RARE: Robustness Assessment and Regularized Enhancement A Study in Function Estimation and Symbolic Regression

Masoud Ebrahimi[1], Qamar Alfalouji[2]
[1]Mälardalen University, Sweden
[2]TU Graz, Austria

Abstract— **Robustness against noise and input perturbations is a crucial requirement for trustworthy machine learning (ML). We introduce *Robustness Assessment and Regularized Enhancement (RARE)*, a model-agnostic framework that evaluates and improves robustness in regression tasks, including neural and symbolic approaches. RARE centres on a user-specified noise model (e.g., Gaussian or Laplace) to differentiate intended from unintended input variations, yet in practice generalises to unseen noise. Its four components are: (i) a *significant-pattern operator* that per feature synthesizes a single representative noise—worst-case in our tests, with best- or average-case also supported; (ii) a *robustness assessor* that measures the ratio of input deviation induced by significant patterns to output variation, analogous to estimating the inverse of local Lipschitz constants of the learned function; (iii) a *data-augmentation module* that increases training exposure to noise using the significant patterns; and (iv) a *regulariser* that embeds the metric in the loss to penalize low robustness. Because RARE targets the Lipschitz behaviour of the function under learning rather than the model's, it neither requires white-box access to model parameters nor assumes continuity or differentiability of the learning method. Consequently, RARE applies to off-the-shelf models such as Random Forests (RF) and even End-to-End Transformer-based Symbolic Models (E2E). Unlike adversarial methods, RARE does not require generating extensive synthetic data to improve robustness. Experiments showed MLPs trained with the regularized loss exhibit 7–33% higher robustness under Gaussian noise and up to a 32.3% improvement under high-variance Laplace noise. Regularized CNNs outperform unregularized CNNs in low-noise conditions under Gaussian noise. A regularized symbolic equation learner converges in as few as 3,000 epochs for some cases, while the non-regularized one converged in 20,000 epochs. Finally, we accelerated Meta's E2E, cutting fitting time by 58.69% across all tested equations.**

I. Introduction

In safety-critical domains such as autonomous driving, medical imaging, and high-frequency trading, even small unintended input perturbations can trigger severe consequences [1]–[4]. In Machine Learning (ML), robustness is the ability of a model to maintain acceptable performance against noisy and adversarial inputs in an operational environment [5]. Therefore, robustness is a critical requirement for ML models deployed in real-world applications. Despite this, most ML pipelines quantify robustness by measuring how accuracy- and precision-based metrics vary under noisy scenarios, overlooking the noise properties; e.g., measuring noise variations and their distributions.

In this paper, we introduce *Robustness Assessment and Regularized Enhancement (RARE)*, a unified framework for assessing and enhancing robustness in regression tasks, including function estimation with neural networks, random forests, and even transformer-based symbolic regression. RARE centers on a user-specified noise model (e.g., Gaussian or Laplace) to differentiate intended from unintended input variations. More specifically, a variation in the input space is considered unintended if it is captured by the noise model and intended otherwise. RARE's four components are:

1) Significant pattern operator: This is the core of RARE, that constructs intended and unintended variations in the input space. Given a user-chosen noise model (Gaussian, Laplace, etc.), we derive for each input feature a *single* significant pattern that captures the worst-case (alternatively best- or average-case) deviation allowed by that distribution. These significant patterns represent the limits of unintended variations in the input space. We determine the boundaries of desired and undesired deviations under a noise model in the output space similarly. The model's robustness is assessed against the intended input and desired output boundaries represented by significant input-output patterns. These patterns are computed once and reused throughout assessment and training, yielding a data footprint independent of dataset size.

2) Robustness assessor: We define robustness as the ratio between the undesired deviations in model output and the magnitude of unintended input variations governed by significant patterns. Mathematically, this value approximates a *local* Lipschitz constant of the *function under learning*, not of the model architecture; hence, no gradients, Jacobians, or spectral norms are required.

3) Data-augmentation module: Because one pattern per feature already captures worst-case noise, augmenting the training set with exactly these samples suffices to reproduce the benefits of adversarial and GAN-based schemes while avoiding their data explosion.

4) Regulariser: The assessed robustness score is embedded directly in the objective function, penalising regressors proportional to their sensitivity to noise and steering optimisation toward solutions with higher empirical robustness.

We conducted experimental evaluations across a diverse set of equations from AI-Feynman [6], [7] and Deep Symbolic Regression (DSR) [8] to asses the robustness of Random Forests (RF), Linear Regression (LR), Multi-Layer Perceptrons (MLPs), Convolutional NNs (CNNs), and End-to-End symbolic regression with transformers (E2E) [9]. Experiments with Gaussian, Uniform, and Laplace noise at varying intensities showed significant robustness variability across regression methods, with RF consistently outperforming others. We also conducted experiments to enhance robustness by training with the regularized loss function $\mathrm{MSE}_{\mathcal{P}}$ that incorporates the robustness measure as a penalty term in the original Mean Squared Error MSE loss function. We compared the performance of MLPs and CNNs trained using MSE against those trained with $\mathrm{MSE}_{\mathcal{P}}$, evaluating them in terms of validation MSE, robustness score $\mathcal{R}$, and training time. Models trained with the regularized loss function show significantly improved robustness, reduced validation errors, and faster convergence. Additionally, we extended our experiments to symbolic regression by using the Equation Learner (DSR-EQL) in [8], modifying its fitting objective from maximizing the standard R^2 to minimizing the proposed $\mathrm{MSE}_{\mathcal{P}}$. This led to faster convergence and improved robustness, while maintaining baseline performance on clean data. Finally, we introduced a noise-aware training data augmentation by incorporating the extracted significant patterns. Both training methods improved robustness and often significantly reduced training and fitting times, while maintaining or even enhancing performance on clean data.

A. Related Work

To the best of our knowledge, RARE is the first framework to unify robustness assessment and training in regression tasks in a model-agnostic way. This subsection establishes the relation of RARE to, and distinguishes it from, existing efforts in ML robustness in the following areas.

1) Accuracy- and Precision-based Robustness: In the context of ML robustness—primarily in classification tasks—methods like adversarial training [4] or randomized smoothing [10] have been proposed to improve robustness, but they mostly rely on accuracy and precision-based metrics. Accuracy-based indicators treat every misclassification or regression error equally; they never reveal *why* the error occurs or whether it could have been predicted from the statistics of the environment. Without an explicit noise model, intended signal variations and unintended corruptions become indistinguishable, leaving practitioners with no actionable levers for improvement [4], [11]–[13].

2) Lipschitz Constant: Bounding a network's Lipschitz constant can certify that no small perturbation flips a classifier's decision [14], [15], yet three practical barriers remain. First, the bound hinges on architectural continuity and differentiability, excluding high-performing but non-smooth learners such as random forests, boosted trees, or transformer-based symbolic regressors. Second, computing or even approximating a tight constant for such models is rarely feasible. Third, a universal bound ignores the *directional* nature of real-world noise: audio sensors are dominated by Gaussian noise, imaging pipelines by low-frequency blur, and finance feeds by Laplace-like shocks.

3) Sensitivity Analysis: Variance-decomposition tools (e.g., Sobol and total-order indices) attribute output spread to inputs [16]. They do not, however, distinguish deterministic sensitivity from stochastic corruption. This is crucial as high sensitivity to input variations does not directly translate to low robustness guarantees and vice versa. For instance, an estimator of $y = e^x$ is extremely sensitive to x yet can still be robust if it learns to denoise x then extrapolate y.

4) Data Augmentation: Adversarial training, GAN-driven augmentation, and additive-noise injection (e.g., randomized smoothing) can certify robustness and improve stability but require orders-of-magnitude more synthetic data, GPU time, and hyper-parameter tuning [4], [10], [17]. For large language-model regressors or evolutionary symbolic engines, such expansion is often prohibitive.

B. Contributions

RARE is a unified framework that tackles the above limitations for regression tasks with MLPs, CNNs, LR, RF, and various symbolic regression methods.

1) Lipschitz Constant: RARE's robustness score approximates a local Lipschitz constant of the function under learning, not of the model's architecture. By moving the Lipschitz constant from the model to the function under learning, we can assess a regressor's robustness without requiring differentiability or continuity. This is a key distinction, as it allows RARE to be applied to non-smooth learners (e.g., random forests, boosted trees, and transformer-based symbolic regressors) without requiring gradients or Jacobians. Moreover, by making the noise model explicit, Lipschitz constants become directional, allowing for a more accurate assessment of robustness.

2) Efficient Data-Augmentation: Significant patterns are representative of the worst-case perturbation allowed by the noise model defining the boundary between intended and unintended variations. One such representative perturbation per feature replaces many synthetic adversarial samples. Experimental evaluations showed that augmenting the training set with these representative perturbations results in improvements akin to those achieved by adversarial training, but with significantly less data and time.

3) Universally Applicable Regulariser: RARE's regulariser is universally applicable to any regression task, including function estimation with neural networks, and fitting evolutionary algorithms or transformer-based symbolic regressors. Experimental evaluations show consistent robustness and performance gains on multi-layer perceptrons, convolutional networks, and Meta's E2E transformer-based symbolic regressor with reduced training time.

The remainder of the paper reads as follows. Section II formalises noise models and notation. Sections III and IV detail the assessment and enhancement procedures. Section V specifies experimental settings. Section VI reports results. Section VII concludes with limitations and future work.

II. Preliminaries

We aim to evaluate the robustness against input perturbations in regression task. Let $\mathbf{X} \in \mathbb{R}^{n\times l}$ and $\mathbf{Y} \in \mathbb{R}^{m\times l}$ denote input and output samples, respectively. The vector $\mathbf{x}_i$ refers to the i^{th} input variable across all samples, and $\mathbf{y}_j$ to the j^{th} output variable. We denote the set of all variables in $\mathbf{X}$ as $\text{vars}(\mathbf{X})$, and use $\mathbf{x}_i \in \mathbf{X}$ and $\mathbf{x}_i \in \text{vars}(\mathbf{X})$ interchangeably. We define $\mathbf{C} = [\mathbf{X}\ \mathbf{Y}] \in \mathbb{R}^{(n+m)\times l}$ as the concatenated data matrix, where $\mathbf{c}_i = \mathbf{x}_i$ for $i \leq n$, and $\mathbf{c}_i = \mathbf{y}_{i-n}$ for $i > n$. Given any variable $\mathbf{v} \in \mathbb{R}^l$, we denote by $\mathbf{X}[\mathbf{x}/\mathbf{v}]$ a copy of $\mathbf{X}$ where $\mathbf{x}$ is replaced with $\mathbf{v}$. Let $f : \mathbf{X} \to \mathbf{Y}$ be the function under learning, and $\mathcal{M} : \mathbb{R}^n \to \mathbb{R}^m$ be a learned regressor estimating f.

Definition 1 (Noise Model). A noise model is defined as $\mathcal{N} = \langle \mathcal{D}, \theta, \mathcal{P}, k \rangle$, where $\mathcal{D}$ is a probability distribution representing the noise type, such as Gaussian Distribution $\mathtt{G}(\mu, \sigma^2)$, $\theta \in \mathbb{R}^2$ is a distribution-specific parameters, explicitly given as $\theta = \{\mu, \sigma^2\}$ for Gaussian distribution, $\theta = \{a, b\}$ for Unifrom distribution, and $\theta = \{\mu, b\}$ for Laplace distribution, $\mathcal{P}$ explicitly scales the magnitude of noise (as percentage scaling), and k is the number of noise samples used for robustness evaluation. Of note, $\mathbf{N} = \mathcal{N}(l)$ denotes k noise of length l sampled from $\mathcal{D}(\theta)$, scaled by $\mathcal{P}$.

The above definition is consistent with the literature [17]–[19]. Finally, for $\mathcal{D}(\theta)$, we consider Gaussian $\mathtt{G}(\mu, \sigma^2)$, Laplace $\mathcal{L}(\mu, b)$, and Uniform $\mathtt{U}(a, b)$ distributions.

Definition 2 (Noise Modulation). Given $\mathbf{X}$ and $\mathcal{N}$, let $\mathbf{N}_i \in \mathbb{R}^{k \times l}$ be the k noise vectors for $\mathbf{x}_i \in \mathbf{X}$. We define $\hat{\mathbf{X}}_i \in \mathbb{R}^{k \times l}$ by

$$\hat{\mathbf{X}}_i = \begin{bmatrix} \mathbf{x}_i + \mathbf{n}_1 & \dots & \mathbf{x}_i + \mathbf{n}_k \end{bmatrix} \text{ for } \mathbf{n}_p \in \mathbf{N}_i \; .$$

We then define $\hat{\mathbf{X}} \in \mathbb{R}^{k \times n \times l}$ by

$$\hat{\mathbf{X}} = \begin{bmatrix} \hat{\mathbf{X}}_1 & \dots & \hat{\mathbf{X}}_n \end{bmatrix} \text{ for } \hat{\mathbf{X}}_i \text{ of } \mathbf{x}_i \in \mathbf{X} \; .$$

Given a regressor $\mathcal{M}$, $\mathbf{X}$, and $\mathcal{N}$, the output $\mathcal{M}(\hat{\mathbf{X}}_i) \in \mathbb{R}^{k \times m \times l}$ denotes the results of applying $\mathcal{M}$ to k copies of $\mathbf{X}$, where only $\mathbf{x}_i$ is replaced with a noisy version $\hat{\mathbf{x}}_i^p \in \hat{\mathbf{X}}_i$. More formally,

$$\mathcal{M}(\hat{\mathbf{X}}_i) = \left[\mathcal{M}\big(\mathbf{X}\left[\mathbf{x}_i/\hat{\mathbf{x}}_i^p\right]\big)\right]_{p=1}^{k}$$

$\mathcal{M}(\hat{\mathbf{X}}_i)$ has k estimations for $\mathbf{y}_j \in \mathbf{Y}$, denoted by $\hat{\mathbf{Y}}_j$. Likewise, We define $\mathcal{M}(\hat{\mathbf{X}})$ denotes applying $\mathcal{M}$ to k perturbed copies of $\mathbf{X}$, where all features $\mathbf{x}_i \in \mathbf{X}$ are simultaneously replaced with their p^{th} noisy counterparts in respective $\hat{\mathbf{X}}_i$. Formally,

$$\mathcal{M}(\hat{\mathbf{X}}) = \left[\mathcal{M}\big(\mathbf{X}\left[\mathbf{x}_i/\hat{\mathbf{x}}_i^p\right]_{i=1}^{n}\big)\right]_{p=1}^{k}$$

Given a target set $\mathbf{T} \subseteq \mathbf{X}$, we define $\mathcal{M}(\hat{\mathbf{T}})$ analogously to $\mathcal{M}(\hat{\mathbf{X}})$, applying perturbations only to variables in $\mathbf{T}$. For brevity, we define $\hat{\mathbf{Y}} = \mathcal{M}(\hat{\mathbf{T}})$, where $\hat{\mathbf{Y}}$ implicitly assumes target set from context when not explicitly stated.

III. Robustness Testing

Robustness testing evaluates *how much* $\mathcal{M}$'s output changes when its input is perturbed by noise. Assuming a noise model $\mathcal{N} = \langle \mathcal{D}, \theta, \mathcal{P}, k \rangle$ that systematically perturbs $\mathbf{X}$ to $\hat{\mathbf{X}}$, we aim to quantify how changes in $\mathbf{X}$—drawn from $\mathcal{N}$—propagate through $\mathcal{M}$ and affect its output.

The *Lipschitz continuity* provides an upper bound on the model's sensitivity to input perturbations. Smaller Lipschitz constants imply greater robustness (i.e., large input changes cause minimal output changes) while larger Lipschitz constants imply higher sensitivity and lower robustness.

A. Significant Patterns

Evaluating robustness over all noisy samples $\hat{\mathbf{X}}$ is computationally impractical. Instead, we introduce the *significant patterns* as a compact summary that captures the essential impact of noise on inputs and outputs. Specifically, for each variable $\mathbf{x}_i \in \mathbf{X}$ and its perturbed set $\hat{\mathbf{X}}_i$, we derive a *significant pattern* from $\hat{\mathbf{X}}_i$ that captures the desired characteristic of deviations induced by $\mathcal{N}$. This pattern serves as a surrogate for $\hat{\mathbf{X}}_i$ in robustness assessments, thereby reducing the computational cost.

Definition 1 (Significant Pattern Operator). Given $\mathbf{x}_i \in \mathbf{X}$, we define the significant pattern $\mathcal{G}(\mathbf{x}_i, \hat{\mathbf{X}}_i, \Omega)$ as:

$$\mathcal{G}(\mathbf{x}_i, \hat{\mathbf{X}}_i, \Omega) = \Omega\left(\delta(\mathbf{x}_i, \min(\hat{\mathbf{X}}_i^\top)), \delta(\mathbf{x}_i, \max(\hat{\mathbf{X}}_i^\top))\right),$$

where $\min(\hat{\mathbf{X}}_i^\top)$ and $\max(\hat{\mathbf{X}}_i^\top)$ represent the element-wise minimum and maximum deviations of $\hat{\mathbf{X}}_i$ around $\mathbf{x}_i$; $\delta : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}^l$ is an element-wise distance function (e.g., absolute difference); $\Omega : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}^l$ aggregates the deviations (default: MAX).

Analogously, $\mathcal{G}(\mathbf{y}_j, \hat{\mathbf{Y}}_j, \Omega)$ is defined for outputs. A maximal boundary deviation from the clean $\mathbf{x}_i$ under the noise model $\mathcal{N}$ is represented by $\mathcal{G}(\mathbf{x}_i, \hat{\mathbf{X}}_i, \max)$.[1]

Inverse-Lipschitz Robustness Using the significant pattern operator, we quantify robustness as the ratio of input perturbations to resulting output deviations:

$$\mathcal{R}_{ij} = \frac{\Delta(\mathbf{x}_i, \mathcal{G}(\mathbf{x}_i))}{\Delta(\mathbf{y}_j, \mathcal{G}(\mathbf{y}_j))}.$$

where $\Delta(u, v) : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}$ denotes a chosen distance function[2] (e.g., Euclidean norm) measuring the distance between vectors u and v, and the measure $\mathcal{R}_{ij}$ quantifies the input perturbation required to induce a unit change in the output; that is, a higher ratio $\mathcal{R}_{ij}$ indicates greater robustness. We adopt the inverted local Lipschitz constant so that robustness is positively oriented; higher is better, making the measure easier to interpret and compare. $\mathcal{R}_{ij}$ is computed using a single worst-case representative marking the boundaries between intended and unintended variations in the input-space represented by $\hat{\mathbf{X}}_i$. Since $\hat{\mathbf{X}}_i$ is generated from a finite number of perturbed samples, both $\mathcal{G}(\mathbf{x}_i)$ and $\mathcal{G}(\mathbf{y}_i)$ provide rough boundaries in input and output spaces influenced by $\mathcal{N}$; hence, $\mathcal{R}_{ij}$ provides an approximation of the ideal robustness measure. As $k \to \infty$, the $\mathcal{R}_{ij}$ converges to the true expectation, which is akin to estimating the function under learning's local Lipschitz constant inverted: large input deviations result in minimal output instability, indicating that the model is resilient to perturbations.

B. Multiple Inputs, Single Output

When multiple input variables affect a single output $\mathbf{y}_j$, we weight their contributions differently based on statistical importance and sensitivity. These weights are summarized in a *weight matrix* W.

[1]As of this point, $\mathcal{G}(\mathbf{x}_i, \hat{\mathbf{X}}_i) = \mathcal{G}(\mathbf{x}_i, \hat{\mathbf{X}}_i, \max)$ unless otherwise stated. $\mathcal{G}(\mathbf{x}_i)$ assumes the second argument from the context.

[2]Please do not confuse Δ with element-wise δ in Definition 1.

Definition 2 (Weight Matrix). Given $\mathbf{X}$ and $\mathbf{Y}$, a weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ quantifies the effect of perturbations in $\text{vars}(\mathbf{X})$ on $\text{vars}(\mathbf{Y})$. Each entry w_{ji} quantifies how perturbations in input $\mathbf{x}_i$ affect output $\mathbf{y}_j$.

1) Computing Weight Matrix We can estimate $\mathbf{W}$ using the conditional density $p_{\mathbf{y}_j}(\mathbf{y}_j \mid \mathbf{X})$, which reflects the likelihood of an output $\mathbf{y}_j$ given all inputs $\mathbf{X}$, together with the marginal density $p_{\mathbf{x}_i}(\mathbf{x}_i)$ that captures the distribution of the input $\mathbf{x}_i$. This is analogous to using the Radon-Nikodym derivative, where $p_{\mathbf{x}_i}$ and $p_{\mathbf{y}_j}$ act as derivatives of their respective probability measures wrt. a base Lebesgue measure on $\mathbb{R}^n$ defined for these PDFs.[3]

Since different values of $\mathbf{x}_i$ might be affected differently by perturbations and the impact of these perturbations on $\mathbf{y}_j$ may not be uniform across all values of $\mathbf{x}_i$, we incorporate the term $\ell_p\big(\mathbf{x}_i, \mathcal{G}(\mathbf{x}_i)\big)$ to quantify the deviation of a representative perturbed input $\mathcal{G}(\mathbf{x}_i)$ from its baseline $\mathbf{x}_i$. This term serves as a sensitivity metric, capturing how variations in $\mathbf{x}_i$ influence $\mathbf{y}_j$. By integrating the product of these components over the domain of $\mathbf{x}_i$, we scale input values by both their likelihood and their perturbation sensitivity. This integration yields an aggregate measure of the effect of perturbations in $\mathbf{x}_i$ on $\mathbf{y}_j$, analogous to how the Radon-Nikodym derivative relates one measure to another.[4] Formally, we define w_{ji} as

$$w_{ji} = \int p_{\mathbf{y}_j}(\mathbf{y}_j \mid \mathbf{X})\, p_{\mathbf{x}_i}(\mathbf{x}_i)\, \ell_p\big(\mathbf{x}_i, \mathcal{G}(\mathbf{x}_i)\big)\, d\mathbf{x}_i\,.$$

This method is especially useful where the relationships between $\mathbf{X}$ and $\mathbf{Y}$ are uncertain or complex, as it rigorously weights all contributing factors by their likelihood, thereby capturing the full variability inherent in the data.

2) Feature Normalization We apply min–max normalization to all $\mathbf{x}_i \in \mathbf{X}$ and their corresponding $\mathcal{G}(\mathbf{x}_i)$, using the $\min(\mathbf{x}_i)$ and $\max(\mathbf{x}_i)$. Analogously, we apply min–max normalization to all outputs $\mathbf{y}_j \in \mathbf{Y}$ and their corresponding $\mathcal{G}(\mathbf{y}_j)$. This consistent scaling preserving relative differences and ensuring stable numerical computations.

Definition 3 (Robustness Measure). Given $\mathbf{X}$, an output variable $\mathbf{y}_j \in \mathbf{Y}$, a weight matrix $\mathbf{W}$, a noise model $\mathcal{N}$, and a regressor $\mathcal{M}$, we define the robustness measure $\mathcal{R}_j$ as:

$$\mathcal{R}_j(\mathbf{X}, \mathbf{y}_j, \mathcal{N}, \mathcal{M}) = \frac{\sum_{i=1}^{n} w_{ji} \cdot \Delta\big(\mathbf{x}_i, \mathcal{G}(\mathbf{x}_i)\big)}{\Delta\big(\mathbf{y}_j, \mathcal{G}(\mathbf{y}_j)\big)}, \tag{1}$$

where $\Delta : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}$ is a distance function.

A greater $\mathcal{R}_j$ indicates that $\mathbf{y}_j$ remains relatively stable under noisy inputs, implying higher robustness. Conversely, smaller values suggest greater sensitivity, as even moderate input perturbations lead to significant variations in $\mathbf{y}_j$.

[3]The measure-theoretic foundations, including the role of Radon-Nikodym derivatives and Lebesgue integration in defining probability densities, are beyond the scope of this paper.

[4]The integration is performed in the sense of a Lebesgue integral, which rigorously handles a broad class of measurable functions, including those with discontinuities or complex behavior.

IV. Training Robust Models

This section presents a robustness enhancement procedure. We introduce two complementary strategies: (1) a *regularized loss function* that penalizes sensitivity to input perturbations using the introduced robustness measure, and (2) a noise-aware data augmentation scheme incorporating significant patterns. By exposing the model to challenging perturbations, this approach enhances robustness while preserving accuracy. In both cases, the trained model achieves an optimal trade-off between performance and robustness.

A. Regularized Loss Function

Given a baseline model $\mathcal{B}$ and its robustness value $\mathcal{R}_\mathcal{B}$, we can train a more robust target model $\mathcal{T}$ using a penalized loss function $L_\mathcal{P}$ that incorporates $\mathcal{R}_\mathcal{B}$ to ensure $\mathcal{R}_\mathcal{B} \leq \mathcal{R}_\mathcal{T}$ without compromising accuracy. We define $L_\mathcal{P}$ as follows:

$$L_\mathcal{P}(\mathcal{B}, \mathcal{T}, \mathbf{X}, \mathbf{Y}, \mathcal{N}) = L + L \cdot ({}^{\mathcal{R}_\mathcal{B}}/_{\mathcal{R}_\mathcal{T}}) \tag{2}$$

where L denotes the original loss of $\mathcal{T}$ (e.g., MSE), and $\mathcal{R}_\mathcal{T}$ is the robustness value of $\mathcal{T}$. If no baseline is available, we assume $\mathcal{R}_\mathcal{B} = 1$. The formulation balances penalties and rewards: if $\mathcal{R}_\mathcal{T} < \mathcal{R}_\mathcal{B}$, the ratio ${}^{\mathcal{R}_\mathcal{B}}/_{\mathcal{R}_\mathcal{T}}$ exceeds 1, penalizing weaker robustness; otherwise, the ratio decreases, deflating $L_\mathcal{P}$ and promoting stronger robustness.[5]

Convexity of $\mathbf{L}_\mathcal{P}$: If L is convex and non-negative, and $\mathcal{R}_\mathcal{T}$ is strictly positive and affine or concave, then $L_\mathcal{P}$ remains convex; ensuring optimization over $L_\mathcal{P}$ is tractable and stable.[6]

B. Noise-Aware Training

Given input samples $\mathbf{X}$, output samples $\mathbf{Y}$, and a noise model $\mathcal{N}$, we can synthesize an augmented dataset by introducing a new variable per $\mathbf{x}_i \in \mathbf{X}$ such that $\mathbf{X}_\text{aug} = \big[\mathbf{x}_i \quad \mathcal{G}(\mathbf{x}_i)\big]_1^n$. The aim is to enhance the robustness by training $\mathcal{M}$ on $\big[\mathbf{X}_\text{aug} \quad \mathbf{Y}\big]$, but for balanced exposure, the samples are duplicated: the first half uses the actual significant patterns as the new features, while the second half has it with the original $\mathbf{x}_i$. That is, $\mathbf{X}_\text{aug} = \mathbf{X}_\text{aug} \cup \big[\mathbf{x}_i \quad \mathbf{x}_i\big]_1^n$, and similarly $\mathbf{Y}_\text{aug} = \mathbf{Y} \cup \mathbf{Y}$. The final training dataset $[\mathbf{X}_\text{aug}, \mathbf{Y}_\text{aug}]$ includes both original and noise-augmented samples, allowing the model to learn robust mappings. To ensure clean accuracy is preserved, we keep the validation set noise-free. This dual exposure trains models to generalize well even under challenging input conditions. By combining robustness-driven regularization with targeted noise augmentation, this training strategy yields models that are more resilient to real-world variations in input data.

V. Experiments

RARE's code is hosted at `https://github.com/3brahimi/RARE`.

A. Baseline Expressions

We use a set of AI-Feynman equations that covers diverse mathematical functions (e.g., exponential, trigonometric, etc.) [6], [7] and DSR-EQL [8] (enumerated from IV.1 to IV.10) as our benchmark.

Each equation's output serves as ground truth for computing a baseline robustness measure $\mathcal{R}_\mathcal{B}$. To calculate $\mathcal{R}_\mathcal{B}$,

[5]For brevity, if $L = \text{MSE}$, we denote $L_\mathcal{P}$ by $\text{MSE}_\mathcal{P}$.

[6]For details, see Section VIII-E in the appendix of the full manuscript at `https://github.com/3brahimi/RARE/blob/main/full-paper.pdf`.

we first use the equation's symbolic expression itself to estimate the expected true output of the generated data. The input and output data are then used to derive the robustness according to (3), where $\mathcal{M}$ is the equation's symbolic expression. In both testing and training, the goal of the models is to achieve higher robustness ($\mathcal{R} \geq \mathcal{R}_{\mathcal{B}}$) without sacrificing accuracy. The full benchmark table and details on the adjustments made during the evaluation are provided in the supplementary materials Table II.

B. Data Generation

Clean Data: The specifications of AI-Feynman's equation's inputs and outputs are provided by a repository of functions in [6], [7]. For DSR-EQL equations, input data is generated on the domain $\mathbf{x}_i \in [-1, 1]$ as mentioned in [8]. Input variables are quantized into equidistant points within their specified domain, and outputs are computed by evaluating each function over these points. When multiple variables exist, we form a mesh grid of inputs and optionally downsample them to manage data size. This clean dataset is then split into 80% training, 10% validation, and 10% testing.

Noisy Data: We applied additive noise according to Definition 2. For testing, we define six noise models $\mathcal{N}_0$ to $\mathcal{N}_5$ (Section 1) and for robustness training we use one noise model during training $\mathcal{N}_{\text{train}}$ and one during testing $\mathcal{N}_{\text{test}}$ (Section 1). For each noise models, we generate five noise matrices ensuring variability in noise application. The same seeds are used across experiments for consistency, and results are averaged over these five noisy datasets.

C. Weights Estimation

In RARE, the weight matrix $\mathbf{W}$ (see Definition 2) quantifies each input's influence on the output. In practice, we estimate these weights using the Total Order Index (TOI) from the `SALib` library [20]. TOI measures not only the direct impact of each input variable on the output but also their interactions, providing a comprehensive assessment of feature importance. Since TOI values may not sum to one, often due to interaction effects that distribute variance among inputs, we normalize them for comparability. Finally, for noise-aware training (Section IV-B), the weights assigned to newly added variables, representing $\mathcal{G}(\mathbf{x}_i)$, are set to zero, as these do not correspond to original function inputs.

D. Testing Setup

1) Noise Models: For robustness testing, we define various noise models $\mathcal{N}_i$ with $k = 40$ as follows:

$\mathcal{N}$	$\mathcal{D}$	$\mathcal{P}$	θ	
$\mathcal{N}_0$	$\mathtt{G}$	1	$\mu = 0$	$\sigma^2 \in \{n/5\}_1^5$
$\mathcal{N}_1$	$\mathtt{U}$	1	$a = 0$	$b \in \{n/5\}_1^5$
$\mathcal{N}_2$	$\mathcal{L}$	1	$\mu = 0$	$b \in \{n/5\}_1^5$
$\mathcal{N}_3$	$\mathtt{G}$	$\{n/10\}_1^{10}$	$\mu = 0$	$\sigma^2 = 1$
$\mathcal{N}_4$	$\mathtt{U}$	$\{n/10\}_1^{10}$	$a = 0$	$b = 1$
$\mathcal{N}_5$	$\mathcal{L}$	$\{n/10\}_1^{10}$	$\mu = 0$	$b = 1$

Of note, following [21], we also employed the Central Limit Theorem (CLT) to determine $k = 40$ for assessment.

2) Regression Methods: We evaluate the robustness of various methods including LR, RF, MLP, CNN. The RF uses 100 estimators and is trained using MSE loss. The MLP has two hidden layers, while the CNN uses two 1D convolutional layers with $\tanh$ activation. We used Adam optimizer for both with a learning rate of 0.001 and an early stopping based on the validation MSE. The number of neurons and filter sizes vary based on the target function. We also consider the E2E model with the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) optimization algorithm proposed in [9], which leverages the self-attention mechanism to capture long-range dependencies in the input data, making it well-suited for complex function-estimations.

E. Training Setup

1) Noise Models: For robustness training, across all experiments, we use noise model $\mathcal{N}_{\text{train}} = \langle \mathtt{G}, \mathcal{P} = 100\%, k = 20, \theta : \{\mu = 0, \sigma^2 = 0.5\} \rangle$. The noise matrices for robustness evaluation of the trained models, are generated using $\mathcal{N}_{\text{test}} = \langle \mathtt{G}, \mathcal{P} = 100\%, k = 40, \theta : \{\mu = 0, \sigma^2 = 0.3\} \rangle$. Similarly, we used CLT to determine $k = 20$ for robustness enhancement and $k = 40$ for assessing robustness.

2) Regression Methods: We use MLP and CNN with architectures identical to those used in testing. For the DSR models, we use the DSR-EQL framework, which integrates symbolic regression into deep learning using a relaxed L_0 regularization [8]. For the E2E, retraining from scratch is computationally expensive, so we initialize it with a pre-trained model (available in [22]) and apply $\text{MSE}_{\mathcal{P}}$ during the evaluation step, replacing the original R^2 (coefficient of determination) objective. This loss is applied after the model generates symbolic expressions and associated constants, guiding the refinement step (such as constant optimization) to improve the model's performance.

3) Evaluation Metrics: We compare the results of the two types of training, clean training on $[\mathbf{X} \quad \mathbf{Y}]$, and noise-aware training on augmented data $[\mathbf{X}_{\text{aug}} \quad \mathbf{Y}_{\text{aug}}]$. The loss functions used are MSE and $\text{MSE}_{\mathcal{P}}$ defined in Eq. (2). The training procedure is repeated 11 times, and the best model is selected based on the validation MSE and $\mathcal{R}$. Each final model is tested against noise samples drawn from $\mathcal{N}_{\text{test}}$. Reported metrics in Section VI-B include validation MSE, $\mathcal{R}$, and the number of training epochs.

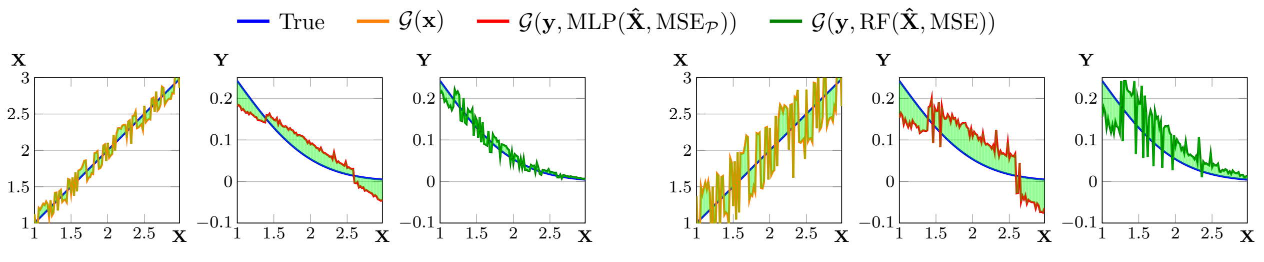
VI. Experimental Evaluation

In this section, we first present the robustness assessment results. Next, we report our experimental evaluation of robustness enhancement training strategies.

A. Robustness Testing Results

1) Extracted Significant patterns: Figure 1 shows examples of the extracted significant patterns for the input variable $\mathbf{x}$ and the output of equation I.6.20a. It shows that as noise increases on the input, the significant patterns $\mathcal{G}(\mathbf{x})$ deviate further from the original values, forming a tube that illustrates the spread of the perturbed values around the input and how this variation propagates to the output $\mathcal{G}(\mathbf{y})$. This spread reflects the model's robustness under varying noise levels. The distance of these patterns from the clean inputs and outputs quantifies robustness measure $\mathcal{R}$ in Definition 3.

2) Models Robustness: Figure 2 compares cumulative $\mathcal{R}$ scores across RF, LR, CNN, MLP, and E2E, for equation



$\mathcal{N}_3(\mathcal{P} = 10\%)$ $\mathcal{N}_3(\mathcal{P} = 30\%)$

Fig. 1: Extracted $\mathcal{G}(\mathbf{x})$ and $\mathcal{G}(\mathbf{y})$ for $\mathcal{N}_3$ with percentages 10% and 30% estimated using MLP and RF for Equation I.6.20a.

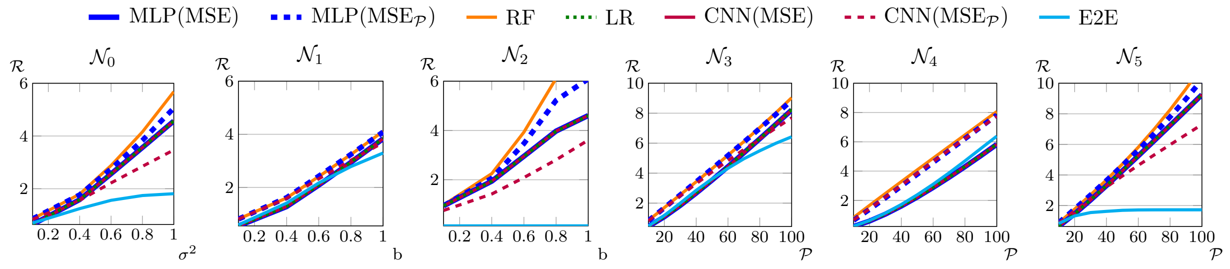


Fig. 2: Cumulative $\mathcal{R}$ for different ML models using different testing noise models

I.6.20a against noise models defined in Section 1. Using MLP(MSE$_{\mathcal{P}}$) and CNN(MSE$_{\mathcal{P}}$), we denote MLP and CNN models trained using MSE$_{\mathcal{P}}$ and MSE as training and validation losses.

Figure 2 shows that RF consistently achieves the highest robustness across noise distributions, because it averages 100 decision trees, reducing variance and smoothing over challenging noise conditions like Laplace ($\mathcal{N}_2$ and $\mathcal{N}_5$). In contrast, gradient-based methods like MLP, CNN, LR, and E2E exhibit higher sensitivity to abrupt or large perturbations, as distributions with sharp or frequent deviations, such as Laplace, can destabilize their training dynamics. Although E2E handles uniform noise well ($\mathcal{N}_1, \mathcal{N}_4$), it underperformed against challenging conditions like Laplace ($\mathcal{N}_5$), underscoring the need for robustness enhancing training.

B. Robustness Training Results

1) MLP and CNN results: Table I reports the validation MSE, robustness $\mathcal{R}$, and training epochs for the trained models using two training regimes, clean $[\,\mathbf{X} \quad \mathbf{Y}\,]$ vs. noise-aware $[\,\mathbf{X}_{\text{aug}} \quad \mathbf{Y}_{\text{aug}}\,]$. Here, the equation is the baseline—i.e., $\mathcal{R}_{\mathcal{B}} = \mathcal{R}_{eq}$)—-and the goal is to compare the robustness of the trained model $\mathcal{R}$ with that of the baseline equation. For each training setup, both MSE and MSE$_{\mathcal{P}}$ were used to be optimized during the training process. The tables included here show results for a selected subset of equations. The full set of results is provided in supplementary materials in Tables III and IV for completeness.

As presented in Table I, for both MLP and CNN architectures, training with MSE$_{\mathcal{P}}$ consistently yields lower validation MSE compared to MSE-based training. In several cases (e.g. equations I.6.20a, I.6.20, I.14.3, and IV.10) MLPs trained with MSE$_{\mathcal{P}}$ show a substantial reduction in validation error—sometimes by an order of magnitude—while maintaining or improving robustness $\mathcal{R}$. Notably, for I.6.20 and I.14.3, $\mathcal{R}$ remains stable, indicating that improvements in accuracy do not come at the expense of robustness. Similarly, CNNs trained with MSE$_{\mathcal{P}}$ display robust performance, with significant improvements in $\mathcal{R}$ under both clean and noise-aware training. For equations I.6.20a and I.12.4, robustness increases are particularly notable, reinforcing the general trend that regularization via MSE$_{\mathcal{P}}$ aids not only in fitting clean data but also in handling noise. Looking at Fig. 2, we can see that training with MSE$_{\mathcal{P}}$ and using $\mathcal{N}_{\text{train}}$ enhances robustness against other types of noise, highlighting the generalizability of this approach.

2) Clean vs. Noise-aware Training: Comparing clean $[\,\mathbf{X} \quad \mathbf{Y}\,]$ and noise-aware $[\,\mathbf{X}_{\text{aug}} \quad \mathbf{Y}_{\text{aug}}\,]$ training in Table I reveals that incorporating noisy examples during training further boosts robustness $\mathcal{R}$. For MLPs, noise-aware training coupled with MSE$_{\mathcal{P}}$ consistently enhances robustness, as seen in I.6.20 and I.12.4, while CNNs exhibit similar improvements in equations like I.12.4 and I.34.8. These results highlight the advantage of early noise exposure, which is consistent with data augmentation strategies in other machine learning domains. However, the improvements in $\mathcal{R}$ from noise-aware training are not uniformly large across all cases. This indicates that clean-data training when combined with MSE$_{\mathcal{P}}$ can achieve significant robustness. The ability of MSE$_{\mathcal{P}}$-based training to generalize effectively—even without noise augmentation—may be attributed to its regularization properties, which likely help the model learn smoother decision boundaries less sensitive to perturbations.

3) Comparison with RF Robustness: RF trained with MSE initially demonstrated superior robustness under most noise models, as shown in Fig. 2, but MLPs trained with MSE$_{\mathcal{P}}$ narrow this gap by achieving comparable $\mathcal{R}$ values in high-noise distributions (e.g., Laplace $\mathcal{N}_2$ and $\mathcal{N}_5$). This highlights the ability of the regularized loss to bridge the robustness gap without relying on ensemble methods. For CNNs, $\mathcal{R}$ declines under MSE$_{\mathcal{P}}$ training compared to RF against some noise models. However, they consistently achieve robustness exceeding the baseline requirement ($\mathcal{R} >$

TABLE I: Results of MLP, CNN, and DSR-EQL models trained by MSE and $\text{MSE}_{\mathcal{P}}$ on $[\,\mathbf{X} \quad \mathbf{Y}\,]$ and $[\,\mathbf{X}_{\text{aug}} \quad \mathbf{Y}_{\text{aug}}\,]$.

			Validation **MSE**				$\mathcal{R}$				**Epochs**			
	Equation Number	$\mathcal{R}_{\text{eq}}$	MSE + $[\,\mathbf{X} \ \mathbf{Y}\,]$	$\text{MSE}_{\mathcal{P}}$ + $[\,\mathbf{X} \ \mathbf{Y}\,]$	MSE + $[\,\mathbf{X}_{\text{aug}} \ \mathbf{Y}_{\text{aug}}\,]$	$\text{MSE}_{\mathcal{P}}$ + $[\,\mathbf{X}_{\text{aug}} \ \mathbf{Y}_{\text{aug}}\,]$	MSE + $[\,\mathbf{X} \ \mathbf{Y}\,]$	$\text{MSE}_{\mathcal{P}}$ + $[\,\mathbf{X} \ \mathbf{Y}\,]$	MSE + $[\,\mathbf{X}_{\text{aug}} \ \mathbf{Y}_{\text{aug}}\,]$	$\text{MSE}_{\mathcal{P}}$ + $[\,\mathbf{X}_{\text{aug}} \ \mathbf{Y}_{\text{aug}}\,]$	MSE + $[\,\mathbf{X} \ \mathbf{Y}\,]$	$\text{MSE}_{\mathcal{P}}$ + $[\,\mathbf{X} \ \mathbf{Y}\,]$	MSE + $[\,\mathbf{X}_{\text{aug}} \ \mathbf{Y}_{\text{aug}}\,]$	$\text{MSE}_{\mathcal{P}}$ + $[\,\mathbf{X}_{\text{aug}} \ \mathbf{Y}_{\text{aug}}\,]$
MLP	I.6.20a	0.78	$5e^{-4}$	$2e^{-5}$	$5e^{-4}$	$6e^{-6}$	0.91	0.85	0.9	0.77	2000	1000	1800	600
	I.6.20	1.00	$7e^{-4}$	$2e^{-5}$	$6e^{-4}$	$9e^{-5}$	1.5	1.5	1.5	1.6	600	215	300	150
	I.12.4	0.0001	$1e^{-2}$	$3e^{-5}$	$2e^{-4}$	$5e^{-5}$	1.4	1.4	2	2.1	400	400	400	360
	I.14.3	1.3	$2e^{-1}$	$8e^{-4}$	$3e^{-1}$	$2e^{-2}$	1.4	1.5	1.4	1.5	1000	1000	1000	690
	IV.10	0.08	$1e^{-3}$	$1e^{-4}$	$6e^{-4}$	$2e^{-5}$	1.15	1.2	1.15	1.19	400	224	1160	587
CNN	I.6.20a	0.78	$4e^{-1}$	$2e^{-6}$	$4e^{-1}$	$3e^{-6}$	0.96	0.8	0.91	0.9	350	230	275	150
	I.12.4	0.0001	$1e^{-4}$	$8e^{-5}$	$1e^{-4}$	$6e^{-5}$	1.83	1.6	2	1.7	773	140	409	120
	I.34.8	0.0001	$2e^{-1}$	$2e^{-1}$	$3e^{-1}$	$2e^{-1}$	1.4	1.3	1.6	1.7	2500	790	1500	515
	I.14.3	1.3	$2e^{-1}$	$1e^{-2}$	$4e^{-1}$	$1e^{-2}$	1.4	1.42	1.47	1.54	2000	780	400	200
	IV.10	0.08	$3e^{-4}$	$2e^{-5}$	$1e^{-4}$	$1e^{-5}$	1.15	1.15	1.6	1.7	1500	700	1300	625
DSR-EQL	IV.1	1.0	$1.88e^{-2}$	$1.85e^{-2}$	—	—	1.5	1.4	—	—	20000	3000	—	—
	IV.2	0.28	$2e^{-4}$	$7e^{-4}$	—	—	0.5	0.54	—	—	20000	20000	—	—
	IV.3	0.28	$4e^{-4}$	$3e^{-5}$	—	—	0.4	2.6	—	—	20000	20000	—	—
	IV.4	0.66	0.398	0.4	—	—	1.3	1.8	—	—	20000	4000	—	—
	IV.5	1.14	$3e^{-4}$	$2.5e^{-4}$	—	—	2	3.4	—	—	20000	20000	—	—
	IV.6	0.67	$6.3e^{-2}$	$6.1e^{-2}$	—	—	1.7	1.3	—	—	20000	10000	—	—
	IV.7	0.54	$3.5e^{-3}$	$3.4e^{-3}$	—	—	4.2	4.3	—	—	20000	20000	—	—
	IV.8	0.58	$5e^{-4}$	$3e^{-4}$	—	—	0.4	0.5	—	—	20000	20000	—	—
	IV.9	0.15	0.37	0.4	—	—	3.4	3.4	—	—	20000	20000	—	—
	IV.10	0.08	$6e^{-2}$	$6e^{-2}$	—	—	5.7	5.9	—	—	20000	3000	—	—

$\mathcal{R}_{eq}$) and maintain significantly lower MSE values compared to both MLPs and CNNs trained with MSE. This makes them highly effective for tasks where accuracy is prioritized, without compromising robustness.

4) DSR Results: Table I reports the results of training DSR-EQL using MSE and $\text{MSE}_{\mathcal{P}}$, showing testing MSE, robustness $\mathcal{R}$, and training epochs. The results demonstrate that training with $\text{MSE}_{\mathcal{P}}$ consistently improves $\mathcal{R}$ compared to training with MSE, without significantly compromising accuracy. For example, in equations IV.3 and IV.5, training with $\text{MSE}_{\mathcal{P}}$ boosts robustness $\mathcal{R}$ from 0.413 to 2.641 and from 2.025 to 3.394, respectively, while maintaining or lowering MSE values. Additionally, these improvements in $\mathcal{R}$ also surpass the robustness of the baseline expressions $\mathcal{R}_{eq}$ (in Table II). The reduced training epochs in cases like IV.1 and IV.10 further suggest that $\text{MSE}_{\mathcal{P}}$ guides optimization more efficiently, enabling faster convergence while leveraging the model's capacity to represent complex symbolic relationships. However, minor trade-offs appear in cases like IV.4, where the MSE slightly increases, from 0.38 to 0.4, while $\mathcal{R}$ improves from 1.3 to 1.8.

5) Training time and Convergence: A key advantage of the proposed $\text{MSE}_{\mathcal{P}}$ is its faster and more stable convergence compared to standard MSE. In many equations (e.g., I.6.20, I.25.3, and I.32.5), the $\text{MSE}_{\mathcal{P}}$-trained models converge in fewer epochs or exhibit more stable convergence than their MSE-only counterparts, as shown in all results in Table I. Figure 3 compares the validation MSE progress across epochs for models trained with MSE vs. $\text{MSE}_{\mathcal{P}}$ as training loss. This figure shows that models trained with $\text{MSE}_{\mathcal{P}}$ consistently converge faster and achieve lower MSE values, reinforcing the efficiency of the proposed loss function.

6) E2E Results: Using $\text{MSE}_{\mathcal{P}}$ as the fitting function to minimize during the refinement stage in E2E transformer-

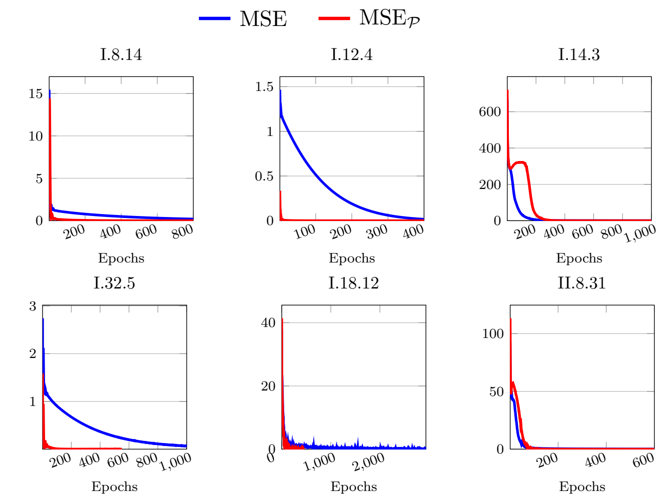


Fig. 3: Validation MSE curves for MLP using the two training loss functions MSE and $\text{MSE}_{\mathcal{P}}$ with $\mathcal{N}_{\text{train}}$

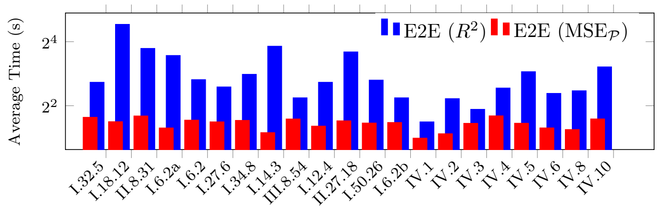


Fig. 4: Average fitting time for E2E using R^2 vs. $\text{MSE}_{\mathcal{P}}$

based model instead of maximizing R^2 maintains the same accuracy and robustness; however, it significantly shortens the refinement stage as shown in Fig. 4. The penalty term in $\text{MSE}_{\mathcal{P}}$ helps smooth the optimization landscape by penalizing deviations between the target and baseline models. This smoothing effect reduces oscillations and fluctuations that can occur when optimizing for R^2, which is sensitive to small prediction errors and can lead to instability or slower convergence (in some cases, fitting the model with R^2 fails to converge entirely). With smoother transitions, $\mathrm{MSE}_{\mathcal{P}}$ improves the efficiency of the BFGS algorithm by reducing the excessive backtracking during line searches. As indicated in Fig. 4, this leads to faster and more stable refinement, (e.g. for equation I.18.12 using $\mathrm{MSE}_{\mathcal{P}}$ reduced the time from 23 seconds to 3 seconds and an average improvement of 58.69% across all equations) making $\mathrm{MSE}_{\mathcal{P}}$ a valuable choice when time efficiency is crucial. The missing equations in this figure (from Table II) highlight cases where $\mathrm{E2E}(R^2)$ did not converge at all.

VII. Conclusion and Future Work

We presented a unified framework for evaluating and improving robustness for regression task in ML and applied it to function estimation. Central to our approach is the concept of significant patterns, which encapsulates input–output deviations under noise. These patterns enable a novel robustness measure that captures input–output sensitivity as a volumetric ratio of input perturbations and output deviations. Through experiments on AI-Feynman and DSR-EQL equations under Gaussian, Uniform, and Laplace noise, we observed substantial variation in robustness across models, with Random Forests being the most robust among baselines. To enhance robustness during training, we proposed a regularized loss that penalizes noise sensitivity wrt. our robustness measure. Models trained with this loss exhibited up to 33% improvement in the robustness score compared to training with MSE alone, reduced validation error, and faster convergence. For example, convergence time for DSR-EQL was reduced from 20,000 to 3,000 epochs in some cases, while the transformer-based E2E model achieved an average of 58.69% reduction in wall-clock fitting time across equations.

In summary, RARE delivers on its promise of a practical framework for evaluating and improving robustness in regression tasks. By leveraging input-output significant patterns, RARE defines a distance-based robustness measure that enables efficient, model-agnostic robustness assessment without requiring gradients or architecture-specific parameters. The proposed regularized loss function effectively penalizes the model's sensitivity to noise, guiding the training towards more robust solutions. Combined with a lightweight noise-aware data augmentation strategy based on significant patterns, RARE enhances model robustness without the need for high data or computational cost.

Experimental evaluations confirm that RARE consistently improves the robustness score against a variety of noise distributions across diverse regression methods, including NNs, and symbolic regressors. Consequently, RARE fulfills its objectives of enabling efficient robustness evaluation, improving operational performance under noise, and accelerating convergence through targeted robustness-aware regularization and augmentation.

Future work will extend the robustness metric to real-world dynamic noise by incorporating time-dependent perturbation models and adaptive noise-aware training. To address privacy, we plan to integrate differential privacy mechanisms directly into the robustness-regularized loss.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR (Poster)*, 2014.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.

[3] N. Carlini, G. Katz, C. Barrett, and D. L. Dill, "Provably minimally-distorted adversarial examples," *arXiv:1903.10484*, 2019.

[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR (Poster)*, 2018.

[5] ISO/IEC JTC 1/SC 7 Committee, "Systems and Software Engineering – Vocabulary," *ISO/IEC/IEEE 24765:2017*, 2017.

[6] M. Tegmark, "AI Feynman: a physics-inspired method for symbolic regression," 2024, [Online; accessed 27-February-2024].

[7] S. Udrescu and M. Tegmark, "AI feynman: a physics-inspired method for symbolic regression," *CoRR*, vol. abs/1905.11481, 2019.

[8] S. Kim, P. Y. Lu, S. Mukherjee, M. Gilbert, L. Jing, V. Ceperic, and M. Soljacic, "Integration of neural network-based symbolic regression in deep learning for scientific discovery," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 9, pp. 4166–4177, 2021.

[9] P. Kamienny, S. d'Ascoli, G. Lample, and F. Charton, "End-to-end symbolic regression with transformers," in *NeurIPS*, 2022.

[10] H. Salman, J. L. Yang, H. Zhang, P. Zhang, C.-J. Hsieh, and A. Madry, "Provably robust deep learning via adversarially trained smoothed classifiers," in *NeurIPS*, vol. 32, 2019.

[11] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2017, pp. 39–57.

[12] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *NeurIPS*, 2019, pp. 125–136.

[13] Z. Shi, H. Zhang, K. Chang, M. Huang, and C. Hsieh, "Robustness verification for transformers," in *ICLR*, 2020.

[14] M. Hein and M. Andriushchenko, "Formal guarantees on the robustness of a classifier against adversarial manipulation," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[15] M.-M. Zühlke and D. Kudenko, "Adversarial robustness of neural networks from the perspective of lipschitz calculus: A survey," *ACM Computing Surveys*, vol. 57, no. 6, pp. 1–41, 2025.

[16] J. T. Wassell, "Sensitivity analysis in practice, andrea saltelli," *Technometrics*, vol. 47, no. 2, p. 236, 2005.

[17] J. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 1310–1320.

[18] L. L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*. Addison-Wesley, 1991.

[19] A. V. Oppenheim, *Discrete-time signal processing*. Pearson Education India, 1999.

[20] J. Herman and W. Usher, "SALib: An open-source python library for sensitivity analysis," *The Journal of Open Source Software*, vol. 2, no. 9, 2017.

[21] M. Ebrahimi, Q. Alfalouji, and M. Basirat, "Validity and robustness of denoisers: A proof of concept in speech denoising," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 33, pp. 650–665, 2025.

[22] P.-A. Kamienny, S. d'Ascoli, G. Lample, and F. Charton, "End-to-end symbolic regression with transformers (github repository)," , 2022, accessed: February 10, 2025.

[23] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[24] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, 2009.

[25] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, 1997.

Supplementary Material

RARE: Robustness Assessment and Regularized Enhancement
A Study in Function Estimation and Symbolic Regression

This document contains supplementary material accompanying the paper *Robustness Assessment and Regularized Enhancement (RARE): A study in Function Estimation and Symbolic Regression*. It provides:

- Additional methodology and proofs supporting the robustness assessment methodology;
- Formal definitions and derivations for approximations, multi-output generalizations, and extending the robustness measure to a robustness metric;
- An extended analysis of the loss regularization and its convexity;
- Full experimental setup, including full tables for baseline equations and complete and detailed results.

All definitions and results herein are referenced to sections in the main text for continuity.

VIII. Additional details for Robustness Testing

Section III introduced the methodology to assess the robustness of function estimators under perturbations, focusing on the computation and evaluation of individual output robustness. This section extends the original framework by providing an example of sensitivity to input perturbations in weight computation and introducing key refinements, such as an approximation to reduce computational cost, a generalization to handle multiple outputs, and a robustness metric. By introducing these refinements, we enable efficient, scalable, and comparative robustness evaluation across models and tasks.

A. Sensitivity to Input Perturbations

The effect of perturbations in $\mathbf{x}_i$ on $\mathbf{y}_j$ may vary; that is, there exist inputs $\mathbf{x}_i$ whose approximate weight is low, yet when perturbed they can cause larger-than-expected variations in $\mathbf{y}_j$. To address this, we incorporate the term $\ell_p\big(\mathbf{x}_i, \mathcal{G}(\mathbf{x}_i)\big)$ to measure the deviation of a representative perturbed input $\mathcal{G}(\mathbf{x}_i)$ from its baseline $\mathbf{x}_i$. This ℓ_p-norm serves as a sensitivity metric, capturing how variations in $\mathbf{x}_i$ influence $\mathbf{y}_j$.

Example.*Suppose we wish to estimate a function $f\colon\ \mathbf{X} \to \mathbf{Y}$ from an input sample set $\mathbf{X}$ and an output sample set $\mathbf{Y}$, where $\mathbf{x} \in \mathbf{X}$ is the only input variable and $\mathbf{y} \in \mathbf{Y}$ is the only output variable. For instance, consider the function*

$$f(\mathbf{x}) = \begin{cases} \mathbf{x}^2, & \textit{if } \mathbf{x} < 0, \\ 2\,\mathbf{x}, & \textit{if } \mathbf{x} \geq 0. \end{cases}$$

Assume that the input in $\mathbf{X}$ are predominantly sampled such that $p_{\mathbf{x}}(\mathbf{x} < 0)$ is high and $p_{\mathbf{x}}(\mathbf{x} \approx 0)$ is low. Although the weight assigned to values near $\mathbf{x} = 0$ may be low due to their rarity, this region is critical as f undergoes a sharp transition at $\mathbf{x} = 0$, amplifying the effect of small perturbations. Even small perturbations around $\mathbf{x} = 0$ can lead to significant changes in the output $f(\mathbf{x})$, shifting the behavior from quadratic to linear. The sensitivity metric $\ell_p\big(\mathbf{x}, \mathcal{G}(\mathbf{x})\big)$ quantifies how perturbations deviate from their baseline, highlighting regions where even small changes significantly impact $f(\mathbf{x})$.

B. Approximating the Robustness Measure

In Section III, we defined $\mathcal{R}_j$ as a robustness measure for individual outputs. However, its computation requires $k \times l$ calls to $\mathcal{M}$ to derive $\mathcal{G}(\mathbf{y}_j)$, where k is the number of noise samples and l is the dataset size of $\mathbf{X}$ and $\mathbf{Y}$.

To mitigate this, we introduce an approximate robustness measure $\acute{\mathcal{R}}_j$, which significantly reduces computational cost while maintaining accuracy, as demonstrated in [21] for a more complex real-world scenario. The approximation is valid under the assumption that the distance between the output $\mathbf{y}_j$ and the model's prediction on the perturbed input $\mathcal{M} \circ \mathcal{G}(\mathbf{x}_i)$ closely approximates the distance to the worst-case output perturbation $\mathcal{G}(\mathbf{y}_j)$. This reduces the required number of model evaluations to a single evaluation per output variable. More formally, if $\Delta(\mathbf{y}_j, \mathcal{M} \circ \mathcal{G}(\mathbf{x}_i)) \approx \Delta(\mathbf{y}_j, \mathcal{G}(\mathbf{y}_j))$, then $\acute{\mathcal{R}}_j$ is defined as:

$$\acute{\mathcal{R}}_j(\mathbf{X}, \mathbf{y}_j, \mathcal{N}, \mathcal{M}) = \frac{\sum_{i=1}^{n} w_{ji} \cdot \Delta(\mathbf{x}_i, \mathcal{G}(\mathbf{x}_i))}{\Delta(\mathbf{y}_j, \mathcal{M} \circ \mathcal{G}(\mathbf{x}_i))}\ . \tag{3}$$

C. Generalizing to Multiple Outputs

Many real-world systems involve multiple correlated output variables. To extend our robustness measure to such systems, we introduce a generalized robustness measure that aggregates the individual robustness scores across all m outputs using the ℓ_p norm. This formulation provides a structured assessment of overall model robustness across multiple outputs.

Definition 1 (Robustness of Multi-Variable Output). Given $\mathbf{X} \in \mathbb{R}^{n \times l}$, $\mathbf{Y} \in \mathbb{R}^{m \times l}$, a noise model $\mathcal{N}$, and a model $\mathcal{M}$, the overall robustness measure $\mathcal{R}$ is defined as the ℓ_p norm of the individual robustness scores $\mathcal{R}_j$ for all $\mathbf{y}_j \in \mathbf{Y}$:

$$\mathcal{R}(\mathbf{X}, \mathbf{Y}, \mathcal{N}, \mathcal{M}) = \left(\sum_{j=1}^{m} |\mathcal{R}_j(\mathbf{X}, \mathbf{y}_j, \mathcal{N}, \mathcal{M})|^p \right)^{\frac{1}{p}}\ . \tag{4}$$

D. Robustness Metric

To facilitate systematic comparisons between different models, we define a robustness metric that quantifies the difference in robustness between two estimators $\mathcal{M}_1$ and $\mathcal{M}_2$.

Definition 2 (Robustness Metric). Given two estimators $\mathcal{M}_1$ and $\mathcal{M}_2$, a noise model $\mathcal{N}$, and clean inputs $\mathbf{X}$, the robustness metric $\nabla(\mathcal{M}_1, \mathcal{M}_2)$ is defined as:

$$\nabla(\mathcal{M}_1, \mathcal{M}_2) = \Delta\left(\mathcal{R}(\mathcal{M}_1, \mathcal{N}), \mathcal{R}(\mathcal{M}_2, \mathcal{N})\right),$$

where $\mathcal{R}(\mathcal{M}, \mathcal{N})$ quantifies the robustness of $\mathcal{M}$ under $\mathcal{N}$. The function Δ is a valid distance measure, ensuring that ∇ inherits metric properties.

This metric enables a direct and structured comparison of the robustness of different models under identical noise conditions.

E. Convexity of Regularized Loss

We define the penalized loss function as

$$L_{\mathcal{P}}(\mathcal{B}, \mathcal{T}, \mathbf{X}, \mathbf{Y}, \mathcal{N}) = L(\theta) + L(\theta) \cdot \frac{\mathcal{R}_{\mathcal{B}}}{\mathcal{R}_{\mathcal{T}}(\theta)}, \tag{5}$$

where

- $L(\theta)$ is the original loss (e.g., MSE) of the target model $\mathcal{T}$, parameterized by θ,
- $\mathcal{R}_{\mathcal{B}}$ is the robustness metric of a baseline model (a constant), and
- $\mathcal{R}_{\mathcal{T}}(\theta)$ is the robustness metric of the target model $\mathcal{T}$, assumed to be strictly positive for all θ.

To establish the convexity of $L_{\mathcal{P}}$, we require that both $L(\theta)$ and the penalty term

$$L(\theta) \cdot \frac{\mathcal{R}_{\mathcal{B}}}{\mathcal{R}_{\mathcal{T}}(\theta)}$$

are convex. Since the specific forms of $L(\theta)$ and $\mathcal{R}_{\mathcal{T}}(\theta)$ depend on the application, we assume:

A1. The loss function $L(\theta)$ is convex and nonnegative.

A2. The robustness metric $\mathcal{R}_{\mathcal{T}}(\theta)$ is strictly positive and is chosen such that its reciprocal,

$$\frac{1}{\mathcal{R}_{\mathcal{T}}(\theta)},$$

is convex. For instance, if $\mathcal{R}_{\mathcal{T}}(\theta)$ is affine or concave over its domain, then its reciprocal is convex on $\{\theta : \mathcal{R}_{\mathcal{T}}(\theta) > 0\}$.

a) Admissibility of A1 and A2.

Many standard loss functions satisfy **A1** [23], [24]. Moreover, the robustness measure $\mathcal{R}$ defined in (1) is nonnegative when the distance function Δ is appropriately chosen. Regarding **A2**, the behavior of $\mathcal{R}_{\mathcal{T}}(\theta)$ during training is as follows: early in training, when model accuracy is low, the denominator of $\mathcal{R}_{\mathcal{T}}(\theta)$ in (1) is large relative to the numerator, so $\mathcal{R}_{\mathcal{T}}(\theta)$ is low. As training proceeds and accuracy improves, the denominator decreases, causing $\mathcal{R}_{\mathcal{T}}(\theta)$ to increase (exhibiting concave up behavior); later, as the increase slows and $\mathcal{R}_{\mathcal{T}}(\theta)$ levels off, it may begin to decrease due to a trade-off between accuracy and generalization towards perturbations (exhibiting concave down behavior) until the next leveling phase. This pattern continues until convergence.

Given these admissible assumptions, we now demonstrate the convexity of $L_{\mathcal{P}}$:

1. **Convexity of $L(\theta)$:** By **A1**, $L(\theta)$ is convex in θ.
2. **Convexity of the Penalty Term:** Suppose
$$Q(\theta) \triangleq \frac{L(\theta)}{\mathcal{R}_{\mathcal{T}}(\theta)}.$$
Since $L(\theta)$ is convex and nonnegative and **A2** ensures that $1/\mathcal{R}_{\mathcal{T}}(\theta)$ is convex, standard results from convex analysis (or an argument using the perspective function [23], [25]) imply that $Q(\theta)$ is convex. Multiplying by the positive constant $\mathcal{R}_{\mathcal{B}}$ preserves convexity, so the penalty term $L(\theta) \cdot (\mathcal{R}_{\mathcal{B}}/\mathcal{R}_{\mathcal{T}}(\theta))$ is convex.
3. **Sum of Convex Functions:** Since $L_{\mathcal{P}}(\theta)$ is the sum of the convex functions, it follows that $L_{\mathcal{P}}(\theta)$ is convex.

For further reference on convexity in optimization problems, see [23].

IX. Experiment

A. Baseline Equations and Adjustments

This section presents the full list of the benchmark equations used in the experiments. Table II contains equations from AI-Feynman and DSR-EQL selected to cover a variety of mathematical functions, including exponential, trigonometric, and polynomial forms.

During initial experiments, we observed high error values when training the DSR-EQL model on some of its benchmark equations, particularly the equation x^3, which had a low success rate reported in [8]. To address this, we replaced x^3 with the equation xyz in Table II, which provided more reliable and robust evaluations. This adjustment ensures a comprehensive evaluation of both accuracy and robustness across diverse functional forms. For ease of reference, we have numbered the equations in DSR-EQL benchmark from IV.1 to IV.10. The tables below provide the full details of the equations used in our experiments.

B. Results

The tables in this section provide the complete training results for both CNN and MLP as target models $\mathcal{T}$, evaluated using the baseline equations listed in Table II. The results compare models trained using MSE and the proposed regularized loss function $\text{MSE}_{\mathcal{P}}$ under clean and noise-aware training setups. Key metrics include the validation MSE, robustness measure $\mathcal{R}$, and the training time in terms of the number of epochs. These detailed tables highlight the improvements in robustness and efficiency achieved through robustness-aware optimization, complementing the main text discussion in Section VI-B.

C. Trends Across Equations Groups

To interpret the performance trends (from Tables III and IV) more clearly, we group the benchmark equations (in Table II) by their mathematical structure and average the results within each group. This summarizes how different types of equations respond to robustness regularization and noise-aware training. The six resulting groups are defined as follows:

- **Exponential:** Equations containing exponential terms, such as Gaussian or logistic functions.
- **Trigonometric:** Expressions involving sine or cosine.
- **Linear / Polynomial:** Simple additive or multiplicative polynomial expressions, often low-degree and structurally separable.
- **Rational Expressions:** Equations with rational forms, including divisions and inverse relationships, often associated with physical rates or potentials.
- **Geometric Expressions:** Geometric expressions involving norms or distance functions, typically combining multiple dimensions through squared sums or roots.
- **Physics-derived:** expressions derived from physical laws, such as electric field energy or density, involving domain-specific constants.

TABLE II: Benchmarking equations from AI-Feynman [6], [7] and DSR-EQL [8].

	Equation Number	Formula	$\mathcal{R}_{\mathbf{eq}}$
AI-Feynman	I.6.20a	$y = \exp\left(\theta^2/2\right)/\sqrt{2\pi}$	0.78
	I.6.20	$y = \exp\left(\theta^2/2\sigma^2\right)/\sqrt{2\pi\sigma}$	1.0
	I.6.20b	$y = \exp\left((\theta-\theta_1)^2/2\sigma^2\right)/\sqrt{2\pi\sigma}$	0.42
	I.8.14	$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$	0.9
	I.12.2	$F = (q_1 q_2)\,/\,\left(4\pi\epsilon r^2\right)$	$2e^{-6}$
	I.12.4	$E = q_1/\,\left(4\pi\epsilon r^2\right)$	0.0001
	I.14.3	$U = mgz$	1.3
	I.18.12	$\tau = rF\sin(\theta)$	1
	I.25.3	$V = q/C$	0.05
	I.27.6	$f = (d_1 d_2)\,/\,(d_2 + nd_1)$	0.01
	I.32.5	$P = \left(q^2 a^2\right)/\left(6\pi\epsilon C^3\right)$	0.001
	I.34.8	$w = (qvB)\,/p$	0.0001
	I.34.27	$E = h\omega$	1.2
	I.50.26	$x = \mathbf{x}_1\left[\cos(\omega t) + \alpha\cos(\omega t)^2\right]$	1.06
	II.8.7	$E = \left(3q^2\right)/\,(20\pi\epsilon d)$	0.016
	II.8.31	$E_{\text{den}} = \left(\epsilon E_f^2\right)/2$	1.07
	II.27.18	$E_{\text{den}} = \epsilon E_f^2$	1.1
	III.8.54	$p_\gamma = \sin\left(Et/h\right)^2$	1
DSR-EQL	IV.1	x	1
	IV.2	x^2	0.28
	IV.3	xyz	0.28
	IV.4	$\sin(2\pi x)$	0.66
	IV.5	xy	1.14
	IV.6	$1/\left(1 + \exp^{(-10x)}\right)$	0.67
	IV.7	$(xy/2) + (z/2)$	0.54
	IV.8	$\exp^{(-x^2)}$	0.58
	IV.9	$x^2 + \sin(2\pi y)$	0.15
	IV.10	$x^2 + y - 2z$	0.08

Table V lists the full assignment of equations to these groups, and the plots in Fig. 5 show how the average trend of the validation (MSE), robustness ($\mathcal{R}$), and training convergence (epochs) vary across training setups for each equation group and model type.

Figure 5 reveals several observations:

- **Error reduction:** All groups benefit from $\text{MSE}_{\mathcal{P}}$ compared to standard MSE, with further improvements when combined with data augmentation. The Exponential and Rational groups exhibit the most significant gains.
- **Improved robustness:** Results show that $\mathcal{R}$ increases when using $\text{MSE}_{\mathcal{P}}$, particularly when combined with noise-aware training. This is most pronounced in Trigonometric and Exponential equations, which tend to be more sensitive to input perturbations.
- **Faster convergence:** The number of training epochs is consistently reduced when using $\text{MSE}_{\mathcal{P}}$ and augmentation. This effect is strongest in the Physics-derived and Rational groups, where training dynamics tend to be less stable under noise.

These results confirm that robustness-aware training not only enhances model robustness but also improves generalization and convergence speed across structurally diverse equations. The consistency of these benefits highlights the general applicability of the proposed RARE framework.

TABLE III: Training results for baseline expressions with $\mathcal{B} =$ Eq. and $\mathcal{T} =$ MLP, estimating equations listed in Table II.

			Validation MSE				$\mathcal{R}$				Epochs			
	Equation Number	$\mathcal{R}_{\text{eq}}$	$\text{MSE}+[\,\mathbf{X}\;\;\mathbf{Y}\,]$	$\text{MSE}_{\mathcal{P}}+[\,\mathbf{X}\;\;\mathbf{Y}\,]$	$\text{MSE}+[\,\mathbf{X}_{\text{aug}}\;\;\mathbf{Y}_{\text{aug}}\,]$	$\text{MSE}_{\mathcal{P}}+[\,\mathbf{X}_{\text{aug}}\;\;\mathbf{Y}_{\text{aug}}\,]$	$\text{MSE}+[\,\mathbf{X}\;\;\mathbf{Y}\,]$	$\text{MSE}_{\mathcal{P}}+[\,\mathbf{X}\;\;\mathbf{Y}\,]$	$\text{MSE}+[\,\mathbf{X}_{\text{aug}}\;\;\mathbf{Y}_{\text{aug}}\,]$	$\text{MSE}_{\mathcal{P}}+[\,\mathbf{X}_{\text{aug}}\;\;\mathbf{Y}_{\text{aug}}\,]$	$\text{MSE}+[\,\mathbf{X}\;\;\mathbf{Y}\,]$	$\text{MSE}_{\mathcal{P}}+[\,\mathbf{X}\;\;\mathbf{Y}\,]$	$\text{MSE}+[\,\mathbf{X}_{\text{aug}}\;\;\mathbf{Y}_{\text{aug}}\,]$	$\text{MSE}_{\mathcal{P}}+[\,\mathbf{X}_{\text{aug}}\;\;\mathbf{Y}_{\text{aug}}\,]$
AI Feynmann	I.6.20a	0.78	$5e^{-4}$	$2e^{-5}$	$5e^{-4}$	$6.8e^{-6}$	0.91	0.85	0.9	0.77	2000	1000	1800	600
	I.6.20	1.0	$7e^{-4}$	$2.2e^{-5}$	$6e^{-4}$	$9.8e^{-5}$	1.5	1.5	1.5	1.6	600	215	300	150
	I.6.20b	0.42	$9e^{-4}$	$9.4e^{-5}$	$7e^{-4}$	$1e^{-4}$	1.3	1.1	1.3	1.3	1000	450	500	350
	I.8.14	0.9	$1e^{-1}$	$5e^{-4}$	$1e^{-1}$	$6e^{-4}$	1.2	1.2	1.4	1.4	800	490	400	400
	I.12.2	$2e^{-6}$	$9e^{-2}$	$1e^{-4}$	$3e^{-3}$	$5e^{-4}$	1.27	1.12	1.47	1.5	700	470	700	330
	I.12.4	0.001	$1e^{-2}$	$3.2e^{-5}$	$2e^{-4}$	$5.3e^{-5}$	1.4	1.4	2	2.1	400	400	400	360
	I.14.3	1.3	$2e^{-1}$	$8e^{-3}$	$3e^{-1}$	$2e^{-2}$	1.4	1.4	1.5	1.5	1000	1000	1000	690
	I.18.12	1.0	$2e^{-1}$	$4e^{-2}$	$1e^{-1}$	$4e^{-2}$	1.1	1.1	1.1	1.2	2000	1300	1700	800
	I.25.3	$5e^{-2}$	$1e^{-2}$	$7e^{-4}$	$4e^{-3}$	$5e^{-4}$	1.0	1.0	1.0	1.0	300	200	300	200
	I.27.6	$1e^{-2}$	$1e^{-2}$	$1e^{-4}$	$3e^{-3}$	$4e^{-4}$	1.3	1.3	1.4	1.5	400	230	300	130
	I.32.5	0.001	$4e^{-2}$	$2e^{-3}$	$1e^{-2}$	$4e^{-3}$	0.9	1.05	1.2	1.4	1000	320	1000	200
	I.34.8	0.3	$4e^{-1}$	$4e^{-3}$	$2e^{-1}$	$8e^{-3}$	1.4	1.8	1.6	2.1	1500	1000	1500	750
	I.34.27	1.2	$9e^{-2}$	$1e^{-3}$	$6e^{-2}$	$1e^{-4}$	1.4	1.36	1.4	1.4	700	540	700	400
	I.50.26	1.06	$1.5e^{-2}$	$4e^{-2}$	$4e^{-2}$	$1e^{-2}$	1.3	1.5	1.5	1.8	3200	2900	4000	3300
	II.8.7	0.016	$4e^{-3}$	$2e^{-4}$	$6e^{-4}$	$3e^{-4}$	1.1	1.1	1.2	1.2	500	330	500	170
	II.8.31	1.07	$1e^{-1}$	$1e^{-3}$	$4e^{-2}$	$1e^{-3}$	1.3	1.3	1.3	1.33	600	600	300	120
	II.27.18	1.1	$2e^{-1}$	$2e^{-3}$	$2e^{-1}$	$2e^{-3}$	1.2	1.3	1.3	1.33	800	690	800	580
	III.8.54	1.0	$1e^{-1}$	$1e^{-1}$	$1e^{-1}$	$1e^{-1}$	2.7	2.8	2.6	2.8	2400	1100	1600	1500
DSL-EQL	IV.1	1.0	$5e^{-5}$	$1e^{-5}$	$2e^{-5}$	$4e^{-6}$	1.0	1.03	1.0	1.0	1060	420	1500	290
	IV.2	0.28	$1e^{-3}$	$7e^{-5}$	$1e^{-4}$	$9e^{-6}$	0.42	0.47	0.38	0.4	1055	300	3500	828
	IV.3	0.28	$1e^{-3}$	$2e^{-4}$	$6e^{-4}$	$3e^{-6}$	0.7	0.8	0.6	0.7	800	200	1100	200
	IV.4	0.66	$7e^{-2}$	$2e^{-2}$	$1e^{-3}$	$2e^{-4}$	0.45	0.47	0.3	0.38	640	360	4000	1190
	IV.5	1.14	$5e^{-4}$	$5e^{-5}$	$1e^{-4}$	$1e^{-5}$	1.3	1.4	1.2	1.29	540	182	2200	500
	IV.6	0.67	$2e^{-4}$	$2e^{-5}$	$1e^{-4}$	$2e^{-6}$	0.56	0.56	0.56	0.56	1900	304	2800	650
	IV.7	0.54	$7e^{-4}$	$5e^{-5}$	$2e^{-4}$	$1e^{-5}$	1.0	1.14	1.07	1.05	470	200	1223	560
	IV.8	0.58	$9e^{-4}$	$4e^{-4}$	$1e^{-4}$	$3e^{-5}$	0.86	0.9	0.81	0.83	1130	370	2800	870
	IV.9	0.15	$4e^{-3}$	$3e^{-4}$	$1e^{-3}$	$3e^{-5}$	0.44	0.42	0.4	0.53	660	340	1700	460
	IV.10	0.08	$1e^{-3}$	$1e^{-4}$	$6e^{-4}$	$2e^{-5}$	1.15	1.2	1.15	1.19	400	224	1160	587

TABLE IV: Training results for baseline expressions with $\mathcal{B} = $ Eq. and $\mathcal{T} = $ CNN, estimating equations listed in Table II.

			Validation MSE				$\mathcal{R}_{\mathcal{T}}$				Epochs			
	Equation Number	$\mathcal{R}_{\text{eq}}$	$\text{MSE}+[\mathbf{X}\ \mathbf{Y}]$	$\text{MSE}_{\mathcal{P}}+[\mathbf{X}\ \mathbf{Y}]$	$\text{MSE}+[\mathbf{X}_{\text{aug}}\ \mathbf{Y}_{\text{aug}}]$	$\text{MSE}_{\mathcal{P}}+[\mathbf{X}_{\text{aug}}\ \mathbf{Y}_{\text{aug}}]$	$\text{MSE}+[\mathbf{X}\ \mathbf{Y}]$	$\text{MSE}_{\mathcal{P}}+[\mathbf{X}\ \mathbf{Y}]$	$\text{MSE}+[\mathbf{X}_{\text{aug}}\ \mathbf{Y}_{\text{aug}}]$	$\text{MSE}_{\mathcal{P}}+[\mathbf{X}_{\text{aug}}\ \mathbf{Y}_{\text{aug}}]$	$\text{MSE}+[\mathbf{X}\ \mathbf{Y}]$	$\text{MSE}_{\mathcal{P}}+[\mathbf{X}\ \mathbf{Y}]$	$\text{MSE}+[\mathbf{X}_{\text{aug}}\ \mathbf{Y}_{\text{aug}}]$	$\text{MSE}_{\mathcal{P}}+[\mathbf{X}_{\text{aug}}\ \mathbf{Y}_{\text{aug}}]$
AI Feynmann	I.6.2a	0.78	$4e^{-1}$	$2.4e^{-6}$	$4e^{-1}$	$3.5e^{-6}$	0.96	0.8	0.9	0.9	350	230	275	150
	I.6.2	1.0	$6e^{-4}$	$2e^{-5}$	$6e^{-4}$	$8e^{-5}$	1.45	1.37	1.54	1.46	487	227	280	152
	I.6.2b	0.42	$7e^{-4}$	$2e^{-4}$	$7e^{-4}$	$1e^{-4}$	1.3	1.2	1.35	1.3	1000	390	757	273
	I.8.14	0.9	$4e^{-2}$	$1e^{-3}$	$1e^{-2}$	$2e^{-2}$	1.2	1.36	1.17	1.17	1000	130	950	60
	I.12.2	$2e^{-6}$	$1e^{-2}$	$8e^{-5}$	$1e^{-3}$	$2e^{-4}$	1.42	1.5	1.37	1.37	1000	240	855	130
	I.12.4	0.001	$1e^{-4}$	$8e^{-5}$	$1e^{-4}$	$6.7e^{-5}$	1.83	1.6	2.0	1.7	773	140	409	120
	I.14.3	1.3	$2e^{-1}$	$1e^{-2}$	$4e^{-1}$	$1e^{-2}$	1.4	1.42	1.47	1.54	2000	780	400	200
	I.18.12	1.0	$2e^{-1}$	$1e^{-1}$	$2e^{-1}$	$2e^{-1}$	0.41	0.42	0.5	0.5	4000	3600	4000	3000
	I.25.3	0.05	$8e^{-3}$	$8e^{-4}$	$2e^{-3}$	$2e^{-4}$	1.08	1.11	1.04	1.06	300	180	300	150
	I.27.6	0.01	$1e^{-3}$	$1e^{-4}$	$1e^{-3}$	$2e^{-4}$	1.2	1.3	1.28	1.28	950	163	612	120
	I.32.5	0.001	$3e^{-2}$	$7e^{-4}$	$1e^{-2}$	$2e^{-3}$	0.8	0.9	0.95	1.0	1000	782	1000	235
	I.34.8	0.03	$2e^{-1}$	$1e^{-2}$	$2e^{-1}$	$9e^{-2}$	1.4	1.3	1.6	1.7	2500	790	1500	515
	I.34.27	1.2	$3e^{-2}$	$6e^{-4}$	$2e^{-2}$	$8e^{-4}$	1.35	1.35	1.35	1.4	1000	475	1000	317
	I.50.26	1.06	$6e^{-2}$	$6e^{-2}$	$1e^{-1}$	$4e^{-2}$	0.99	1.65	1.35	1.55	4300	1370	1300	570
	II.8.7	0.016	$4e^{-3}$	$7.9e^{-5}$	$5e^{-4}$	$1e^{-4}$	1.2	1.09	1.06	1.05	500	270	500	180
	II.8.31	1.07	$1e^{-1}$	$2e^{-3}$	$1e^{-2}$	$2e^{-3}$	1.28	1.28	1.29	1.32	600	515	600	120
	II.27.18	1.1	$2e^{-1}$	$2e^{-3}$	$4e^{-2}$	$2e^{-3}$	1.27	1.28	1.31	1.32	800	500	800	224
	III.8.54	1.0	$1e^{-1}$	$1e^{-1}$	$1e^{-1}$	$1e^{-1}$	2.7	1.7	2.5	1.7	730	470	540	720
DSL-EQL	IV.1	1.0	$4e^{-5}$	$2e^{-6}$	$1.5e^{-5}$	$2e^{-6}$	1.0	1.03	1.0	1.0	760	130	1800	240
	IV.2	0.28	$2e^{-3}$	$2e^{-4}$	$1e^{-4}$	$1e^{-5}$	0.38	0.46	0.38	0.4	820	260	3200	700
	IV.3	0.28	$1e^{-3}$	$5e^{-4}$	$6e^{-4}$	$2e^{-4}$	0.7	0.74	0.6	0.7	800	100	1200	600
	IV.4	0.66	$6e^{-2}$	$6e^{-2}$	$1e^{-3}$	$7e^{-4}$	0.47	1.0	0.25	0.36	540	200	3200	1050
	IV.5	1.14	$4e^{-4}$	$6e^{-5}$	$1e^{-4}$	$1e^{-5}$	1.3	1.4	1.27	1.3	490	150	1600	500
	IV.6	0.67	$6e^{-4}$	$7e^{-5}$	$9e^{-5}$	$2e^{-6}$	0.56	0.56	0.56	0.56	1100	320	3000	430
	IV.7	0.54	$4e^{-4}$	$2e^{-4}$	$9e^{-5}$	$1e^{-5}$	1.07	1.18	1.0	1.0	470	120	1500	540
	IV.8	0.58	$4e^{-3}$	$2e^{-4}$	$1e^{-4}$	$2e^{-5}$	0.86	0.9	0.8	0.82	700	270	2300	820
	IV.9	0.15	$4e^{-3}$	$8e^{-5}$	$1e^{-3}$	$2e^{-5}$	0.41	0.44	0.5	0.5	570	800	1600	500
	IV.10	0.08	$3e^{-4}$	$2e^{-5}$	$1e^{-4}$	$1.7e^{-5}$	1.15	1.15	1.15	1.2	1500	700	1300	625

TABLE V: Benchmark equations (in Table II) groupings used for aggregated analysis.

Group	**Equations**
Exponential	I.6.20a, I.6.20, I.6.20b, IV.6, IV.8
Trigonometric	I.18.12, I.50.26, III.8.54, IV.4, IV.9
Linear	IV.1, IV.2, IV.3, IV.5, IV.7, IV.10, I.14.3, I.34.27
Rational	I.12.2, I.12.4, I.25.3, I.27.6, I.32.5, I.34.8, II.8.7
Geometric	I.8.14
Physics-derived	II.8.31, II.27.18

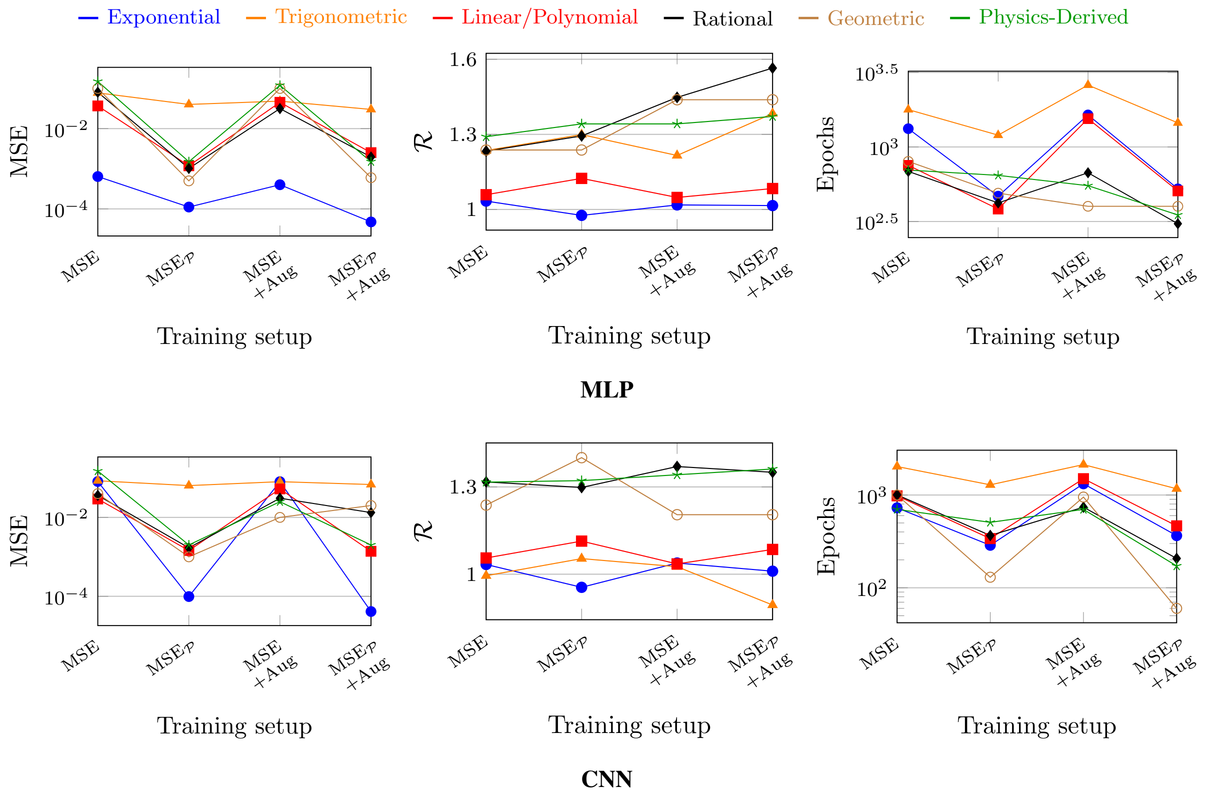


Fig. 5: Average validation MSE, $\mathcal{R}$, and convergence epochs over equations in defined groups in Table V across all training setups: $[\,\mathbf{X} \quad \mathbf{Y}\,]$ with MSE (MSE), $[\,\mathbf{X} \quad \mathbf{Y}\,]$ with MSE$_\mathcal{P}$ (MSE$_\mathcal{P}$), noise-aware $[\,\mathbf{X}_{\text{aug}} \quad \mathbf{Y}_{\text{aug}}\,]$ with MSE (MSE + Aug), and $[\,\mathbf{X}_{\text{aug}} \quad \mathbf{Y}_{\text{aug}}\,]$ with MSE$_\mathcal{P}$ (MSE$_\mathcal{P}$ + Aug) for MLP (top) and CNN (bottom) models.