MODEL-CONSTRAINED DEEP LEARNING APPROACHES FOR INVERSE PROBLEMS[*]

TAN BUI-THANH[†]

Abstract. Deep Learning (DL), in particular deep neural networks, by design is purely data-driven and in general does not require physics. This is the strength of DL but also one of its key limitations when applied to science and engineering problems in which underlying physical properties—such as stability, conservation, and positivity—and desired accuracy need to be achieved. DL methods in their original forms is not capable of respecting the underlying mathematical models or achieving desired accuracy even in big-data regimes. On the other hand, many data-driven science and engineering problems, such as inverse problems, typically have limited experimental or observational data, and DL would overfit the data in this case. Leveraging information encoded in the underlying mathematical models, we argue, not only compensates missing information in low data regimes but also provides opportunities to equip DL methods with the underlying physics and hence obtaining higher accuracy. This short paper introduces several model-constrained DL approaches—including both feed-forward DNN and autoencoders—that learn not only information hidden in the training data but also in the underlying mathematical models to solve inverse problems. We present and provide intuitions for our formulations for general nonlinear problems. For linear inverse problems and linear networks, the first order optimality conditions show that our model-constrained DL approaches can learn information encoded in the underlying mathematical models, and thus can produce consistent or equivalent inverse solutions, while naive purely data-based counterparts cannot.

Key words. Inverse problem, optimization, model-constrained, deep learning, deep neural network.

1. Introduction.

Inverse problems are pervasive in scientific discovery and decision making for complex, natural, engineered, and societal systems. They are perhaps the most popular mathematical approaches for *enabling predictive scientific simulations* that *integrate observational/experimental data, simulations and/or models* [20, 13, 27]. Many engineering and sciences systems are governed by parametrized partial differential equations (PDE). Computational PDE-constrained inverse problems faces not only the ill-posed nature—namely, non-existence, non-uniqueness, instability of inverse solutions—but also the computational expense of solving the underlying PDE. Computational inverse methods typically require the PDE to be solved at many realizations of parameter and the cost is an increasing function of the parameter dimension. The fast growth of this cost is typically associated with the curse of the dimensionality. Inverse problems for practical complex systems [1, 20, 14, 5, 15] however possess this high dimensional parameter space challenge. Thus, mitigating the cost of repeatedly solving the underlying PDE has been one of paramount importance in computational inverse problems.

The field of Machine Learning (ML) typically refers to computational and statistical methods for automated detection of meaningful patterns in data [2, 26, 19]. While Deep Learning (DL) [9], a subset of machine learning, has proved to be state-of-the-art methods in many fields of computer sciences such as computer vision, speech

[*]Submitted to the editors DATE.

Funding: This work was partially funded by the National Science Foundation awards NSF-1808576 and NSF-CAREER-1845799; by the Defense Thread Reduction Agency award DTRA-M1802962; by the Department of Energy award DE-SC0018147; and by 2020 ConTex award; and by 2018 UT-Portugal CoLab award.

[†]Department of Aerospace Engineering and Engineering Mechanics, The Oden Institute for Computational Engineering and Sciences, UT Austin, Austin, Texas (tanbui@ices.utexas.edu, https://users.oden.utexas.edu/~tanbui/).

recognition, natural language processing, etc. Its presence in the scientific computing community is, however, mostly limited to off-the-shelf applications of deep learning. Unlike classical scientific computational methods, such as finite element methods [6, 4, 8], in which solution accuracy and reliability are guaranteed under regularity conditions, standard DL methods are often far from providing reliable and accurate predictions for science and engineering applications. The reason is that though approximation capability of deep learning, e.g. via Deep Neuron Networks (DNN), is as good as classical methods in approximation theory [7, 11, 18, 12], DL accuracy is hardly attainable in general due to limitation in training. It has been shown that the training problem is highly nonlinear and non-convex, and that the gradient of loss functions can explode or vanish [10], thus possibly preventing any gradient-based optimization methods from reliably converging to a minimizer. Even when converged, the prediction of the (approximate) optimal deep learning model can be prone to over-fitting and can have poor generalization error.

Many data-driven inverse problems in science and engineering problems have limited experimental or observational data, e.g. due to the cost of placing sensors (e.g. each oil well can cost million dollars) or the difficulties of placing sensors in certain regions (e.g. deep ocean bottoms). DL, by design, does not require physics, but data. This is the strength of DL. It is also the key limitation to science and engineering problems in which underlying physics need to be respected (e.g. stability, conservation, positivity, etc) and higher accuracy is required. In this case, purely data-based DL approaches are prone to over-fitting and thus incapable of respecting the physics or providing desired accuracy. Similar to least squares finite element methods [3], we can train DL solution constrained by the PDE residual as a regularization [25, 22, 23, 24, 29, 28, 16, 21]). The resulting DL models can learn solutions that attempt to make the PDE residual small. This physics-informed neural network (PINN) approach directly approximates the PDE solution in infinite dimensional spaces such as L^1 or L^2. While universal approximation results (see, e.g., [7, 11, 18, 12]) could ensure any desired accuracy with sufficiently large number of neurons, practical network architectures are moderate in both depth and width, and hence the number of weights and biases. Therefore, the accuracy of PINN can be limited. Moreover, for *parametrized PDEs*—that are pervasive in design, control, optimization, inference, and uncertainty quantification—training a PINN that is generalized for both spatial accuracy and accuracy in high-dimensional parameter spaces is not trivial [17].

For inverse problems, the object of interest is not the solution of parametrized PDE, per say, but some observable integrated Quantity of Interest (QoI) of the solution. Since the solution depends on the parameter, QoI is a function of parameter. The mapping from parameter to some observable QoI is known as the parameter-to-observable map. Unlike the solution, QoI does not depend on spatial or temporal variables but only on the parameter. Moreover, for most of engineering and sciences application the dimension of QoI or observational data is typically much less than that of the solution.

The paper is organized as follows. section 2 describes notations. In section 3 we briefly introduce nonlinear inverse problems, and a data-driven naive DL approach (NDL) and solution for linear network and linear inverse problem are presented. The goal of section 4 is to present a model-constrained DNN (MCDNN) approach designed to learn the inverse map while being constrained by the forward map of the underlying discretized PDE. Then, we present two model-constrained decoder approaches to learn the inverse map in section 5 and a model-constrained encoder approach in section 6. We conclude the paper with future research directions in section 7.

2. Notations.

- Boldface lower cases are for (column) vectors.
- Uppercase letters are for matrices. I is the identity matrix, whose size is clear from the context in which it appears.
- n_{t} is the number of training scenarios/data.
- m is dimension (number of rows) of the parameter vector $\boldsymbol{u}$.
- n is dimension (number of rows) of each observable data vector $\boldsymbol{y}$.
- $U \in \mathbb{R}^{m \times n_{\mathrm{t}}}$ is the parameter matrix concatenating available parameter $\boldsymbol{u}$.
- $Y \in \mathbb{R}^{n \times n_{\mathrm{t}}}$ is the data matrix concatenating available data $\boldsymbol{y}$.
- $\{U, Y\}$ or $\{Y, U\}$ is the training data set.
- $\mathbb{1} \in \mathbb{R}^{n_{\mathrm{t}}}$ is the column vector with all ones.
- $\|\cdot\|$ denotes standard Euclidean norm for vectors and Frobenius norm for matrices.
- $\boldsymbol{u}$ is the vector of parameters in the underlying parametrized PDEs.
- $\boldsymbol{w}$ is the solution of the underlying parametrized PDEs.
- $\boldsymbol{\theta}$ is the vector of all weights and biases of DNN.
- $\mathcal{G}$ denotes general nonlinear parameter-to-observable or forward map.
- G is preserved to denote a linear parameter-to-observable or forward map.
- $\boldsymbol{y}$ denotes the vector observational data.

3. Forward and inverse problems. We denote by $\boldsymbol{u} \in \mathbb{R}^m$ the parameters sought in the inversion, by $\boldsymbol{w} \in \mathbb{R}^s$ the forward states, by $\mathcal{G} : \mathbb{R}^s \to \mathbb{R}^n$ the forward map (computing some observable quantity of interest), and by $\boldsymbol{y} \in \mathbb{R}^n$ the observations. For the clarity of the exposition, we assume there is no observation noise or error, thus

$$\boldsymbol{y} := \mathcal{G}\left(\boldsymbol{w}\left(\boldsymbol{u}\right)\right) + \boldsymbol{\eta}, \tag{3.1}$$

where $\boldsymbol{\eta}$ takes into account the possibility that $\mathcal{G}\left(\boldsymbol{w}\left(\boldsymbol{u}\right)\right)$ is not adequate and/or there are noises/errors in observational data. The parameter-to-observable (PtO) map is the composition of the forward map $\mathcal{G}$ and the states, i.e. $\mathcal{G} \circ \boldsymbol{w}$. However for simplicity of the exposition, we do not distinguish it from the forward map and thus we also write $\mathcal{G} : \mathbb{R}^m \ni \boldsymbol{u} \mapsto \mathcal{G}\left(\boldsymbol{u}\right) := \mathcal{G}\left(\boldsymbol{w}\left(\boldsymbol{u}\right)\right) \in \mathbb{R}^n$. The forward states are the solution of the forward equation

$$\mathcal{F}\left(\boldsymbol{u}, \boldsymbol{w}\right) = \boldsymbol{f}, \tag{3.2}$$

Assume that (3.2) is well-posed so that, for a given set of parameters $\boldsymbol{u}$, one can (numerically) solve for the corresponding forward states $\boldsymbol{w} = \boldsymbol{w}\left(\boldsymbol{u}\right) := \mathcal{F}^{-1}\left(\boldsymbol{f}\right)$. In the forward problem, we compute observational data $\boldsymbol{y}$ via (3.1) given a set of parameter $\boldsymbol{u}$. In the inverse problem, we seek to determine the unknown parameter $\boldsymbol{u}$ given some observational data $\boldsymbol{y}$, that is, we wish to construct the inverse of $\mathcal{G}$. Since m is typically (much) larger than for many practical problems, the parameter-to-observable map $\mathcal{G}$ is not invertible even when $\mathcal{G}$ is linear. The inverse task is thus ill-posed and notoriously challenging as a solution for $\boldsymbol{u}$ may not exist (non-existence), even when it may, it is not unique (non-uniqueness) nor continuously depends on the data $\boldsymbol{y}$ (instability). An approximate solution is typically sought via (either deterministically or statistically) regularization.

Given the popularity of emerging machine learning, in particular deep learning, methods, we may attempt to apply pure data-driven DL, or naive DL (NDL), to learn the (ill-posed) inverse of $\mathcal{G}$, e.g.,

$$\min_{\mathbf{b},W} \frac{1}{2}\left\|U - \Psi\left(Y, W, \mathbf{b}\right)\right\|^2 + \frac{\alpha_1}{2}\left\|W\right\|^2 + \frac{\alpha_2}{2}\left\|\mathbf{b}\right\|^2, \tag{3.3}$$

where Ψ is DNN with weight matrix W and bias vector $\mathbf{b}$ and the last two terms are regularizations for weights and biases with nonnegative regularization parameters α_1 and α_2. This approach completely disregards the underlying mathematical model (3.1)-(3.2). In particular, even for linear inverse problem—$\boldsymbol{y} = G\boldsymbol{u}$ and there is no error in computing the data so that $Y = GU$—and linear DNN, it is not clear if the NDL approach (3.3) could recover a solution of the original inverse problem

$$\min_{\boldsymbol{u}} \left\|\boldsymbol{y}^{obs} - G\boldsymbol{u}\right\|^2 \tag{3.4}$$

in an interpretable sense. Indeed, suppose that we choose a linear activation function such that the DNN model $\Psi\left(Y, W, \mathbf{b}\right)$ for leaning the inverse map can be written as $WY + B$, where $B := \mathbf{b}\mathbb{1}^T$. It is easy to see that the optimal weight W^0 and bias $\mathbf{b}^0$ for (3.3) is given as

$$W^0 = \left[GU\left(I - \frac{1}{n_{\mathrm{t}} - \alpha_2}\mathbb{1}\mathbb{1}^T\right)U^TG^T + \alpha_1 I\right]^{-1} U\left(I - \frac{1}{n_{\mathrm{t}} - \alpha_2}\mathbb{1}\mathbb{1}^T\right)U^TG^T$$
$$\mathbf{b}^0 = \frac{1}{1 - \alpha_2/n_{\mathrm{t}}}\left(I - W^0G\right)\overline{\boldsymbol{u}},$$

where $\overline{\boldsymbol{u}} := \frac{1}{n_{\mathrm{t}}}U\mathbb{1}$. Thus the NDL inverse solution for a given testing/observational data $\boldsymbol{y}^{obs}$ is given by

$$\boldsymbol{u}^{\mathrm{NDL}} = W^0\boldsymbol{y}^{obs} + \mathbf{b}^0, \tag{3.5}$$

which is not clearly seen as a solution to the original inverse problem (3.4) in some sense.

This could be claimed as an advantage of the DNN approach. However, DNN can be seen as an "interpolation" method and thus can generalize well only for scenarios that have been seen in or are closed to the training data set $\{U, Y\}$. This implies a possible enormous amount of training data to learn the inverse of a highly nonlinear forward model (3.1)-(3.2). In practical sciences and engineering problems, this extensive data regime is unfortunately rarely the case due to the cost of placing sensors (e.g. each oil well can cost million dollars) or the difficulties in placing sensors in certain regions (e.g. deep ocean bottoms). *In order for a DNN to generalize well in insufficient data regimes, it should be equipped with information encoded in the forward model* (3.1)-(3.2) *that is not covered in the data set.* In other words, it is natural to require DNN to be aware of the underlying mathematical models (or discretizations) in order for it to be a reliable and meaningful tool for sciences and engineering applications. The question is how to inform DNN about the underlying models?

In the following, as an effort to train DNN to learn not only information hidden in the training data but also the underlying models, we explore several model-aware DNN approaches to learning the inverse of PtO map $\mathcal{G}$. Though the approaches are designed for general nonlinear problems, our focus here is on linear PtO map as it provides insights into if learning the inverse map via model-aware DNN is at all possible.

4. Model-Constrained Deep Neural Network (MCDNN) for learning the inverse map. We propose the learn the inverse map via DNN constrained by the forward map as

$$\min_{\mathbf{b},W} \frac{1}{2}\left\|U - \Psi\left(Y, W, \mathbf{b}\right)\right\|^2 + \frac{\alpha}{2}\left\|Y - \mathcal{G}\left(\Psi\left(Y, W, \mathbf{b}\right)\right)\right\|^2, \tag{4.1}$$

where Ψ is DNN learning the map from observable data $\boldsymbol{y}$ to parameter $\boldsymbol{u}$ with weight matrix W and bias vector $\mathbf{b}$. Unlike the naive pure data-driven DNN approach (3.3), (4.1) makes the DNN Ψ aware that the training data is generated by the forward map $\mathcal{G}$. This is done by requiring the output of the DNN—approximate unknown parameter $\boldsymbol{u}$ for a given data $\boldsymbol{y}$ as the input—when pushed through the forward model reproduces the data $\boldsymbol{y}$. The model-aware term $\frac{\alpha}{2}\left\|Y - \mathcal{G}\left(\Psi\left(Y, W, \mathbf{b}\right)\right)\right\|^2$ can be considered as a physics-informed regularization approach for DNN (compared to the non-physical regularizations in (3.3)).

To provide some intuition for our MC-DL approach let us choose a linear activation function such that the DNN model $\Psi\left(Y, W, \mathbf{b}\right)$ for leaning the inverse map can be written as $WY + B$, where $B := \mathbf{b}\mathbb{1}^T$. We also assume that the forward map is linear, i.e., $\boldsymbol{y} = G\boldsymbol{u}$ and there is no error in computing the data so that $Y = GU$. For linear inverse problem with linear DNN, the model-constrained training problem (4.1) becomes

$$\min_{\mathbf{b},W} \frac{1}{2}\left\|U - \left(WY + B\right)\right\|_F^2 + \frac{\alpha}{2}\left\|Y - G\left(WY + B\right)\right\|_F^2, \tag{4.2}$$

where the subscript F denotes the standard Frobenius norm.

Lemma 4.1. *Suppose that Y has full row rank. The optimal solution W^I and $\mathbf{b}^I$ of the DNN training problem (4.2) satisfies*

$$\mathbf{b}^I = \left(I + \alpha G^T G\right)^{-1}\left[I - \overline{U}\overline{Y}^T\left(\overline{Y}\overline{Y}^T\right)^{-1} G\right]\overline{\boldsymbol{u}},$$
$$W^I = \left(I + \alpha G^T G\right)^{-1}\left[\overline{U}\overline{Y}^T\left(\overline{Y}\overline{Y}^T\right)^{-1} + \alpha G^T\right],$$

where $\overline{\boldsymbol{u}} := \frac{1}{n_t}U\mathbb{1}$ and $\overline{\boldsymbol{y}} := \frac{1}{n_t}Y\mathbb{1}$ are the column-average of the training parameters and data, $\overline{Y} := Y - \overline{\boldsymbol{y}}\mathbb{1}^T$, and $\overline{U} := U - \overline{\boldsymbol{u}}\mathbb{1}^T$.

Corollary 4.2. *Suppose that Y has full row rank and $\alpha > 0$. For a given testing/observational data $\boldsymbol{y}^{obs}$, the DNN inverse solution $\boldsymbol{u}^{MCDL}$ of (4.2) is given by*

$$\boldsymbol{u}^{MCDL} = \left(I + \alpha G^T G\right)^{-1}\left[\overline{\boldsymbol{u}} + \overline{U}\overline{Y}^T\left(\overline{Y}\overline{Y}^T\right)^{-1}\left(\boldsymbol{y}^{obs} - \overline{\boldsymbol{y}}\right) + \alpha G^T\boldsymbol{y}^{obs}\right]$$

which is exactly the solution of the following regularized linear inverse problem

$$\min_{\boldsymbol{u}} \frac{1}{2}\left\|\boldsymbol{y}^{obs} - G\boldsymbol{u}\right\|^2 + \frac{1}{2\alpha}\left\|\boldsymbol{u} - \boldsymbol{u}_0\right\|^2,$$

where

$$\boldsymbol{u}_0 = \overline{\boldsymbol{u}} + \overline{U}\overline{Y}^T\left(\overline{Y}\overline{Y}^T\right)^{-1}\left(\boldsymbol{y}^{obs} - \overline{\boldsymbol{y}}\right).$$

The results of Corollary 4.2 shows that the MDDL inverse solution $\boldsymbol{u}^{\mathrm{MCDL}}$ is equivalent to a Tikhonov-regularized inverse solution with a special reference parameter $\boldsymbol{u}_0$ that depends on the training set $\{U, Y\}$ and the given observational data $\boldsymbol{y}^{obs}$. In other words, *the modal-constrained deep learning MCDNN approach provides data-informed Tikhonov-regularized inverse solutions.*

5. Model-Constrained Decoder (MCdecoder) for learning the inverse map.

Let us denote by $\Psi_{\text{e}}\left(\cdot, W_{\text{e}}, \mathbf{b}_{\text{e}}\right)$ the encoder with weight W_{e} and bias $\mathbf{b}_{\text{e}}$, and by $\Psi_{\text{d}}\left(\cdot, W_{\text{e}}, \mathbf{b}_{\text{e}}\right)$ the decoder with weight W_{d} and bias $\mathbf{b}_{\text{d}}$. We wish to train a model-constrained decoder (MCdecoder) in the following sense

$$
\begin{aligned}
\min_{\mathbf{b}_{\text{e}}, W_{\text{e}}, \mathbf{b}_{\text{d}}, W_{\text{d}}} \frac{\alpha}{2}\left\|Y - \Psi_{\text{e}}\left(U, W_{\text{e}}, \mathbf{b}_{\text{e}}\right)\right\|^2 + \frac{1}{2}\left\|U - \Psi_{\text{d}}\left(\Psi_{\text{e}}\left(U, W_{\text{e}}, \mathbf{b}_{\text{e}}\right), W_{\text{d}}, \mathbf{b}_{\text{d}}\right)\right\|^2 + \\
\frac{\beta}{2}\left\|Y - \mathcal{G}\left(\Psi_{\text{d}}\left(\Psi_{\text{e}}\left(U, W_{\text{e}}, \mathbf{b}_{\text{e}}\right), W_{\text{d}}, \mathbf{b}_{\text{d}}\right)\right)\right\|^2,
\end{aligned}
\tag{5.1}
$$

where the first term forces the encoder to map the parameter $\boldsymbol{u}$ to observation $\boldsymbol{y}$. The second term requires the decoder after taking the encoder output as its input reproduce the parameter. The third term is to ensure that the autoencoder system cannot be arbitrary but to obey the forward map. That is, the output of the decoder, which approximates the parameter, after going through the underlying forward map must reproduce the data. Thus, *unlike standard autoencoder approach which is purely data-driven, our proposed autoencoder has the parameter as its input, the observational data as its latent variable, and its output aware of the forward map.* For linear NN and linear inverse problem, the MCdecoder formulation (5.1) reduces to

$$
\begin{aligned}
\min_{\mathbf{b}_{\text{e}}, W_{\text{e}}, \mathbf{b}_{\text{d}}, W_{\text{d}}} \frac{1}{2}\left\|U - W_{\text{d}}\left(W_{\text{e}}U + B_{\text{e}}\right) - B_{\text{d}}\right\|_F^2 + \frac{\alpha}{2}\left\|Y - W_{\text{e}}U - B_{\text{e}}\right\|_F^2 + \\
\frac{\beta}{2}\left\|Y - GW_{\text{d}}\left(W_{\text{e}}U + B_{\text{e}}\right) - GB_{\text{d}}\right\|_F^2,
\end{aligned}
\tag{5.2}
$$

where $B_{\text{e}} := \mathbf{b}_{\text{e}}\mathbb{1}^T$ and $B_{\text{d}} := \mathbf{b}_{\text{d}}\mathbb{1}^T$.

Definition 5.1 (Equivalent inverse solution). *An inverse solution $\hat{\boldsymbol{u}}$ is equivalent to the true underlying parameter $\boldsymbol{u}^*$ if*

$$
\mathcal{G}\left(\hat{\boldsymbol{u}}\right) = \mathcal{G}\left(\boldsymbol{u}^*\right)
$$

Lemma 5.2. *The following combination of ${W_{\text{e}}}^{MCdecoder}$, $\mathbf{b}_{\text{e}}^{MCencoder}$, ${W_{\text{d}}}^{MCdecoder}$, $\mathbf{b}_{\text{d}}^{MCdecoder}$ satisfying*

$$
\mathbf{b}_{\text{e}}^{MCdecoder} = 0, \quad \mathbf{b}_{\text{d}}^{MCdecoder} = 0, \quad {W_{\text{e}}}^{MCdecoder} = G, \quad \text{and } {W_{\text{d}}}^{MCdecoder}G = I,
$$

is a stationary point of the lost function in the MCdecoder formulation (5.2).

That is, *the encoder weight matrix ${W_{\text{e}}}^{MCdecoder}$ in Lemma 5.2 is exactly the forward map G and the decoder weight matrix ${W_{\text{d}}}^{MCdecoder}$ is a left inverse of the forward map.* In this case, given a new data $\boldsymbol{y}^{obs}$, the decoder returns an equivalent inverse solution.

A variant of (5.1) is given by

$$
\begin{aligned}
\min_{\mathbf{b}_{\text{e}}, W_{\text{e}}, \mathbf{b}_{\text{d}}, W_{\text{d}}} J := \frac{1}{2}\left\|U - \Psi_{\text{d}}\left(\Psi_{\text{e}}\left(U, W_{\text{e}}, B_{\text{e}}\right), W_{\text{d}}, B_{\text{d}}\right)\right\|^2 + \\
\frac{\beta}{2}\left\|\Psi_{\text{e}}\left(U, W_{\text{e}}, B_{\text{e}}\right) - \mathcal{G}\left(\Psi_{\text{d}}\left(\Psi_{\text{e}}\left(U, W_{\text{e}}, B_{\text{e}}\right), W_{\text{d}}, B_{\text{d}}\right)\right)\right\|^2,
\end{aligned}
\tag{5.3}
$$

which, for linear neural network and linear inverse problem, becomes

$$\min_{\mathbf{b}_\mathrm{e}, W_\mathrm{e}, \mathbf{b}_\mathrm{d}, W_\mathrm{d}} J := \frac{1}{2}\left\|U - W_\mathrm{d}\left(W_\mathrm{e}U + B_\mathrm{e}\right) - B_\mathrm{d}\right\|^2 + \frac{\beta}{2}\left\|W_\mathrm{e}U + B_\mathrm{e} - GW_\mathrm{d}\left(W_\mathrm{e}U + B_\mathrm{e}\right) - GB_\mathrm{d}\right\|^2, \tag{5.4}$$

Lemma 5.3. *The following combination of* ${W_\mathrm{e}}^{MCdecoder}$, $\mathbf{b}_\mathrm{e}^{\mathsf{MCencoder}}$, ${W_\mathrm{d}}^{MCdecoder}$, *and* $\mathbf{b}_\mathrm{d}^{\mathsf{MCdecoder}}$ *satisfying*

$$\mathbf{b}_\mathrm{e}^{\mathsf{MCdecoder}} = \mathcal{N}\left({W_\mathrm{d}}^{MCdecoder}\right), \quad \mathbf{b}_\mathrm{d}^{\mathsf{MCdecoder}} = 0, \quad G \times {W_\mathrm{d}}^{MCdecoder} = I, \quad \text{and } {W_\mathrm{e}}^{MCdecoder} \times {W_\mathrm{d}}^{MCdecoder} = I,$$

is a stationary point of the lost function in the MCdecoder formulation (5.4). *Here,* $\mathcal{N}(\cdot)$ *denotes the null space. In addition, if*

That is, the decoder weight matrix ${W_\mathrm{d}}^{\mathrm{MCdecoder}}$ is a right inverse for both the forward map G and the encoder weight matrix ${W_\mathrm{e}}^{\mathrm{MCdecoder}}$.

Definition 5.4 (Consistent inverse solution). *An inverse solution* $\hat{\mathbf{u}}$ *is consistent if it reproduces the data* $\boldsymbol{y}^{obs}$ *when pushed through the forward map* $\mathcal{G}$, *i.e.,*

$$\mathcal{G}\left(\hat{\boldsymbol{u}}\right) = \boldsymbol{y}^{obs}.$$

The trained decoder in Lemma 5.3 thus provides consistent inverse solutions.

Remark 5.5. Though we are interested in the inverse solution, the decoder Ψ_d, designed in (5.1)–(5.3) as an approximate inverse map, is the primary object of interest. Should approximating the forward map $\mathcal{G}(\boldsymbol{u})$ be also a goal (such as for forward propagation of uncertainty), the encoder $\Psi_\mathrm{e}(\cdot, W_\mathrm{e}, \mathbf{b}_\mathrm{e})$, once trained, can be used as an approximate forward map by design. In fact for linear inverse problems and linear neural networks, Lemma 5.2 shows that the encoder could be exactly learn the forward map in formulation (5.1) while Lemma 5.3 shows that the encoder could indirectly learn (since the decoder is a right inverse for both the forward map G and the encoder weight matrix ${W_\mathrm{e}}^{\mathrm{MCdecoder}}$) the forward map in formulation (5.3).

6. Model-constrained Encoder (MCencoder) for learing the inverse map.

Recall in section 5 we train the decoder to learn the inverse map. The training is regularized by requiring the encoder to behave similar to the forward map. In this section, we reverse the autoencoder structure, that is, we train the encoder to learn the inverse map and the training is regularized by requiring the decoder to behave like the forward map. Similar to (5.1) we wish to train a model-constrained encoder by solving the following optimization problem.

$$\min_{\mathbf{b}_\mathrm{e}, W_\mathrm{e}, \mathbf{b}_\mathrm{d}, W_\mathrm{d}} \frac{\alpha}{2}\left\|U - \Psi_\mathrm{e}\left(Y, W_\mathrm{e}, B_\mathrm{e}\right)\right\|^2 + \frac{1}{2}\left\|Y - \Psi_\mathrm{d}\left(\Psi_\mathrm{e}\left(Y, W_\mathrm{e}, B_\mathrm{e}\right), W_\mathrm{d}, B_\mathrm{d}\right)\right\|^2 + \frac{\beta}{2}\left\|Y - \mathcal{G}\left(\Psi_\mathrm{e}\left(Y, W_\mathrm{e}, B_\mathrm{e}\right)\right)\right\|^2, \tag{6.1}$$

where the first term forces the decoder, as a surrogate to the inverse map, to transform observation $\boldsymbol{y}$ to parameter $\boldsymbol{u}$. The second term requires the decoder, after taking the encoder output as its input, reproduce the observation. The third term is to ensure that the autoencoder system cannot be arbitrary but be constrained the underlying forward map. That is, the output of the encoder, which approximates the parameter, after going through the forward map must reproduce the data. For linear NN and linear inverse problem, the MCencoder formulation (5.3) becomes

$$\min_{\mathbf{b}_\mathrm{e}, W_\mathrm{e}, \mathbf{b}_\mathrm{d}, W_\mathrm{d}} J := \frac{1}{2}\left\|Y - W_\mathrm{d}\left(W_\mathrm{e} Y + B_\mathrm{e}\right) - B_\mathrm{d}\right\|_F^2 + \frac{\alpha}{2}\left\|U - W_\mathrm{e} Y - B_\mathrm{e}\right\|_F^2 + \frac{\beta}{2}\left\|Y - G W_\mathrm{e} Y - G B_\mathrm{e}\right\|_F^2, \tag{6.2}$$

THEOREM 6.1. *At least one combination* ${W_\mathrm{e}}^{MCencoder}, \mathbf{b}_\mathrm{e}^{\mathsf{MCencoder}}, {W_\mathrm{d}}^{MCencoder}$, *and* $\mathbf{b}_\mathrm{d}^{\mathsf{MCencoder}}$ *satisfying*

$$\begin{aligned}
\mathbf{b}_\mathrm{e}^{\mathsf{MCencoder}} &= \overline{\boldsymbol{u}} - {W_\mathrm{e}}^{MCencoder}\overline{\boldsymbol{y}},\\
\mathbf{b}_\mathrm{d}^{\mathsf{MCencoder}} &= \overline{\boldsymbol{y}} - {W_\mathrm{d}}^{MCencoder}\overline{\boldsymbol{u}},\\
{W_\mathrm{e}}^{MCencoder} &= {W_\mathrm{e}}^{MCencoder}{W_\mathrm{d}}^{MCencoder}{W_\mathrm{e}}^{MCencoder},\\
{W_\mathrm{e}}^{MCencoder} &= {W_\mathrm{e}}^{MCencoder} G {W_\mathrm{e}}^{MCencoder}.\\
{W_\mathrm{d}}^T\left[I - W_\mathrm{d} W_\mathrm{e}\right] &= \alpha\left[W_\mathrm{e} - \overline{UY}^T\left(\overline{YY}^T\right)^{-1}\right] + \beta G^T\left(I - G W_\mathrm{e}\right),
\end{aligned}$$

is a stationary point of the optimization problem (6.2). *In addition, if* Y *has full row rank, all stationary points, including optimal solutions, obey the above four identities. Clearly, Any* $\left({W_\mathrm{e}}^{MCencoder}, {W_\mathrm{d}}^{MCencoder}\right)$ *satisfying*

$$\begin{aligned}
G{W_\mathrm{e}}^{MCencoder} &= I,\\
{W_\mathrm{d}}^{MCencoder}{W_\mathrm{e}}^{MCencoder} &= I.
\end{aligned}$$

is also a valid stationary point of (6.2).

Theorem 6.1 says that, once trained, the encoder weight matrix ${W_\mathrm{e}}^{\mathrm{MCencoder}}$ of the MCencoder formulation (6.2) can have forward map G as its generalized inverse and the decoder weight matrix ${W_\mathrm{d}}^{\mathrm{MCencoder}}$ behaves similar to the forward map in the sense that both are generalized inverses of ${W_\mathrm{e}}^{\mathrm{MCencoder}}$. In particular, this holds true at optimal solutions when the data is sufficiently rich, i.e., Y has full row rank. Note that this is not an entirely impractical assumption as the number of rows of Y is typically much smaller than the number of rows of U. Theorem 6.1 also indicates that the encoder weight matrix being a right inverse of the forward map G and the decoder weight matrix ${W_\mathrm{d}}^{\mathrm{MCencoder}}$ is a favorable possibility. In this case, we have

$$G\left({W_\mathrm{e}}^{\mathrm{MCencoder}}\boldsymbol{y}^{obs} + \mathbf{b}_\mathrm{e}^{\mathsf{MCencoder}}\right) = \boldsymbol{y}^{obs} = G\boldsymbol{u}^*,$$

which shows that the encoder is an approximate inverse map whose inverse solution is consistent as it exactly reproduces the observation $\boldsymbol{y}^{obs}$ when pushed through the forward map.

7. Conclusions.

We argue that in order for a DNN to generalize well in insufficient data regimes, it should be equipped with information encoded in the underlying mathematical modles that is not covered in the data set. In other words, it is natural to require DNN to be aware of the underlying mathematical models (or discretizations) in order for it to be a reliable and meaningful tool for sciences and engineering applications. To this end, we have presented several model-constrained DL approaches—using both feed-forward DNN and autoencoders—to learn inverse solutions while being aware of the forward model under consideration. The first order optimality conditions for the proposed model-constrained DL approaches can respect the physics. In particular, They can provide consistent or equivalent inverse solutions of the original inverse problems. Ongoing work is to investigate the second order optimality conditions and to extend the result to nonlinear inverse problems.

Acknowledgments. We would like to thank Sheroze Sheriffdeen, Jonathan Wittmer, Hwan Goh, and Co Tran for fruitful discussions.

REFERENCES

[1] O. M. Alifanov. *Inverse Heat Transfer Problems.* Springer Verlag, Berlin, Heidelberg, New-York, 1994.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg, 2006.

[3] Pavel Bochev and Max Gunzburger. *Least-Squares Finite Element Methods*, volume 166. 01 2006.

[4] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods.* Springer Verlag, Berlin, Heidelberg, New York, second edition, 2002.

[5] Tan Bui-Thanh, Carsten Burstedde, Omar Ghattas, James Martin, Georg Stadler, and Lucas C. Wilcox. Extreme-scale UQ for Bayesian inverse problems governed by PDEs. In *SC12: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2012. Gordon Bell Prize finalist, http://users.ices.utexas.edu/%7Etanbui/PublishedPapers/sc12.pdf.

[6] P. G. Ciarlet. *The finite element method for elliptic problems*, volume 40 of *Classics in Applied Mathematics.* SIAM (SIAM), Philadelphia, PA, 2002. Reprint of the 1978 original [North-Holland, Amsterdam; MR0520174 (58 #25001)].

[7] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

[8] Alexandre Ern and Jean-Luc Guermond. *Theory and Practice of Finite Elements*, volume 159 of *Applied Mathematical Sciences.* Spinger-Verlag, 2004.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[10] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[11] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[12] Jesse Johnson. Deep, skinny neural networks are not universal approximators. In *International Conference on Learning Representations*, 2019.

[13] Jari Kaipio and Erkki Somersalo. *Statistical and Computational Inverse Problems*, volume 160 of *Applied Mathematical Sciences.* Springer-Verlag, New York, 2005.

[14] Dimitri Komatitsch, Jeroen Ritsema, and Jeroen Tromp. The spectral-element method, Beowulf computing, and global seismology. *Science*, 298:1737–1742, 2002.

[15] Matthieu Lefebvre, Ebru Bozda, Henri Calandra, Judy Hill, Wenjie Lei, Daniel Peter, Norbert Podhorszki, David Pugmire, Herurisa Rusmanugroho, James Smith, and Jeroen Tromp. A data centric view of large-scale seismic imaging workflows. *Supercomputing (SC) 13*, 2013. Invited paper.

[16] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[17] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G. Johnson. Physics-informed neural networks with hard constraints for inverse design, 2021.

[18] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[19] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.
[20] Dean S. Oliver, Albert C. Reynolds, and Ning Liu. *Inverse theory for petroleum reservoir characterization and history matching*. Cambidge University Press, 2008.
[21] Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.
[22] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686 – 707, 2019.
[23] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125 – 141, 2018.
[24] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Machine learning of linear differential equations using gaussian processes. *Journal of Computational Physics*, 348:683 – 693, 2017.
[25] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.
[26] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA, 2014.
[27] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, Philadelphia, PA, 2005.
[28] Rohit K. Tripathy and Ilias Bilionis. Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of Computational Physics*, 375:565 – 588, 2018.
[29] Yibo Yang and Paris Perdikaris. Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, 2019.