Optimising Transfer Learning in Multi-fidelity Neural Networks

Lucca Pereira Martins, lp573

Summary

Neural Networks can be used as surrogate models to map the inputs of an engineering problem to outputs of interest. The networks, once trained, are computationally inexpensive when compared to the systems they describe, allowing for repeated evaluations and increased flexibility. However, accurate training often requires extensive quantities of data, which can scale to prohibitive volumes as network complexity increases. As such, we explore the application and optimization of transfer learning techniques using datasets of varying fidelity.

We explore two inductive transfer learning approaches and will extend the research to evaluate transductive transfer learning. We present a numerical example to illustrate the ability of bi-fidelity models in real-world applications. Finally, we will investigate multi-fidelity approaches that maximise surrogate model accuracy whilst finding the optimal number of fidelities.

Background

Neural networks (NNs) imitate biological neurons and have an impressive ability to learn nonlinear and complex functions. We can train large neural networks to accurately describe the response of a physical system, known as surrogate modelling. Notable advantages include faster design exploration and optimization, increased computational efficiency and accurate uncertainty propagation from inputs to outputs. There are however some limitations. There can often be a lack of clarity when enforcing physical constraints or governing equations, something which Physics-Informed Neural Networks (PINNs) attempts to address [1]. The most limiting drawback is the substantial – often prohibitive – data requirements for training said networks. Datasets for scientific and engineering applications are often limited in size due to costs associated with the physical experiments or the computational complexity of high-fidelity simulations [2]. By integrating data from models of different fidelities into the neural network training process, we may be able to reduce the volumes of data necessary to reach accurate surrogate estimations.

Multiple models are often available to describe the same physical system. High-fidelity models refer to ones with higher levels of accuracy but a (typically) higher computational cost. On the other hand, low-fidelity models have generally low levels of accuracy but can generate large datasets at little computational cost. Differing levels of fidelity can be achieved by, for example, adjusting mesh size in a Finite Element Model (FEM) or including another spatial dimension, say from 2D to 3D. In a multi-fidelity approach, the availability of models of varying fidelities is exploited to achieve a high level of accuracy in the overall prediction.

The key starting point in multi-fidelity modelling is discovering and exploiting both linear and non-linear relationships between low- and high-fidelity models. We seek additive and multiplicative coefficients to understand the correlation between these models, such as:

$$\boldsymbol{y}_H = \rho(\boldsymbol{x}_L)\boldsymbol{y}_L + \delta(\boldsymbol{x}_L) \tag{1}$$

Where y_L and y_H are the vector low- and high-fidelity model predictions, respectively, $\rho(x)$ is the multiplicative correlation component and $\delta(x)$ is the additive correlation component, for some low-fidelity input x_L. We can generalise this bi-fidelity approach to more degrees:

$$\hat{\boldsymbol{y}}_N = \sum_{i=1}^{N} \rho(\boldsymbol{x}_i)\boldsymbol{y}_i + \delta(\boldsymbol{x}_i) \tag{2}$$

Where $\hat{\boldsymbol{y}}_N$ is our multi-fidelity prediction for N degrees of fidelity. Clearly, multi-fidelity models governed by the above are restricted to only handling linear relationships. To capture nonlinear correlation, we introduce an unknown linear/nonlinear mapping function $F(\cdot)$ and describe the following autoregressive scheme:

$$\begin{aligned} \hat{\boldsymbol{y}}_N &= \sum_{i=1}^{N} F(\boldsymbol{y}_i) + \delta(\boldsymbol{x}_i) \\ \Rightarrow \hat{\boldsymbol{y}}_N &= \sum_{i=1}^{N} \mathcal{F}(\boldsymbol{x}_i, \boldsymbol{y}_i) \end{aligned} \tag{3}$$

We can further deconstruct $\mathcal{F}(\cdot)$ into its linear and nonlinear parts:

$$\hat{\boldsymbol{y}}_N = \sum_{i=1}^{N} \mathcal{F}_l(\boldsymbol{x}_i, \boldsymbol{y}_i) + \mathcal{F}_{nl}(\boldsymbol{x}_i, \boldsymbol{y}_i) \tag{4}$$

By understanding the nature of $\mathcal{F}_l$ and $\mathcal{F}_{nl}$ we can apply transfer learning (TL) to use knowledge gained from solving one problem to solve both similar/related problems (inductive TL) and unrelated problems (transductive TL). We will explore both. For example, a trained neural network (e.g. a low-fidelity model) can be used to help train, or improve, another neural network (e.g. a high-fidelity model). There are significant advantages to transfer learning over standard learning, namely a smaller initial training error, faster convergence and similar (or smaller) validation error when using smaller datasets.

Methodology

We first consider the bi-fidelity case and start with a network trained using the low-fidelity dataset $\mathcal{D}_l = \{(\boldsymbol{x}_{l,i}, \boldsymbol{y}_{l,i})\}_{i=1}^{N_l}$ to generate an initial surrogate model. The most common neural network is the feed-forward neural network (FNN), also known as the multilayer perceptron (MLP) [3]. We subsequently *adapt* this network based on the (smaller) high-fidelity training dataset $\mathcal{D}_h = \{(\boldsymbol{x}_{h,i}, \boldsymbol{y}_{h,i})\}_{i=1}^{N_h}$, where typically $N_h \ll N_l$. Note that $\boldsymbol{x}_l, \boldsymbol{x}_h \in \boldsymbol{x}$ where $\boldsymbol{x}$ is a set of k-dimensional inputs we subsample from. Furthermore, dimensionality reduction or a sensitivity study is often performed to minimise k and thus reduce computational expense [4].

Adaptation can take many forms. One approach is to fix both the architecture and parameters of the low-fidelity network then map its output through a high-fidelity network to a high-fidelity prediction. This approach relies on the assumption that learning the mapping between the low- and high-fidelity prediction is simpler than learning the mapping between the model inputs and the quantity of interest. We name this approach Bi-fidelity Prediction Mapping (BFPM). Another approach is to fix the architecture of the low-fidelity network but update the network parameters using the high-fidelity data. We name this approach Bi-fidelity Weighted Learning (BFWL). Both approaches are pursued (Figure 1).

We investigate how inductive TL can be applied to optimise bi-fidelity models under both BFPM and BFWL. We will then compare the performance of both under transductive TL to assess the flexibility and transferability of both networks. Finally, we expand our research to the multi-fidelity case, generating N databases with unique fidelities. Thus the approaches are now Multi-fidelity Prediction Modelling (MFPM) and Multi-fidelity Weighted Learning (MFWL). We want to maximise model accuracy with respect to degrees of fidelity, N, and more specifically, understand the model's capacity to learn $\mathcal{F}(\cdot)$ and find the optimal N.

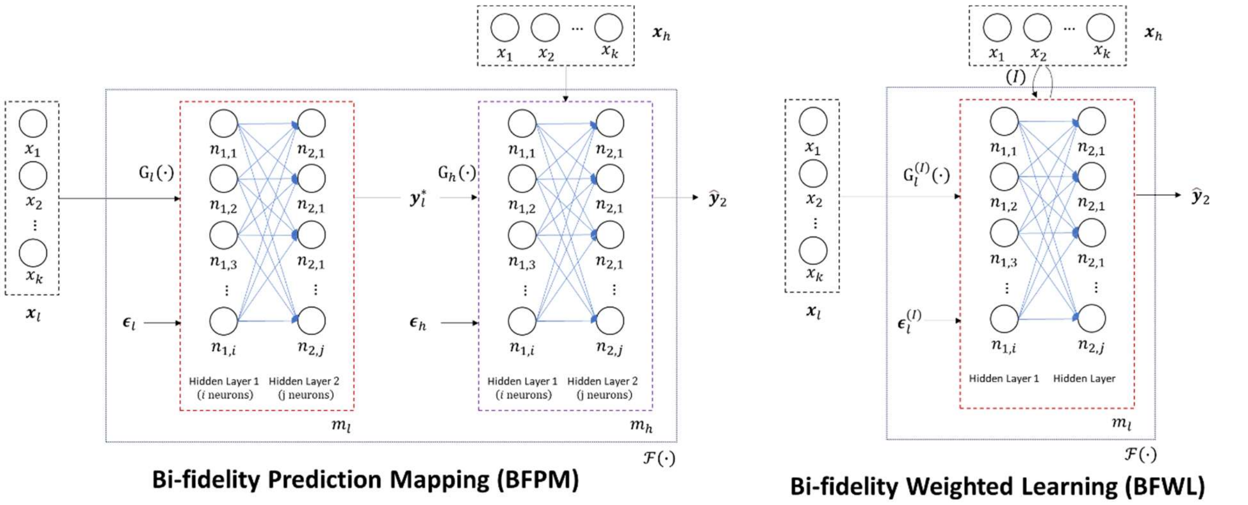

Figure 1a & 1b) BFPM and BFWL Transfer Learning approaches for neural networks using notation explained throughout text.

Numerical Example

We investigate the application of surrogate neural networks in geotechnical engineering, a division where multi-fidelity data is prevalent. We focus on deep excavations. Computer-aided design (CAD) and other forms of modelling are pivotal during the designing and construction of excavation projects. Projects in dense urban areas can have disastrous consequences and a high risk-to-life. Furthermore, CAD models can be inflexible, computationally expensive and hard to update 'on-the-fly'; site conditions and soil parameters are often refined as more data is gathered. As such, surrogate neural networks of CAD models can be trained using multi-fidelity experimental and simulation data to produce accurate and fast predictions, facilitating real-time decision making on-site.

Data was gathered from the excavation and construction of the (completed) PROTOS EFW project in Chester, UK, a waste-processing facility combined with a power plant. A key stage was the excavation of a 40m long, 18m wide and 12m deep waste storage bunker. Popular geotechnical CAD software Plaxis2D was used to model the excavation and construction stages (Figure 2). The model acts as a digital twin for the site and was calibrated using experimental data from in-built sensors lining the bunker. Whilst discussing the model itself is outside the

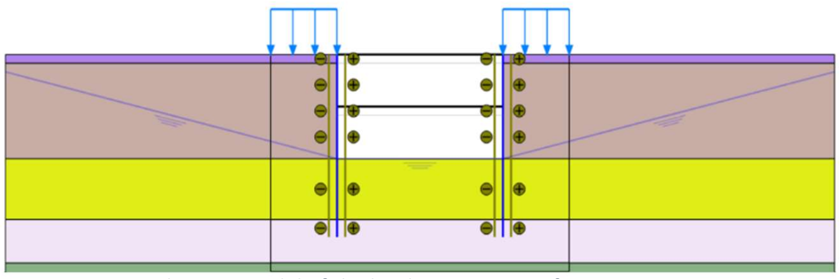
Figure 2: Plaxis2D model of the bunker excavation for PROTOS EFW project

x

scope of our research, it is essential to note the importance of basing our multi-fidelity database on provenly accurate CAD models and reliable experimental data. For more detail, refer to [5].

We are particularly interested in understanding the relationship between constituent soil parameters $\boldsymbol{x} \in \{x_1, \dots, x_k\}$ (such as soil cohesion value, friction angle or Young's modulus) and the resulting maximum deformations, moments and stresses $\boldsymbol{y}$ of the retaining wall at a certain depth d after ample consolidation. Effective stability analysis can only be undertaken once relevant settlement has occurred and transient pore water pressures dissipate.

$$\boldsymbol{y} = G(\boldsymbol{x}) + \boldsymbol{\epsilon} \tag{5}$$
$$\boldsymbol{x} \coloneqq \{x_1, x_2, \dots, x_k, d\} \tag{6}$$

Where $\boldsymbol{\epsilon}$ accounts for any bias in the relationship and $G(\cdot)$ is an unknown linear/nonlinear mapping function between inputs and outputs, not be confused with $\mathcal{F}(\cdot)$ which maps low- and high-fidelity predictions. We begin by performing a sensitivity study to identify the 6 most important input parameters when measuring $\boldsymbol{y}$. This alleviates computational expense without compromising on accuracy. These parameters are Young's modulus and friction angle for the two most prevalent soil types, cohesion value for the cohesive soil and depth d.

Next, we create N multi-fidelity datasets by simulating datapoints $(\boldsymbol{x_i}, \boldsymbol{y_i})$ using the Plaxis2D model. First, consider the $N = 2$, bi-fidelity case. We generate $N_l = 676$ low-fidelity datapoints using a coarse mesh (932 elements, fig.3a), $\mathcal{D}_l$, and $N_h = 290$ high-fidelity datapoints using a finer mesh (8231 elements, fig.3b), $\mathcal{D}_h$. Note $\mathcal{D}_l, \mathcal{D}_h \in \mathcal{D}$. We can now begin creating our surrogate neural networks.

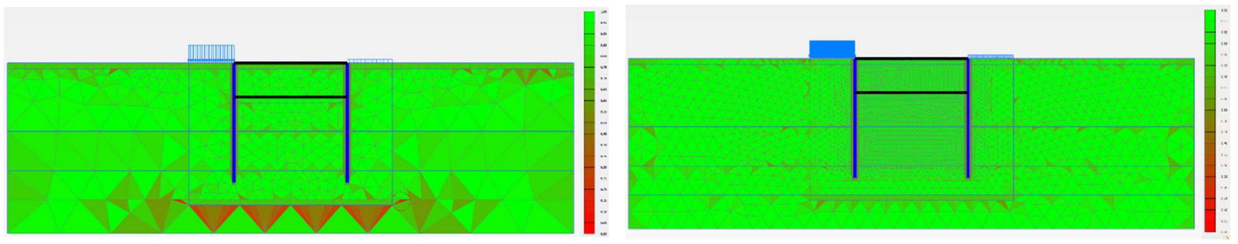
Figure 3a & 3b) Plaxis2D model with coarse mesh and fine mesh, respectively. Quality indicator from 'good' (green) to 'bad' (red) referring to suitability of individual elements in FEM model. Note the coarse mesh has multiple 'bad' elements, as expected.

An important step for both BFPM and BFWL is to first optimize then fix the architecture of the low-fidelity network, m_l. Should we build our multi-fidelity approach based on an excessively inaccurate low-fidelity model, any improvements in overall prediction accuracy may not be thanks to the inclusion of higher-fidelity data, but rather merely mediocre progress from an extremely low benchmark. We seek to maximise the information gained from $\mathcal{D}_l$ first.

$$\underset{G_l(\cdot),\boldsymbol{\epsilon}_l}{\operatorname{argmin}} \frac{1}{N_l^{TEST}} \sum_{i=1}^{N_l^{TEST}} \left| \boldsymbol{y}_{l,i}^{TEST} - \boldsymbol{y}_{l,i}^{PRED} \right|^2 \quad , \quad \boldsymbol{y}_l^{PRED} = G_l(\boldsymbol{x}_l) + \boldsymbol{\epsilon}_l \tag{7}$$

We build our low-fidelity MLP regression model and tune the appropriate hyperparameters (such as neuron number, activation function, maximum iteration and optimization algorithm). We conclude an 8,6 architecture is best (hence $m_l: i = 8, j = 6$ referring to figure 1). This is combined with the 'Adam' optimization algorithm and the Rectified Linear Unit (ReLU) activation function. For brevity, we do not offer a complete explanation of these here, please refer to [1] & [2].

Figure 4 shows the metrics we optimized when determining the ideal size of hidden layer 1 of m_l. A similar study was conducted for hidden layer 2. We note that iteration time appears arbitrary and as such is insignificant for our problem thus far. From figure 5, we can see 1000 epochs are more than sufficient for the model to converge. We can now fix the architecture of m_l and adapt it using $\mathcal{D}_h$ in line with BFPM and BFWL:

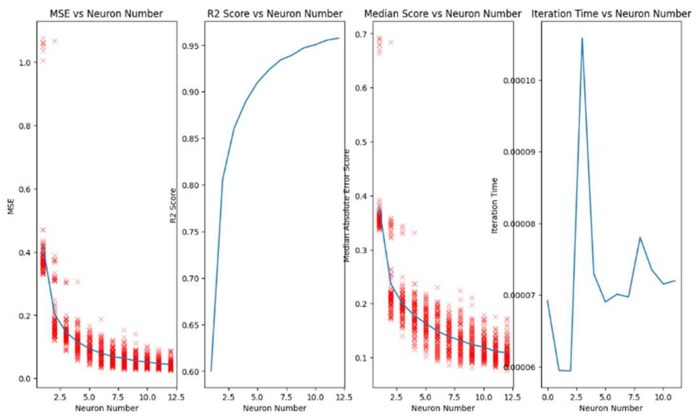

Figure 4: Optimizing neuron number in a single hidden layer of m_l against specific metrics. Red crosses indicate results for singular model iterations, note the overall high variance.

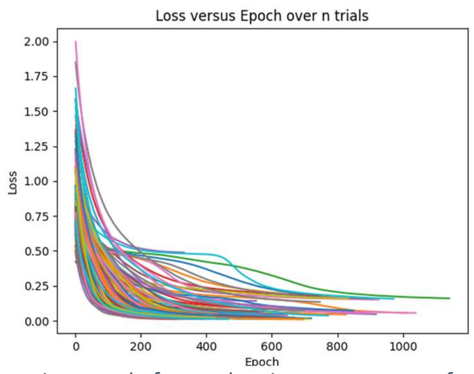

Figure 5: Loss, or error, against epoch, for m_l showing convergence after approximately 1000 epochs.

<u>Bi-fidelity Prediction Mapping (BFPM)</u>

We pass high-fidelity inputs $\boldsymbol{x}_h$ through the low-fidelity model m_l to obtain predictions $\boldsymbol{y}_l^*$:

$$\boldsymbol{y}_l^* = G_l(\boldsymbol{x}_h) + \boldsymbol{\epsilon}_l \tag{8}$$

We then train our high-fidelity model m_h using both $\boldsymbol{x}_h$ and $\boldsymbol{y}_l^*$ (m_h: $i = 18, j = 0$ ref. fig.1a).

$$\boldsymbol{y}_h^* = G_h(\boldsymbol{x}_h, \boldsymbol{y}_l^*) + \boldsymbol{\epsilon}_h \tag{9}$$

$$\boldsymbol{y}_h^* = G_h\big(\boldsymbol{x}_h, (G_l(\boldsymbol{x}_h) + \boldsymbol{\epsilon}_l)\big) + \boldsymbol{\epsilon}_h$$

$$\widehat{\boldsymbol{y}}_2 = \boldsymbol{y}_h^* = \mathcal{F}(\boldsymbol{x}_l, \boldsymbol{x}_h) \tag{10}$$

<u>Bi-fidelity Weighted Learning (BFWL)</u>

We utilise partial fitting to update the low-fidelity model m_l by presenting it with new training data: the high-fidelity inputs $\boldsymbol{x}_h$ (ref fig1b). This efficient form of incremental learning allows models to learn input mini-batches and respond accordingly by slightly adjusting weights and biases. This can be done repeatedly for I iterations, but we choose $I = 1$. Building on eq. 5:

$$\widehat{\boldsymbol{y}}_2 = G(\boldsymbol{x}_l) + \boldsymbol{\epsilon}|_{x_h} \Rightarrow \widehat{\boldsymbol{y}}_2 = G'(\boldsymbol{x}_l, \boldsymbol{x}_h) + \boldsymbol{\epsilon}' \tag{11}$$

Results and Discussion

We define the following as suitable metrics for assessing the performance of our models and approaches between predictions $\hat{\boldsymbol{y}}_2$ and true values $\boldsymbol{y}$: Mean Squared Error (MSE), Median Squared Error (MedSE), Coefficient of Determination during training and testing (R2 Train and R2 Test, respectively). We pass the low- and high-fidelity datasets $\mathcal{D}_l$ and $\mathcal{D}_h$ through the low-fidelity model m_l to set benchmarks. We also pass the unseparated dataset $\mathcal{D}$ through m_l.

Table 1: Performance of low- and high-fidelity surrogate neural networks for relevant metrics

	MSE	*MedSE*	*R2 Train*	*R2 Test*
$m_l(\mathcal{D}_l)$	0.080	0.161	0.919	0.909
$m_l(\mathcal{D}_h)$	0.209	0.218	0.881	0.810
$m_l(\mathcal{D})$	0.072	0.158	0.935	0.918
BFPM	0.051	0.148	0.950	0.931
BFWL	0.043	0.116	0.950	0.952

We start by discussing the benchmarks. As expected, the combined dataset $\mathcal{D}$ achieves a better overall score (meaning a lower MSE and MedSE with a higher R2 Train/Test) than the individual datasets $\mathcal{D}_l$ and $\mathcal{D}_h$. We note $m_l(\mathcal{D}_h)$ achieves relatively worse scores; this makes sense as $N_h \ll N_l$. More interestingly, we can report the success of both approaches, Bi-fidelity Prediction Mapping (BFPM) and Bi-fidelity Weighted Learning (BFWL), in improving the accuracy of predictions against the undifferentiated dataset $\mathcal{D}$. The improvement can be deemed significant. Note also that BFWL appears to perform better than BFPM on all metrics. Regarding the performance of BFWL, we noticed that by repeatedly partially fitting the model over I iterations with $\mathcal{D}_h$, we converge to extremely low values of MSE, around 0.01.

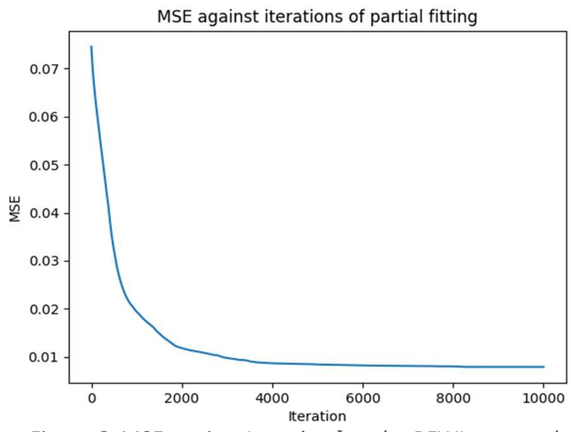

Figure 6: MSE against Iteration I under BFWL approach

Conclusion and Future Work

Both approaches show great promise in delivering high-accuracy, multi-fidelity models. Whilst BFWL appears the better of the two, we must first consider higher degrees of fidelity as well as model efficiency and computational cost when concluding which is best. Furthermore, we will investigate the MFPM and MFWL models on transductive TL by exploring a different construction site, subject to similar inputs and outputs. We hope to determine how many degrees of fidelity are both realistic and beneficial when modelling real-world problems of this nature. In doing so, we may significantly reduce the quantity of data needed to train surrogate neural networks.

References

[1] P. Zhang, Z. Yin, Y. Jin, J. Yang, B. Sheil, *Physics-informed Multi-fidelity Residual Neural Networks for Hydromechanical Modelling of Granular Soil and Foundation Considering Internal Erosion* (2022); [2] S. De, J. Britton, M. Reynold, R. Skinner, K. Jansen, A. Doostan, *On Transfer Learning of Neural Networks using Bi-fidelity data fir uncertainty propagation* (2019); [3] M. Popescu, V. Balas, L. Popescu, N. Mastorakis, *Multilayer Perceptron and Neural Networks* (2009); [4] L. van der Maaten, E. Postma, H. van den Herik, *Dimensionality Reduction: A Comparative Review* (2008); [5] P. Hensman, *Investigation of Soil-Structure Interaction for Deep Excavations* (2023)