Autoinverse: Uncertainty Aware Inversion of Neural Networks

Navid Ansari
Max Planck Institute for Informatics
Saarbrücken, Germany
nansari@mpi-inf.mpg.de

Hans-Peter Seidel
Max Planck Institute for Informatics
Saarbrücken, Germany
hpseidel@mpi-sb.mpg.de

Nima Vahidi Ferdowsi
Max Planck Institute for Informatics
Saarbrücken, Germany
nvahidi@mpi-inf.mpg.de

Vahid Babaei
Max Planck Institute for Informatics
Saarbrücken, Germany
vbabaei@mpi-inf.mpg.de

Abstract

Neural networks are powerful surrogates for numerous forward processes. The inversion of such surrogates is extremely valuable in science and engineering. The most important property of a successful neural inverse method is the performance of its solutions when deployed in the real world, i.e., on the *native forward process* (and not only the learned surrogate). We propose *Autoinverse*, a highly automated approach for inverting neural network surrogates. Our main insight is to seek inverse solutions in the vicinity of reliable data which have been sampled form the forward process and used for training the surrogate model. *Autoinverse* finds such solutions by taking into account the predictive uncertainty of the surrogate and minimizing it during the inversion. Apart from high accuracy, *Autoinverse* enforces the feasibility of solutions, comes with embedded regularization, and is initialization free. We verify our proposed method through addressing a set of real-world problems in control, fabrication, and design. Our code and data are available at: https://gitlab.mpi-klsb.mpg.de/nansari/autoinverse

1 Introduction

"... optimizing for the wrong thing — offering prayers to the bronze snake of data rather the larger force behind it." [6]

With the deep learning breakthrough during the last decade, data-driven modeling using neural network based *surrogates* is now a standard practice in science and engineering. These surrogates strive to imitate the behavior of a *native forward process* (NFP), such as a physics simulation, which maps a *design* into its *performance*[1]. While forward processes are essential for troubleshooting and analysis, oftentimes their ultimate application lies in their inversion, i.e., the reverse process of mapping performances into designs. Despite the recent progress in inversion of neural networks within multiple disciplines [35, 16, 13, 37], a fundamental unaddressed question is still standing out. Inversion of a surrogate model, fitted to the data sampled from the NFP, is ultimately different than the inversion of the NFP itself. The source of this gap could be explicit, such as the noise in

[1] In the applications showcased in this paper (fabrication-oriented design and robotics), the term *design* refers to the space where the input to the forward process is characterized and parameterized and *performance* refers to the parameterized space of desired properties. Commonly, *hidden state* or *parameters*, and *measurement* or *goal* are used interchangeably with design and performance, respectively

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

measurements, or implicit, such as the poor sampling of the NFP. Although the obtained solutions from inverting the surrogate can be evaluated on the NFP, none of the current neural inversion methods offers a tailored solution for addressing this important gap.

Our main insight in this work is to *expect* and *account* for any potential mismatch between the data, and consequently the surrogate, on the one hand and the NFP on the other. Our proposed method, `Autoinverse`, realizes this vision by taking into account the predictive uncertainty of the surrogate and minimizing it during the inversion. Therefore, the inverted solutions avoid the unreliable regions within the training data.

We show that our `Autoinverse` strategy can augment existing neural inversion methods (both optimization-based and architecture-based approaches) with uncertainty compensation in a simple and practical manner. `Autoinverse` closes the gap between the surrogate and the NFP *not* through attempting a perfect fit of the surrogate to the NFP, an onerous task, but by finding inverse solutions in the vicinity of the reliable training data where the surrogate and the NFP are most similar. Neural inverse methods equipped with `Autoinverse` outperform their counterparts significantly on both standard data sampled from the NFP and imperfect data, e.g., those corrupted by noise. Apart from high accuracy, `Autoinverse` enforces the feasibility of solutions, comes with embedded regularization (freeing the inversion approaches from hand-crafted regularizations based on domain knowledge), and is initialization-free. It achieves all these properties in a highly automated manner and only with a light, intuitive tuning.

2 Related work

Neural Network Inversion We can divide neural network inversion approaches into two main categories. First, *inverse architectures*, where we compute a network architecture that takes a given performance and maps it into a (distribution of) design(s). Second, *direct optimization* [33], where we optimize for a design such that it produces the desired performance. Although the simplest inverse architecture can be attempted by training a neural network in the reverse direction, it fails because of the one-to-many nature of the mapping. *Tandem* networks use an inverse architecture by employing a pre-trained forward network in order to compute a consistent loss. The tandem approach has been developed independently across different disciplines [23, 32, 33, 39] dating back (at least) to Tominaga [34]. Many inverse architectures try to model the conditional posterior, $p(x|y)$, using variational methods [24, 19] based on (conditional) variational auto-encoders [20]. These networks condition the design on the target performance and yield a distribution of solutions from which multiple samples could be drawn. Kruse et al. [21] show that the invertible neural networks (INNs), built upon normalizing flows [10], give the highest accuracy in terms of both surrogate error and design posterior compared to a wide range of inverse architectures.

When using direct optimization methods, gradient-based optimizers can be readily used as the neural surrogate is differentiable. Ren et al. [30] use stochastic gradient descent via backpropagation with respect to the design variables, and present a highly accurate and practical method. They benchmark their method, dubbed as *neural adjoint* (*NA*), against a set of inverse architectures and obtain significantly more accurate solutions. Sun et al. [33] showed a similar approach except using a quasi-Newton method for optimization. Ansari et al. [2] push forward in this direction by demonstrating that, for piecewise linear neural networks, e.g., those with ReLU activation, the direct optimization can be formulated as a mixed-integer linear program (MILP) and thus obtain *globally* optimal solutions. While optimization methods are very accurate, their main disadvantage is their performance as they can be orders of magnitude slower than inverse architectures.

Neural Networks and Predictive Uncertainty While neural networks are ubiquitous in almost all branches of natural sciences, their weakness at quantifying predictive uncertainty impedes their use in crucial applications. Using a Bayesian formalism [4], Bayesian neural networks (BNNs) [28, 9, 26], given the training data and a prior over network's parameters, compute the posterior distribution of the parameters. Having computed the posterior, the predictive uncertainty can be computed. The inference step in computing BNNs is known to be computationally hard [17]. This explains the popularity of simpler methods for estimating predictive uncertainty, such as Monte Carlo dropout [11] and Deep Ensembles [22]. Deep Ensembles strikes a good balance between simplicity and practicality on the one hand and predictive performance on the other hand (Section 3.3). One of the main advantages of the Deep Ensembles is its capability to predict aleatoric and epistemic uncertainty separately.

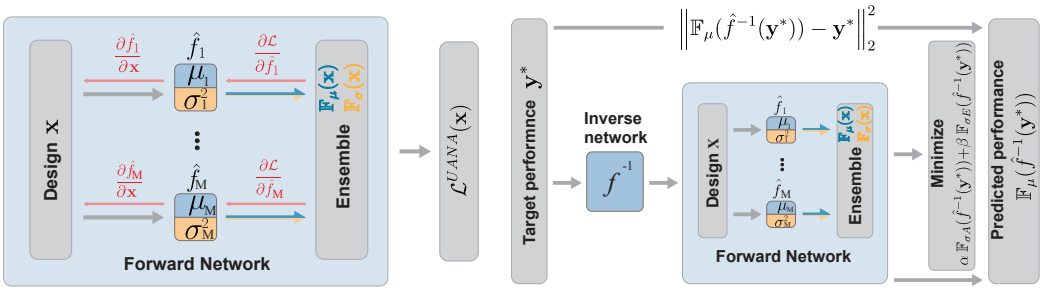


Figure 1: By using a deep ensemble predictive uncertainty estimator as the forward model we can make conventional inversion methods uncertainty aware. On the left we can see `UANA` and on the right `UA-tandem` architecture.

Aleatoric and epistemic uncertainty carry different information regarding the surrogate [18] and considering both of them improves the quality of the neural inversion (Section 4.3). Neural networks capable of predictive uncertainty are increasingly adopted in many different applications, such as reinforcement learning and active learning [12]. As we shall see, neural inversion is yet another domain that takes advantage of this trend.

3 Method

`Autoinverse` is an easy-to-implement technique for augmenting neural inverse methods with uncertainty awareness. `Autoinverse` achieves this goal by, first, training a surrogate capable of predictive uncertainty [25]. Second, relying on this trained surrogate and using a novel inversion cost function, `Autoinverse` finds accurate designs with minimal uncertainty. We apply `Autoinverse` on two inverse methods belonging to the two main neural inversion categories, i.e., optimization- and architecture-based in Sections 3.1 and 3.2, respectively. As we will see in Section 3.3, we rely on established methods [22] to train networks equipped with predictive uncertainty.

3.1 Uncertainty aware neural adjoint (`UANA`)

Given a pretrained neural surrogate $\hat{f}(\cdot)$, neural adjoint (`NA`) [30] is an inverse method that uses the cost function $\mathcal{L}^{NA}(\cdot)$ to push designs $\mathbf{x}$ to have a performance $\hat{f}(\mathbf{x})$ as close as possible to a desired performance $\mathbf{y}^*$:

$$\mathcal{L}^{NA}(\mathbf{x}) := \arg\min_{\mathbf{x}} \left\| \hat{f}(\mathbf{x}) - \mathbf{y}^* \right\|_2^2. \tag{1}$$

`NA` uses gradient descent to iteratively reduce the cost function, a scheme much like training neural networks but with the input as the optimization variable instead of network's weights and biases. Equation 2 shows a single `NA` iteration with δ as the step size:

$$\mathbf{x}^z = \mathbf{x}^{z-1} - \delta\left(\frac{\partial \mathcal{L}^{NA}}{\partial \hat{f}} \times \frac{\partial \hat{f}}{\partial \mathbf{x}}\right). \tag{2}$$

`Autoinverse` proposes to perform the inversion using a pretrained BNN. We use Deep Ensembles [22] made of M neural networks capable of a prediction $\mathbb{F}_{\mu}(\mathbf{x})$, as well as its *aleatoric* $\mathbb{F}_{\sigma A}(\mathbf{x})$ and *epistemic* $\mathbb{F}_{\sigma E}(\mathbf{x})$ uncertainties (Section 3.3). Aleatoric uncertainty increases as the noise level in the training data increases. Epistemic uncertainty measures the uncertainty in the model. `Autoinverse` modifies `NA` such that we obtain solutions $\mathbf{x}$ that have performances close to the target performance $\mathbf{y}^*$ *while* resulting in small aleatoric and epistemic uncertainties:

$$\mathcal{L}^{UANA}(\mathbf{x}) := \arg\min_{\mathbf{x}} \|\mathbb{F}_{\mu}(\mathbf{x}) - \mathbf{y}^*\|_2^2 + \alpha\, \mathbb{F}_{\sigma A}(\mathbf{x}) + \beta\, \mathbb{F}_{\sigma E}(\mathbf{x}) \tag{3}$$

We introduce α and β as hyperparameters to adjust the relative significance of aleatoric and epistemic uncertainties, respectively.

Equation 4 shows how one iteration of UANA requires the back-propagation using the ensemble of all gradients of M networks of Deep Ensembles:

$$\mathbf{x}^z = \mathbf{x}^{z-1} - \delta \sum_{m=1}^{M} \left(\frac{\partial \mathcal{L}^{UANA}}{\partial \hat{f}_m} \times \frac{\partial \hat{f}_m}{\partial \mathbf{x}} \right) \tag{4}$$

where $\hat{f}_m$ represents one of the networks in the ensemble. Figure 1(a) depicts this collective procedure where each individual network in the ensemble votes for the direction where updating the design will lead to the maximal accuracy and minimal uncertainty.

3.2 Uncertainty aware tandem (UA-tandem)

Tandem is a representative of architecture-based methods in which we train an inverse network $\hat{f}^{-1}(\cdot)$ in a manner resembling the encoder-decoder architecture ([32–34]). Unlike the encoder-decoder approach, we start with training the forward model $\hat{f}(\cdot)$. Then we freeze the trainable parameters of $\hat{f}(\cdot)$ and train the inverse model $\hat{f}^{-1}(\cdot)$ in the position of the encoder in order to decrease the cost function:

$$\mathcal{L}^T(\hat{f}^{-1}(\mathbf{y}^*)) := \arg\min_{\hat{f}^{-1}(\cdot)} \left\| \hat{f}(\hat{f}^{-1}(\mathbf{y}^*)) - \mathbf{y}^* \right\|_2^2. \tag{5}$$

Once $\hat{f}^{-1}(\cdot)$ is trained we can simply query it to find designs with our desired performances:

$$\hat{f}^{-1}(\mathbf{y}^*) = \mathbf{x}. \tag{6}$$

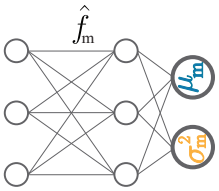
The uncertainty-aware tandem (UA-tandem) follows the same procedure except that it replaces $\hat{f}(\cdot)$ with $\mathbb{F}_\mu(\cdot)$. Additionally, it includes the uncertainties in the loss:

$$\mathcal{L}^{UAT}(\hat{f}^{-1}(\mathbf{y}^*)) := \arg\min_{\hat{f}^{-1}(\cdot)} \left\| \mathbb{F}_\mu(\hat{f}^{-1}(\mathbf{y}^*)) - \mathbf{y}^* \right\|_2^2 + \alpha\, \mathbb{F}_{\sigma A}(\hat{f}^{-1}(\mathbf{y}^*)) + \beta\, \mathbb{F}_{\sigma E}(\hat{f}^{-1}(\mathbf{y}^*)). \tag{7}$$

Figure 1(b) depicts the architecture of UA-tandem.

3.3 Predictive uncertainty using Deep Ensembles

Deep Ensembles comprises of an ensemble of M neural networks $\hat{f}_m$ each capable of a prediction $\mu_m(\mathbf{x})$ and its associated uncertainty $\sigma_m(\mathbf{x})$ in form of a Gaussian distribution $\mathcal{N}(\mu_m(\mathbf{x}), \sigma_m(\mathbf{x}))$. The cost function for training each network in the ensemble is the negative log likelihood [29]:



$$\mathcal{L}_m^{NLL} := \frac{\log(\sigma_m^2(\mathbf{x}))}{2} + \frac{(\mathbf{y}^* - \mu_m(\mathbf{x}))^2}{2\sigma_m^2(\mathbf{x})}. \tag{8}$$

Intuitively, in case of the aleatoric uncertainty $\mu_m(\mathbf{x})$ fails to reliably predict $\mathbf{y}^*$. Hence, $\sigma_m^2(\cdot)$ must increase to reduce the loss while the first term ensures $\sigma_m^2(\cdot)$ does not diverge to infinity.

The next step is to ensemble the results of all the networks into a single prediction and a single uncertainty. Deep Ensembles [22] models the ensemble as a single Gaussian distribution $\mathcal{N}(\mathbb{F}_\mu(\mathbf{x}), \mathbb{F}_\sigma(\mathbf{x}))$ approximating the mixture of M previously computed Gaussian distributions

$$\mathbb{F}_\mu(\mathbf{x}) := \frac{1}{M} \sum_m \mu_m(\mathbf{x}), \tag{9a}$$

$$\mathbb{F}_\sigma(\mathbf{x}) = \frac{1}{M} \sum_m (\sigma_m^2(\mathbf{x}) + \mu_m^2(\mathbf{x})) - \mathbb{F}_\mu^2(\mathbf{x}). \tag{9b}$$

The uncertainty of the ensemble can be decomposed into two input-dependent uncertainties, i.e., aleatoric $\mathbb{F}_{\sigma A}(\mathbf{x})$ and epistemic $\mathbb{F}_{\sigma E}(\mathbf{x})$ through slight modification to Equation 9b [18].

$$\mathbb{F}_{\sigma}(\mathbf{x}) := \alpha\, \mathbb{F}_{\sigma A}(\mathbf{x}) + \beta\, \mathbb{F}_{\sigma E}(\mathbf{x}), \tag{10a}$$

$$\mathbb{F}_{\sigma A}(\mathbf{x}) := \frac{1}{M} \sum_{m} \sigma_m^2(\mathbf{x}), \tag{10b}$$

$$\mathbb{F}_{\sigma E}(\mathbf{x}) := \frac{1}{M} \sum_{m} (\mu_m^2(\mathbf{x}) - \mathbb{F}_{\mu}^2(\mathbf{x})). \tag{10c}$$

This enables us to control the behavior of the neural inversion by tuning the relative importance of these two uncertainties.

4 Evaluation

We evaluate the performance of `Autoinverse` through experimenting with the existing neural inverse methods and their uncertainty-aware counterparts. In the paper, we focus on `NA` and `UANA` while `tandem` and `UA-tandem` are evaluated mainly in the Appendix. We perform the evaluation on a set of applications in robotics, and fabrication-oriented computational design. We report the error by running the experiments 3 times to capture the variations. More details are provided in each case study and in Appendix (Section A).

Equations 3 and 7 have two hyperparameters (α and β) that keep the balance between the MSE, the aleatoric uncertainty and the epistemic uncertainty. We observe that with a relatively larger epistemic weight (β) we obtain better results. Exploiting this intuition, we tune these parameters for 3 different sets of values for $\{\alpha, \beta\}$: $\{\{0.1, 1\}, \{1, 10\}, \{10, 100\}\}$. We then make two finer step depending on the outcome of the former evaluation and choose the best set of weights. We keep this budget of 5 experiment runs for the rival methods as well. Typically, we use 10% of the target performance for tuning our inverse methods.

4.1 Experiments

Multi-joint robot is a simple inverse kinematics problem which is being used as a standard test for neural inverse problems [21, 30]. In this problem, the *design* is the 1D position of the base of the multi-joint robot along with the 1D rotation of its three joints. The inverse problem concerns finding a combination of positions and angles for the base and the joints such that the tip of the arm lands the target position, i.e., the *performance*. We follow [30] for setting up this experiment and its corresponding analytical equation used as the NFP. The training data consists of 10,000 pairs of samples generated by randomly sampling the NFP.

Spectral printer Spectral printing enables digital fabrication of the object's appearance faithfully ([14, 5]). Unlike reproduction of the color (e.g., RGB), reproducing the spectrum ensures that the original and the duplicate remain visually similar independent of the color of the light source. Spectral printing has various important applications specially in fine art reproduction using both 2D and 3D printing [27, 1, 32]. Deep neural networks are becoming the main computational tool for modeling the spectral printing process specially when dealing with a large number of inks. The final objective is to find the correct ink densities at each pixel that can best estimate the target 31D spectrum.

In this experiment, we create the NFP by simulating a printer using an ensemble of 20 neural networks. The design space comprises the ink densities and our `spectral printer` NFP predicts the spectrum of the resulting color. We use real, measured data from an 8-channel printer with 8 EPSON inks including Cyan (C), Magenta (M), Yellow (Y), Black (K), Light black (LK), Light light black (LLK), Light Cyan (LC), and Light Magenta (LM). The light inks, added to the standard CMYK to improve the print quality, introduce significant multi-modality. All networks in the ensemble NFP are trained on 40,000 printed patches [1] consisting of different ink-density combinations and their corresponding spectra. The ensembling is intended for an accurate NFP and is independent of our use of Deep Ensembles for computing the uncertainty. The visual nature of `spectral printer` makes it attractive for analyzing different methods. We release this NFP to the public to add another example to the neural inversion testbed.

Table 1: The NFP and surrogate errors (mean $\pm$ STD) of different neural inverse methods on `multi-joint robot` for 1000 target locations.

Error	NA	UANA	tandem	UA-tandem	MINI	INN
NFP	$(3.24 \pm 0.51) \times 10^{-4}$	$(\mathbf{3.21 \pm 1.48}) \times \mathbf{10^{-6}}$	$(4.42 \pm 1.56) \times 10^{-3}$	$(\mathbf{4.04 \pm 0.38}) \times \mathbf{10^{-5}}$	1.6×10^{-3}	$(9.48 \pm 0.021) \times 10^{-3}$
Surrogate	$(1.99 \pm 0.05) \times 10^{-8}$	$(9.13 \pm 6.08) \times 10^{-7}$	$(8.58 \pm 3.00) \times 10^{-6}$	$(7.10 \pm 0.64) \times 10^{-6}$	0	$(2.04 \pm 0.017) \times 10^{-13}$

Soft robots are made of soft, flexible materials. This unique property has made them suitable candidates for interaction with humans in applications like minimally invasive surgery and advanced prosthetics [7]. Unlike `multi-joint robot` with a limited number of rotating joints, each segment of the soft robot is a potential actuator that through their contraction and expansion can determine the robot's final shape. This inverse kinematics problem is typically solved through partial differential equations ([38]). In order to accelerate the solve time of this inverse kinematics problem, Sun et al. [33] proposed a neural surrogate modeling of the problem and its inversion via `tandem`. The design space in this problem consists of the contraction or expansion of 40 controllable soft edge segments. The superposition of all the actuations determines the final deformation position of the soft robot via the position of its 206 vertices.

We use an FEM-based simulation ([38, 15]) as our NFP. We model the relationship between the actuations and the final shape of the soft robot with a neural-network surrogate. Our goal is to solve the neural inversion to find a suitable set of actuations (design) that brings the tip of the soft robot to the target position (performance). The training data consists of 50,000 samples queried by random sampling the actuation with an expansion ratios between -0.2 and 0.2 [33]. The designs are then evaluated on the FEM-based NFP to calculate their deformations.

4.2 Quantitative comparison of different neural inversion methods

We evaluate the accuracy of a set of neural inversion methods on `multi-joint robot` in terms of both the NFP and the surrogate errors. The surrogate error is the difference between the 're-prediction' of the obtained inverse solution by the surrogate neural network and the target performance. The NFP error is the difference between the target and the performance of the generated design evaluated on the NFP. In addition to the core methods described so far, we evaluate mixed-integer neural inversion (`MINI`) [2], and the invertible neural network (`INN`) [3].

Table 1 summarizes the inversion results on 1000 randomly sampled target locations for the multi-joint robotic arm. We keep our evaluation fair by setting the capacity of the neural surrogates comparable wherever possible. For instance, all methods except `MINI` have around 3 million parameters (see Appendix, Table 3 for more details). We also used equal computational resources for the tuning. Methods with hyperparameters, like `UANA` and `UA-tandem`, are tuned in 5 stages by hand. Alternatively, the methods without hyperparameters (`NA, tandem`) are given $5\times$ extra budget for inversion: We run `NA` and `tandem` $5\times$ and choose the model that generates the best NFP error. We repeat this process $3\times$ and report the standard error. `MILP` and `INN` are fundamentally different methods. `MILP` finds the global optimum and thus does not need tuning. `INN` has a latent space which we can sample to generate diverse designs. We sample INN's latent space 1024 times for all 1000 targets, evaluate them on the NFP, and report the best NFP error. We repeat this process three times to generate the standard error.

Table 1 shows the outstanding accuracy of the inverse methods that adopt `Autoinverse`, i.e., `UANA` and `UA-tandem`, in terms of the NFP error. It also demonstrate how even a perfect surrogate error (e.g., `MINI`) does not guarantee accurate solutions when tested on the NFP. A second look at Table 1 reveals further interesting insights. Although `NA` obtains notably lower surrogate error than its uncertainty-aware counterpart (`UANA`), it performs significantly worse in terms of the NFP error. The main reason for this trend is that, when optimizing the surrogate, `UANA` is not only concerned with finding accurate designs leading to a small accuracy gap between the target and candidate performances, i.e., the surrogate error, but also with those designs featuring low uncertainty (through the uncertainty term in Equation 10). Therefore, `UANA` achieves high accuracy in terms of the essential NFP error at the cost of worsening the inconsequential surrogate error. Furthermore, `UA-tandem`, for example, achieves better performance than `NA`. This is a remarkable result for an architecture-based

Table 2: The distribution of ink densities (≥ 0.4) after the inversion of `spectral printer` using `UANA`. Once we insert noise into LC channel or sample it sparsely, `Autoinverse` detects and avoids it. STD has been rounded to the nearest integer.

Model	dataset	NFP error	C	M	Y	K	LC	LM	LK	LLK
UANA	Standard	$(6.30 \pm 0.031) \times 10^{-3}$	186 ± 2	67 ± 4	63 ± 6	3 ± 0	437 ± 2	356 ± 7	26 ± 3	348 ± 5
	Sparse	$(5.64 \pm 0.017) \times 10^{-3}$	316 ± 1	60 ± 2	59 ± 1	2 ± 0	$\mathbf{0 \pm 0}$	326 ± 5	25 ± 4	323 ± 11
	Noisy	$(6.13 \pm 0.026) \times 10^{-3}$	263 ± 1	67 ± 4	34 ± 2	1 ± 0	$\mathbf{0 \pm 0}$	276 ± 3	29 ± 3	378 ± 5
NA	Standard	$(1.57 \pm 0.001) \times 10^{-1}$	1895 ± 13	1060 ± 26	1795 ± 18	162 ± 11	865 ± 8	1396 ± 30	179 ± 15	2378 ± 19
	Sparse	$(1.34 \pm 0.007) \times 10^{-1}$	905 ± 6	606 ± 7	1515 ± 8	137 ± 6	1604 ± 11	2130 ± 17	294 ± 15	2312 ± 24
	Noisy	$(1.47 \pm 0.002) \times 10^{-1}$	1192 ± 14	988 ± 10	1128 ± 10	55 ± 6	1029 ± 23	948 ± 21	283 ± 12	1742 ± 37

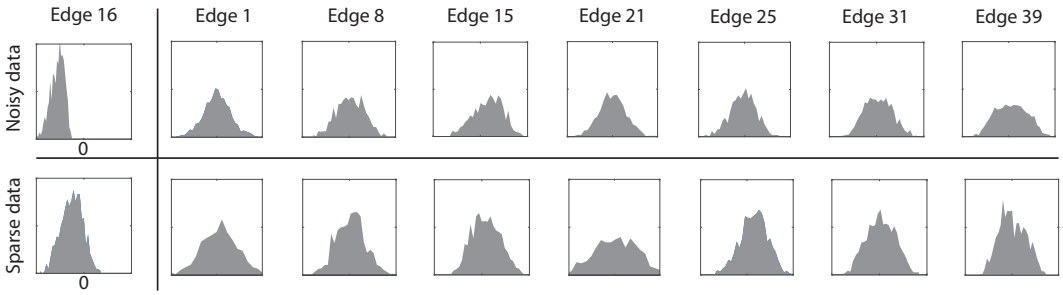


Figure 2: Distribution of actuation values for `soft robot`. Edge 16 is corrupted by noise (top) and sampled sparsely (bottom) in positive range. While other randomly chosen edges feature both negative and positive actuations, `UANA` produces solutions without (or with less) positive actuations for Edge 16.

method given it is significantly faster than the optimization-based `NA`. We report extensively the time performance and the details of the training for this and the next experiments in the Appendix.

4.3 Neural inversion in the presence of imperfect data

One of the main advantages of `Autoinverse` appears in scenarios where the training data suffers from noise, e.g., measurement noise or poor sampling, e.g., sparsity in some regions. We evaluate the performance of `Autoinverse` on imperfect training data using `NA` and `UANA` (`tandem` and `UA-tandem` are evaluated under the same configuration in the Appendix Section C).

Locally sparse data We sample both `soft robot` and `spectral printer` NFPs (Section 4.1) in a way that the data does not contain any samples from one of the inputs in a specific interval. For `spectral printer`, we would like to find ink densities to reproduce the spectra of the colors in the painting in Figure 3 (made of 3568 distinct color spectra). We sample the printer channels at 0 (no ink), 0.05, 0.1, 0.5, and 1 (full ink) densities to form the *standard* training data (with no known uncertainty). We create a partially *sparse* dataset similar to the standard one except for the Light Cyan (LC) channel for which we only have samples at 0, 0.05 and 0.1. Table 2 shows that while for the standard dataset `UANA` finds inverse solutions that include the LC channel frequently (437 times), for the sparse dataset it avoids this channel *completely* and compensate for it using the Cyan channel. `UANA` is able to avoid this channel as the epistemic uncertainty increases in sparse regions of dataset (see Appendix Section C.4 for more details).

In `soft robot` we sample the 16th (among 40) controllable edge only in the negative range (contraction only). We then use the trained network to invert 1000 test samples. Figure 2 shows the distribution of each edge for 1000 inversion tasks. We have plotted the distribution for the 16th edge as well as for 7 other randomly chosen edges. As evident from Figure 2 bottom row, `UANA` is highly reluctant to choose designs with positive actuations for this edge.

Locally noisy dataset With the same problem configuration as before, we would like to test the robustness of `Autoinverse` on a dataset locally corrupted with noise. We start with a standard dataset and inject Gaussian noise $\mathcal{N}(0,\ 0.1)$ to the spectrum of the samples with more than 0.4 LC density. Table 2 shows how after introducing noise to the LC, the network avoids that channel and compensates it by using more Cyan instead. In `soft robot` we corrupt the final shape of the soft

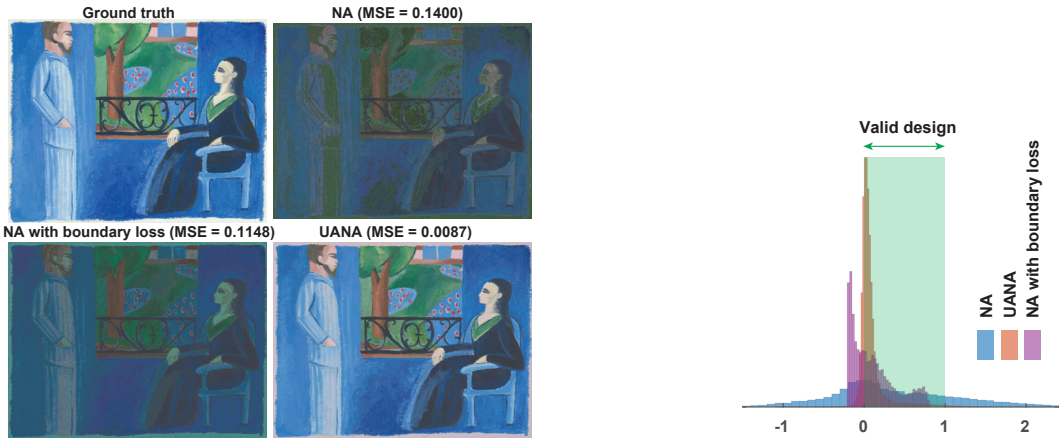


Figure 3: In painting reproduction, `UANA` outperforms `NA` with and without a boundary loss term. The distribution of the inverse solutions is shown on right.

robot with Gaussian noise $\mathcal{N}(0,\ 0.5)$ for all the shapes where the 16th edge has positive actuation. As we can see in Figure 2 (top row), the inversion has completely avoided positive actuations for this edge.

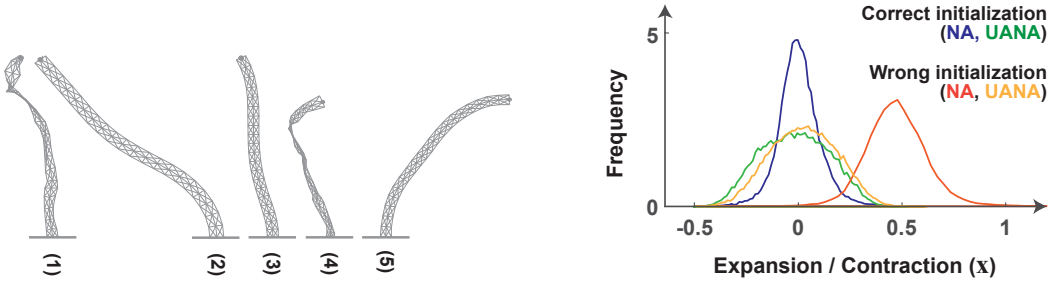
4.4 Autoinverse brings AutoML to neural inversion

Autoinverse incorporates feasibility. Deep Ensembles produces high epistemic uncertainty outside the distribution of the training data. This includes the regions where the NFP is not defined and thus not sampled. For example, ink densities outside $[0, 1]$ are not printable. If such cases arise during the inversion, they are clipped to $[0, 1]$ [1]. In such regions, networks in the ensemble do not agree and thus the epistemic uncertainty increases. `Autoinverse` automatically avoids these unfeasible regions. In order to simulate the result of the inversion, after clipping the out-of-range densities we feed them into the NFP. As evident from Figure 3 left, inversion via NA results in a poor reproduction of the original painting. `UANA`, on the other hand, achieves spectacular reproduction accuracy. This is explained by the plot in Figure 3 right, showing the distribution of ink densities obtained from both methods.

Autoinverse has built-in regularization. We incorporate regularization into inversion methods in order to obtain solutions that, among other purposes, agree with the observations and follow a certain statistical distribution. Oftentimes, regularization is case-specific, requires human knowledge, and comes with unexpected side effects. Here we show that `Autoinverse` follows the distribution of the training data naturally by taking into account the epistemic uncertainty. We validate this point using both `spectral printer` and `soft robot` experiments. We show that `Autoinverse` without any explicit regularization performs better or on par with its counterpart inversion methods equipped with regularization.

In the `spectral printer` experiment, we evaluate the effect of the *boundary loss*, originally proposed as a generic regularization for `NA` to limit the designs within a box constraint (see [30] and Appendix Section D). The boundary loss is added to Equation 1 and weighted using a hyperparameter. We tune this parameter with the same tuning budget we allocate for tuning the uncertainty weights (5 set of inversions on evaluation data). In Figure 3, we observe that although `NA` with boundary loss improves the distribution of ink densities within the valid region ($[0, 1]$), it still trails the regularization-free `UANA` significantly.

Regularizing the `soft robot` is less intuitive as the superposition of all actuations determines the final shape and whether it is physically plausible. In [33], the regularization is a smoothness term that keeps actuation values near each other (see Appendix Section D). Figure 4(a)(1) demonstrates how `NA` fails without regularization to control the robot with a reasonable deformation. Once the regularization is added to `NA`, the designs become physically meaningful (Figure 4(a)(2)). Figure 4(a)(3) shows how `UANA` performs comparably without any regularization.



(a) (b)

Figure 4: On the left we can see the distribution of actuations calculated by `UANA` and `NA` with two different initialization: one initialization near the training data distribution and one far from it. On the right we can see a range of randomly chosen soft robot shapes calculated by different methods with different regularization and initialization.

Autoinverse is initialization-free. Sensitivity to the initialization is a widely known issue in non-convex optimizations. Despite equipping `NA` with the smoothness term in `soft robot`, Figure 4(b) demonstrates how an incorrect initialization can result in solutions with seemingly good surrogate and regularization loss (see Appendix Section D) but in the infeasible region of the design space (Figure 4(a)(4)). `UANA` with the same incorrect initialization *and* without any regularization, leads to soft robot designs that reach the target location accurately and produce plausible deformations (Figure 4(a)(5)). For `UANA`, the solver starts with reducing the main contributor to the optimization objective, i.e., the epistemic uncertainty. Once a region with a reasonably small uncertainty is reached, the accuracy term starts to take effect and a desirable solution within the valid range of the design space is found.

4.5 Ablation studies

Separating the ensembling effect Although in both `NA` and `UANA` we use surrogates with similar capacity (network size), one could argue that the higher performance of `UANA` comes from the ensemble architecture of its surrogate. We detach the impact of ensembling and uncertainty awareness on the inversion performance by implementing `NA ensemble` where, instead of a single, large forward neural network, it uses an ensemble of networks. The inversion procedure is identical to `NA` (see Appendix Section E and Figure 5). We employ `NA ensemble` in `multi-joint robot` with the same configurations as in Section 4.2. The surrogate and NFP error for `NA ensemble` are $(3.30 \pm 0.59) \times 10^{-9}$ and $(1.17 \pm 0.32) \times 10^{-4}$, respectively. Comparing these values with those in Table 1, `NA ensemble` shows a slight improvement over `NA` but is significantly outperformed by `UANA`.

Diversity of activation functions in Deep Ensembles Deep Ensembles uses a similar set of networks with identical activation functions [22]. In practice, we observe that a diverse set of activation layers leads to a better performance of `Autoinverse`. Different activation functions generate different behaviours, show higher disagreement where the training data is under-represented and, thus, result in an accurate estimation of the landscape of epistemic uncertainty [36]. For all experiments in this work, we use a diverse range of activation functions (see Appendix Section A). As an ablation, we run `UANA` on `spectral printer` using sparse data (same configuration as Section 4.3) but with ReLU as the only activation layer. In contrast to Table 2, `UANA` (with ReLU only activation) does not completely avoid the sparse domain and delivers 19 ± 4 solutions that contain LC with densities ≥ 0.4.

5 Discussion

The `Autoinverse` cost function is multi-objective. In addition to finding a single solution through a weighted combination of these objectives (what we have seen so far), we capture the trade-off between accuracy and uncertainty using the Pareto front (see Appendix Section F). `Autoinverse`

is an inversion *strategy* that could be applied to various inverse methods, especially those that concern imitation of an original process (dubbed as NFP in our work). We look forward to see `Autoinverse` adopted for more inverse architectures beyond `tandem`, and more direct optimizations beyond first-order `NA`.

Predicting the uncertainty using Deep Ensembles [22] is not the most efficient solution. This is because multiple forward networks are trained in two stages (once for the mean, and once jointly for mean and variance). It is highly interesting to integrate the more efficient Monte Carlo dropout [11, 18] into `Autoinverse`.

References

[1] Navid Ansari, Omid Alizadeh-Mousavi, Hans-Peter Seidel, and Vahid Babaei. Mixed integer ink selection for spectral reproduction. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020.

[2] Navid Ansari, Hans-Peter Seidel, and Vahid Babaei. Mixed integer neural inverse design. *arXiv preprint arXiv:2109.12888*, 2021.

[3] Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJed6j0cKX`.

[4] José M Bernardo and Adrian FM Smith. *Bayesian theory*, volume 405. John Wiley & Sons, 2009.

[5] Yongda Chen, Roy S Berns, and Lawrence A Taplin. Six color printer characterization using an optimized cellular yule-nielsen spectral neugebauer model. *Journal of Imaging Science and Technology*, 48(6):519–528, 2004.

[6] Brian Christian and Tom Griffiths. *Algorithms to live by: The computer science of human decisions*. Macmillan, 2016.

[7] Matteo Cianchetti, Cecilia Laschi, Arianna Menciassi, and Paolo Dario. Biomedical applications of soft robotics. *Nature Reviews Materials*, 3(6):143–153, 2018.

[8] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6 (2):182–197, 2002.

[9] John S Denker and Yann LeCun. Transforming neural-net output levels to probability distributions. In *Proceedings of the 3rd International Conference on Neural Information Processing Systems*, pages 853–859, 1990.

[10] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[11] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[12] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

[13] Konstantinos Gavriil, Ruslan Guseinov, Jesús Pérez, Davide Pellis, Paul Henderson, Florian Rist, Helmut Pottmann, and Bernd Bickel. Computational design of cold bent glass façades. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020.

[14] Roger David Hersch and Frédérique Crété. Improving the yule-nielsen modified neugebauer model by dot surface coverages depending on the ink superposition conditions. In *Color Imaging X: Processing, Hardcopy, and Applications*, volume 5667, pages 434–447. SPIE, 2005.

[15] Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[16] Jiaqi Jiang, Mingkun Chen, and Jonathan A Fan. Deep neural networks for the evaluation and design of photonic devices. *Nature Reviews Materials*, 6(8):679–700, 2021.

[17] Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. Hands-on bayesian neural networks–a tutorial for deep learning users. *arXiv preprint arXiv:2007.06823*, 2020.

[18] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

[19] Yashar Kiarashinejad, Sajjad Abdollahramezani, and Ali Adibi. Deep learning approach based on dimensionality reduction for designing electromagnetic nanostructures. *npj Computational Materials*, 6(1):1–12, 2020.

[20] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[21] Jakob Kruse, Lynton Ardizzone, Carsten Rother, and Ullrich Köthe. Benchmarking invertible architectures on inverse problems. *arXiv preprint arXiv:2101.10763*, 2021.

[22] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

[23] Dianjing Liu, Yixuan Tan, Erfan Khoram, and Zongfu Yu. Training deep neural networks for the inverse design of nanophotonic structures. *ACS Photonics*, 5(4):1365–1369, 2018.

[24] Wei Ma, Feng Cheng, Yihao Xu, Qinlong Wen, and Yongmin Liu. Probabilistic representation and inverse design of metamaterials based on a deep generative model with semi-supervised learning strategy. *Advanced Materials*, 31(35):1901111, 2019.

[25] David JC MacKay. Probable networks and plausible predictions-a review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 6(3):469, 1995.

[26] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors. *Machine Learning: An Artificial Intelligence Approach, Vol. I.* Tioga, Palo Alto, CA, 1983.

[27] Peter Morovič, Ján Morovič, Jordi Arnabat, and Juan Manuel García-Reyero. Revisiting spectral printing: A data driven approach. In *Color and Imaging Conference*, volume 2012, pages 335–340. Society for Imaging Science and Technology, 2012.

[28] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[29] David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

[30] Simiao Ren, Willie Padilla, and Jordan Malof. Benchmarking deep inverse models over time, and the neural-adjoint method. *arXiv preprint arXiv:2009.12919*, 2020.

[31] Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic, and Georg Martius. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. *arXiv preprint arXiv:2203.09168*, 2022.

[32] Liang Shi, Vahid Babaei, Changil Kim, Michael Foshey, Yuanming Hu, Pitchaya Sitthi-Amorn, Szymon Rusinkiewicz, and Wojciech Matusik. Deep multispectral painting reproduction via multi-layer, custom-ink printing. *ACM Trans. Graph.*, 37(6):271:1–271:15, December 2018.

[33] Xingyuan Sun, Tianju Xue, Szymon Rusinkiewicz, and Ryan P Adams. Amortized synthesis of constrained configurations using a differentiable surrogate. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=wdIDt--oLmV`.

[34] Shoji Tominaga. Color control using neural networks and its application. In *Color Imaging: Device-Independent Color, Color Hard Copy, and Graphic Arts*, volume 2658, pages 253–260. International Society for Optics and Photonics, 1996.

[35] Hsiao-Yu Fish Tung, Adam W Harley, William Seto, and Katerina Fragkiadaki. Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4364–4372. IEEE, 2017.

[36] Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning*, pages 9690–9700. PMLR, 2020.

[37] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. *Advances in neural information processing systems*, 28, 2015.

[38] Tianju Xue, Alex Beatson, Sigrid Adriaenssens, and Ryan Adams. Amortized finite element analysis for fast pde-constrained optimization. In *International Conference on Machine Learning*, pages 10638–10647. PMLR, 2020.

[39] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...

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

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

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

Appendix

A Standard error calculation, tuning, and implementation details

In this section we provide more details on different inverse methods we use in the paper. As many of the inverse methods we use in the paper have a stochastic component, we perform all following experiments 3 times and report the standard error. Since we have a budget of 5 runs for tuning the hyperparameters of `Autoinverse` methods, we allocate similar or higher resources for other methods in order to make the comparisons fair.

`NA` To avoid local minima, we run 50 *solves* of inversion with random initialization (each having up to 2000 iterations). As an alternative to tuning, we run `NA` 5 times and report the best NFP error. To accelerate the inversion in this paper we perform batch optimization. Thus, we can increase the number of target samples (up to GPU memory limit) without impacting the inversion time noticeably.

`NA ensemble` The configurations for `NA ensemble` is similar to `NA` except for the forward model. Unlike `NA`, we have an ensemble of networks ($\mathbb{F}_\mu$) that comprises the forward model of `NA ensemble`:

$$\mathbb{F}_\mu(\mathbf{x}) := \frac{1}{M} \sum_m \mu_m(\mathbf{x}). \tag{11}$$

Unlike `UANA`, single networks in the ensemble are incapable of predicting uncertainty. The cost function for `NA ensemble` is therefore defined as:

$$\mathcal{L}^{NA_{en}}(\mathbf{x}) := \arg\min_x \|\mathbb{F}_\mu(\mathbf{x}) - \mathbf{y}^*\|_2^2 \tag{12}$$

Similar to `UANA`, the back propagation is based on an ensemble of gradients coming from all the single networks in the ensemble (Figure 5).

$$\mathbf{x}^z = \mathbf{x}^{z-1} - \delta \sum_{m=1}^{M} \left(\frac{\partial \mathcal{L}^{NA_{en}}}{\partial \hat{f}_m} \times \frac{\partial \hat{f}_m}{\partial \mathbf{x}}\right) \tag{13}$$

Similar to `NA`, we run `NA ensemble` 5 times and pick the best results with respect to the NFP error.

`tandem` Since `tandem` does not explicitly possess any hyperparameter, instead of tuning, we evaluate 5 pre-trained inverse models (each initialized differently) and select the one that has the best NFP error on 10% of the target data. We then perform the inversion on the remaining target data using the best-performing inverse model.

`UANA` **and** `UA-tandem` In Section 4.5 of paper we showed how using a diverse range of activation functions in the ensemble network improves the quality of epistemic uncertainty and consequently the results of the inversion. For training the surrogates used in `UANA`, `UA tandem` and `NA ensemble` we use 10 networks in the ensemble with the following activations $\mathbb{F}_\mu$:

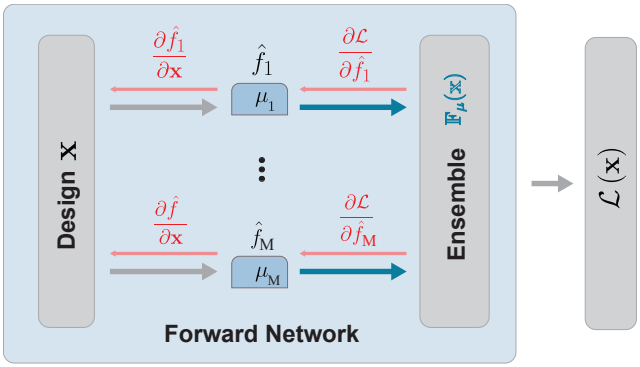


Figure 5: NA ensemble architecture

Table 3: Training details of different neural surrogate models used in inverse methods for `multi-joint robot`.

Network's name	Sub-networks name	Trainable parameters	Layer configuration
`INN`	-	3727416	[30, 3]
`NA`	-	3204302	100, 1000, 1500, 1000, 100
`NA ensemble`	Forward networks	351802×10	100, 500, 500, 100
`UANA`	μ networks	351802×10	100, 500, 500, 100
	σ networks	20902×10	100, 100, 100
`Tandem`	Forward network	3204302	100, 1000, 1500, 1000, 100
	Inverse network	113804	100, 250, 250, 100
`UA-tandem`	μ networks	351802×10	100, 500, 500, 100
	σ networks	20902×10	100, 100, 100
	Inverse network	117108	100, 250, 250, 100
`MINI`	-	10802	100, 100

- Tanh $\times 2$
- ReLU $\times 2$
- CELU $\times 2$
- LeakyReLU $\times 2$
- ELU
- Hardswish

We use ReLU activation functions for all other methods.

`INN` For each one of the 1000 target performances we randomly sample the latent space of the INN architecture [30] 1024 times. Thus, we end up with 1024 designs (for a single target) and evaluate all designs on the NFP and report the best error as the NFP error. For the surrogate error, we report the average forward loss.

`MINI` `MINI` is based on a mixed-integer optimization which is capable of finding the globally optimum solution [2]. This method is deterministic and every run returns the same solution, as a result we do not report the standard error for `MINI`.

Hardware To have a fair comparison we run all the methods on the same GPU machine for evaluating time performances. We used an NVIDIA TITAN X GPU for time evaluation. For other evaluations we used a GPU cluster. Training the forward models is trivially parallelizable. Moreover, we can parallelize 50 iterations of `NA` and `UANA` and aggregate the data in a post-processing step and choose the best results *based on the surrogate error*. Nevertheless, we are reporting our computation time assuming both training and inversion are performed serially on a single GPU.

B Training details for neural surrogate models in `multi-joint robot` (Table 1 in paper)

In Table 3 and Table 4, we can compare the capacity, training time, and accuracy of the neural network surrogates used for the inversion of `multi-joint robot`. We keep similar training capacity for all methods except `MINI`. `MINI` uses a combinatorial optimization and is not scalable to large networks [2]. We trained a smaller surrogate for `MINI` but at the same time we monitored its training loss to lie within a reasonable range.

C Details for 'Neural inversion in the presence of imperfect data'

C.1 Counterpart results of `tandem` and `UA-tandem` for `spectral printer`

Table 5 presents the results of `spectral printer`, similar to Section 4.3 (Table 2) of the paper but comparing `tandem` and `UA-tandem`. As evident from the table, we obtain similar performance gain

Table 4: [Continued] Training details of different neural surrogate models used in inverse methods for `multi-joint robot`.

Network's name	Sub-networks name	Total training time (s)	Total inversion time (s)	Training loss
`INN`	-	-	$1024 \times (1.3 \times 10^{-2})$	2.10×10^{-2}
`NA`	-	190	374	3.33×10^{-5}
`NA ensemble`	single	1650	426	3.91×10^{-6}
`UANA`	μ networks	1650	1075	2.34×10^{-6}
	σ networks	192		
`Tandem`	Forward network	190	3.8×10^{-3}	3.33×10^{-5}
	Inverse network	181		
`UA-tandem`	μ networks	1650	3.8×10^{-3}	2.34×10^{-6}
	σ networks	192		
	Inverse network	1109		
`MINI`	-	150	2.98×10^{4}	4.89×10^{-4}

Table 5: The distribution of ink densities (≥ 0.4) after the inversion of `spectral printer` using `UA-tandem`. Once we insert noise into LC channel or sample it sparsely, `Autoinverse` detects and avoids it. STD has been rounded to nearest integer.

Model	Data set	NFP error	C	M	Y	K	LC	LM	LK	LLK
`UA-tandem`	Standard	$(2.62 \pm 0.488) \times 10^{-3}$	180 ± 12	48 ± 2	13 ± 4	3 ± 1	174 ± 8	36 ± 12	0 ± 0	0 ± 0
	Sparse	$(2.34 \pm 0.097) \times 10^{-3}$	291 ± 11	43 ± 0	15 ± 2	3 ± 1	$\mathbf{0 \pm 0}$	89 ± 16	0 ± 0	0 ± 0
	Noisy	$(5.16 \pm 0.423) \times 10^{-3}$	242 ± 2	63 ± 0	18 ± 1	1 ± 0	$\mathbf{0 \pm 0}$	64 ± 32	8 ± 1	29 ± 2
`tandem`	Standard	$(3.51 \pm 2.903) \times 10^{-2}$	159 ± 9	40 ± 10	0 ± 0	12 ± 9	255 ± 124	187 ± 256	5 ± 4	1070 ± 1514
	Sparse	$(2.18 \pm 0.840) \times 10^{-2}$	208 ± 22	36 ± 3	0 ± 0	17 ± 7	52 ± 30	20 ± 9	14 ± 19	0 ± 0
	Noisy	$(3.82 \pm 0.810) \times 10^{-2}$	174 ± 29	60 ± 32	39 ± 42	71 ± 9	303 ± 171	192 ± 239	114 ± 123	62 ± 74

when augmenting `tandem` with uncertainty awareness. Similar to `UANA`, `UA-tandem` has completely avoided the LC channel.

C.2 Soft robot actuation distribution

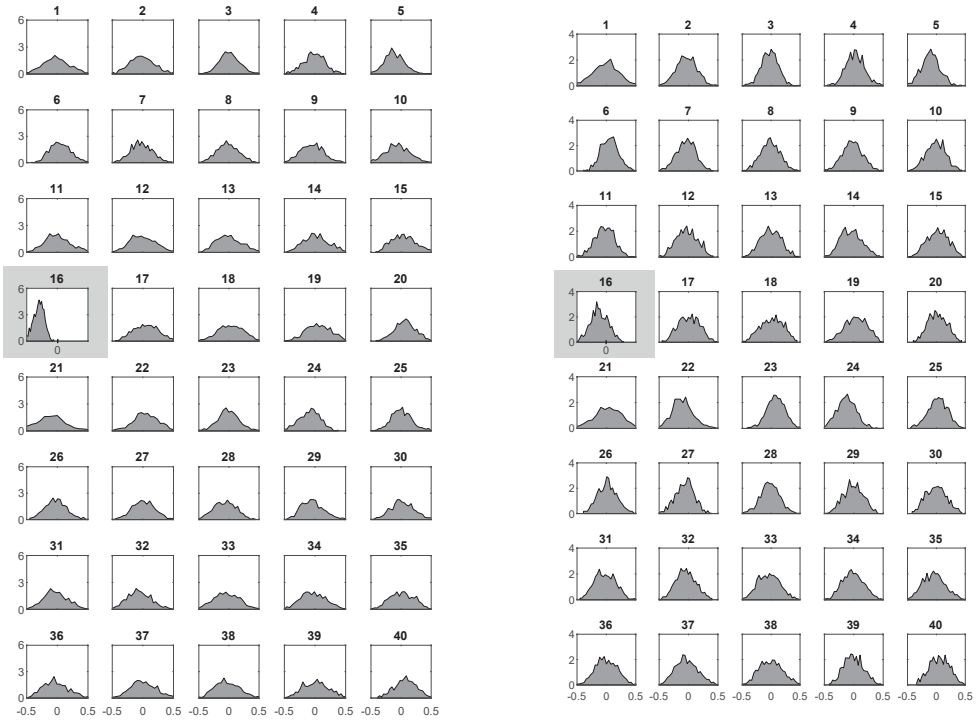
Figure 2 in the paper showed the actuation distribution of only 8 soft robot edges (Section 4.3 in paper). In Figure 6 we show the distribution of actuation for *all* edges computed by `UANA` on surrogates learned using partially noisy and partially sparse data. We repeat this experiment using `UA-tandem` to emphasize on the generality of `Autoinverse` (Figure 7).

C.3 Training details of surrogates used for `spectral printer` and `soft robot`

Tables 6 and 7 show the layer configuration, number of trainable parameters, the training and inversion time, and the training loss of different surrogate models used in `spectral printer`. Tables 8 and 9 show the layer configuration, number of trainable parameters, the training and inversion time, and the training loss of different surrogate models used in `soft robot`. From the tables we can observe that

Table 6: Training details of different neural surrogate models used in inverse methods for `spectral printer`.

Network's name	Sub-networks name	Trainable parameters	Layer configuration
`NA`	-	905931	100, 500, 800, 500, 100
`NA ensemble`	Forward networks	64531×10	100, 100, 200, 100, 100
`UANA`	μ networks	64531×10	100, 100, 200, 100, 100
	σ networks	24231×10	100, 100, 100
`Tandem`	Forward network	905931	100, 500, 800, 500, 100
	Inverse network	117108	100, 250, 250, 100
`UATandem`	μ networks	64531×10	100, 100, 200, 100, 100
	σ networks	24231×10	100, 100, 100
	Inverse network	117108	100, 250, 250, 100



(a) 16th edge is noisy in the positive actuation range.

(b) The training data does not contain any samples with positive actuation of the 16th edge.

Figure 6: The actuation distribution of all the edges for inversion on both noisy and sparse data via `UANA`.

Table 7: [Continued] Training details of different neural surrogate models used in inverse methods for `spectral printer`.

Network's name	Sub-networks name	Training time	Inversion time	Training loss
`NA`	-	295	300	4.45×10^{-6}
`NA ensemble`	Forward networks	441	563	3.38×10^{-6}
`UANA`	μ networks	441	1.15×10^{3}	3.44×10^{-6}
	σ networks	240		
`Tandem`	Forward network	295	1.07×10^{-2}	4.45×10^{-6}
	Inverse network	260		
	μ networks	441		
`UATandem`	σ networks	240	1.04×10^{-2}	3.44×10^{-6}
	Inverse network	1.08×10^{3}		

(a) 16th edge is noisy in the positive actuation range.

(b) The training data does not contain any samples with positive actuation of the 16th edge.

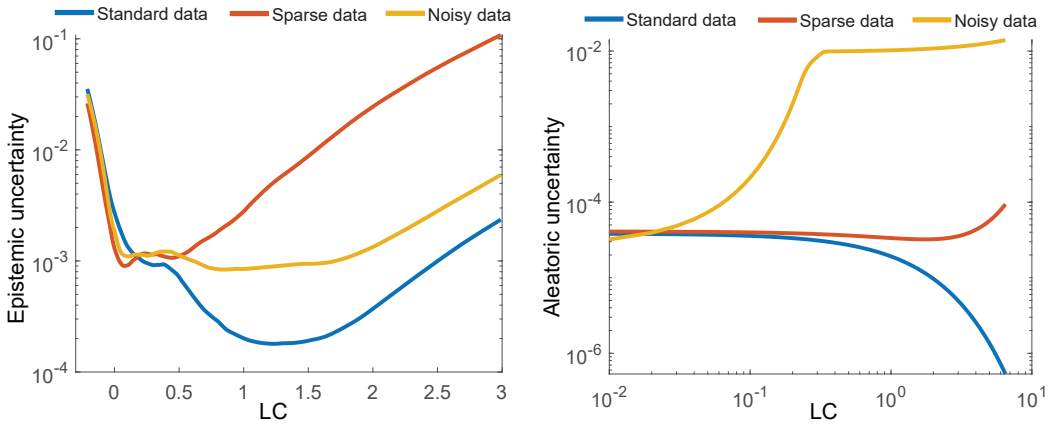
Figure 7: The actuation distribution of all the edges for inversion on both noisy and sparse data via `UA-tandem`.

Table 8: Training details of different neural surrogate models used in inverse methods for `soft robot`.

Network's name	Sub-networks name	Trainable parameters	Layer configuration
`NA`	-	45506206	2000, 5000, 5000, 2000
`UANA`	μ networks	3227606×10	300, 1500, 1500, 300
	σ networks	45106×10	100, 100, 100
`Tandem`	Forward network	45506206	2000, 5000, 5000, 2000
	Inverse network	3227440	300, 1500, 1500, 300
`UATandem`	μ networks	3227606×10	300, 1500, 1500, 300
	σ networks	45106×10	100, 100, 100
	Inverse network	3227440	300, 1500, 1500, 300

Table 9: [Continued] Training details of different neural surrogate models used in inverse methods for `soft robot`.

Network's name	Sub-networks name	Total training time (s)	Total inversion time (s)	Training loss
`NA`	-	5428	2250	2.63×10^{-4}
`UANA`	μ networks	16940	2950	2.39×10^{-5}
	σ networks	16290		
`Tandem`	Forward network	5428	2.90×10^{-1}	2.63×10^{-4}
	Inverse network	1740		
`UATandem`	μ networks	16940		
	σ networks	16290	1.65×10^{-1}	2.39×10^{-5}
	Inverse network	9458		



(a) (b)

Figure 8: The landscape of the aleatory and epistemic uncertainty.

the training capacity of different surrogates models is comparable. Furthermore, the training accuracy of all models is similar.

C.4 Epistemic and aleatoric loss behaviour for spectral printing experiment

Epistemic uncertainty The key to handling design feasibility is the epistemic uncertainty (Equation 3(c) paper). We know that the scarcity of data results in higher epistemic uncertainty [18]. On the other hand, by definition, we do not have any infeasible or out-of-range data points in our dataset. Hence, if we query a network with an infeasible or out-of-range input, we will get high uncertainty for the prediction. We use this trend to avoid such samples in the inversion. Figure 8(a) shows the trend of the epistemic uncertainty values for `spectral printer` (Section 4.3, Table 2 in paper). For that experiment, we ran the inversion using `UANA` on 3 different datasets: standard, noisy, and sparse and observed how the problematic ink channel (LC) is avoided. Figure 8(a) demonstrates *why* that ink channel is avoided. In Figure 8(a), we set all ink channels except LC to 0 while increasing the values of LC ink density from 0 (the x-axis of the plot).

As expected, for all three datasets moving away from the feasible region (between 0 and 1) increases the epistemic uncertainty. When trained for the sparse data (red curve), where the LC channel has not been sampled after 0.4, the epistemic uncertainty starts to increase earlier.

Outside the feasible region, each network in the ensemble has to extrapolate as it has not been trained in those regions. Consequently, the predictions of ensemble networks diverge. The divergence of the networks increases the epistemic uncertainty and, during the inversion, the uncertainty aware methods can reject solutions in these regions.

Aleatoric uncertainty Figure 8(b) demonstrates the behavior of the aleatoric uncertainty of the surrogate used for the same experiment (`UANA` on `spectral printer`). Similarly, to generate the plots, we set all the ink channels to 0 and change the values of the Light Cyan ink densities. As evident from Figure 8(b), the level of uncertainty increases significantly for the noisy dataset, while for sparse and standard datasets aleatoric uncertainty is at least two orders of magnitude smaller. The increase of aleatoric uncertainty for the noisy data helps `UANA` avoid any samples from those regions (Tables 2 in the paper and Table 5).

D Details for 'Autoinverse brings AutoML to neural inversion'

Spectral printer Complementary to Figure 4 of the paper, in Figure 9 we compare the inversion performance using a diverse range of inverse methods. Here we clearly see that basic methods, such as `NA` and `tandem` fail spectacularly in computing feasible designs.

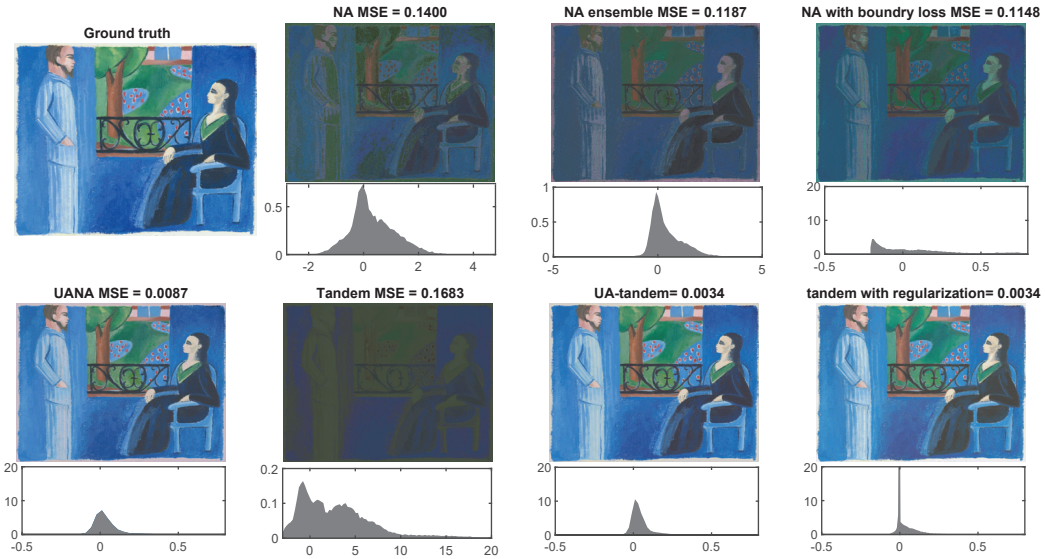


Figure 9: Spectral reproduction of a ground-truth painting using different inverse methods (`spectral printer`). Apart form the reproduction quality, we show the distribution of the computed *designs*, i.e., ink densities. Note that the feasible ink density range is $[0, 1]$.

Boundary loss is a semi-generic regularization, suitable for handling box constraints [30]:

$$\mathcal{L}_{bnd} = \mathrm{Re}\,LU\left(|x - \mu_x| - \frac{1}{2}R_x\right) \tag{14}$$

where R_x is the value range of design samples in the dataset and μ_x is their average. Incorporating the boundary loss in `NA` in Figure 9 results in an improvement in the distribution of the ink intensities. However, the results are still far from acceptable.

Note how applying hand-crafted regularization (ink intensity regularization) [32] improves the quality of `tandem` significantly (`tandem with regularization` in Figure 9). `UANA` and `UA-tandem` however perform comparably without any regularization.

Soft robot objective for the Neural Adjoint method The objective function for `soft robot` inversion comprises of two terms, one is responsible for bringing the tip of the robot to the target and the other one ($\mathcal{R}(\mathbf{x}^{in})$) guarantees the deformations to remain physical [33].

$$\mathcal{L}(\mathbf{x}^{in}) := \left\|\mathbf{x}_i^{out} - \mathbf{t}\right\|_1 + \lambda \cdot \mathcal{R}(\mathbf{x}^{in}) \qquad i \in [123,\ 124]\,, \tag{15}$$

$$\mathcal{R}(\mathbf{x}^{in}) := \sum_{\substack{1<i<n, i\neq n/2,\\ i\neq n/2+1}} \left(\frac{\mathbf{x}_{i+1}^{in} - \mathbf{x}_i^{in}}{2} - \frac{\mathbf{x}_i^{in} - \mathbf{x}_{i-1}^{in}}{2}\right)^2. \tag{16}$$

where $\mathbf{t}$ represents the target location and $\mathbf{x}_i^{out}$ represents the position of all 206 vertices of soft robot, among which $i \in [123,\ 124]$ represent the position of its tip. Also, λ adjusts the importance of the smoothness term and $\mathcal{R}(\mathbf{x}^{in})$ regulates the actuation of the flexible edges ($\mathbf{x}^{in}$) to insure that the deformation of the robot is physical.

Sensitivity to initialization (`soft robot`) In Section 4.4 of the paper, we learned how hand-crafted regularization improves the quality of the designs. Despite having regularization, `soft robot` inversion using `NA` fails when initialized with values far from feasible region. This is evident from both the irregular robot shapes (Figure 3(b) in the paper) and the distribution of the actuation (Figure 10) computed using **regularized** `NA` but with *wrong* initialization. At the same time, `UANA` without any regularization and with a wrong initialization produces plausible robot shapes (Figure 3(b) in the paper) and actuation distribution (centered around 0). Interestingly, the smoothness score for

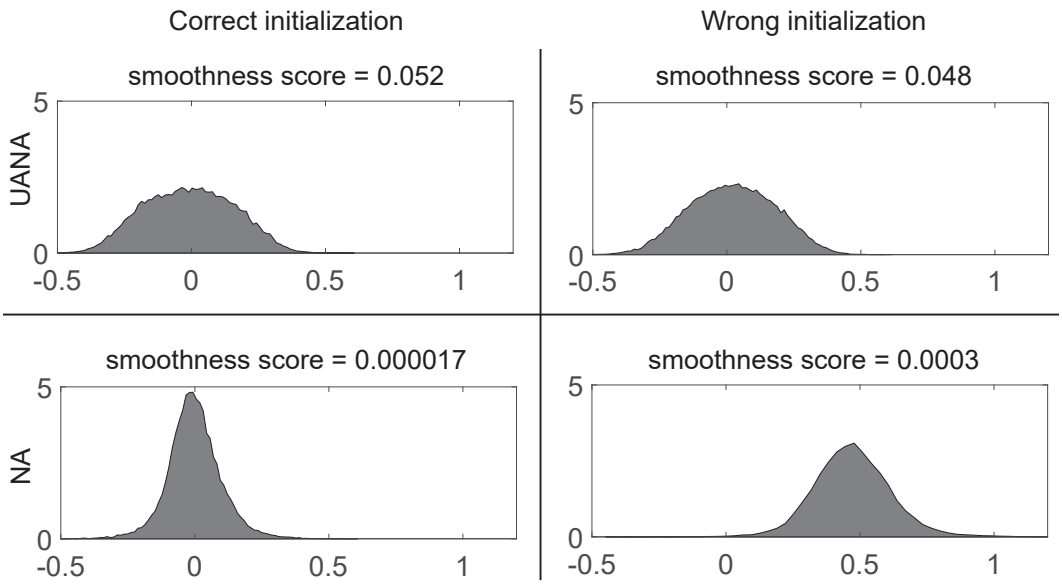
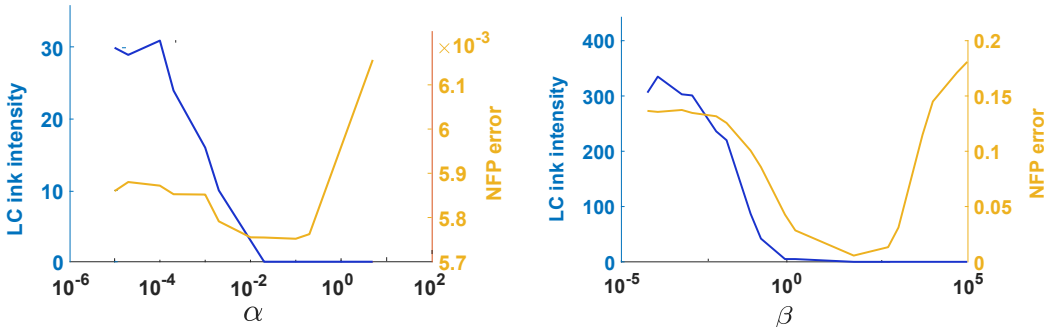


Figure 10: The effect of wrong initialization for inversion of `soft robot` using `NA` with regularization and two different initialization. Note the robustness of `UANA` without any form of regularization.



(a) (b)

Figure 11: Stability of `Autoinverse` within a wide range of its hyperparameters α and β.

clearly failed shapes obtained from `NA` with regularization, are for both initializations very small (Figure 10). This shows how even regularization can be multi-modal and fall into a wrong local minima and generate designs with nonphysical shapes (Figure 3(b)).

E Details for 'Ablation studies'

As discussed in Section 4.5 of the paper, the power of `UANA` lies in its uncertainty awareness and not the ensembling process. We include the result of `NA ensemble` for paitning reproduction in Figure 9. While we observe a marginal improvement of `NA ensemble` over `NA`, it is clearly outperformed by `UANA`.

Sensitivity to uncertainty weights `Autoinverse` is extremely stable when tuning its hyperparameters, i.e., uncertainty weights (α and β in Equation 3 in paper). This ensures that a light hyperparameter tuning is enough for obtaining reasonable results. We evaluate this behavior by using a wide range of α and β values spanning over 5 orders of magnitude. This ablation is performed using `UANA`. In the ablation of aleatoric (α) and epistemic (β) weight, we have used the noisy and epistemic data of `spectral printer`, respectively. When evaluating α we keep β fixed at the tuned value. Alternatively for the evaluation of β, α is constant at the tuned value.

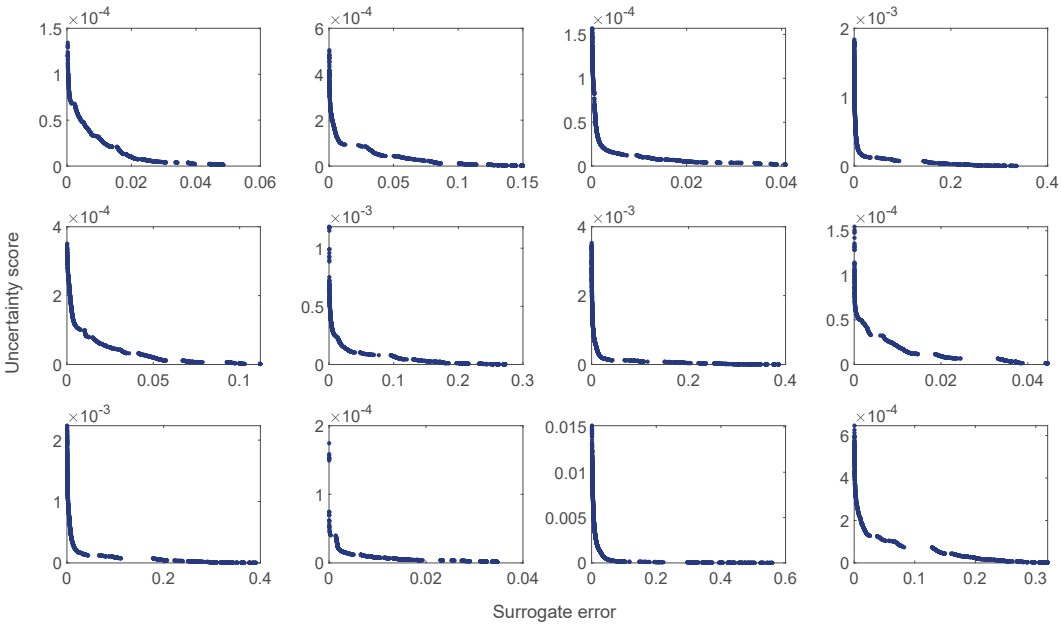


Figure 12: Pareto front for 12 randomly chosen targets.

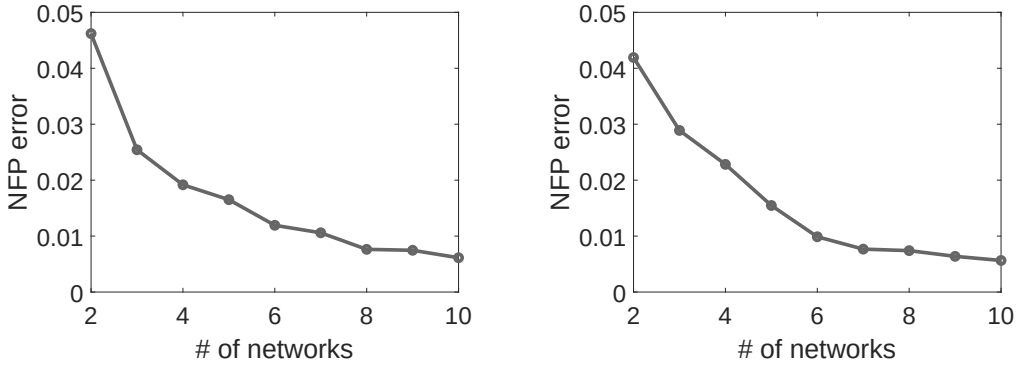
The most interesting fact about Figures 11(a) and 11(b), is the correlation of the α and β with the NFP error. This correlation means that adjusting the importance of these weights on the surrogate model directly improves the quality of the inversion in reality. Interestingly, we can observe the robustness of `Autoinverse` against the variation of α and β, such that for a range of around 3 orders of magnitude the NFP loss remains stable around a desirable value. The larger the weights, `Autoinverse` chooses less and less samples in the problematic regions (LC channel density larger than 0.4). This trend continues with very large uncertainty weights. However, these weights cannot be indefinitely increased as the MSE term of the objective (Equations 6 and 10 in the paper) will be undermined and inversion's NFP error increases.

F Pareto front of accuracy versus uncertainty

We calculate the Pareto front for 12 randomly chosen targets from the spectral printer experiment with the standard dataset (Figure 12). We use the NSGA II [8], an evolutionary algorithm that samples our forward BNN to discover the Pareto front iteratively. The uncertainty score in this experiment is the weighted sum of aleatoric and epistemic uncertainty. We set the values of the weights on the tuned values on the inversion task. The population size and the number of generations in this experiment are 1000 and 100, respectively. Figure 12 suggests that the losses of uncertainty aware inversion are conflicting such that for example reducing the MSE loss will lead to the increase of the uncertainty score.

G Ablation of the number of networks in the ensemble

In this experiment we investigate the importance of the accuracy of the calculated uncertainties on the final NFP error. We have trained deep ensemble networks with a varying number of networks in the ensemble. The networks are trained on the spectral printer experiment for both noisy and sparse datasets. As evident in Figure 13, by increasing the number of networks in the ensemble the NFP error improves significantly. This trend indicates the importance of accurate prediction of the uncertainties.



(a) The effect of increasing the number of networks in the ensemble trained on the noisy data.

(b) The effect of increasing the number of networks in the ensemble trained on the sparse data.

Figure 13: Evaluating the inversion performance using Deep Ensembles with different number of sub-networks.

H Implementation

In practice, training the ensemble networks directly with negative log likelihood loss (Equation 8) is challenging [31]. Instead, following [29], we take a 3-step procedure for implementing deep ensemble predictive uncertainty. We start with training an ensemble of conventional networks with diverse activation functions and MSE as its loss. These networks are in fact the initialization of $\mathbb{F}_\mu(\cdot)$. The next step is training $\mathbb{F}_\sigma(\cdot)$ and fine tuning $\mathbb{F}_\mu(\cdot)$ jointly with the negative log likelihood loss (Equation 8). Finally, we replace $\hat{f}(\cdot)$ with $\mathbb{F}_\mu(\cdot)$ and incorporate $\mathbb{F}_\sigma(\cdot)$ in the `Autoinverse` loss.