RID-Noise: Towards Robust Inverse Design under Noisy Environments Supplementary Material

Jia-Qi Yang[1], Ke-Bin Fan[2], Hao Ma[2], and De-Chuan Zhan[1]

[1]State Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing, China
{yangjq,zhandc}@lamda.nju.edu.cn
[2]Research Institute of Superconductor Electronics (RISE),
School of Electronic Science and Engineering,
Nanjing University, Nanjing, China
kebin.fan@nju.edu.cn, mf20230071@smail.nju.edu.cn

December 6, 2021

Proof of Property 1 and Property 2

First, we review the definitions for convenience. Given a target $\mathbf{y}_{\textbf{target}}$, the robustness objective of a parameter $\mathbf{x}'$ can be defined by the expected loss

$$\mathscr{L}_{p(\mathbf{y}|\mathbf{x})}(\mathbf{x}', \mathbf{y}_{\textbf{target}}) = \mathbb{E}_{\mathbf{y}' \sim p(\mathbf{y}|\mathbf{x}')} \boldsymbol{\ell}(\mathbf{y}', \mathbf{y}_{\textbf{target}}) \tag{1}$$

The target agnostic robustness $\mathscr{R}$ is defined by

$$\mathscr{R}_{p(\mathbf{y}|\mathbf{x})}(\mathbf{x}') = \mathscr{L}_{p(\mathbf{y}|\mathbf{x})}(\mathbf{x}', \mathscr{F}_{p(\mathbf{y}|\mathbf{x})}(\mathbf{x}')) \tag{2}$$

$$\mathscr{F}_{p(\mathbf{y}|\mathbf{x})}(\mathbf{x}') = \mathbb{E}_{\mathbf{y}' \sim p(\mathbf{y}|\mathbf{x}')} \mathbf{y}' \tag{3}$$

The mean squared error (MSE) $\boldsymbol{\ell}(\mathbf{y}, \mathbf{y}_{\textbf{target}})$ can be defined by:

$$\boldsymbol{\ell}(\mathbf{y}, \mathbf{y}_{\textbf{target}}) = ||\mathbf{y} - \mathbf{y}_{\textbf{target}}||_2^2 \tag{4}$$

$$= \sum_i (\mathbf{y}_i - \mathbf{y}_{\textbf{target}_i})^2 \tag{5}$$

Note that the MSE implemented in most deep learning frameworks further takes mean value along the $\mathbf{y}$ dimension, that is

$$\boldsymbol{\ell}(\mathbf{y}, \mathbf{y}_{\textbf{target}}) = \frac{1}{d_y} ||\mathbf{y} - \mathbf{y}_{\textbf{target}}||_2^2 \tag{6}$$

$$= \frac{1}{d_y} \sum_i (\mathbf{y}_i - \mathbf{y}_{\textbf{target}_i})^2 \tag{7}$$

However, since the estimated robustness $\mathscr{r}$ is normalized (see Algorithm 1, line [8-9]), thus invariant to the scale of $\mathbf{y}$, we analysis with the MSE loss in Eq. (5).

Property 1. *With the MSE loss*

$$\mathscr{L}_{p(\mathbf{y}|\mathbf{x})}(\mathbf{x}', \mathbf{y}_{\textit{target}}) = \mathscr{R}_{p(\mathbf{y}|\mathbf{x})}(\mathbf{x}') + \mathscr{B}(\mathbf{x}', \mathbf{y}_{\textit{target}}) \tag{8}$$

$$\mathscr{B}(\mathbf{x}', \mathbf{y}_{\textit{target}}) = ||\mathscr{F}_{p(\mathbf{y}|\mathbf{x})}(\mathbf{x}') - \mathbf{y}_{\textit{target}}||_2^2 \tag{9}$$

Proof.

$$\begin{aligned}
&\mathscr{L}_{p(\mathbf{y}|\mathbf{x})}(\mathbf{x}', \mathbf{y}_{\textbf{target}}) && (10)\\
&= \mathbb{E}_{\mathbf{y}'\sim p(\mathbf{y}|\mathbf{x}')}\mathscr{C}(\mathbf{y}', \mathbf{y}_{\textbf{target}}) && (11)\\
&= \mathbb{E}_{\mathbf{y}'\sim p(\mathbf{y}|\mathbf{x}')}||\mathbf{y}' - \mathbf{y}_{\textbf{target}}||_2^2 && (12)\\
&= \mathbb{E}_{\mathbf{y}'\sim p(\mathbf{y}|\mathbf{x}')}||\mathbf{y}' - \mathscr{F}(\mathbf{x}') + \mathscr{F}(\mathbf{x}') - \mathbf{y}_{\textbf{target}}||_2^2 && (13)\\
&= \mathbb{E}_{\mathbf{y}'\sim p(\mathbf{y}|\mathbf{x}')}\sum(\mathbf{y}'_i - \mathscr{F}(\mathbf{x}')_i + \mathscr{F}(\mathbf{x}')_i - \mathbf{y}_{\textbf{target}_i})^2 && (14)\\
&= \sum\mathbb{E}_{\mathbf{y}'\sim p(\mathbf{y}|\mathbf{x}')}(\mathbf{y}'_i - \mathscr{F}(\mathbf{x}')_i + \mathscr{F}(\mathbf{x}')_i - \mathbf{y}_{\textbf{target}_i})^2 && (15)\\
&= \sum\mathbb{E}_{\mathbf{y}'\sim p(\mathbf{y}|\mathbf{x}')}((\mathbf{y}'_i - \mathscr{F}(\mathbf{x}')_i)^2 + (\mathscr{F}(\mathbf{x}')_i - \mathbf{y}_{\textbf{target}_i})^2 && (16)\\
&+ (\mathbf{y}'_i - \mathscr{F}(\mathbf{x}')_i)(\mathscr{F}(\mathbf{x}')_i - \mathbf{y}_{\textbf{target}_i})) && (17)\\
&= \sum\mathbb{E}(\mathbf{y}'_i - \mathscr{F}(\mathbf{x}')_i)^2 + \mathbb{E}(\mathscr{F}(\mathbf{x}')_i - \mathbf{y}_{\textbf{target}_i})^2 && (18)\\
&+ \mathbb{E}(\mathbf{y}'_i - \mathscr{F}(\mathbf{x}')_i)(\mathscr{F}(\mathbf{x}')_i - \mathbf{y}_{\textbf{target}_i}) && (19)\\
&= \sum\mathbb{E}(\mathbf{y}'_i - \mathscr{F}(\mathbf{x}')_i)^2 + \mathbb{E}(\mathscr{F}(\mathbf{x}')_i - \mathbf{y}_{\textbf{target}_i})^2 && (20)\\
&= \mathbb{E}\sum(\mathbf{y}'_i - \mathscr{F}(\mathbf{x}')_i)^2 + \mathbb{E}\sum(\mathscr{F}(\mathbf{x}')_i - \mathbf{y}_{\textbf{target}_i})^2 && (21)\\
&= \mathbb{E}||\mathbf{y}' - \mathscr{F}(\mathbf{x}')||_2^2 + ||\mathscr{F}(\mathbf{x}') - \mathbf{y}_{\textbf{target}}||_2^2 && (22)\\
&= \mathbb{E}_{\mathbf{y}'\sim p(\mathbf{y}|\mathbf{x}')}||\mathbf{y}' - \mathscr{F}(\mathbf{x}')||_2^2 + ||\mathscr{F}(\mathbf{x}') - \mathbf{y}_{\textbf{target}}||_2^2 && (23)\\
&= \mathscr{R}_{p(\mathbf{y}|\mathbf{x})}(\mathbf{x}') + \mathscr{B}(\mathbf{x}', \mathbf{y}_{\textbf{target}}) && (24)
\end{aligned}$$

Where the Eq. (19) is eliminated since

$$\begin{aligned}
&\mathbb{E}(\mathbf{y}'_i - \mathscr{F}(\mathbf{x}')_i)(\mathscr{F}(\mathbf{x}')_i - \mathbf{y}_{\textbf{target}_i}) && (25)\\
=&(\mathscr{F}(\mathbf{x}')_i - \mathbf{y}_{\textbf{target}_i})\mathbb{E}(\mathbf{y}'_i - \mathscr{F}(\mathbf{x}')_i) && (26)\\
=&(\mathscr{F}(\mathbf{x}')_i - \mathbf{y}_{\textbf{target}_i})(\mathbb{E}\mathbf{y}'_i - \mathscr{F}(\mathbf{x}')_i) && (27)\\
=&0 && (28)
\end{aligned}$$

□

The Property 1 indicates that the robust design objective Eq. (1) can be decomposed into the sum of a target agnostic robustness $\mathscr{R}$ and a bias term $\mathscr{B}$.

Property 2. *Training a forward model $f_{\textit{forward}}(\mathbf{x})$ with the MSE loss converges to the mean function $\mathscr{F}(\mathbf{x})$.*

Proof. First, minimizing the MSE loss with respect to several targets converges to their mean value, that is

$$\arg\min_{\mathbf{x}} \sum_{i=1}^{n} ||\mathbf{x} - \mathbf{x}_i||_2^2 = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \tag{29}$$

since the convexity and the only local minimum is given by

$$\frac{\partial \sum_{i=1}^{n} ||\mathbf{x} - \mathbf{x}_i||_2^2}{\partial \mathbf{x}} = 2n\mathbf{x} - 2\sum \mathbf{x}_i = 0 \tag{30}$$

Secondly, the mean value of $\mathbf{y}_i$ given $\mathbf{x}_i$ is an unbiased estimation of $\mathbb{E}_{p(\mathbf{y}|\mathbf{x}_i)}\mathbf{y} = \boldsymbol{\mathcal{F}}(\mathbf{x}_i)$, thus Property 2 holds. □

Utilizing Property 2, we train a feed-forward neural network $f_{\text{forward}}(\mathbf{x})$ to estimate the mean function $\boldsymbol{\mathcal{F}}(\mathbf{x})$.

Discussion on alternative of normalizing flows

Conditional variational auto-encoders (cVAEs) can also be used in our RID-Noise framework. A cVAE has an encoder modeling $q(z|x, y)$ and a decoder modeling $p(x|z, y)$, where q is a variational distribution which approximates the true posterior $p(z|x, y)$. In the robust inverse design setting, x is a design parameter, y is a design target, and z is the latent representation of x given corresponding y. The log-likelihood can be bounded with the evidence lower bound (ELBO):

$$\begin{aligned} \log p(x|y) &\geq -KL(q(z|x, y), p(z)) + \mathbb{E}_q \left[\log p(x|z, y)\right] \\ &= \textbf{ELBO}(x, y) \end{aligned}$$

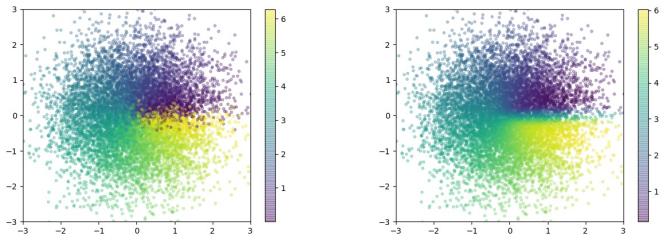
Thus our weighted negative log-likelihood (WNLL) loss function can be modified to

$$\begin{aligned} \text{WNLL}(D; \theta) &= -\sum_{i=1}^{n} w(\mathbf{x}_i, \mathbf{y}_i, D) \log p(\mathbf{x}_i|\mathbf{y}_i; \theta) \\ &\leq -\sum_{i=1}^{n} w(\mathbf{x}_i, \mathbf{y}_i, D)\textbf{ELBO}(\mathbf{x}_i, \mathbf{y}_i) \end{aligned}$$

However, we found that cVAEs performs worse than cINNs in RID-Noise, which could be due to that cINNs can optimize exact weighed likelihood directly instead of minimizing an upper bound.

Additional Experiments

With the page limit, we exhibit some additional experiments in the following sections.



(a) Data distribution. (b) $f_{\text{forward}}(\mathbf{x})$

Figure 1: Visualize the $f_{\text{forward}}(\mathbf{x})$. It is clear that the $f_{\text{forward}}(\mathbf{x})$ predicts a blurred $\mathbf{y}$ of data distribution, especially around the $y = 0$ area. The f_{forward} function successfully learned to predict the mean value of $\mathbf{y}$ even though the original data distribution is such noisy.

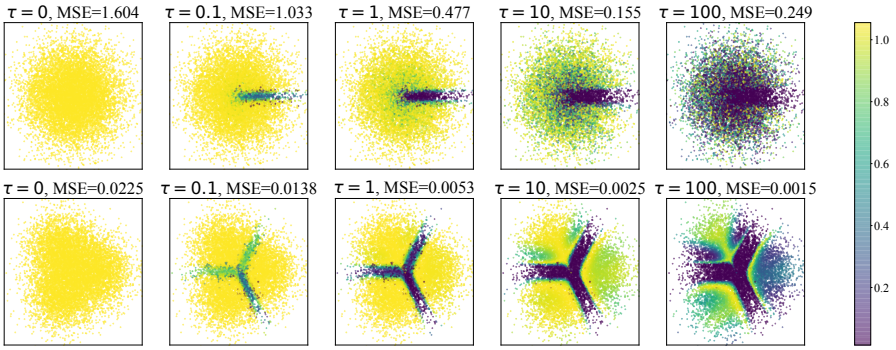


Figure 2: The relationship between weight distribution and value of τ in the Radian task and Clusters task.

Visualize f_{forward}

Does $f_{\text{forward}}(\mathbf{x})$ really learned $\mathscr{F}(\mathbf{x})$ as suggested in Property 2? We plot the training data distribution and the predicted $\mathbf{y}$ value to show that $f_{\text{forward}}(\mathbf{x})$ successfully learned to predict the mean value function $\mathscr{F}(\mathbf{x})$, as depicted in Figure 1.

Visualize the Weights

We visualize the distribution of the learned weights $\boldsymbol{w}$ to show how it works in Figure 2 and Figure 3. We can infer that by tuning the value of τ, we can get rid of more noisy training points gradually.

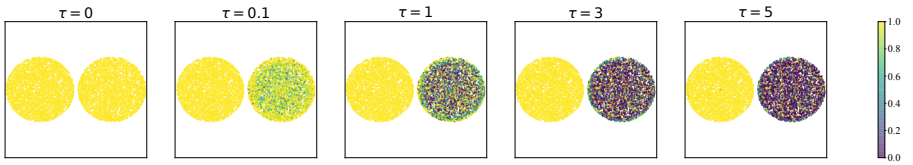


Figure 3: The relationship between weight distribution and value of τ in the Radius task.

Visualizing of Toy Tasks

We visualized the generated distribution of cGAN, cVAE, cINN and RID-Noise in the Clusters task in Figure 4. The effect of τ in the Radian task and the Clusters task are depicted in Figure 5 and Figure 6 respectively.

Reproducibility

The Robust Inverse Design Benchmarks

All three benchmark tasks can be denoted by a deterministic forwarding function $g(\mathbf{x}) = \mathbf{y}$, where $\mathbf{x}$ denotes the corresponding design parameters, and $\mathbf{y}$ is the response vector. We construct stochastic problems by adding random noises to the input $\mathbf{x}$ and output $\mathbf{y}$ of the function $g(\mathbf{x}) = \mathbf{y}$.

We define the stochastic problem with $\mathbf{x}$ dependent noise:

$$g_{n_x}(\mathbf{x}) = g(\mathbf{x} + \boldsymbol{n}_x(\mathbf{x}, \epsilon)) \tag{31}$$

A stochastic problem with $\mathbf{y}$ dependent noise is defined by

$$g_{n_y}(\mathbf{x}) = g(\mathbf{x}) + \boldsymbol{n}_y(g(\mathbf{x}), \epsilon) \tag{32}$$

A stochastic problem with $\mathbf{x}$ dependent noise as well as $\mathbf{y}$ dependent noise is defined by

$$g_{n_{xy}}(\mathbf{x}) = \tilde{\mathbf{y}} + \boldsymbol{n}_y(\tilde{\mathbf{y}}, \epsilon) \tag{33}$$

$$\tilde{\mathbf{y}} = g(\mathbf{x} + \boldsymbol{n}_x(\mathbf{x}, \epsilon)) \tag{34}$$

Where ϵ is a random noise vector sampled from a normal distribution. The noise functions $\boldsymbol{n}_x$, $\boldsymbol{n}_y$ and $\boldsymbol{n}_{xy}$ are constructed to emphasize the parameter selection power of robust design methods for each task.

For example, the $\boldsymbol{n}_x$ in the Kinematics task adds smaller noise when the arm is higher than horizon, thus the robust design method should learn to prefer to push the arm higher. That is, the $\mathbf{x}$ is transformed using the following **x_noise_fn**

```
def x_noise_fn(x):
    x = np.copy(x)
    x1 = np.copy(x[:, 1])
    s = np.copy(x1 < 0)
```

```
    x1[s] = x1[s] + np.random.randn(*x1[s].shape) * 0.5
    x[:, 1] = x1
    return x
```

The n_y in the Kinematics task also adds smaller noise when the arm is higher than horizon. That is, the $\mathbf{y}$ is transformed using the following **y_noise_fn**

```
def y_noise_fn(y):
    y = np.copy(y)
    y1 = np.copy(y[:, 1])
    s = y1 < 0
    _noise = np.random.randn(*y1.shape) * 0.5
    y1[s] = y1[s] + _noise[s]
    y[:, 1] = y1
    return y
```

When we apply both the $\mathbf{x}$ dependent noise and $\mathbf{y}$ dependent noise, the data is transformed by

```
_x = x_noise_fn(x)
_y = forward(_x)
y = y_noise_fn(_y)
```

We also designed problem specific noise functions for the Ballistics task and the Meta-Material task. For example, the $\mathbf{x}$ dependent noise of Ballistics task is related to the velocity of the ball; the $\mathbf{y}$ dependent noise of Meta-Material task is related to the value of the low frequency magnitude. We generate 10000 data points for training and 10000 data points for testing in Kinematics and Ballistics task; 20000 for training and 10000 for testing for the Meta-Material task.

All the experiments are conducted on a single work station with Nvidia GTX-2080Ti*2, Intel(R) Xeon(R) CPU E5-2620 v4, and 120 Gb memory. The forward simulator and noise functions are provided in the code supplementary material. For more details, please refer to the code implementation provided in the supplementary material.

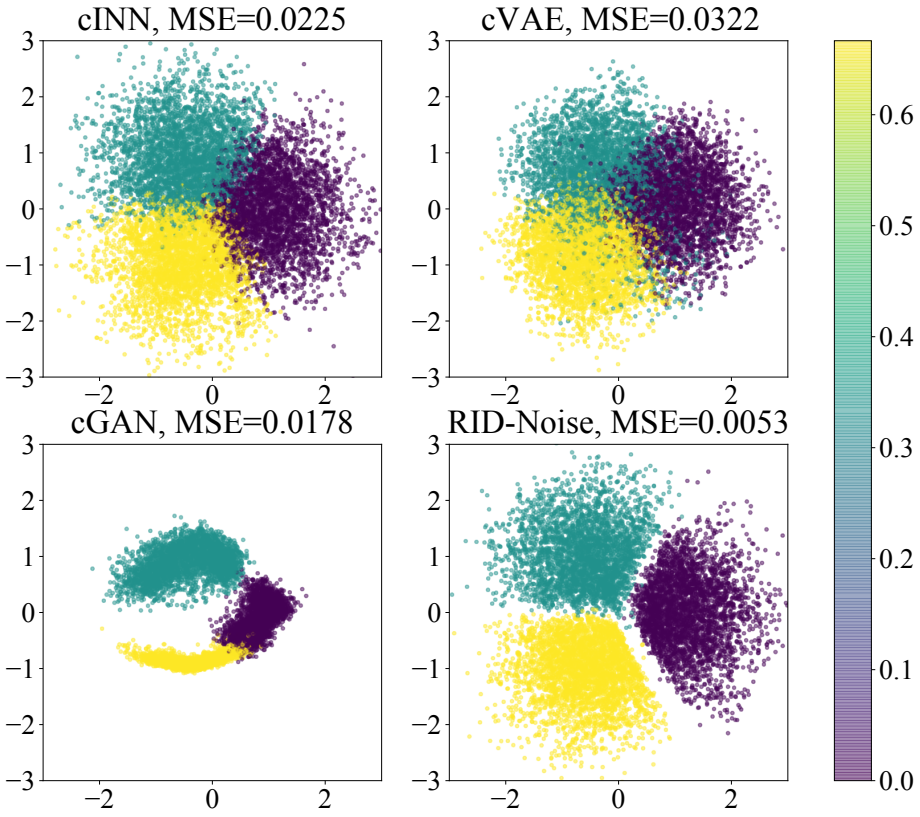


Figure 4: Comparison of inverse design methods on the Clusters task. The cVAE and cINN can manage to fit the noisy data distribution, thus lead to large error. The cGAN struggles to fit the noise and leads to a twisty distribution. Only RID-Noise learns the data distribution while keeps a low MSE.

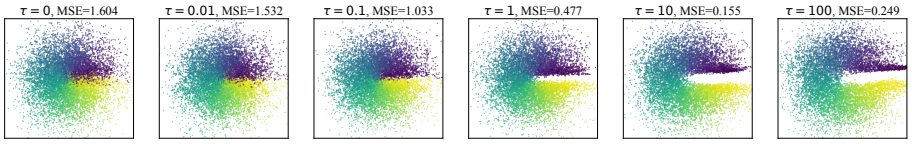


Figure 5: Effect of τ in the Radian task.

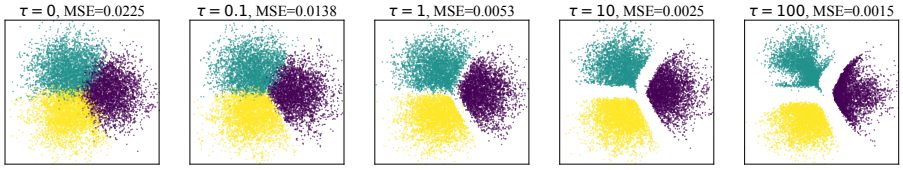


Figure 6: Effect of τ in the Clusters task.