Design of Loss Functions for Solving Inverse Problems using Deep Learning

Jon Ander Rivera[1,2], David Pardo[1,2,3], and Elisabete Alberdi[1]

[1] University of the Basque Country (UPV/EHU), Leioa, Spain
[2] BCAM-Basque Center for Applied Mathematics, Bilbao, Spain
[3] IKERBASQUE, Basque Foundation for Science, Bilbao, Spain

Abstract. Solving inverse problems is a crucial task in several applications that strongly affect our daily lives, including multiple engineering fields, military operations, and/or energy production. There exist different methods for solving inverse problems, including gradient based methods, statistics based methods, and Deep Learning (DL) methods. In this work, we focus on the latest. Specifically, we study the design of proper loss functions for dealing with inverse problems using DL. To do this, we introduce a simple benchmark problem with known analytical solution. Then, we propose multiple loss functions and compare their performance when applied to our benchmark example problem. In addition, we analyze how to improve the approximation of the forward function by: (a) considering a Hermite-type interpolation loss function, and (b) reducing the number of samples for the forward training in the Encoder-Decoder method. Results indicate that a correct design of the loss function is crucial to obtain accurate inversion results.

Keywords: Deep Learning · Inverse Problems · Neural Network.

1 Introduction

Solving inverse problems [17] is of paramount importance to our society. It is essential in, among others, most areas of engineering (see, e.g., [3, 5]), health (see, e.g. [1]), military operations (see, e.g, [4]) and energy production (see, e.g. [11]). In multiple applications, it is necessary to perform this inversion in real-time. This is the case, for example, of geosteering operations for enhanced hydrocarbon extraction [2, 10].

Traditional methods for solving inverse problems include gradient based methods [13, 14] and statistics based methods (e.g., Bayesian methods [16]). The main limitation of these kind of methods is that they lack an explicit construction of the pseudo-inverse operator. Instead, they only *evaluate* the inverse function for a given set of measurements. Thus, for each set of measurements, we need to perform a new inversion process. This may be time consuming.

Deep Learning (DL) seems to be a proper alternative to overcome the aforementioned problem. With DL methods, we explicitly build the pseudo-inverse operator rather than only evaluating it. Recently, the interest on performing

inversion using DL techniques has grown exponentially (see, e.g., [9, 15, 18, 19]). However, the design of these methods is still somehow *ad hoc* and it is often difficult to encounter a comprehensive road map to construct robust Deep Neural Networks (DNNs) for solving inverse problems.

One major problem when designing DNNs is the error control. Several factors may lead to deficient results. Such factors include: poor loss function design, inadequate architecture, lack of convergence of the optimizer employed for training, and unsatisfactory database selection. Moreover, it is sometimes elusive to identify the specific cause of poor results. Even more, it is often difficult to asses the quality of the results and, in particular, determine if they can be improved.

In this work, we take a simple but enlightening approach to elucidate and design certain components of a DL algorithm when solving inverse problems. Our approach consists of selecting a simple inverse benchmark example with known analytical solution. By doing so, we are able to evaluate and quantify the effect of different DL design considerations on the inversion results. Specifically, we focus on analyzing a proper selection of the loss function and how it affects to the results. While more complex problems may face additional difficulties, those observed with the considered simple example are common to all inverse problems.

The remainder of this article is as follows. Section 2 describes our simple model inverse benchmark problem. Section 3 introduces several possible loss functions. Section 4 shows numerical results. Finally, Section 5 summarizes the main findings.

2 Simple inverse benchmark problem

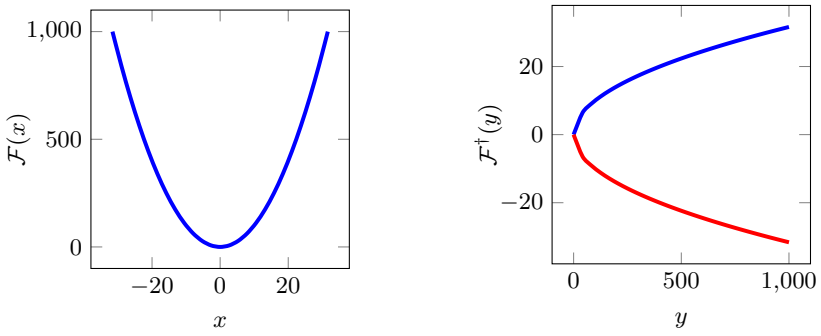
We consider a benchmark problem with known analytical solution. Let $\mathcal{F}$ be the forward function and $\mathcal{F}^\dagger$ the pseudo-inverse operator. We want our benchmark problem to have more than one solution since this is one of the typical features exhibited by inverse problems. For that, we need $\mathcal{F}$ to be non-injective. We select the non-injective function $y = \mathcal{F}(x) = x^2$, whose pseudo-inverse has two possible solutions: $x = \mathcal{F}^\dagger(y) = \pm\sqrt{y}$. (See Figure 1). The objective is to design a NN that approximates one of the solutions of the inverse problem.

2.1 Database and data rescaling

We consider the domain $\Omega = [-33, 33]$. In there, we select a set of 1000 equidistant numbers. The corresponding dataset of input-output pairs $\{(x_i, \mathcal{F}(x_i))\}_{i=1}^{1000}$ is computed analytically.

In some cases, we perform a change of coordinates in our output dataset. Let's name $\mathcal{R}$ the linear mapping that goes from the output of the original dataset into the interval [0,1]. Instead of approximating function $\mathcal{F}$, our NN will approximate function $\mathcal{F}^{\mathcal{R}}$ given by

$$\mathcal{F}^{\mathcal{R}} := \mathcal{R} \circ \mathcal{F}. \tag{1}$$



(a) Forward solution. (b) Inverse solutions.

Fig. 1: Benchmark problem.

In the cases we perform no rescaling, we select $\mathcal{R} = \mathcal{I}$, where $\mathcal{I}$ is the identity mapping.

3 Loss functions

We consider different loss functions. The objective here is to discern between adequate and poor loss functions for solving the proposed inverse benchmark problem.

We denote as $\mathcal{F}_\varphi$ and $\mathcal{F}_\theta^\dagger$ the NN approximations of the forward function and the pseudo-inverse operator, respectively. Weights φ and θ are the parameters to be trained (optimized) in the NN. Each value within the set of weights is a real number.

In a NN, we try to find the weights φ^* and θ^* that minimize a given loss function L. We express our problem mathematically as

$$(\varphi^*, \theta^*) = \arg\min_{\varphi,\theta} L(\varphi, \theta). \tag{2}$$

Loss based on the missfit of the inverse data: We first consider the traditional loss function:

$$L_1(\theta) = \left\| \mathcal{F}_\theta^{\dagger\,\mathcal{R}}(y) - x \right\|. \tag{3}$$

Theorem 1. *Solution of minimization problem* (2) *with the loss function given by Eq.* (3) *has analytical solution for our benchmark problem in both the* l_1 *norm*

$$||x||_1 = \sum_{i=1}^{n} |x_i| \tag{4}$$

and the l_2 *norm*

$$||x||_2 = \sum_{i=1}^{n} (x_i)^2. \tag{5}$$

These solutions are such that:

- *For* l_1, ${\mathcal{F}_{\theta^*}^{\dagger}}^{\mathcal{R}}(y) = x$, *where* x *is any value in the interval* $[-\sqrt{y}, \sqrt{y}]$.
- *For* l_2, ${\mathcal{F}_{\theta^*}^{\dagger}}^{\mathcal{R}}(y) = 0$.

Proof. We first focus on norm $\|\cdot\|_1$. We minimize the loss function:

$$\sum_{i \in I} |{\mathcal{F}_{\theta}^{\dagger}}^{\mathcal{R}}(y_i) - x_i|, \tag{6}$$

where $I = \{1, ..., N\}$ denotes the training dataset. For the exact pseudo-inverse operator ${\mathcal{F}_{\theta}^{\dagger}}^{\mathcal{R}}$, we can express each addend of (6) as follows:

$$|{\mathcal{F}_{\theta}^{\dagger}}^{\mathcal{R}}(y_i) - x_i| = \begin{cases} -2x_i, \text{ if } x_i \leq -\sqrt{y_i}, \\ 0, \quad \text{if } -\sqrt{y_i} \leq x_i \leq \sqrt{y_i}, \\ 2x_i, \quad \text{if } x_i \geq \sqrt{y_i}. \end{cases} \tag{7}$$

Taking the derivative of Eq. (6) with respect to x_i, we see in view of Eq. (7) that the loss function for the exact solution attains its minimum at every point $x_i \in [-\sqrt{y_i}, \sqrt{y_i}]$.

In the case of norm $\|\cdot\|_2$, for each value of y we want to minimize:

$$\sum_{i \in I} \left({\mathcal{F}_{\theta}^{\dagger}}^{\mathcal{R}}(y_i) - x_i \right)^2. \tag{8}$$

Again, for the exact pseudo-inverse operator ${\mathcal{F}_{\theta}^{\dagger}}^{\mathcal{R}}$, we can express each addend of Eq. (8) as:

$$(-\sqrt{y_i} - x_i)^2 + (\sqrt{y_i} - x_i)^2. \tag{9}$$

Taking the derivative of Eq. (8) with respect to x_i and equaling it to zero, we obtain:

$$2x_i + \sqrt{y_i} - \sqrt{y_i} = 0 \Rightarrow x_i = \frac{\sqrt{y_i} - \sqrt{y_i}}{2} = 0. \tag{10}$$

Thus, the function is minimized when the approximated value is 0. □

Observation: Problem of Theorem 1 has infinite solutions in the l_1 norm. In the l_2 norm, the solution is unique; however, it differs from the two desired exact inverse solutions.

Loss based on the missfit of the effect of the inverse data: As seen with the previous loss function, it is inadequate to look at the misfit in the inverted space. Rather, it is desirable to search for an inverse solution such that after applying the forward operator, we recover our original input. Thus, we consider the following modified loss function, where $\mathcal{F}^{\mathcal{R}_1}$ corresponds to the analytic forward function:

$$L_2(\theta) = \left\| \mathcal{F}^{\mathcal{R}_1}({\mathcal{F}_{\theta}^{\mathcal{R}_2}}^{\dagger}(y)) - y \right\|. \tag{11}$$

Unfortunately, computation of $\mathcal{F}^{\mathcal{R}_1}$ required in L_2 involves either (a) implementing $\mathcal{F}^{\mathcal{R}_1}$ in a GPU, which may be challenging in more complex examples, or (b) calling $\mathcal{F}^{\mathcal{R}_1}$ as a CPU function multiple times during the training process. Both options may considerably slow down the training process up to the point of making it impractical.

Encoder-Decoder loss: To overcome the computational problems associated with Eq. (11), we introduce an additional NN, named $\mathcal{F}_{\varphi}^{\mathcal{R}_1}$, to approximate the forward function. Then, we propose the following loss function:

$$L_3(\varphi,\theta) = \left\|\mathcal{F}_{\varphi}^{\mathcal{R}_1}(x) - y\right\| + \left\|\mathcal{F}_{\varphi}^{\mathcal{R}_1}(\mathcal{F}_{\theta}^{\dagger\,\mathcal{R}_2}(y)) - y\right\|. \tag{12}$$

Two NNs of this type that are being simultaneously trained are often referred to as Encoder-Decoder [6, 12].

Two-steps loss: It is also possible to train $\mathcal{F}_{\varphi}^{\mathcal{R}_1}$ and $\mathcal{F}_{\theta}^{\dagger\,\mathcal{R}_2}$ separately. By doing so, we diminish the training cost. At the same time, it allows us to separate the analysis of both NNs, which may simplify the detection of specific errors in one of the networks. Our loss functions are:

$$L_{4.1}(\varphi) = \left\|\mathcal{F}_{\varphi}^{\mathcal{R}_1}(x) - y\right\| \tag{13}$$

and

$$L_{4.2}(\theta) = \left\|\mathcal{F}_{\varphi^*}^{\mathcal{R}_1}(\mathcal{F}_{\theta}^{\dagger\,\mathcal{R}_2}(y)) - y\right\|. \tag{14}$$

We first train $\mathcal{F}_{\varphi}^{\mathcal{R}_1}$ using $L_{4.1}$. Once $\mathcal{F}_{\varphi}^{\mathcal{R}_1}$ is fixed (with weights φ^*), we train $\mathcal{F}_{\theta}^{\dagger\,\mathcal{R}_2}$ using $L_{4.2}$.

4 Numerical Results

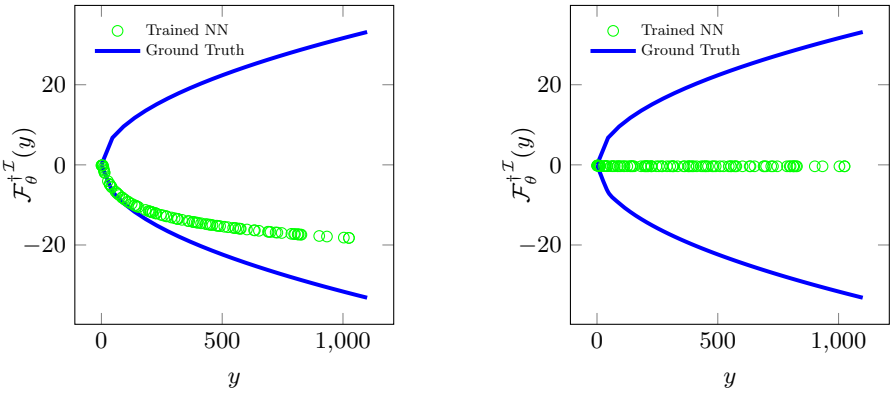
We consider two different NNs. The one approximating the forward function has 5 fully connected layers [8] with ReLU activation function [7]. The one approximating the inverse operator has 11 fully connected layers with ReLU activation function. ReLU activation function is defined as

$$\mathrm{ReLu}(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0. \end{cases} \tag{15}$$

These NN architectures are "overkilling" for approximating the simple benchmark problem studied in this work. Moreover, we also obtain results for different NN architectures, leading to identical conclusions that we omit here for brevity.

4.1 Loss function analysis

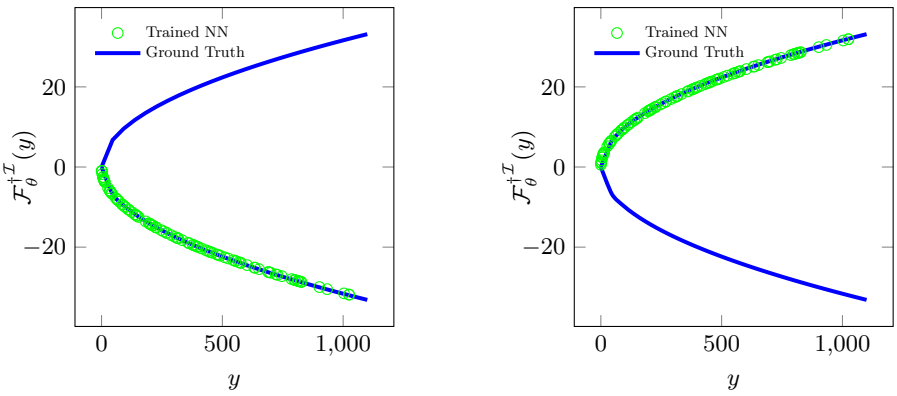
Loss based on the missfit of the inverse data: We produce two models using norms l_1 and l_2, respectively. Figure 2 shows the expected disappointing results (see Theorem 1). The approximated NN values (green circles) are far from the true solution (blue line). From an engineering point of view, the recovered solution is worthless. The problem resides on the selection of the loss function.



(a) Norm l_1. (b) Norm l_2.

Fig. 2: Predicted ($\mathcal{F}_\theta^{\mathcal{I}\dagger}$, green circles) vs exact ($\mathcal{F}^{\mathcal{I}\dagger}$, blue line) inverse solutions evaluated over the testing dataset.

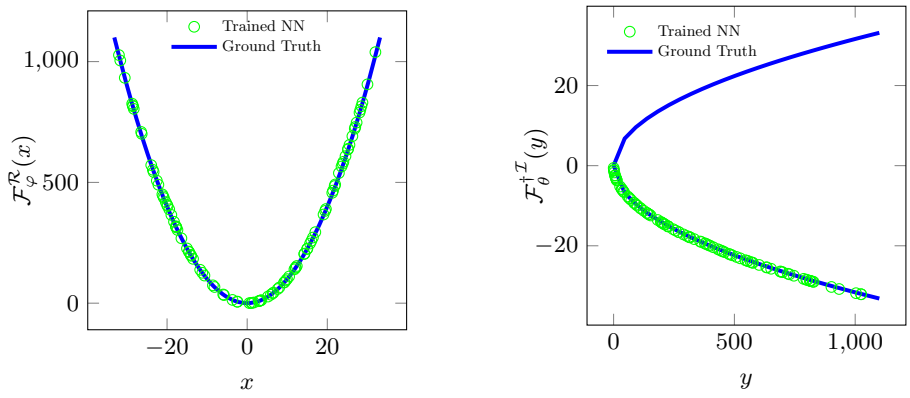
Loss based on the missfit of the effect of the inverse data: Figure 3 shows the real values of y (ground truth) vs their predicted pseudo-inverse values. The closer the predicted values are to the blue line, the better the result from the NN. We now observe an excellent match between the exact and approximated solutions. However, as mention in Section 3, this loss function entails essential limitations when considering complex problems.



(a) Norm l_1. (b) Norm l_2.

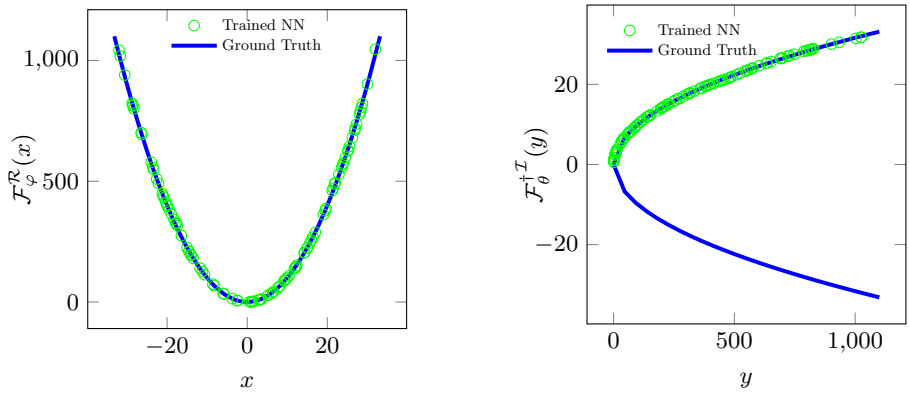
Fig. 3: Solution of the pseudo-inverse operator approximated by the NN.

Encoder-Decoder loss: Figure 4 shows the results for norm l_1 and Figure 5 for norm l_2. We again recover excellent results, without the limitations provided by loss function L_2. Coincidentally, different norms recover different solution branches of the inverse problem. Note that in this problem, it is possible to prove that the probability of recovering either of the solution branches is identical.



(a) Forward function. (b) Pseudo-inverse operator.

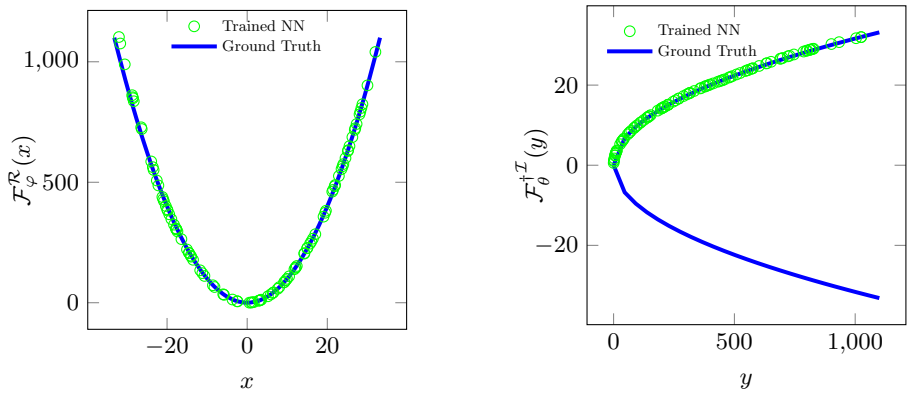
Fig. 4: Exact vs NN solutions using loss function L_3 and norm l_1.



(a) Forward function. (b) Pseudo-inverse operator.

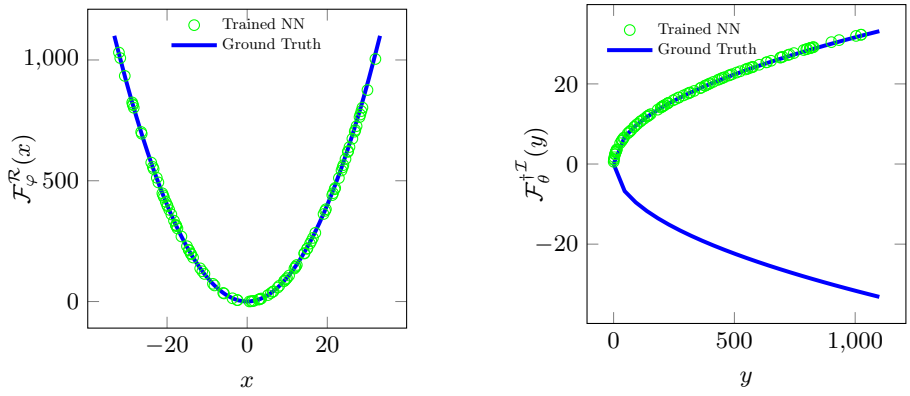
Fig. 5: Exact vs NN solutions using loss function L_3 and norm l_2.

Two-steps loss: Figures 6 and 7 show the results for norms l_1 and l_2, respectively. The approximations of forward function and pseudo-inverse operator are accurate in both cases.



(a) Forward function. (b) Pseudo-inverse operator.

Fig. 6: Exact vs NN solutions using loss functions $L_{4.1}$ and $L_{4.2}$ and norm l_1.



(a) Forward function. (b) Pseudo-inverse operator.

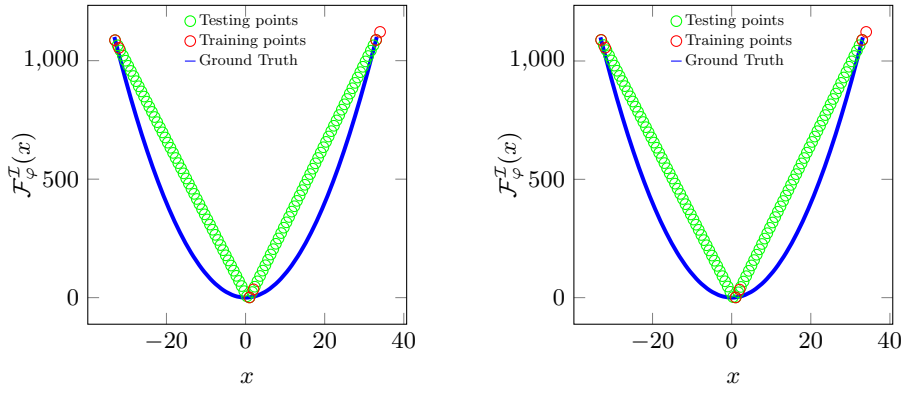
Fig. 7: Exact vs NN solutions using loss functions $L_{4.1}$ and $L_{4.2}$ and norm l_2.

4.2 Hermite-type loss functions

We now consider the two-steps loss function and we focus only on the forward function approximation given by Eq. (13). This is frequently the most time

consuming part when solving an inverse problem with NNs. In this section, we analyze different strategies to work with a reduced dataset, which entails a dramatic reduction of the computational cost. We consider a dataset of three input-outputs pairs $(x, y) = \{(-33, 1089), (1, 1), (33, 1089)\}$.

Figure 8 shows the results for norms l_1 and l_2. Training data points are accurately approximated. Other points are poorly approximated.



(a) Norm l_1. (b) Norm l_2.

Fig. 8: Results of the NN that approximates the forward function. Red points correspond to the evaluation of the training dataset x and to $x+1$. Green points correspond to the evaluation of a testing dataset.

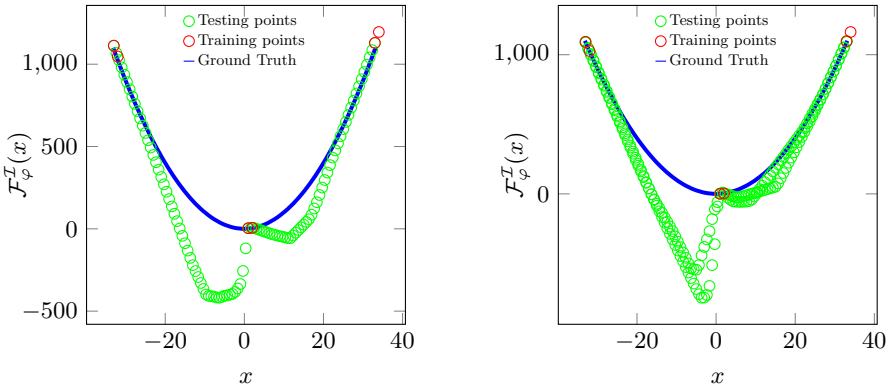
To improve the approximation, we introduce another term to the loss function. We force the NN to approximate the derivatives at each training point. This new loss is:

$$||\mathcal{F}_\varphi^{\mathcal{I}}(x) - y|| + \left|\left| \frac{\mathcal{F}_\varphi^{\mathcal{I}}(x+\epsilon) - \mathcal{F}_\varphi^{\mathcal{I}}(x)}{\epsilon} - \frac{\partial \mathcal{F}^{\mathcal{I}}(x)}{\partial x} \right|\right|. \tag{16}$$

From a numerical point of view, the term that approximates the first derivatives could be very useful. If we think about x as a parameter of a Partial Differential Equation (PDE), we can efficiently evaluate derivatives via the adjoint problem.

Figure 9 shows the results when we use norms l_1 and l_2 for the training. For this benchmark problem, we select $\epsilon = 1$. Thus, to approximate derivatives, we evaluate the NN at the points $x + 1$.

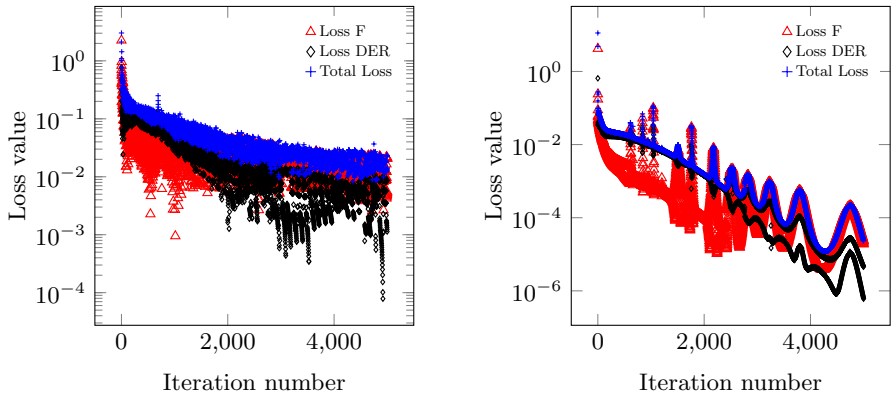
We observe that points nearby the training points are better approximated via Hermite interpolation, as expected. However, the entire approximation still lacks accuracy and exhibits undesired artifacts due to an insufficient number of training points. Thus, while the use of Hermite interpolation may be highly beneficial, especially in the context of certain PDE problems or when the derivatives are easily accessible, there is still a need to have a sufficiently dense database of



(a) Norm l_1. (b) Norm l_2.

Fig. 9: Results of the NN that approximates the forward function. Red points correspond to the evaluation of the training dataset x and to $x+1$. Green points correspond to the evaluation of a testing dataset.

sampling points. Figure 10 shows the evolution of the terms composing the loss function.



(a) Norm l_1. (b) Norm l_2.

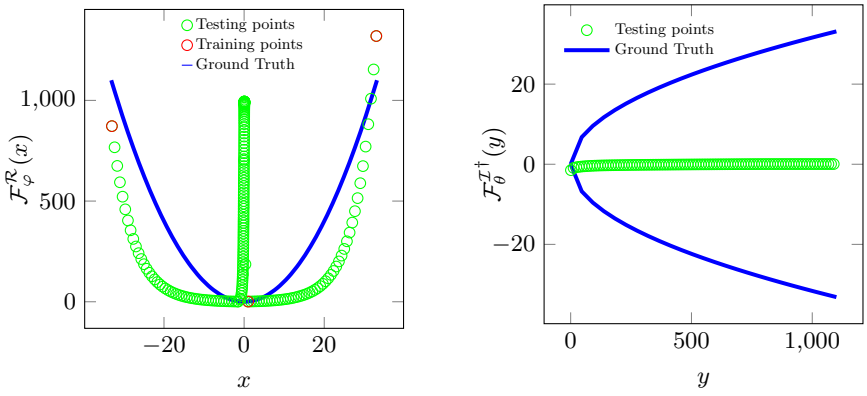
Fig. 10: Evolution of the loss value when we train the NN that approximates $\mathcal{F}_\varphi^{\mathcal{I}}$ using as loss Eq. (16). "Loss F" corresponds to the loss of the first term of Eq. (16). "Loss DER" corresponds to the loss of the second term of Eq. (16). "Total Loss" corresponds to the total value of Eq.(16).

4.3 Loss function with a reduced number of samples for the forward training

We now consider an Encoder-Decoder loss function, as described in Eq. (12). The objective is to minimize the number of samples employed to approximate the forward function since producing such database is often the most time-consuming part in a large class of inverse problems governed by PDEs.

We employ a dataset of three input-output pairs $\{(-33, 1089), (1, 1), (33, 1089)\}$ for the first term of Eq. (12) and a dataset of 1000 values of y obtained with an equidistant distribution on the interval $[0, 1089]$ for the second term of Eq. (12).

Figure 11 shows the results of the NNs trained with norm l_1. Results are disappointing. The forward function is far from the blue line (real forward function), specially nearby zero. The forward function leaves excessive freedom for the training of the inverse function. This allows the inverse function to be poorly approximated (with respect the to real inverse function).



(a) Forward function. (b) Pseudo-inverse operator.

Fig. 11: Exact vs NN solutions using loss function L_3, norm l_1, and a reduced number of samples for the forward evaluation.

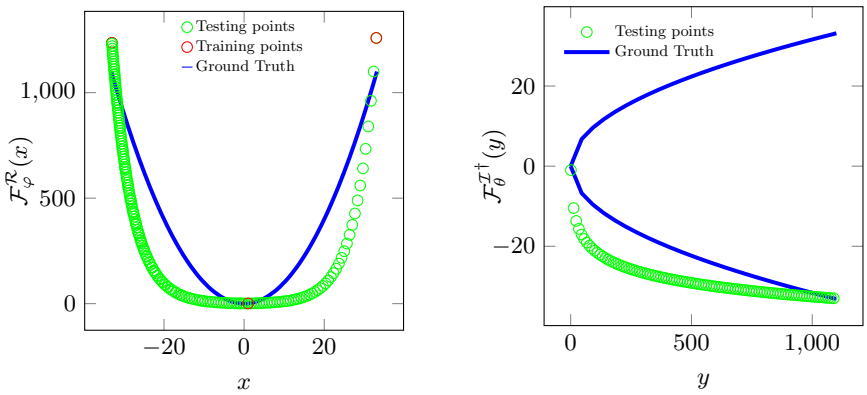
In order to improve the results, we train the NNs adding a regularization term to Eq. (12). We add the following regularization term maximizing smoothness on $\mathcal{F}_\varphi^{\mathcal{R}}$:

$$L_{3.1}(\varphi, \theta) = ||\mathcal{F}_\varphi^{\mathcal{R}}(x) - y|| + ||\mathcal{F}_\varphi^{\mathcal{R}}(\mathcal{F}_\theta^{\mathcal{I}\dagger}(y)) - y|| + \left|\left| \frac{\mathcal{F}_\varphi^{\mathcal{R}}(x+\epsilon) - \mathcal{F}_\varphi^{\mathcal{R}}(x)}{\epsilon} \right|\right|. \quad (17)$$

We evaluate this regularization term over a dataset of 1000 samples obtained with an equidistant distribution on the interval $[-33, 33]$ and we select $\epsilon = 1$.

Figure 12 shows the results of the NN. Now, the forward function is better approximated around zero. Unfortunately, the approximation is still inaccurate,

indicating the need for additional points on the approximation. Figure 13 shows the evolution of the terms composing the loss function. The loss values associated with the first and the second terms are minimized. The loss corresponding to the regularization term remains as the largest one.



(a) Forward function. (b) Pseudo-inverse operator.

Fig. 12: Exact vs NN solutions using loss function L_3, norm l_1, and a reduced number of samples for the forward evaluation.

5 Conclusions

We analyze different loss functions for solving inverse problems. We demonstrate via a simple numerical benchmark problem that some traditional loss functions are inadequate. Moreover, we propose the use of an Encoder-Decoder loss function, which can also be divided into two loss functions with a one-way coupling. This enables to decompose the original DL problem into two simpler problems.

In addition, we propose to add a Hermite-type interpolation to the loss function when needed. This may be especially useful in problems governed by PDEs where the derivative is easily accessible via the adjoint operator. Results indicate that Hermite interpolation provides enhanced accuracy at the training points and in the surroundings. However, we still need a sufficient density of points in our database to obtain acceptable results.

Finally, we evaluate the performance of the Encoder-Decoder loss function with a reduced number of samples for the forward function approximation. We observe that the forward function leaves excessive freedom for the training of the inverse function. To partially alleviate that problem, we incorporate a regularization term. The corresponding results improve, but they still show the need for additional training samples.

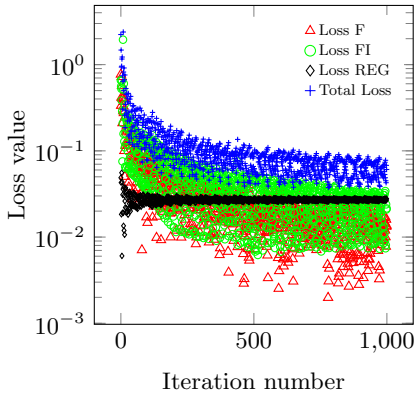


Fig. 13: Evolution of the loss value for Encoder-Decoder method trained with loss function $L_{3.1}$ and norm l_1. "Loss F" corresponds to the loss of the first term of Eq.(17). "Loss FI" corresponds to the loss of the second term of Eq. (17). "Loss REG" corresponds to the loss of the third term of Eq. (17). "Total Loss" corresponds to the total value of Eq. (17).

References

1. Albanese, R.A.: Wave propagation inverse problems in medicine and environmental health. In: Chavent, G., Sacks, P., Papanicolaou, G., Symes, W.W. (eds.) Inverse Problems in Wave Propagation. pp. 1–11. Springer New York, New York, NY (1997)
2. Beer, R., Dias, L., da Cunha, A., Coutinho, M., Schmitt, G., J. Seydoux, C.M., Legendre, E., Yang, J., Li, Q., da Silva, A., Ferraris, P., Barbosa, E., Guedes, A.: Geosteering and/or reservoir characterization the prowess of new generation LWD tools. Society of Petrophysicists and Well-Log Analysts (SPWLA) 51st Annual Logging Symposium (2010)
3. Bonnet, M., Constantinescu, A.: Inverse problems in elasticity. Inverse Problems **21**(2), R1–R50 (feb 2005). https://doi.org/10.1088/0266-5611/21/2/r01
4. Broquetas, A., Palau, J., Jofre, L., Cardama, A.: Spherical wave near-field imaging and radar cross-section measurement. IEEE Transactions on Antennas and Propagation **46**(5), 730–735 (1998)
5. Burczyński, T., Beluch, W., Dugosz, A., Orantek, P., Nowakowski, M.: Evolutionary methods in inverse problems of engineering mechanics. In: Inverse Problems in Engineering Mechanics II, pp. 553 – 562. Elsevier Science Ltd, Oxford (2000). https://doi.org/https://doi.org/10.1016/B978-008043693-7/50131-8, http://www.sciencedirect.com/science/article/pii/B9780080436937501318
6. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation (2014)
7. Hara, K., Saito, D., Shouno, H.: Analysis of function of rectified linear unit used in deep learning. In: 2015 International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2015)

8. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
9. Jin, Y., Wu, X., Chen, J., Huang, Y.: Using a physics-driven deep neural network to solve inverse problems for LWD azimuthal resistivity measurements pp. 1–13 (06 2019)
10. Li, Q., Omeragic, D., Chou, L., Yang, L., Duong, K.: New directional electromagnetic tool for proactive geosteering and accurate formation evaluation while drilling (2005)
11. Liu, G., Zhou, B., Liao, S.: Inverting methods for thermal reservoir evaluation of enhanced geothermal system. Renewable and Sustainable Energy Reviews **82**, 471 – 476 (2018). https://doi.org/https://doi.org/10.1016/j.rser.2017.09.065, http://www.sciencedirect.com/science/article/pii/S1364032117313175
12. Mao, X.J., Shen, C., Yang, Y.B.: Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections (2016)
13. Neto, A.S., Soeiro, F.: Solution of implicitly formulated inverse heat transfer problems with hybrid methods. In: Computational Fluid and Solid Mechanics 2003, pp. 2369 – 2372. Elsevier Science Ltd, Oxford (2003). https://doi.org/https://doi.org/10.1016/B978-008044046-0.50582-0
14. Oberai, A.A., Gokhale, N.H., o, G.R.F.: Solution of inverse problems in elasticity imaging using the adjoint method. Inverse Problems **19**(2), 297–313 (feb 2003). https://doi.org/10.1088/0266-5611/19/2/304
15. Puzyrev, V.: Deep learning electromagnetic inversion with convolutional neural networks. Geophysical Journal International **218**, 817–832 (05 2019). https://doi.org/10.1093/gji/ggz204
16. Stuart, A.M.: Inverse problems: A bayesian perspective. Acta Numerica **19**, 451–559 (2010). https://doi.org/10.1017/S0962492910000061
17. Tarantola, A.: Inverse Problem Theory and Methods for Model Parameter Estimation. Society for Industrial and Applied Mathematics, USA (2004)
18. Xu, Y., Sun, K., Xie, H., Zhong, X., Mirto, E., Feng, Y., Hong, X., Schlumberger: Borehole resistivity measurement modeling using machine-learning techniques (2018)
19. Zhu, G., Gao, M., Kong, F., Li, K.: A fast inversion of induction logging data in anisotropic formation based on deep learning. IEEE Geoscience and Remote Sensing Letters **PP**, 1–5 (01 2020). https://doi.org/10.1109/LGRS.2019.2961374