	0.00 ± 0.00	19.94 ± 0.00	19.99 ± 0.00	20.39 ± 0.16	20.21 ± 0.05
	35	85	(-100, +100)	0.00 ± 0.00	19.98 ± 0.00	19.99 ± 0.00	20.62 ± 0.19	20.19 ± 0.03
	35	140	(-200, +200)	19.99 ± 0.00	19.99 ± 0.00	19.99 ± 0.00	20.73 ± 0.22	20.20 ± 0.02
Rastrigin	20	4	(-5, +5)	0.00 ± 0.00	0.00 ± 0.00	312.78 ± 0.00	115.86 ± 29.93	45.14 ± 6.85
	25	13	(-15, +15)	0.00 ± 0.00	0.00 ± 0.00	$1.8e3 \pm 0.00$	$1.1e3 \pm 210.4$	20.06 ± 0.00
	30	35	(-50, +50)	0.00 ± 0.00	$1.3e4 \pm 0.00$	$1.3e4 \pm 0.00$	$7.1e3 \pm 1.9e3$	$1.7e3 \pm 182.6$
	35	80	(-100, +100)	0.00 ± 0.00	$4.6e4 \pm 0.00$	$9.9e4 \pm 0.00$	$6.7e4 \pm 7.5e3$	$9.7e3 \pm 1.6e3$
	35	120	(-150, +150)	$1.3e4 \pm 0.00$	$6.1e4 \pm 0.00$	$2.9e5 \pm 0.00$	$1.2e5 \pm 1.5e4$	$3e4 \pm 1.9e4$
Levy	20	6	(-10, +10)	0.09 ± 0.00	0.09 ± 0.00	0.67 ± 0.00	18.66 ± 8.48	2.82 ± 0.70
	25	14	(-20, +20)	0.00 ± 0.00	0.00 ± 0.00	7.68 ± 0.00	189.7 ± 48.31	18.10 ± 4.23
	30	45	(-50, +50)	0.00 ± 0.00	$4.8e3 \pm 0.00$	$4.9e3 \pm 0.00$	$3.9e3 \pm 967.1$	931.7 ± 136.8
	35	65	(-80, +80)	0.00 ± 0.00	$6.8e3 \pm 0.00$	$1.5e4 \pm 0.00$	$8.4e3 \pm 2.1e3$	$1.7e3 \pm 294.4$
	35	125	(-150, +150)	$5.6e3 \pm 0.00$	$1.8e4 \pm 0.00$	$6.9e4 \pm 0.00$	$3.2e4 \pm 5.7e3$	$6.8e3 \pm 1e3$

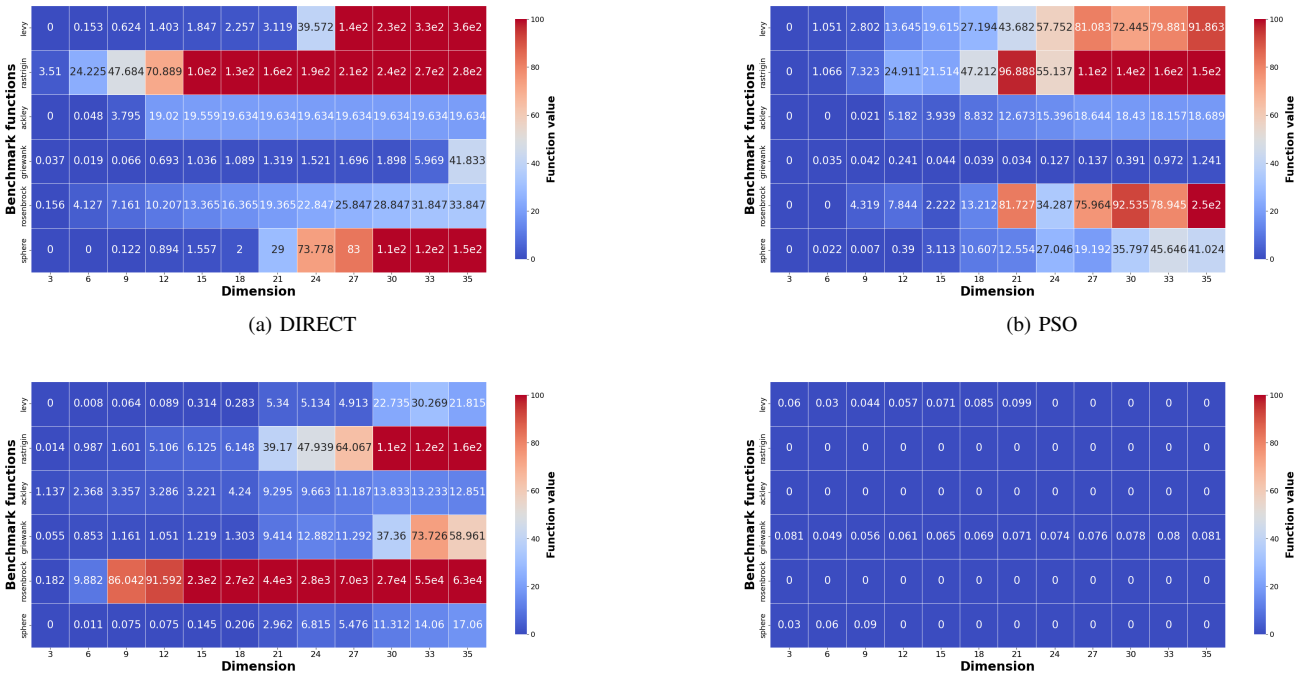
used during training. The Table V includes a range of progressively more difficult test cases designed to assess the method's scalability to complex optimization scenarios. These results indicate that the agent has learned a flexible and scalable optimization strategy that remains effective even under unseen conditions beyond those experienced during training. When tested on 600 randomly sampled out-of-distribution test instances with moderate deviation, the FC-RL agent reached global minimum in nearly 80% of the cases. Furthermore, the deterministic nature of FC-RL ensured that it consistently produces the same solution for a given input, thereby guaranteeing reproducibility and reliability.

Notably, even when the problem dimensionality was increased to 35 and the search-space was doubled or tripled relative to the training configuration, the FC-RL agent remained competitive, outperforming all the baseline methods except in the case of the Griewank function. This exception is likely due to its highly oscillatory landscape and the presence of large flat regions, which can mislead the policy and hinder effective convergence. Additionally, the DDPG RL agent showed significantly poorer performance in higher-dimensional settings often suffering from divergence compared to FC-RL. This highlights the critical role of the meta-network in ensuring scalability and robustness, particularly for out-of-distribution tasks. However, we also observed a degradation in FC-RL performance when the deviations from training conditions became substantial, as it failed to consistently converge to the global minimum.

For intuitive understanding, we generated heatmaps showing the performance of the proposed FC-RL and baseline methods across the selected multi-modal benchmark optimization functions with increasing dimensions up to 35. It can be seen that FC-RL heatmap shown in Fig. 2(d) stays predominantly near global optimum $f(x) < 0.01$, even at the higher dimensions as indicated by cooler dark blue regions. In contrast, the heatmaps of baseline methods in Fig. 2 exhibits increasing amounts of hotter red regions, representing solutions that are away from the global optimum.

V. Discussions

The experimental results showcase that the proposed FC-RL method consistently outperform baseline optimization algorithms, particularly in high dimensions. Notably, FC-RL achieves strong performance on out-of-distribution problems without requiring retraining or hyperparameter tuning, showcasing zero-shot generalization. This behavior can be attributed to the meta-network, which condition the RL policy through feature-wise linear modulation. This introduces an additional degree of freedom, enabling the policy to reconfigure itself to adapt to new problems. Additionally, it facilitates feature invariance, allowing the network learn to



(a) DIRECT (b) PSO

(c) GA (d) FC-RL

Fig. 2: Performance heatmaps across different dimensions and benchmark functions for (a) DIRECT (b) PSO, (c) GA, and (d) FC-RL.

separate universal optimization behavior from task-specific adjustment which allow for better transfer across tasks [12].

However, it is important to acknowledge that the performance of FC-RL degrades when the test instances diverge significantly from the training distribution, highlighting FC-RL's sensitivity to how well its training data reflects the test conditions. Furthermore, like most reinforcement learning methods, FC-RL's effectiveness is sensitive to the number of training episodes, fewer training episodes result in reduced performance. It is also crucial to clarify the scope of FC-RL's generalization capabilities. The policy is explicitly conditioned on the task or function type, which is hard-coded during training. As such, FC-RL does not generalize to novel task types but rather, it generalizes across unseen configurations within known task or function types. This design choice is deliberate and aligns with practical scenarios where variability is morcfte often encountered in configurations than in system types. For example, in the domain of electric motor design, the emergence of entirely new motor types is rare. However, variations in design parameters or configurations within existing motor types occur frequently [22] [23]. Similar trends are observed in controller tuning, where the control architecture remains fixed but parameters must be adjusted to reflect changes in system dynamics, such as varying loads, friction, or delays [24]. In such contexts, FC-RL's generalization across varying configurations of known multiple systems, rather than entirely new systems is both practical and desirable.

In future work, we aim to apply FC-RL to more realistic engineering optimization tasks, with a specific focus on lightweight actuator design. Future directions may also include extending FC-RL with a meta-network that conditions the policy based on function landscape characteristics, to explore its generalization to entirely unseen function types and the efficiency of meta-features extraction in real-world settings.

VI. Conclusion

In this paper, we presented Feature-Conditioned Reinforcement Learning (FC-RL) method for engineering optimization, which often achieves global convergence and zero-shot generalization across high-dimensional problems without retraining or retuning. We demonstrated FC-RL's global convergence ability across six benchmark functions, showing that it often outperforms baseline optimization algorithms on both in-distribution and moderately out-of-distribution test instances. Notably, FC-RL achieved global convergence in nearly 80% of the moderately out-of-distribution cases, underscoring its robustness under unseen and challenging configurations. By removing the need for problem-specific retraining by using meta-network, FC-RL points toward generalizable optimization pipelines that could shorten design cycles and lower computational costs for engineering applications. Additionally, its deterministic nature ensures that the same input consistently produce the same output, enhancing reproducibility. These results highlight FC-RL's potential for practical implementation in settings such as actuator design, which we will explore in future work

APPENDIX

A. Selected Benchmark Functions

Let $\tilde{\mathbf{x}}$ denote the shifted input vector representing the candidate solution,

1) Sphere Function:

$$f(\mathbf{x}) = \sum_{i=1}^{d} \tilde{x}_i^2 \tag{8}$$

2) Rosenbrock Function:

$$f(\mathbf{x}) = \sum_{i=1}^{d-1} \left[100(\tilde{x}_{i+1} - \tilde{x}_i^2)^2 + (\tilde{x}_i - 1)^2\right] \tag{9}$$

3) Griewank Function:

$$f(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{d} \tilde{x}_i^2 - \prod_{i=1}^{d} \cos\left(\frac{\tilde{x}_i}{\sqrt{i}}\right) + 1 \tag{10}$$

4) Ackley Function:

$$f(\mathbf{x}) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}\tilde{x}_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(c\tilde{x}_i)\right) + 20 + e \tag{11}$$

where $a = 20, \quad b = 0.2,$ *and* $\quad c = 2\pi$

5) Rastrigin Function:

$$f(\mathbf{x}) = 10d + \sum_{i=1}^{d}\left[\tilde{x}_i^2 - 10\cos(2\pi\tilde{x}_i)\right] \tag{12}$$

6) Levy Function:

$$f(\mathbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2\left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_d - 1)^2\left[1 + \sin^2(2\pi w_d)\right] \tag{13}$$

where $w_i = 1 + \frac{\tilde{x}_i - 1}{4}$

B. Baseline Hyperparameters

All baseline algorithms ran for a fixed number of iterations, consistent with the maximum steps used in an FC-RL episode. Each algorithm operated within problem-specific constraints defined by the function domain. Hyperparameters used in baseline methods are specified in Table VI

TABLE VI: Baseline hyperparameters

Optimizer	Key Settings
PSO (PySwarms)	$w = 0.9$, $c_1 = 0.5$, $c_2 = 0.3$; swarm size: 50; no velocity clamping; global-best topology.
GA (PyGAD)	Population size: 50; selection: 'sss'; crossover: single-point; mutation: random (10% genes).
DIRECT (NLopt)	No tunable hyperparameters

REFERENCES

[1] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[2] F. Eichin, R. Werle, and M. van Werkhoven, "Digitalisation in electric motor systems – Part IV: Energy consumption due to the digitalisation of electric motor systems," 4E Electric Motor Systems Platform (EMSA), Tech. Rep., May 2024.

[3] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Hoboken, NJ, USA: John Wiley & Sons, 2009.

[4] E. Özcan and M. Yılmaz, "Particle swarms for multimodal optimization," in *Adaptive and Natural Computing Algorithms*, B. Beliczynski *et al.*, Eds. Berlin, Germany, Springer, 2007, pp. 366–375, doi: 10.1007/978-3-540-71618-1_41.

[5] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Proc. Int. Symp. Stochastic Algorithms*. Berlin, Germany, Springer, Oct. 2009, pp. 169–178.

[6] N. Casas, "Genetic algorithms for multimodal optimization: A review," *arXiv* preprint arXiv:1508.05342, 2015.

[7] W. Cai, L. Yang, and Y. Yu, "Solution of Ackley function based on particle swarm optimization algorithm," in *Proc. 2020 IEEE Int. Conf. Adv. Electr. Eng. Comput. Appl. (AEECA)*, Dalian, China, 2020, pp. 563–566, doi: 10.1109/AEECA49918.2020.9213634.

[8] J. Wang, "Enhancing particle swarm algorithm for multimodal optimization problems," in *Proc. 2017 Int. Conf. Comput. Intell. Inf. Syst. (CIIS)*, Nanjing, China, 2017, pp. 1–6, doi: 10.1109/CIIS.2017.10.

[9] F. Zhang and Z. Chen, "A novel reinforcement-learning-based particle swarm optimization algorithm for better symmetry between convergence speed and diversity," *Symmetry*, vol. 16, no. 10, Art. no. 1290, 2024.

[10] M. W. da Silva Filho, G. A. Barbosa, and P. B. Miranda, "Learning global optimization by deep reinforcement learning," in *Proc. Brazilian Conf. Intelligent Syst.*, Cham, Switzerland: Springer, Nov. 2022, pp. 417–433.

[11] H. Zhang, J. Sun, and Z. Xu, "Learning to be global optimizer," *arXiv* preprint arXiv:2003.04521, 2020.

[12] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, Apr. 2018.

[13] K. Li and J. Malik, "Learning to Optimize," *arXiv* preprint arXiv:1606.01885, 2016.

[14] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv* preprint arXiv:1509.02971, 2015.

[15] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[16] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the Brownian motion," *Phys. Rev.*, vol. 36, no. 5, pp. 823–841, Sep. 1930.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: `http://www.deeplearningbook.org`

[18] M. Z. Naser *et al.*, "A review of 315 benchmark and test functions for machine-learning optimization algorithms and metaheuristics with mathematical and visual descriptions," *arXiv* preprint arXiv:2406.09581, 2024.

[19] L. J. Miranda, "PySwarms: A research toolkit for particle swarm optimization in Python," *J. Open Source Softw.*, vol. 3, no. 21, p. 433, 2018.

[20] A. F. Gad, "PyGAD: An intuitive genetic-algorithm Python library," *Multimedia Tools Appl.*, vol. 83, no. 20, pp. 58029–58042, 2024.

[21] U. Kumar and S. Soman, "Benchmarking NLopt and state-of-the-art algorithms for continuous global optimization via IACOR," *Swarm and Evol. Comput.*, vol. 27, pp. 116–131, 2016.

[22] L. R. Devi, S. Sreekumar, R. Bhakar, Dileep G. and S. Padmanaban, "Electric motor modeling, analysis, and design for E-mobility applications: A state of the art," *e-Prime–Advances Electr. Eng., Electron. Energy*, Art. no. 100985, 2025.

[23] B. Bilgin *et al.*, "Modeling and analysis of electric motors: State-of-the-art review," *IEEE Trans. Transp. Electrific.*, vol. 5, no. 3, pp. 602–617, Sep. 2019.

[24] K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tuning*, 2nd ed. Research Triangle Park, NC, USA, ISA – The Instrumentation, Systems and Automation Society, 1995.