Bauhaus-Universität Weimar

9 Inverse Problems & Deep Learning: Basic Methodology

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Deep Learning in Computational Mechanics – an introductory course, Herrmann et al. 2025

website

book

Bauhaus-Universität Weimar

Contents

9 Inverse Problems

Given a parametrized non-linear partial differential equations of the form

$$\frac{\partial^a u}{\partial^a t} + \mathcal{N}(u;\lambda) = 0, \qquad x \in \Omega, \qquad t \in \mathcal{T}$$

Inverse problems consider the identification of the differential equation given a (partial) solution u

- in the form of the non-linear differential operator $\mathcal{N}(u;\lambda)$
- or the coefficients λ
- a order of derivative in time t

Examples of inverse problems

- X-ray computed tomography: geometry reconstruction using the attenuation of the x-rays
- Calculation of the earth's density from measurements of the variation in the gravity field
- Flaw identification through the disturbance of ultrasonic pulses
- Topology optimization

Inverse problems are often ill-posed. Ill-posed means that the solution is either not unique or a small variation of the input causes a large variation of the output.

Non-unique solution: If the answer is 42 then it could have been composed of 41+1 or 40+2 or….

Bauhaus-Universität Weimar

9.1 Basic Methodology

- Three main methodologies in deep learning

Physics-Informed Neural Networks (e.g. the elastic bar discussed in Chapter 5)

- Minimization of the residual of the partial differential equation
- Sub-method of physics-informed learning (see Chapter 10)

Iterative Forward Solvers

- Minimization of the residual between measurement data and the solution to the differential equation in an alternating fashion
- Enforcement of physical laws by use of classical methods for the solution of differential equations

Data-driven solvers

- Minimization of the residual between predictions and labelled data
- Physics is ”learnt” by NN

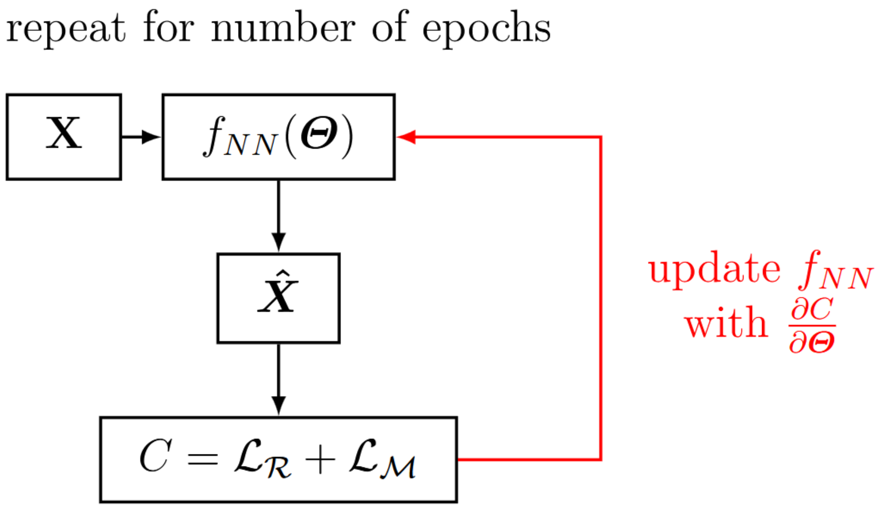
Degree of enforcement of the underlying physics decreases from top to bottom

9.1.1 Physics-Informed Neural Networks

In general the network provides a prediction $\hat{\boldsymbol{X}}$ given an input $\boldsymbol{X}$

Inputs can be, e.g.

- coordinates x, y
- solutions of PDEs u
- solutions of PDEs at previous time-steps u_{t-1}
- physical parameters λ


repeat for number of epochs
$\mathbf{X}$
$f_{NN}(\boldsymbol{\Theta})$
$\hat{\boldsymbol{X}}$
$C = \mathcal{L}_{\mathcal{R}} + \mathcal{L}_{\mathcal{M}}$
update f_{NN} with $\frac{\partial C}{\partial \boldsymbol{\Theta}}$

Predictions can be

- solutions u
- inverse quantities λ
- the non-linear differential operators $\mathcal{N}(u; \lambda)$.

The cost function is composed of

$$\mathcal{L}_M = \frac{1}{n}\sum_{i=1}^{m_M} (\tilde{u}_i - \hat{u}_i)^2 , \qquad \mathcal{L}_R = \frac{1}{m_R}\sum_{i=1}^{m_R} \left(\frac{\partial \hat{u}_i}{\partial t} + \hat{\mathcal{N}}[\hat{u}_i; \hat{\lambda}_i] \right)^2$$

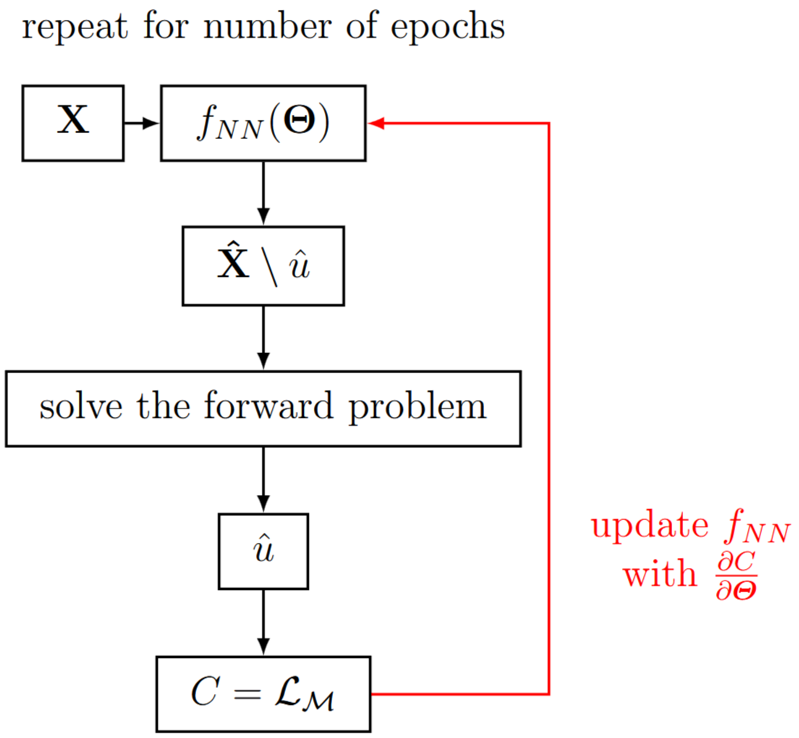
9.1.2 Iterative Forward Solvers

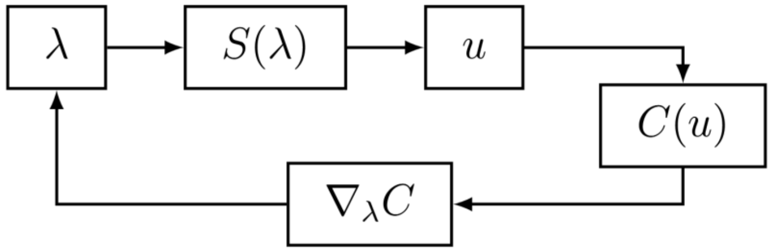
- In general the network provides a prediction $\widehat{\boldsymbol{X}}\backslash\hat{u}$ without the solution to the PDE given an input $\boldsymbol{X}$
- The prediction $\widehat{\boldsymbol{X}}\backslash\hat{u}$ is the used to solve the forward problem with a conventional approach yielding $\hat{u}$
- The predicted solution $\hat{u}$ is used to compute the measurement loss as cost function

$$C = \mathcal{L}_M = \frac{1}{n}\sum_{i=1}^{m_M}(\tilde{u}_i - \hat{u}_i)^2$$

- The physics are indirectly enforced by the forward solver
- Only finds a single solution

Without a neural network $f_{NN}(\boldsymbol{\Theta})$ the scheme is equivalent to standard gradient-based optimization: $\lambda = \widehat{\boldsymbol{X}}\backslash\hat{u}$, $S(\lambda)$ is the forward solver
But what is the point of the neural network?


repeat for number of epochs
$\mathbf{X}$
$f_{NN}(\boldsymbol{\Theta})$
$\hat{\mathbf{X}} \setminus \hat{u}$
solve the forward problem
$\hat{u}$
$C = \mathcal{L}_M$
update f_{NN} with $\frac{\partial C}{\partial \Theta}$

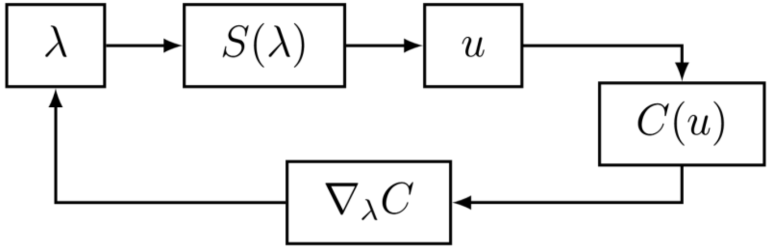


9.1.2.1 Neural Network Ansatz in Iterative Solvers

Standard gradient-based optimizer

- Parametrization of λ with classical ansatz functions (e.g., FEM)

$$\hat{\lambda}(x) \approx \sum_i \lambda_i N_i(x)$$



$$\begin{tikzcd} \lambda \arrow[r] & S(\lambda) \arrow[r] & u \arrow[dr] & \\ & & \nabla_\lambda C \arrow[ull] & C(u) \arrow[l] \end{tikzcd}$$

- Where λ_i are the coefficients and $N_i(x)$ the shape functions
- Given a set of coefficients λ_i the forward solution u is computed with the forward solver $S(\lambda)$
- The quality of λ_i is assessed by the cost function $C(u)$
- The derivative $\nabla_\lambda C$ is used to update the set of coefficients λ_i

$$\lambda_i^{n+1} = \lambda_i^n - \alpha \nabla_\lambda C$$

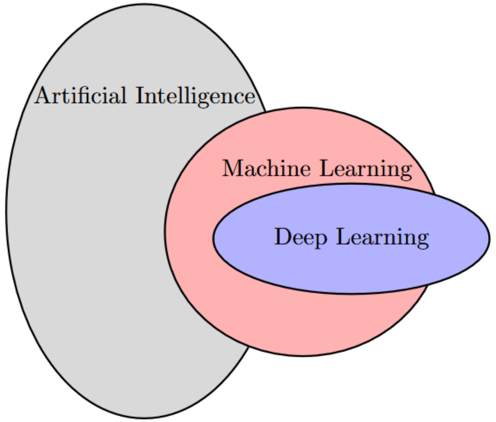
Iterative Forward Solver with Neural Network Ansatz

- Instead of relying on a classical parametrization of λ, a neural network is employed

$$\hat{\lambda}(x) \approx \lambda_{NN}(x; \boldsymbol{\Theta})$$

- The coefficients are thus the neural network parameters $\boldsymbol{\Theta}$

As optimizing the neural network retrieves the parameters $\boldsymbol{\Theta}$ for a single solution and cannot be applied to a different problem, the procedure is **not machine learning**. However, as a neural network is still being optimized, it can still be considered **deep learning**.



Bauhaus-Universität Weimar

9.1.2.2 Motivation for a Neural Network Ansatz

On neural networks for generating better local optima in topology optimization, Herrmann et al. 2024

Consider the Rosenbrock function as toy example

$$g(y_1, y_2) = (1 - y_1)^2 + 100(y_2 - y_1^2)^2$$

Goal is to find the optimum from an initial guess $y_1^{(0)}, y_2^{(0)}$

Standard gradient-based optimizer optimizes design variables y_1, y_2 directly

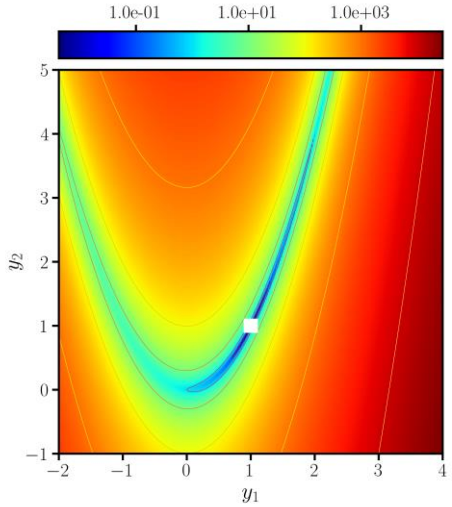
- as optimization is performed directly, we call this linear ansatz

Iterative Forward Solver with Neural Network Ansatz parametrizes y_1, y_2 with a neural network

$$\hat{\lambda} = \begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \end{pmatrix} = f_{NN}(\boldsymbol{\xi}; \boldsymbol{\Theta})$$

Such that the parameters $\boldsymbol{\Theta}$ become the design variables (instead of y_1, y_2)

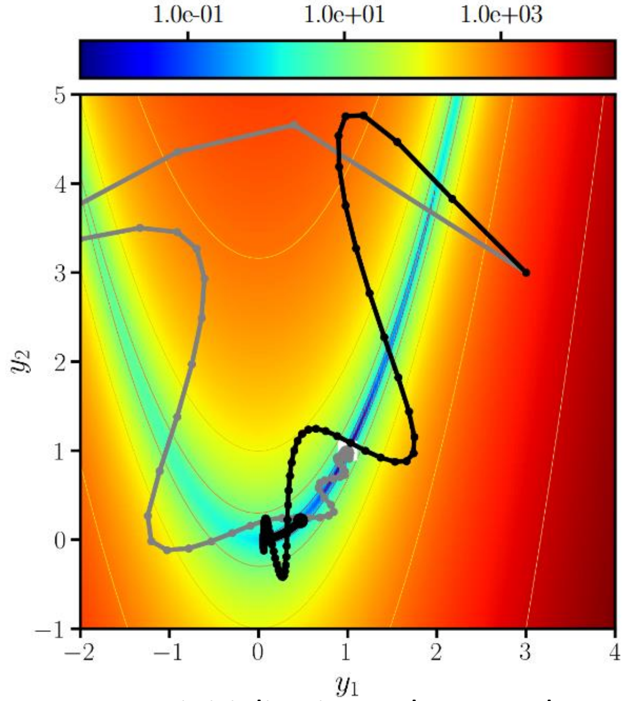
- Enables an overparameterization (as number of parameters can be larger than 2)
- ξ is kept constant throughout the optimization (it is an artifact of the neural network architecture, which requires an input)
- ξ is typically set via random noise

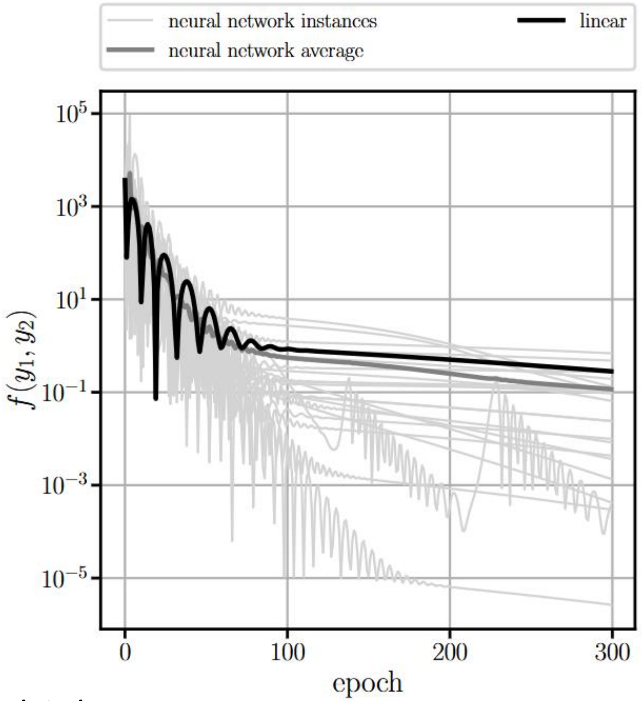


With optimum at $y_1 = 1, y_2 = 1$

9.1.2.2 Motivation for a Neural Network Ansatz

Linear ansatz versus **neural network ansatz** on the Rosenbrock function (after 300 epochs)





Further improvements are possible by reintegrating machine learning via transfer learning (see 9.2.5.1)

- For most initializations, the neural network is better
- Advantage is only possible when using the Adam optimizer for both parametrizations

Exercises

- E.31 Neural Network Ansatz on Optimization Benchmarks (C)
  - Compare a neural network ansatz with a linear ansatz on four basic optimization benchmarks (Rosenbrock, Rastrigin, Ackley, Levy) using different optimizers (gradient descent with momentum, AdaGrad, RMSprop, Adam, and L-BFGS).

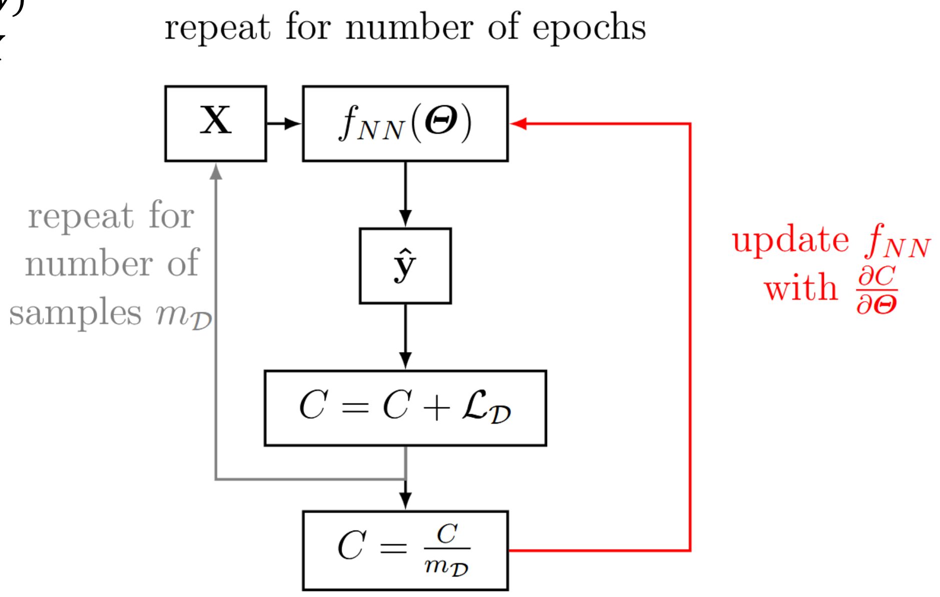
9.1.3 Data-Driven Solvers

In data-driven solvers, a labelled data-set of a physical problem is known as

$$(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{y}})$$

- The neural network predicts the label $\hat{\boldsymbol{y}}$ given an input $\boldsymbol{X}$
- This process is repeated over the whole dataset
- The cost function is the error between the predictions and the labelled data

$$C = \frac{1}{m_D} \sum_{i=1}^{m_D} (\tilde{\boldsymbol{y}}_i - \hat{\boldsymbol{y}}_i)^2$$


repeat for number of epochs
X
$f_{NN}(\boldsymbol{\Theta})$
$\hat{\mathbf{y}}$
$C = C + \mathcal{L}_{\mathcal{D}}$
$C = \frac{C}{m_{\mathcal{D}}}$
repeat for number of samples $m_{\mathcal{D}}$
update f_{NN} with $\frac{\partial C}{\partial \boldsymbol{\Theta}}$

Data driven solvers…

- require an off-line training phase (“expensive”)
- learn multiple solutions
- deliver fast predictions in the on-line phase (“cheap”)
- require a very large amount of data
- Do not enforce laws of physics. Physics is “observed and learnt”

Conceptually identical with the surrogate model for learning strain distributions in Chapter 3

Bauhaus-Universität Weimar

Contents

Bauhaus-Universität Weimar

9 Inverse Problems & Deep Learning: Basic Methodology

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Deep Learning in Computational Mechanics – an introductory course, Herrmann et al. 2025

website

book