UNIVERSITY OF CAMBRIDGE

Department of Engineering

Future Excavation Construction: Machine Learning and Real-time Monitoring

Author Name: Lucca Pereira Martins

Supervisor: Dr Brian Sheil

Date: 26/05/2024

I hereby declare that, except where specifically indicated, the work submitted herin is my own original work.

Signed LPmartins *date* 26/05/2024

Abstract

“Future Excavation Construction: Machine Learning and Real-time Monitoring”

Lucca Pereira Martins, lp573, 5661F, St John’s College

Deep excavations and geotechnical engineering projects often use computational, surrogate models to imitate physical systems on-site and describe their possible responses. Neural Networks prove useful surrogate models. These projects are highly complex and dangerous, with significant risk to life and to nearby infrastructure. As such, models must be precise and efficient to ensure accurate response predictions and facilitate on-site decision making. Training said models is data-intensive, further complicated by the differing quality of the available data. Multi-fidelity neural networks prove an excellent approach for developing surrogate models in these scenarios. Data of varying fidelities are often available on construction projects, such as site data and computational/simulation data. Although the former is typical of higher quality, it alone is too sparse and insufficient for training, thus combining the available data and leveraging their varying fidelities can lead to increase surrogate model prediction accuracy.

Two transfer learning methods for multi-fidelity neural networks are introduced, multi-fidelity weighted learning (MFWL) and multi-fidelity prediction mapping (MFPM). These are first both validated against an arbitrary example, then their effectiveness is investigated on a real-world deep excavation site, where we predict the deflections of retaining walls during the construction stages of an underground bunker. By using inductive transfer learning between low- and high-fidelity models, we achieve predictions that outperform industry-standard Finite Element models, with MFPM being the more promising method.

1. Introduction & Background

1.1 Problem Statement

Deep excavations and geotechnics are highly complex aspects within civil engineering. There is an abundance of interdependent variables governing the final outcome, from soil parameters and meteorological conditions to transient forces generated by people and plant operating on the construction site. All of these must be reflected in the methodology, adopted for assessing the structural soundness and ultimately, monitored for the delivery of the end-product. There is extensive risk associated with deep excavation projects, particularly those in dense urban areas which require sizeable foundations or basements, for example skyscrapers, sports stadiums or underground transportation networks. Soil or structural failures could put people at risk and cause catastrophic damage to nearby infrastructure and the project itself.

The complexity and danger to life mean such projects rely heavily on intricate computational models to help map and predict the possible outcomes as accurately as possible by manipulating the aforementioned variables. These models typically require calibration against site data, which is challenging and expensive to obtain, whilst the models themselves are equally time consuming to build and validate. Furthermore, high-quality data with which to train/calibrate said models are scarce. Finally, these computational models can easily deviate far off from the real site conditions as time goes on. Being able to improve/recalibrate the models with up-to-date statistics would be advantageous over building entirely new models.

This problem extends into general applications of Machine Learning (ML), where data requirements are often the limiting factor in a ML model's performance. Accuracy is regularly forgone to reduce costs. As such, there is an important need to optimize these surrogate models whilst minimizing the quantity of data necessary to train them. To this end, we propose ML methods which maximise the information gained from available data and seek to outperform standard computational models used in deep excavations. By providing better predictions and facilitating on-site decision making, the models could become common practice in geotechnical engineering.

1.2 Machine Learning and Neural Network Surrogates

Artificial neural networks and deep learning, a subset of supervised machine learning, have significantly improved our ability to computationally model real-world problems, particularly in the field of engineering and geoscience where multivariate problems are common. Neural Networks (NNs) imitate biological neurons and have an impressive ability to learn nonlinear and complex functions. We can train NNs to accurately describe the response of a physical system, known as surrogate modelling, similarly to how computer aided design (CAD) programs are used to accurately simulate real-world scenarios. Notable advantages include faster design exploration and optimization, increased computational efficiency and accurate uncertainty propagation from inputs to outputs. There are, however, some limitations such as convoluted model optimization and a lack of clarity when enforcing physical constraints or governing equations, something which Physics-Informed Neural Networks (PINNs) attempt to address [1]. Feed-forward Neural Networks (FNNs), also known as the Multilayer Perceptron (MLP), are commonly used in Machine Learning (ML) tasks [2], consisting of interconnected input, hidden and output layers (ref. fig.1).

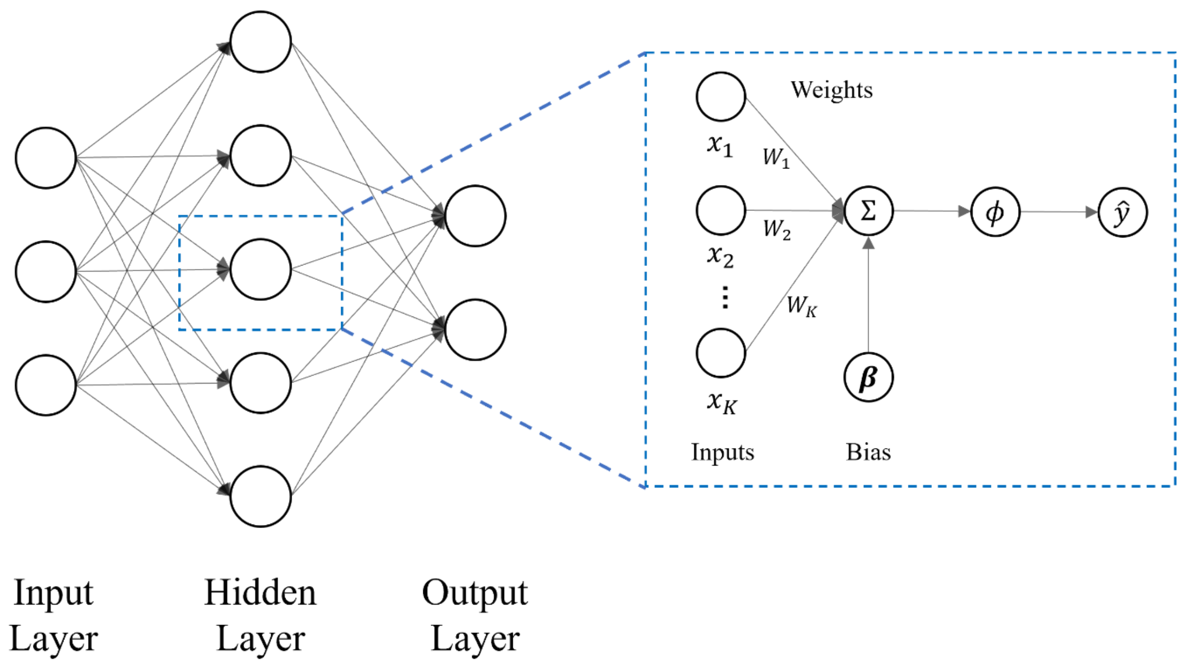


Figure 1: Multilayer Perceptron (MLP) Neural Network with an input layer, single hidden layer and an output layer. Nodal breakdown provided with notation described below.

A model $\mathcal{M}$ trained on dataset $\mathcal{D}$ with N hidden layers, weight matrices $\boldsymbol{W}_i$ and bias vectors $\boldsymbol{\beta}_i$, will predict an output $\hat{\boldsymbol{y}}$ from a given input $\boldsymbol{x}$ (eq.1). Intrinsically, layers are made up of nodes, or neurons, each mapping an input from the previous layer, $\boldsymbol{x}_i$, to an output $\boldsymbol{y}_i$ (eq. 2):

$$\hat{\boldsymbol{y}} \approx \mathcal{M}(\boldsymbol{x}; \{\boldsymbol{W}_i\}_{\mathrm{i}=0}^{N}, \{\boldsymbol{\beta}_i\}_{\mathrm{i}=0}^{N})|_{\mathcal{D}} \tag{1}$$

$$\hat{\boldsymbol{y}} \approx \mathcal{M}(\boldsymbol{x}; \boldsymbol{\theta})|_{\mathcal{D}} \ ; \ \ \boldsymbol{\theta} \coloneqq \{\boldsymbol{W}, \boldsymbol{\beta}\}$$

$$\boldsymbol{x}_{i+1} \leftarrow \hat{\boldsymbol{y}}_i = \phi_i(\boldsymbol{W}_i \boldsymbol{x}_i + \boldsymbol{\beta}_i) \tag{2}$$

Where $\phi_i(\cdot)$ is the i-th hidden layers activation function, often used to explore non-linear mappings. Standard activation functions include the Rectified Linear Unit (ReLU) function, or the sigmoidal logistic function (eq.3).

$$\phi_{ReLU}(z) = \max(0, z) \tag{3.1}$$

$$\phi_{sigmoid}(z) = \frac{1}{1 + e^{-z}} \tag{3.2}$$

The unknown parameters $\boldsymbol{\theta}$ can be learned by minimizing an appropriate loss function $\mathcal{L}$, typically the Mean Squared Error (MSE) between predictions $\hat{\boldsymbol{y}}$ and true values $\boldsymbol{y}$ from a test dataset $\mathcal{D}_{test}$ with N datapoints (eq.6).

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L} \tag{4}$$

$$\mathcal{L}_{MSE} \coloneqq \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{y}_i - \mathcal{M}(\boldsymbol{x}_i; \boldsymbol{\theta}))^2 \quad \forall(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{D}_{test} \tag{5}$$

$$\mathcal{L}_{MSE} \coloneqq \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{y}_i - \widehat{\boldsymbol{y}}_i)^2 \quad \forall(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{D}_{test} \tag{6}$$

$\mathcal{L}$ is minimized through backpropagation, using an optimization algorithm to numerically solve for $\boldsymbol{\theta}^*$. Simple Newtonian methods such as the stochastic gradient descent (SGD) algorithm utilize first order partial derivatives to iteratively update the parameter prediction, until a sensitivity threshold is crossed, and the algorithm terminated (eq.7).

$$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \eta_k \frac{\partial \mathcal{L}_k}{\partial \boldsymbol{\theta}_k} \tag{7}$$

Where η_k is the learning rate at the kth iteration, which can be fixed or adaptable.

More complex Newtonian optimization algorithms such as L-BFGS and Adaptive Moment Estimation (Adam), or evolutionary optimization algorithms such as Grey Wolf Optimizer (GWO), can make use of higher order derivatives and leverage past gradient information. They are often more robust, better suited to large datasets and converge to $\boldsymbol{\theta}^*$ more quickly. Their discussion is out of our scope, although worth understanding, particularly L-BFGS [3].

There is no closed-form solution to determining the ideal network hyperparameters $\boldsymbol{\lambda}^*$, a combination of the nodal parameters $\boldsymbol{\theta}$ and other parameters such as number of hidden layers, number of neurons per hidden layer, activation function, optimization algorithm and epoch, to name but a few. However, knowing $\boldsymbol{\lambda}^*$, or more practically, finding a set of values $\boldsymbol{\lambda} \cong \boldsymbol{\lambda}^*$, is essential to network performance. Common methods include grid search [4], Bayesian inference [5] and using evolutionary algorithms [6], especially for approximating numerical parameter values. Ultimately trial-and-error is the most common approach. This can be cumbersome, time consuming and lacking in robustness, although sound mathematical reasoning can often reduce the exploration time.

1.3 Multi-fidelity Neural Networks (MFNNs)

Multiple surrogate 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, or more practically, such as comparing simulation data to experimental data.

The most limiting drawback when training $\mathcal{M}$ is the substantial – often prohibitive – data requirements. 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. By integrating a multi-fidelity approach into the NN training process, the availability of models of varying fidelities is exploited to reduce the volumes of data needed to achieve valuable surrogate estimates and improve overall prediction accuracy.

The key starting point in multi-fidelity neural networks (MFNNs) 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:

$$\widehat{\boldsymbol{y}}_L \approx \boldsymbol{\mathcal{M}}_L(\boldsymbol{x})\big|_{\mathcal{D}_L=\{(x_{L,i},y_{L,i})\}_{i=1}^{N_L}} \tag{8}$$

$$\widehat{\boldsymbol{y}}_H = \rho(\boldsymbol{x})\widehat{\boldsymbol{y}}_L + \delta(\boldsymbol{x}) \tag{9}$$

Where $\widehat{\boldsymbol{y}}_L$ and $\widehat{\boldsymbol{y}}_H$ are the vector low- and high-fidelity predictions, respectively, $\rho(\boldsymbol{x})$ is the multiplicative correlation component and $\delta(\boldsymbol{x})$ is the additive correlation component, for some vector input $\boldsymbol{x}$. $\mathcal{M}_L$ and $\mathcal{M}_H$ are trained on the low- and high-fidelity datasets $\mathcal{D}_L$ and $\mathcal{D}_H$ with N_L and N_H datapoints, respectively. Typically, $N_H \ll N_L$. We can generalise this bi-fidelity approach to higher degrees:

$$\widehat{\boldsymbol{y}}_N = \sum_{i=1}^{N} \rho(\boldsymbol{x}_i)\widehat{\boldsymbol{y}}_i + \delta(\boldsymbol{x}_i) \tag{10}$$

Where $\widehat{\boldsymbol{y}}_N$ is our multi-fidelity prediction for N degrees of fidelity and $\boldsymbol{x}_i$ is the input at the ith 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:

$$\widehat{\boldsymbol{y}}_N = \sum_{i=1}^{N} F(\widehat{\boldsymbol{y}}_i) + \delta(\boldsymbol{x}_i)$$

$$\Rightarrow \widehat{\boldsymbol{y}}_N = \sum_{i=1}^{N} \mathcal{F}(\boldsymbol{x}_i, \widehat{\boldsymbol{y}}_i) \tag{11}$$

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

$$\widehat{\boldsymbol{y}}_N = \sum_{i=1}^{N} \mathcal{F}_l(\boldsymbol{x}_i, \widehat{\boldsymbol{y}}_i) + \mathcal{F}_{nl}(\boldsymbol{x}_i, \widehat{\boldsymbol{y}}_i) \tag{12}$$

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) [7]. 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) which both represent the same physical system (inductive TL). Transductive TL is more complex; data requirements to learn meaningful patterns across seemingly unrelated tasks are substantial, although it has useful applications in Natural Language Processing (NLP) and computer vision. 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 training datasets.

1.4 Work to Date on MFNNs

Transfer learning is a technique used in training multi-fidelity neural networks and has been shown to outperform standard training approaches for bi-fidelity numerical examples [8]. In [9], a multi-fidelity physics informed deep neural network (MF-PIDNN) is created through using TL to update a low-fidelity model. The model learnt approximate governing equations to provide accurate predictions even in zones with no data. Both concepts could be usefully applied to the field of deep excavations as formula and equations link most variables, some of which might not directly be measurable on site.

Further work on MFNNs include novel physics-informed models [10,11] and recursive co-kriging models [12]. The latter is an improved way of training multi-fidelity surrogates based on Gaussian process regression (kriging), which was first applied to geostatistics. These concentrate on scenarios where knowledge is available from both data and formulae.

Works specific to the application of ML in geotechnical engineering include [13] and [14]. In the former, a wholistic view of how machine learning could be implemented throughout construction process is offered, particularly interesting is the mention of deep learning in improving safety and on-site operation monitoring. The latter focuses on leveraging data collected from tunnel-boring machines to support on-site decision-making through data analysis and patter recognition techniques. Once again, both could assist with monitoring deep-excavations and offer a stepping stone for multi-fidelity methods in the field.

Few practical scenarios have yet been investigated, most MFNN research focuses on introducing new theoretical models and verifying them on example problems. It is important to assess their feasibility on real-world tasks. Deep excavations could greatly benefit from improved computational model prediction accuracy and hence multi-fidelity neural networks. As such, we will address these by investigating MFNNs in geotechnical engineering.

2. Methodology

2.1 Bi-fidelity Prediction Mapping and Bi-fidelity Weighted Learning

We first consider the bi-fidelity case, proposing two TL methods and laying theoretical groundwork for higher degrees of fidelity. To learn $\mathcal{F}_l$ and $\mathcal{F}_{nl}$ (ref. eq.12), we subsequently *adapt* $\mathcal{M}_L$, a low-fidelity MLP surrogate, using the (smaller) high-fidelity training dataset $\mathcal{D}_H$. Note that $\boldsymbol{x}_l, \boldsymbol{x}_h \in \boldsymbol{x}$ where $\boldsymbol{x}$ is a set of K-dimensional inputs we subsample from. Dimensionality reduction or a sensitivity study is often performed to minimise K and thus reduce computational expense.

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, or more effective, than learning the mapping between the combined 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 nodal parameters, $\boldsymbol{\theta} \coloneqq \{\boldsymbol{W}, \boldsymbol{\beta}\}$, using $\mathcal{D}_H$, known as partial fitting. We name this approach Bi-fidelity Weighted Learning (BFWL) (Figure 2). Algorithmically, the model is initialised with the parameters learnt from the low-fidelity data, then incrementally adjusted by means of one or more iterations of backpropagation using the new mini-batches of data (ref. eq.7).

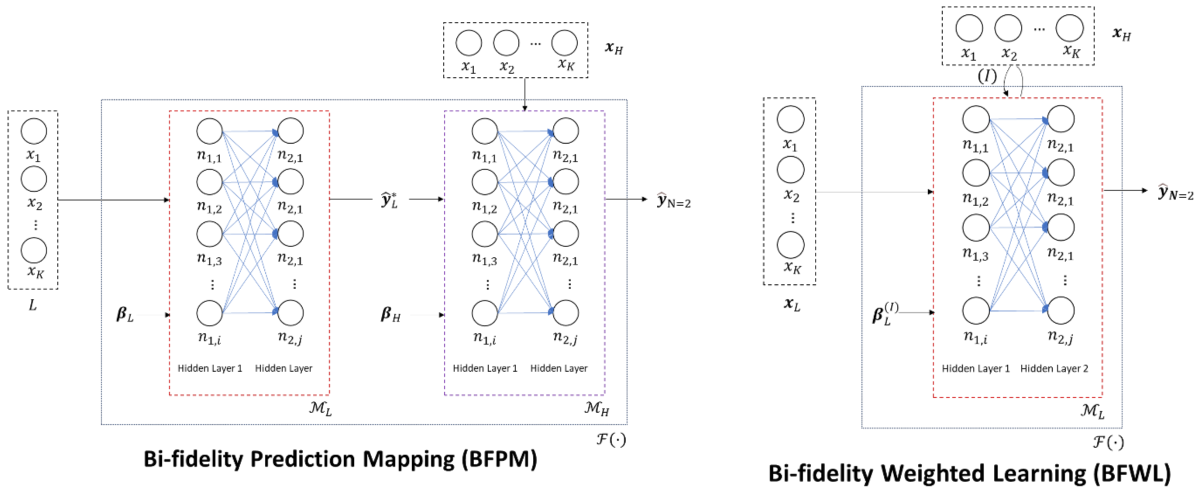


Figure 2: Bi-fidelity Prediction Mapping (BFPM) and Bi-fidelity Weighted Learning (BFWL) approaches for MFNNs.

2.2 Multi-fidelity Prediction Mapping and Multi-fidelity Weighted Learning

We can extend the concepts presented in BFPM and BFWL to the multi-fidelity case, hence Multi-fidelity Prediction Mapping (MFPM) & 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 $\mathcal{F}(\cdot)$ whilst finding the optimal N. In practice, scenarios with large N are rare, with typical values ranging from two to four. Spatial dimensions can provide said variety in fidelity, for example 2D and 3D CAD/Finite Element simulation data combined with 2D and 3D experimental data, as can polynomials with increasing degrees.

Considering further the MFWL approach, note that using a singular network with a fixed architecture can restrict the MFNN's ability to learn the linear and non-linear mappings $\mathcal{F}_l$

and $\mathcal{F}_{nl}$, respectively. Logically, if the underlying NN is excessively simple, for example it uses linear activation functions or too few hidden layers, the model will be incapable of modelling complex relationships, regardless of the degree of fidelity. For this reason, MFPM seems more robust to solving real-world problems, despite it involving numerous networks which must each be trained (often the most expensive part of working with NNs).

2.3 Multi-fidelity Transfer Learning for Continuous Functions with Linear Correlation

We investigate how inductive TL can be applied to optimise multi-fidelity models under both MFPM and MFWL through a numerical example, justifying the preferred method. Consider an instructional example of approximating a one-dimensional continuous function using data from three levels of fidelity with linear relationships. We generate datasets by sampling 20, 10 and 5 independent, uniformly spaced datapoints from the following low-, medium- and high-fidelity functions in the domain $x \in [0,1.5]$:

$$y_L = 0.5(6x-2)^2\sin(12x-4) + 10(x-0.5) - 5 \tag{13.1}$$
$$y_M = 0.8(6x-2)^2\sin(12x-4) + 2(x-0.5) - 1 \tag{13.2}$$
$$y_H = (6x-2)^2\sin(12x-4) + (x-0.5) \tag{13.3}$$

Clearly, we are dealing with the same underlying true function y_H and adding noisy coefficients to reproduce lower fidelities, figure 3 shows how the functions vary notably. This is a common occurrence in ML where the target function cannot be directly observed or sampled from, yet similar, less accurate functions are available.

We first train the low-fidelity model $\mathcal{M}_L$ on the low-fidelity dataset $\mathcal{D}_L$ to set a benchmark and form the foundation for future methods. We use three hidden layers with 20, 15 and 10 neurons, or [20,15,10] for short, and a sigmoid activation function, noting the Mean Squared Error (MSE) and R2 regression score against the true function (ref. table 1). Sufficient hidden layers are necessary to learn non-linear relationships for regression tasks, or non-linear decision boundaries for classification tasks.

To implement Multi-Fidelity Weighted Learning (MFWL), we update the parameters of $\mathcal{M}_L$ by partially fitting it with $\mathcal{D}_M$ and keeping the architecture fixed, thus creating a medium-fidelity model $\mathcal{M}_{M,WL}$ from which we can evaluate test samples. Repeating the same step with $\mathcal{D}_H$, we reach the final MFWL model $\mathcal{M}_{H,WL}$.

$$\mathcal{M}_L(\cdot\,;\boldsymbol{W}_L,\boldsymbol{\beta}_L)|_{\mathcal{D}_L} \rightarrow \mathcal{M}_M(\cdot\,;\boldsymbol{W}_M,\boldsymbol{\beta}_M)|_{\mathcal{D}_L,\mathcal{D}_M} \rightarrow \mathcal{M}_{H,WL}(\cdot\,;\boldsymbol{W}_H,\boldsymbol{\beta}_H)|_{\mathcal{D}_L,\mathcal{D}_M,\mathcal{D}_H} \tag{14}$$

For Multi-fidelity Prediction Mapping (MFPM), we first pass the training data x_M through $\mathcal{M}_L$ and append the results to $\mathcal{D}_M$, from which $\mathcal{M}_M$ is trained:

$$\hat{y}_{L,i} = \mathcal{M}_L\big(x_{M,i}\,;\boldsymbol{W}_L,\boldsymbol{\beta}_L\big)|_{\mathcal{D}_L} \tag{15}$$

$$\mathcal{D}_M^* = \big\{\big([x_{M,i},\ \hat{y}_{L,i}\ ], y_{M,i}\big)\big\}_{i=1}^{N=10} \tag{16}$$

$$\hat{y}_M = \mathcal{M}_M(x_{test}\,;\boldsymbol{W}_M,\boldsymbol{\beta}_M)|_{\mathcal{D}_M^*} \tag{17}$$

The same is repeating using $\mathcal{M}_M$ and $\mathcal{D}_H$ to give $\mathcal{M}_{H,PM}$ which we compare to $\mathcal{M}_{H,WL}$. We find the optimal architecture for $\mathcal{M}_M$ as [10,10] with a sigmoid activation function and $\mathcal{M}_{H,PM}$ as [10] with a ReLU activation function. Notice how the models reduce in complexity as fidelity increases to avoid over-generalisation.

To investigate the efficacy of both methods, we plot the true function y_H and the estimates from $\mathcal{M}_L$, $\mathcal{M}_{H,PM}$ and $\mathcal{M}_{H,WL}$ for $x \in [0,1.5]$ (ref. fig.4). All three models appear to learn the target function to an extent. As expected, the low-fidelity prediction is the least similar to y_H, yet still follows rough trends such as the position of local maxima/minima around $x = 0.7$, 1.0 and 1.3. It struggles most between $x < 0.6$ and $x > 1.4$ purely due to the noisy training data. MFPM appears the dominant multi-fidelity method, more accurately following y_H throughout, passing through $\mathcal{D}_H$ exactly and performing well in areas of sparse training data. There is clear success in both approaches. We also observe both the Mean Squared Error (MSE) and the coefficient of determination (R2) between predictions and y_H (ref. table 1).

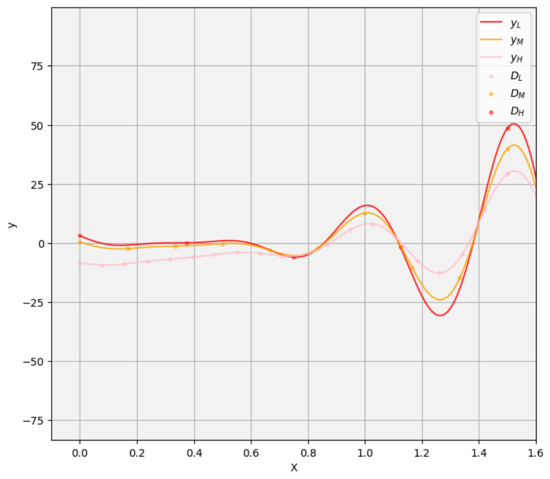


Figure 4: Low-, medium- and high-fidelity functions and the sampled datapoints.

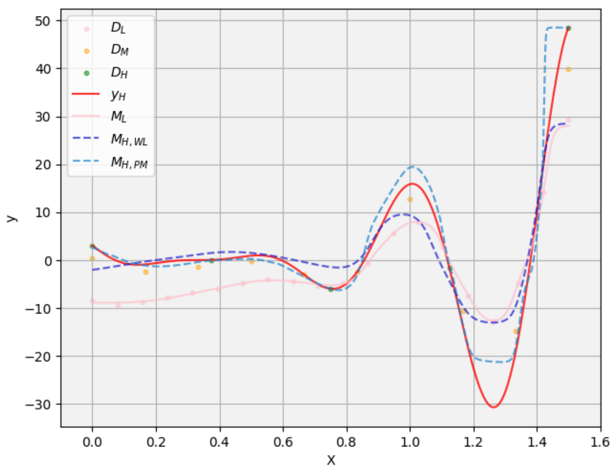


Figure 3: MFPM and MFWL predictions for y_H and the sampled datapoints. Note that $\mathcal{D}_H$ has been changed from red to green for clarity.

Table 1: Quantitative evaluation of MFPM and MFWL predictions and $\mathcal{M}_L$ benchmark against y_H for MSE and R2 metrics.

	MFPM		MFWL	
	MSE	R2	MSE	R2
$\mathcal{M}_L$	76.94	0.46	76.94	0.46
$\mathcal{M}_M$	39.39	0.69	57.92	0.59
$\mathcal{M}_H$	12.14	0.91	43.61	0.69

Both $\mathcal{M}_M$ and $\mathcal{M}_H$ across either method show significant improvement against $\mathcal{M}_L$, with MFPM outperforming MFWL in all cases. There is clear, sound theoretical and experimental backing to support these TL approaches for multi-fidelity neural networks, thus we explore a practical application next.

3. Case Study Details

3.1 The PROTOS EFW Deep Excavation Project

We investigate the application of surrogate neural networks in geotechnical engineering, a division where multi-fidelity data is prevalent, focusing on deep excavations. CAD and other forms of computational modelling such as Finite Element (FE) programs are pivotal during the designing and construction of excavation projects; developments in dense urban areas can have disastrous consequences and a high risk-to-life, hence guaranteeing the maximum possible accuracy is paramount. Furthermore, computational models can be inflexible, mathematically expensive and hard to update 'on-the-fly'; site conditions and soil parameters are often refined and adjusted accordingly as data is gathered. NN surrogates of computational 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 are 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. Prior to excavation, a sheet-piling cofferdam was installed marking the walls of the bunker. Excavation was performed in two stages across 28 days and carefully monitored using multiple sensors installed along/within the cofferdam which measured key variables during the excavation stages. These include inclinometers for lateral deflection, vibrating wire strain gauges for axial loads, strains, bending moments and temperatures, and piezometers for measuring the water level inside and outside of the bunker.

The popular geotechnical software Plaxis2D was used to model the excavation and construction stages (ref. fig.5). Plaxis2D is an all-inclusive Finite Element (FE) program which can manipulate thousands of high-resolution input and output variables, offering a complete range of possible simulations such as groundwater flow, bending moment and stress analyses, or deflection calculations in soils or structures. The digital site model was calibrated using the experimental data from the in-built sensors lining the bunker. Whilst discussing the model itself is outside the scope of our research, it is essential to note the importance of basing our multi-fidelity database on provenly accurate computational models and reliable experimental data. For more detail on the Plaxis2D model refer to [15]. A Plaxis3D model was also created and calibrated. The added dimension increases the accuracy but also significantly increases the computational cost. Running a simulation of all the construction stages in Plaxis2D for the PROTOS EFW site takes approximately 10 minutes, whilst one in Plaxis3D can take hours. One reason is because simulating consolidation phases, a timely process in real-life, is equally timely in computational models. This emphasises the need for accurate surrogate models to aid on-site decision making.

3.2 Developing a Machine Learning Approach

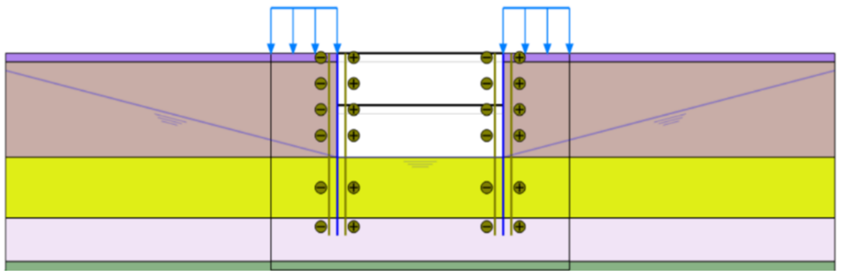


Figure 5: Plaxis2D model of the bunker excavation for PROTOS EFW project.

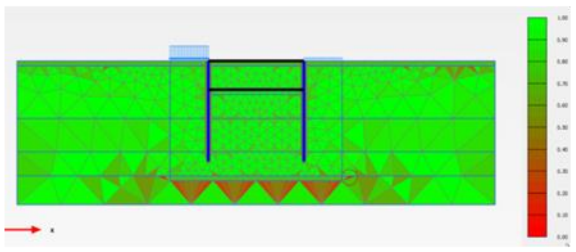
We are particularly interested in understanding the relationship between constituent soil parameters $\boldsymbol{x} \coloneqq [x_1, \dots, x_K]$ (such as soil cohesion value, friction angle or Young's modulus) and the resulting lateral deflections $\boldsymbol{y}$ of the storage bunker's retaining wall at certain depths d. Monitoring deflection profiles and maximum displacements during the construction phase is of interest when ensuring what is delivered complies with specifications, safety requirements and appropriate tolerances. Mistakes or inaccurate predictions could lead to structural failures and potential collapse of the bunker, or excessive settlement damage to nearby infrastructure.

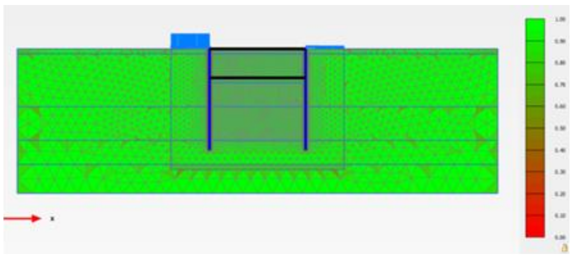
A comprehensive sensitivity study identified the most important input parameters when measuring $\boldsymbol{y}$, thus minimising K and the simulation's computational expense without compromising on accuracy. The six key parameters are Young's modulus and friction angle for the two most prevalent soil types, cohesion value for the cohesive soil (the other prevalent soil type is cohesionless) and depth d. Details regarding the methodology, construction process, sensitivity study and much more are covered in great depth in [15].

A logical and simple first step to assess the use of MFNNs on the PROTOS EFW site is to generate datasets from the calibrated Plaxis2D model using mesh sizes of significantly different coarseness, then train models to predict the maximum deflection at any construction stage along any depth, $y^* \coloneqq \max(\boldsymbol{y})$. As such, the dimension of $\boldsymbol{x}$ reduces to $K = 5$, depth d is not a relevant input parameter since we estimate only the maximum deflection.

3.3 Predicting Maximum Deflection using BFPM and BFWL

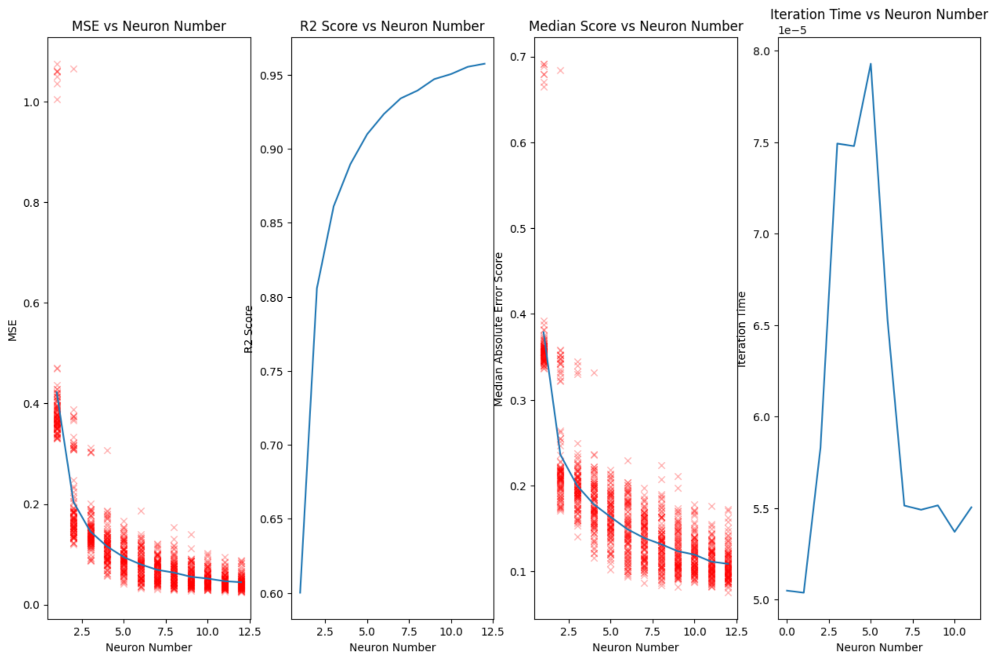
We first consider the bi-fidelity case and generate $N_L = 676$ low-fidelity datapoints using a coarse mesh (932 elements, fig.6a), forming $\mathcal{D}_L$, and $N_H = 290$ high-fidelity datapoints using a finer mesh (8231 elements, fig.6b), forming $\mathcal{D}_H$. Note $\mathcal{D}_L, \mathcal{D}_H \in \mathcal{D}_{train}$ and a similar dataset $\mathcal{D}_{test}$ was created and held out for evaluating the models' performances. We can now begin training our surrogate neural networks. An important step for both BFPM and BFWL is to first optimize then fix the architecture of the low-fidelity network, $\mathcal{M}_L$. Should we develop 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 through tuning the appropriate hyperparameters $\boldsymbol{\lambda}_L$ in $\mathcal{M}_L$. These include number of hidden layers, neuron number in each layer, activation functions, epochs, optimization algorithms and more. We conclude an [8,6] architecture is best with the Adam optimization algorithm and ReLU activation function.





Figures 6a, 6b: Plaxis2D FEM with coarse mesh and fine mesh, respectively. Quality indicator on right-hand side from 'good' (green) to 'poor' (red) referring to the suitability of individual mesh elements. Note that the coarse mesh has several 'poor' elements, as expected.

Figure 7 shows the metrics we optimized when determining the ideal size of hidden layer 1 of $\mathcal{M}_L$ using the elbow method. To counteract randomness, we repeated the Squared Error tests 100 times and plotted the mean (in blue), with each individual evaluation denoted by a red cross. A similar study was conducted for hidden layer 2. We note that, here, iteration time appears arbitrary. We can now fix the architecture of $\mathcal{M}_L$ and adapt it using $\mathcal{D}_H$ in line with BFPM and BFWL. For BFPM, we find the optimal network parameters $\boldsymbol{\lambda}_H^*$ to be a single-layer hidden layer with 18 nodes and a ReLU activation function. For $\mathcal{M}_{H,WL}$, we partially fit the model with a single iteration of $\mathcal{D}_H$.



Figures 7a, 7b, 7c, 7d: Training metrics for $\mathcal{M}_L$ including MSE, R2 score, MedSE and Iteration time (respectively) plotted against neuron number.

We define the following as suitable metrics for assessing the performance of our models and approaches between predictions $\hat{y}$ and true values y^*: Mean Squared Error (MSE), Median Squared Error (MedSE), Coefficient of Determination on training and testing dataset (R2 Train and R2 Test, respectively). We score these on the testing dataset $\mathcal{D}_{test}$ with size N.

$$\mathcal{L}_{MSE} \coloneqq \frac{1}{N}\sum_{i=1}^{N}(y_i^* - \hat{y}_i)^2 \tag{18}$$

$$\mathcal{L}_{MedSE} \coloneqq median\{(y_i^* - \hat{y}_i)^2\}_{i=1}^{N} \tag{19}$$

From table 3, there is clear success in both the BFWL and BFPM approaches: achieving lower Squared Error performances and higher R2 scores, an improvement which can be deemed significant. BFPM outperforms BFWL in this scenario, which combined with its superior ability to learn non-linear relationships and lead in the previous numerical example, sets it as the preferred method.

Table 2: Quantitative evaluation of $\mathcal{M}_L$, BFPM and BFWL predictions on $\mathcal{D}_{test}$ for MSE, MedSE and R2 metrics.

	$\mathcal{L}_{MSE}$	$\mathcal{L}_{MedSE}$	R2 Train	R2 Test
$\mathcal{M}_L(\mathcal{D}_{test})$	0.072	0.158	0.935	0.918
$\mathcal{M}_{H,WL}(\mathcal{D}_{test})$	0.051	0.148	0.950	0.931
$\mathcal{M}_{H,PM}(\mathcal{D}_{test})$	0.043	0.116	0.950	0.952

We next investigate a more involved use of MFNNs in deep excavations. We focus instead on the staged nature of the construction process and train models to learn the deflection profile along the entire retaining wall, specifically attempting to predict the final construction stage deflection shape given low-fidelity simulation data and high-fidelity experimental data from previous stages.

3.4 Pre-Requisites for Multi-Fidelity Learning

During the storage bunker excavation, high resolution inclinometers were used to measure the lateral deflection at 29 discrete points along the depth of the retaining wall. These were spaced at 0.5m intervals, from the top of the sheet pile which sat flush with the surface ($d = 0m$), to the bottom ($d = 14.5m$) of the sheet pile, sitting a little below the excavation floor. Recordings were taken 11 times over the 28 days, with the 11th measurement marking the accepted final deflection. We denote the recording instance α, such that $\alpha \in \{1, \dots, 11\}$ and consider only one wall of the bunker. These experimental datapoints combined with the calibrated Plaxis2D model provide two levels of fidelity for wall deformation.

We generate the low-fidelity dataset $\mathcal{D}_L$ by setting soil parameters $\boldsymbol{x}$, fixing the mesh size, then simulating the Plaxis2D model over all the construction stages, iterating for 20 different values of $\boldsymbol{x}$. Similarly to before, $\boldsymbol{x} \coloneqq [x_1, \dots, x_K]$ where $K = 5$ and the inputs are Young's modulus and friction angle for the two most prevalent soil types and cohesion value for the cohesive soil. Depth d is not necessary as we are modelling the entire deflection profile, $\boldsymbol{y} = [y(d = 0.5i)]_{i=1}^{29}$. We ensure a sufficient, yet representative, range of inputs are considered by sampling the parameters x_k normally, using their accepted mean value $\bar{x}_k$ and the uncertainty from experimental measurements σ_k as the standard deviation. It is important to train the surrogate model on varied data so it can interpolate well.

$$x_{i,k} \sim \mathcal{N}(\bar{x}_k, \sigma_k) \ \forall i \in N_L$$
$$\boldsymbol{x_i} \sim \mathcal{N}(\bar{\boldsymbol{x}}, \boldsymbol{\sigma}) \ \forall i \in N_L \tag{20}$$

We pass the normally distributed inputs through the Plaxis2D model and record the lateral deflections along the wall at the final construction stage, $\alpha = 11$, at the same depth intervals as the experimental data. We could include Plaxis2D data for deflection curves from previous stages, although found this hindered the model (discussed later). The deflection curves are shown below along with the true deflection curve $\boldsymbol{y}_{\alpha=11}$ plotted for reference (see fig.8, *Normalized and absolute deflection curves from the 20 Plaxis2D simulations against* $\boldsymbol{y}^*$). The normalized data is useful to better explain the shape of the curves across fidelity, whilst the absolute curves show the disparity in possible maximum deflection values.

The curves from Plaxis2D simulations follow the same rough shape, with noticeably less spread up to $d = 6m$ and higher spread toward the bottom. We notice how depth at maximum deflection, d^*, occurs around $d^* = 6.5m$ for all inputs. The dataset includes a variety of curves which capture the final stage deflections for possible ground conditions reasonably well. When comparing the Plaxis2D curves to $\boldsymbol{y}_{\alpha=11}$ we notice a large disagreement. The simulation overpredicts maximum displacement, and underpredicts the deflections below d^*. There is evidently room for improvement between the Plaxis2D simulations and the true deflection values.

Finally, we append the construction stage to the input values, hence allowing the model to learn deflection profiles as a function of both soil parameters and time (eq.21). This can be used to assess if the wall is deflecting as expected and within tolerances during excavation.

$$\mathcal{D}_L \coloneqq \{[\boldsymbol{x}_i, \alpha_i = 11], \boldsymbol{y}_i\}_{i=1}^{N_L=20} \tag{21}$$

The experimental, high-fidelity dataset $\mathcal{D}_H$ is more easily assembled using the accepted mean values for soil parameters, $\overline{\boldsymbol{x}}$, and the measured deflections $\boldsymbol{y}$ at the relevant construction stage $\alpha = [1, \dots, 10]$. We define the true output values $\boldsymbol{y}^* \coloneqq \boldsymbol{y}_{\alpha=11}$ and the true input values $\boldsymbol{x}^* = \overline{\boldsymbol{x}}$. There is one site measurement per construction stage, hence 10 datapoints (we leave out the final true shape for testing). Also note that $\mathcal{D}_L, \mathcal{D}_H \in \mathcal{D}$, the undifferentiated, combined dataset thus providing another benchmark to evaluate our MFNN surrogate against.

$$\mathcal{D}_H \coloneqq \{[\overline{\boldsymbol{x}}, \alpha_i = i], \boldsymbol{y}_{\alpha=i}\}_{i=1}^{N_H=10} \tag{22}$$

$$\mathcal{D} \coloneqq \{\mathcal{D}_L, \mathcal{D}_H\}$$

$$\Rightarrow \mathcal{D} \coloneqq \{[\boldsymbol{x}_i, \alpha_i], \boldsymbol{y}_i\}_{i=1}^{N=30} \tag{23}$$

The experimental deflection curves are plotted accordingly (ref. fig.9). The absolute deflection curves show how the profile $y(d)$ evolves with time as further settlement and consolidation occurs. d^* lowers with time, going from approximately $d^* = 6.7m$ to $d^* = 9.6m$. The curves elongate and deflections below d^* increase, shown more clearly in the normalized deflection curves. A key point to highlight is the significant difference between $\boldsymbol{y}_{\alpha=10}$ and $\boldsymbol{y}_{\alpha=11}$, perhaps one of the largest changes in shape and magnitude between adjacent construction stages. Not only is predicting the final deflection shape the most useful, it will clearly also be the most challenging, emphasizing the potential benefits of multi-fidelity methods. There is a slight discontinuity in $\boldsymbol{y}_{\alpha=4}$ and $\boldsymbol{y}_{\alpha=4}^{norm}$ arising from experimental inaccuracy but it can be ignored: the NN will address this automatically.

Comparing the normalized experimental deflection curves for stages 1 and 2, to the Plaxis2D simulation ones, we notice a similarity between $\boldsymbol{y}_{\alpha=1,2}^{exp,norm}$ and $\boldsymbol{y}_{\alpha=11}^{Plx,norm}$. This highlights the Plaxis2D model's inability to accurately predict real-world consolidation, or rather the complicated nature of real-world consolidation. It is dependent on several uncontrollable factors, such as meteorology, which means simulations can only offer rough approximations; surrogate models and experimental data are key to fast, accurate predictions.

Overall, we want to predict 29 outputs (the deflections along the wall) from 6 inputs (the soil parameters and the construction phase). More precisely, we seek to estimate $\boldsymbol{y}^*$ from the previous experimental deflection curves in $\mathcal{D}_H$ and the simulated curves in $\mathcal{D}_L$. Whilst one could assume this mapping might require an extensive neural network, we will show that through BFPM, improvements in prediction can be made with simple NNs.

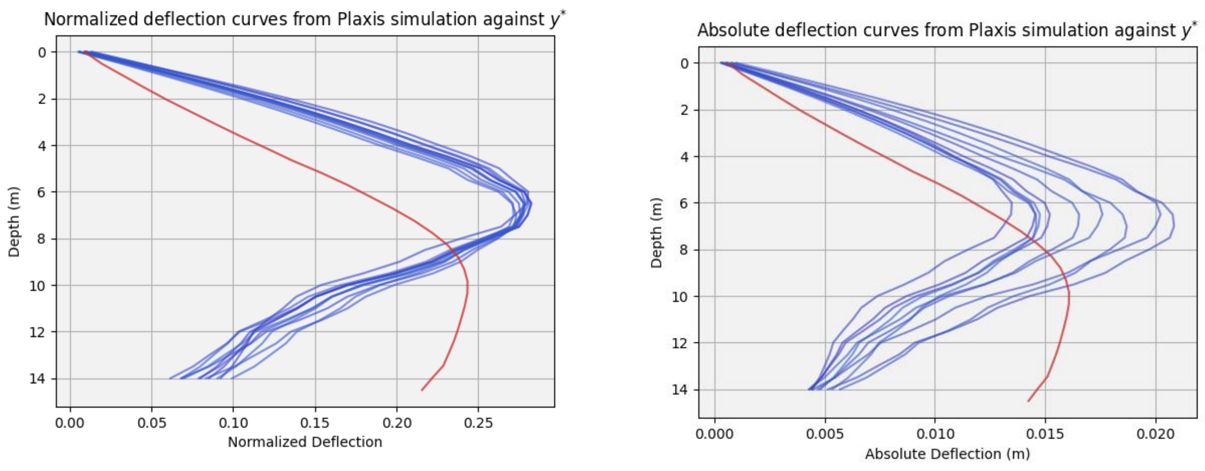


Figure 8a,8b: Normalized and absolute deflection curves from the 20 Plaxis2D simulations against $\boldsymbol{y}^*$(in red)

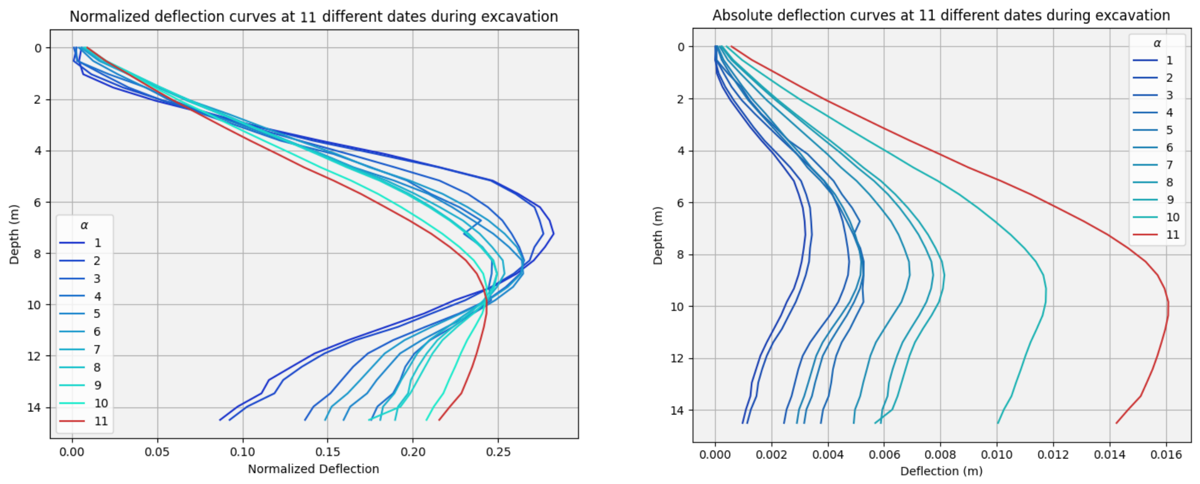


Figure 9a,9b: Normalized and absolute deflection curves from the experimental data against $\boldsymbol{y}^*$(in red)

3.5 Applying the BFPM Approach.

We start by training $\mathcal{M}_L$ on solely the Plaxis2D data $\mathcal{D}_L$. The hyperparameters $\boldsymbol{\lambda}_L$ for $\mathcal{M}_L$ involve a [25] architecture with a logistic activation function and the L-BFGS optimization algorithm. Hyperparameter tuning was conducted in a similar fashion to that addressed in section 2.3. $\mathcal{M}_L$ performs as a surrogate for the Plaxis2D model from which final-stage deflection profiles can be generated for possible variations of soil parameters. We then pass the high-fidelity dataset's inputs $\boldsymbol{x}_{H,i} \in \boldsymbol{x}_H$ through $\mathcal{M}_L$ and append them to $\mathcal{D}_H$, with which we train $\mathcal{M}_H$. This model will focus on learning the relationship between α and $\boldsymbol{y}$:

$$\widehat{\boldsymbol{y}}_{L,i} \approx \mathcal{M}_L\big(\boldsymbol{x}_{H,i}; \boldsymbol{\lambda}_L\big)|_{\mathcal{D}_L}$$

$$\mathcal{D}_H^* = \big\{\big(\big[\overline{\boldsymbol{x}}, \alpha_i = i,\ \widehat{\boldsymbol{y}}_{L,i}\ \big], \boldsymbol{y}_{H,\alpha=i}\big)\big\}_{i=1}^{N_H=10}$$

$$\Longrightarrow \widehat{\boldsymbol{y}}_L^* \approx \mathcal{M}_L(\boldsymbol{x}^*\,; \boldsymbol{\lambda}_L)|_{\mathcal{D}_L} \tag{24}$$

$$\widehat{\boldsymbol{y}}_H^* \approx \mathcal{M}_H(\boldsymbol{x}^*\,; \boldsymbol{\lambda}_H)|_{\mathcal{D}_H^*} \tag{25}$$

For benchmark purposes, we train another model $\mathcal{M}_G$ on the general, combined dataset $\mathcal{D}$. $\boldsymbol{\lambda}_G$ involves a [30,30] architecture with a logistic activation function and the L-BFGS optimization algorithm. Figure 9 shows how the real deflection profiles evolve significantly with time; whilst the normalized shape remains similar. This brings into question the importance of using high-quality data, such as more recent experimental recordings, rather than all the available data.

Let $\mathcal{D}_S$ denote a subset of combined dataset $\mathcal{D}$, thus $\mathcal{D}_S \in \mathcal{D}$, where $\mathcal{D}_S$ constitutes only the last $\hbar$ experimental measurements, excluding the true value, where $\hbar$ can also be optimized:

$$\mathcal{D}_{\hbar} \coloneqq \{[\bar{\boldsymbol{x}}, \alpha_i = i], \boldsymbol{y}_{\alpha=i}\}_{i=11-\hbar}^{N_H=10} \;\; ; \;\; \mathcal{D}_{\hbar} \in \mathcal{D}_H \;\; , \hbar \in \{1, \dots, N_H\} \tag{26}$$

$$\therefore \mathcal{D}_{\hbar=N_H} \equiv \mathcal{D}_H$$

$$\mathcal{D}_S \coloneqq \{\mathcal{D}_L, \mathcal{D}_{\hbar}\} \tag{27}$$

This leads to our final model $\mathcal{M}_S$, trained on the shortened combined dataset $\mathcal{D}_S$ and using the same hyperparameters as $\mathcal{M}_G$, thus $\boldsymbol{\lambda}_S \equiv \boldsymbol{\lambda}_G$. It is important to go into depth about these models as to establish a confident benchmark from which we can highlight the true merits of MFNNs and the BFPM approach. We do so later.

Returning to the notion of transfer learning and following eq.12, we combine the low- and high-fidelity predictions using a function $\mathcal{F}(\cdot\,;\boldsymbol{\zeta})$ in a way that minimizes the loss function $\mathcal{L}$ between the true and predicted deflection profiles, thus learning the function's parameters $\boldsymbol{\zeta}$:

$$\hat{\boldsymbol{y}}_{N=2}^{*} = \mathcal{F}\left(\{\boldsymbol{x}_{L,i}, \boldsymbol{y}_{L,i}\}_{i=1}^{N_L=20}\right) + \mathcal{F}\left(\{\boldsymbol{x}_{H,i}, \boldsymbol{y}_{H,i}\}_{i=1}^{N_H=10}\right) = \mathcal{F}(\mathcal{D}_L) + \mathcal{F}(\mathcal{D}_H) = \mathcal{F}(\hat{\boldsymbol{y}}_L^{*}, \hat{\boldsymbol{y}}_H^{*}; \boldsymbol{\zeta}) \tag{28}$$

From the expected order of the deflection shape, we use a quadratic function for $\mathcal{F}(\cdot\,;\boldsymbol{\zeta})$:

$$\mathcal{F}(\hat{\boldsymbol{y}}_L^{*}, \hat{\boldsymbol{y}}_H^{*}; \boldsymbol{\zeta}) = A\hat{\boldsymbol{y}}_L^{*} + B{\hat{\boldsymbol{y}}_L^{*}}^2 + C\hat{\boldsymbol{y}}_H^{*} + D{\hat{\boldsymbol{y}}_H^{*}}^2 + E\hat{\boldsymbol{y}}_L^{*}\hat{\boldsymbol{y}}_H^{*} + F \tag{29}$$

$$\therefore \boldsymbol{\zeta} \coloneqq [A, B, C, D, E, F] \tag{30}$$

The parameters in $\boldsymbol{\zeta}$ are optimized using a grid search algorithm. We search between [0,5] in increments of 0.1 for each parameter and exhaustively iterate through all possible combinations, following equation 31 to solve for $\boldsymbol{\zeta}^{*}$.

$$\boldsymbol{\zeta}^{*} = \arg\min_{\boldsymbol{\zeta}} \mathcal{L}_{MSE}(\boldsymbol{y}^{*}, \hat{\boldsymbol{y}}_{N=2}^{*}) \tag{31}$$

$$\boldsymbol{\zeta}^{*} = [0.1\,, 0\,, 2.1\,, 0.5\,, 0\,, 0] \tag{32}$$

$$\Longrightarrow \mathcal{F}(\hat{\boldsymbol{y}}_L^{*}, \hat{\boldsymbol{y}}_H^{*}; \boldsymbol{\zeta}^{*}) = 0.1\hat{\boldsymbol{y}}_L^{*} + 2.1\hat{\boldsymbol{y}}_H^{*} + 0.5{\hat{\boldsymbol{y}}_H^{*}}^2 \tag{33}$$

Notice the low- or zero-valued coefficients for variables containing $\hat{\boldsymbol{y}}_L^{*}$, the low-fidelity model's prediction. This is because the low-fidelity model is excessively inaccurate versus the BFPM prediction, in terms of shape, position of d^{*} and absolute deflection values; there is little valuable information in $\hat{\boldsymbol{y}}_L^{*}$ thus a greater weighting is placed on $\hat{\boldsymbol{y}}_H^{*}$. We examine this more closely in the discussion. The function $\mathcal{F}(\cdot\,;\boldsymbol{\zeta})$ combines the desired shape with the correct magnitude. In more complex tasks, another NN may be necessary to best solve for $\boldsymbol{\zeta}^{*}$ which is where PINNs can be beneficial if $\mathcal{D}_H$ is too sparse.

We have hence defined four models, $\mathcal{M}_L$, $\mathcal{M}_H$, $\mathcal{M}_G$ and $\mathcal{M}_S$ and optimized all accordingly. The first, third and fourth models will serve as benchmarks, representing predictions from simulation-only data and from an agglomerated dataset, respectively. We have also defined the optimal mapping function $\mathcal{F}(\cdot\,;\boldsymbol{\zeta}^{*})$ which combines the predictions from $\mathcal{M}_L$ and $\mathcal{M}_H$.

4. Results and Discussion

4.1 Establishing Benchmarks from $\mathcal{M}_L$, $\mathcal{M}_G$ and $\mathcal{M}_S$

The Mean Squared Error (MSE) between true deflections $\boldsymbol{y}^*$ and predictions $\hat{\boldsymbol{y}}^*$ proves a simple and useful metric when evaluating the fit of our models. The elementary operators, low dimensionality of our data, and relatively small size of our datasets keep the evaluation time and computational cost low, facilitating the often-time-consuming processes in hyperparameter optimization such as grid searching. It also provides quantitative backing to visual inspections.

Another useful metric is the cross-correlation between two series: a measure of similarity as a function of the displacement of one relative to the other. A general definition often found in signal processing texts follows:

$$c_k = \sum_n a_{n+k} \cdot \bar{v}_n \tag{34}$$

Where a and v are N-dimensional arrays and $\overline{\boldsymbol{x}}$ denotes the complex conjugate of $\boldsymbol{x}$. Applied to our problem, we find:

$$\mathcal{L}_{CC} = \boldsymbol{y}^* \cdot \overline{\hat{\boldsymbol{y}}^*} = \sum_{k=1}^{K_y = \|\boldsymbol{y}\|_0 = 29} y_k^* \cdot \overline{\hat{y}_k^*} = \sum_{k=1}^{29} y_k^* \cdot \hat{y}_k^* \quad ; \quad y_k^*, \hat{y}_k^* \in \mathbb{R}\ \forall k \tag{35}$$

Where $\|\boldsymbol{x}\|_0$ denotes the L0 norm of $\boldsymbol{x}$, or hence the number of non-zero elements in $\boldsymbol{x}$. This provides another quantitative form of assessment for our predictions. A higher value of $\mathcal{L}_{CC}$ indicates a closer match between the two series. Note that we expect small values of $\mathcal{L}_{CC}$ for all evaluations because $y_k^*, \hat{y}_k^* \ll 1$.

First, we investigate $\mathcal{M}_G$, $\mathcal{M}_S$ and the use of $\hbar$ to filter $\mathcal{D}$ and improve predictions. Typically, there is no need to reduce the training dataset as NNs can assign the necessary weights and biases to effectively nullify training data which it deems a hindrance. Stacking networks, like in the MFPM approach, can assist further. However, due to the small database size, and our desire to keep it small, and the singular network in $\mathcal{M}_G/\mathcal{M}_S$, we address this manually.

When training on $\mathcal{D}$, $\mathcal{M}_G$ places insufficient importance on the feature α_i of $\boldsymbol{x}$, moulding itself excessively on the deflection shapes from initial stages, which we know to be significantly different from $\boldsymbol{y}^*$, suggesting a low-valued $\hbar$ is optimal. Conversely, setting $\hbar$ too low, for example using only the latest experimental recording, or two previous recordings, can place insufficient weight on the experimental data, moulding the prediction too closely to the Plaxis2D curves, which we also know to be significantly different from $\boldsymbol{y}^*$. Through grid searching, we find the optimum, $\hbar^*$, as 7. We plot the predictions from $\mathcal{M}_G$ and $\mathcal{M}_S$, $\hat{\boldsymbol{y}}^*_{\mathcal{M}_G}$ and $\hat{\boldsymbol{y}}^*_{\mathcal{M}_S}$ respectively, against the true deflections $\boldsymbol{y}^*$ (ref. fig.10).

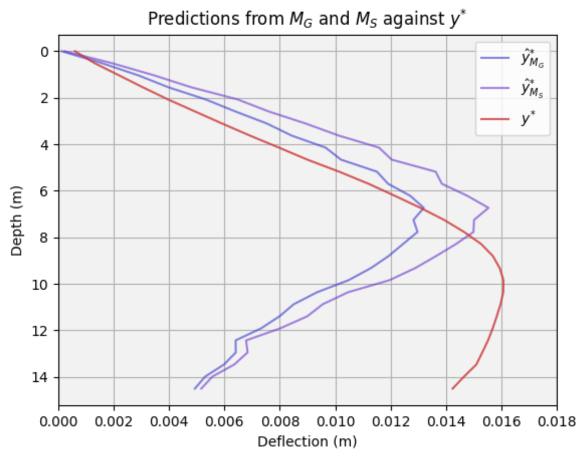


Figure 8: Predicted deflections from models $\mathcal{M}_G$ and $\mathcal{M}_S$ trained on agglomerated datasets $\mathcal{D}$ and $\mathcal{D}_S$, respectively, against $\boldsymbol{y}^*$.

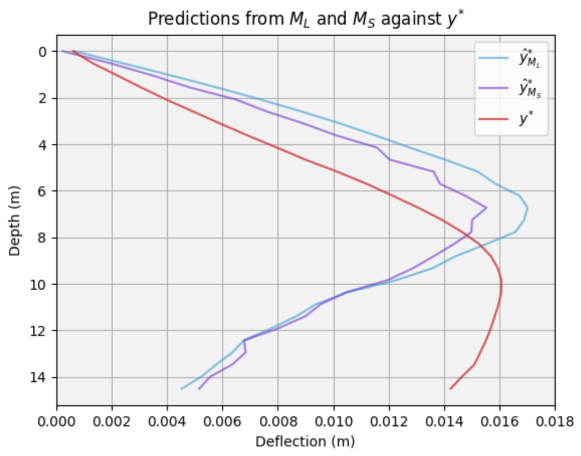


Figure 9: Predicted deflections from models $\mathcal{M}_L$ and $\mathcal{M}_S$ trained on low-fidelity dataset $\mathcal{D}_L$ and agglomerated dataset $\mathcal{D}_S$, respectively, against $\boldsymbol{y}^*$.

Notice the significant improvement that such a simple adjustment can make, with no need to increase the training dataset size or the network complexity. Whilst $\mathcal{M}_S$ acts solely as a benchmark, we have taken the time to optimize it as to, once again, emphasize the importance of evaluating against the best possible predictions which do not use multi-fidelity methods. $\mathcal{M}_S$ outperforms $\mathcal{M}_G$ in all relevant metrics (ref. table 3). There is still room for improvement when we compare either $\hat{\boldsymbol{y}}^*_{\mathcal{M}_G}$ or $\hat{\boldsymbol{y}}^*_{\mathcal{M}_S}$ to $\boldsymbol{y}^*$, particularly for $d > d^*$.

Interestingly, the BFPM model does not suffer from accuracy reduction due to the inclusion of early-stage experimental data. This is likely because the coupling of networks allows it to better place importance on later-construction-stage data. As such, it was trained with the complete experimental dataset $\mathcal{D}_H$ rather than the subset $\mathcal{D}_{\hbar}$.

We evaluate the performance of the low-fidelity model $\mathcal{M}_L$ against the optimal, combined dataset model $\mathcal{M}_S$. The predictions $\hat{\boldsymbol{y}}^*_{\mathcal{M}_S}$ and $\hat{\boldsymbol{y}}^*_L$ are comparable, although neither particularly accurate (ref. fig11).

Note that $\max \hat{\boldsymbol{y}}^*_L \gg \max \hat{\boldsymbol{y}}^*_{\mathcal{M}_S}$. From a safety perspective, it is better to overestimate the maximum deflection and over-specify construction materials, than assume values which risk falling below the true deformation. Thus $\mathcal{M}_L$ is likely preferred to $\mathcal{M}_S$. We go into more quantitative detail later, and first move on to discussing the results of the bi-fidelity approach.

4.2 Evaluating the BFPM Approach

The neural network surrogate $\mathcal{M}_L$ underperforms when predicting final-stage deflection profiles for unfamiliar combinations of soil parameters $\boldsymbol{x}$. There is a clear mismatch between real-world behaviour and theoretical behaviour which Plaxis2D fails to capture. A multi-fidelity approach maximizes the information gain from $\mathcal{D}$ and addresses this gap.

Define the BFPM prediction for $\alpha = 11$ as $\hat{\boldsymbol{y}}^*$, the optimised non-linear mapping from the low-fidelity prediction $\hat{\boldsymbol{y}}^*_L$ and high-fidelity prediction $\hat{\boldsymbol{y}}^*_H$, following on from equation 33:

$$\hat{\boldsymbol{y}}^* := \mathcal{F}(\hat{\boldsymbol{y}}^*_L, \hat{\boldsymbol{y}}^*_H; \boldsymbol{\zeta}^*) \tag{36}$$

Where $\widehat{\boldsymbol{y}}_L^*$ and $\widehat{\boldsymbol{y}}_H^*$ are defined in equations 24 & 25. We plot $\widehat{\boldsymbol{y}}^*$ against the true value $\boldsymbol{y}^*$, the benchmark $\widehat{\boldsymbol{y}}_L^*$ and the penultimate experimental datapoint $\boldsymbol{y}_{\alpha=10}$ (ref. fig.12).

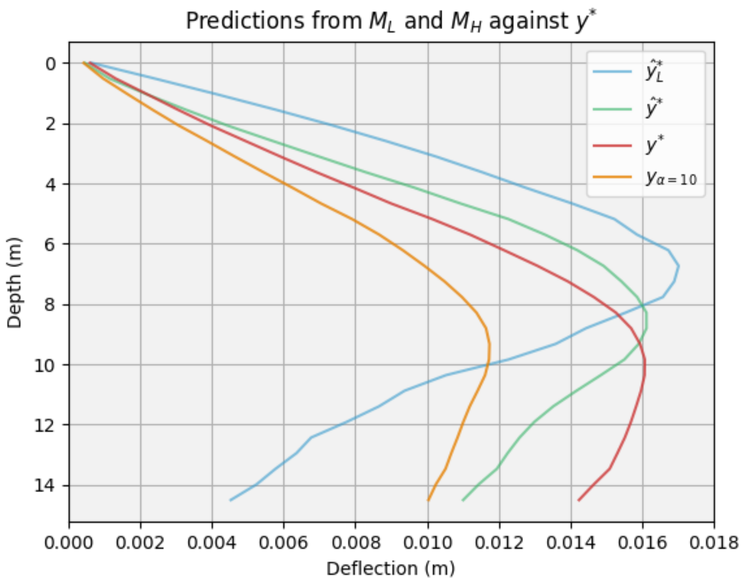


Figure 10: Predicted deflections from models $\mathcal{M}_L$ and $\mathcal{M}_H$ against $\boldsymbol{y}^*$ and $\boldsymbol{y}_{\alpha=10}$.

The BFPM prediction $\widehat{\boldsymbol{y}}^*$ shows clear and significant improvement against $\widehat{\boldsymbol{y}}_L^*$ and $\widehat{\boldsymbol{y}}_{\mathcal{M}_S}^*$, and positively builds from $\boldsymbol{y}_{\alpha=10}$. The shape is more closely aligned with $\boldsymbol{y}^*$, the predicted d^* is closer to the real d^* and the maximum deflection values are also closer (ref. table 3). Whilst $\widehat{\boldsymbol{y}}^*$ for $d > d^*$ is still below that for $\boldsymbol{y}^*$, there is improvement against both $\widehat{\boldsymbol{y}}_L^*$ and $\boldsymbol{y}_{\alpha=10}$.

Table 3: Quantitative evaluation of $\boldsymbol{y}_{\alpha=10}, \mathcal{M}_G, \mathcal{M}_S, \mathcal{M}_L$ and $\mathcal{M}_H$ predictions against $\boldsymbol{y}^*$ for MSE, cross-correlation and key values in deflection curve.

	$\mathcal{L}_{MSE}$	$\mathcal{L}_{CC}$	d^* (m)	$max\,\boldsymbol{y}$ (m)
$\boldsymbol{y}^*$	---	---	9.6	0.0161
$\boldsymbol{y}_{\alpha=10}$	1.109e-05	0.00318	9.3	0.0118
$\widehat{\boldsymbol{y}}_{\mathcal{M}_G}^*$	2.582e-05	0.00298	6.7	0.0132
$\widehat{\boldsymbol{y}}_{\mathcal{M}_S}^*$	2.390e-05	0.00340	6.7	0.0155
$\widehat{\boldsymbol{y}}_{\mathcal{M}_L}^*$	2.969e-05	0.00359	6.7	0.0170
$\widehat{\boldsymbol{y}}^*$	3.377e-06	0.00420	8.6	0.0161

Reviewing table 3 clearly emphasizes the success of multi-fidelity approaches. As mentioned before, the performance of $\mathcal{M}_L$ and $\mathcal{M}_S$ are comparable, as expected. On the other hand, $\mathcal{M}_H$ massively outperforms the other predictions: the MSE is an order smaller, the cross-correlation improves by over 30% from the next-best prediction and the predicted maximum deflection value is closer to the true value. The only aspect where it falls short is in estimating d^*, likely because the inaccuracy in $\mathcal{M}_L$ skews the value irrecoverably. Notice that $\max \widehat{\boldsymbol{y}}^* \equiv \max \boldsymbol{y}^*$. This is coincidence and we do not claim this to be a direct result of applying BFPM.

4.3 General Discussion

Instead of learning the 29 outputs which make up $\boldsymbol{y}$, we also considered first fitting the data to an N-degree polynomial (likely a cubic or quartic) and learning the $N + 1$ parameters instead, where clearly $N + 1 \ll 29$. This would have been a valuable approach to reduce the model complexity if dataset size proved insufficient. However, this was not the case and we thought it unnecessary to introduce another potential source of error through approximation.

A significant area of difficulty was optimizing the networks in $\mathcal{M}_L$ and $\mathcal{M}_H$, particularly the computational cost of iterating through all possible combinations of qualitative and quantitative hyperparameters. This could be reduced by setting reasonable limits to explore or by ruling out certain parameters using mathematical deduction. Whilst network optimization is of paramount importance in any NN task, the focus of our work was on applying novel multi-fidelity methods to real-world scenarios.

Another area of difficulty, although also one common to most ML tasks, was cleaning and preparing the dataset. Extracting results from Plaxis2D was convoluted, although possible to automate. Automation is an important, arguably essential, step when generating datasets from programs, particularly ones with long simulation times. Manipulating units proved trivial. We also considered learning the normalized deflection shapes instead of the absolute values, which would be most beneficial if the difference in elements of $\boldsymbol{y}$ were very small.

4.3.1 Predicting Curves from Earlier Construction Stages

We focused on predicting the deflection profile at the final stage of construction, $\boldsymbol{y}_{\alpha=11}$. This is the most useful and most important profile, and as discussed before, likely the hardest to predict given the significant change from deflection data at the preceding stage(s). The task could undoubtedly have been repeated to instead predict the shape at earlier stages, although using multi-fidelity neural networks would likely have been excessive. For instance, take curves $\boldsymbol{y}_{\alpha=1}$ and $\boldsymbol{y}_{\alpha=2}$ from figure 9b. The difference is so minimal, and in practical terms insignificant, that using a multi-fidelity approach would have hindered more than helped.

Furthermore, the research could have extended to a wider range of measurements beyond deflection, ones which also evolve with time and are key to ensuring safety standards are met. For example, the vibrating wire strain gauges used to monitor the cofferdam also recorded axial loads, strains and bending moments, all of which Plaxis2D could also predict.

4.3.2 Dataset Size

Conversely to the previous numerical example on predicting the maximum deflection at any point during construction (refer to section 3.3), we use a much smaller dataset in our BFPM approach: only 30 datapoints compared to the 966 used in the former. Generating $\mathcal{D}_L$ is time consuming and computationally expensive, whilst gathering more data for $\mathcal{D}_H$ is often unfeasible. Expense significantly increases as the simulations required become more complex, such as using Plaxis3D or the finest possible meshes in FE models.

As such, we seek to maximize prediction accuracy without increasing data requirements, which is shown to be feasible when comparing $\hat{\boldsymbol{y}}^*$ to $\hat{\boldsymbol{y}}^*_{\mathcal{M}_G}$: the same dataset applied in different manners has differing results. We could investigate how much smaller $\mathcal{D}$ can be whilst still generating results of comparable accuracy, or hence minimize N with respect to $\mathcal{L}$. Practically, N_L must be large enough for the model to learn the surrogate mapping between soil parameters $\boldsymbol{x}$ and deflections $\boldsymbol{y}$, whilst N_H must be large enough for the model to learn how the deflection shape evolves with time. As discussed above, the latter could only require a handful of well-selected datapoints.

5. Conclusions

Neural Network (NN) surrogate models are trained on datasets of varying fidelity. Two multi-fidelity learning methods to train said models, Multi-fidelity Weighted Learning (MFWL) and Multi-fidelity Prediction Mapping (MFPM), are presented and investigated, firstly on a trivial numerical example, then for two applications in a real-world deep excavation task. The approaches explore how we can maximise the models' prediction accuracies by optimizing the way in which multi-fidelity data is used. An initial model is trained on the low-fidelity data and is subsequently adapted using the high-fidelity data, outperforming relevant benchmarks such as the model trained on the agglomerated dataset.

The method was particularly useful when monitoring the deflection of a retaining wall during the construction of a deep excavation. The NN surrogate underperforms when predicting final deflection profiles for unfamiliar combinations of soil parameters. There is a clear mismatch between real-world consolidation and the theoretical consolidation which the surrogate is trained on. A combination of Plaxis2D simulation data (representing the low-fidelity dataset) and experimental site data (representing the high-fidelity dataset) was used to train a MFNN through inductive transfer learning (TL) on a Bi-fidelity Prediction Mapping (BFPM) approach to better predict the final deflection profile. The low-fidelity data served to learn the surrogate mapping between soil parameters and deflection shape, whilst the site data learnt the mapping between construction stage and shape. Finally, a non-linear function combined the two and refined the overall prediction. The approach was effective, and the accuracy improved significantly against the original surrogate and benchmarks which did not consider data fidelity. A desirable addition would be Plaxis3D simulation data, offering another level of fidelity with which to further investigating the MFPM approach.

Despite the notable advantages of MFNNs and the BFWL approach, there are still aspects which can be improved or further studied. Transductive TL has exciting properties in improving the generalisation of ML frameworks, envisioning a one-model-fits-all future. To that extent, MFNNs and TL could be used to facilitate the learning of equally important variables such as bending moments or shear forces, particularly if their profiles are similar or related. Alternatively, the approach could be applied to similar construction sites through inductive TL to pretrain new surrogate models. Any system in which there is low- and high-fidelity data, such as a digital twin and site data on a construction site, could likely benefit from multi-fidelity methods, especially if there is a significant disparity between the two.

Secondly, and most importantly, groundwork has been laid to investigate the use of higher levels of fidelity in MFPM and MFWL. Determining how many levels are both realistic and beneficial when modelling real-world problems is of interest, particularly to geotechnical engineering problems. Here, variables are often dependant on time and the quality of data is inconsistent, thus numerous fidelities are available. In doing so, we may reduce the quantity of data needed to train surrogate neural networks whilst also improving their accuracy, thus making deep excavation projects more efficient and safer.

6. 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] M. Popescu, V. E. Balas, L. Perescu-Popescu, N.Mastorakis, *Multilayer Perceptron and Neural Networks* (2009)

[3] D. Liu, J. Nocedal, *On the limited memory BFGS method for large scale optimization* (1989)

[4] J. Bergstra, Y. Bengio, *Random search for hyper-parameter optimization* (2012)

[5] A. Svalova, P. Helm, D. Prangle, M. Rouainia, S. Glendinning, D. J. Wilkinson, *Emulating computer experiments of transport infrastructure slope stability using Gaussian processes and Bayesian inference* (2021)

[6] M. N. A. Raja, S. K. Shukla, *Predicting the settlement of geosynthetic-reinforced soil foundations using evolutionary artificial intelligence technique* (2021)

[7] S. J. Pan, Q. Yang, *A Survey on Transfer Learning* (2010)

[8] S. De, J. Britton, M. Reynolds, R. Skinner, K. Jansen, A. Doostan, *On transfer learning of neural networks using bi-fidelity data for uncertainty propagation* (2019)

[9] S. Chakraborty, *Transfer Learning based multi-fidelity physics informed deep neural network* (2020)

[10] X. Meng, G. Em Karniadakis, *A composite NN that learns from multi-fidelity data: application to function approximation and inverse PDE problems* (2019)

[11] M. Guo, A. Manzoni, M. Amendt, P. Conti, J. S. Hesthavaen, *Multi-fidelity regression using artificial neural networks: Efficient approximation of parameter-dependent output quantities* (2022)

[12] L. Le Gratiet, J. Garnier, *Recursive co-kriging model for design of computer experiments with multiple levels of fidelity* (2014)

[13] Y. Xu, Y. Zhou, P. Sekula, L. Ding, *Machine Learning in construction: from shallow to deep learning* (2021)

[14] B. B. Shiel, M. A. Mooney, S. K. Suryasentana, H. Zhu, *Machine Learning to inform tunnelling operations: recent advances and future trends* (2020)

[15] P. Hensman, *Investigation of soil-structure interaction for deep excavations* (2023)

7. Appendix

7.1 Risk Assessment

Risks outlined at the beginning of the project reflected well on the hazards encountered during the project. No changes were required.