Acta Geotechnica (2023) 18:1681–1698
https://doi.org/10.1007/s11440-022-01706-2

RESEARCH PAPER

Ready-to-use deep-learning surrogate models for problems with spatially variable inputs and outputs

Xuzhen He[1] · Haoding Xu[1] · Daichao Sheng[1]

Received: 24 September 2021 / Accepted: 9 September 2022 / Published online: 23 September 2022

Abstract

Data-driven intelligent surrogate models gain popularity recently. Particularly in Monte-Carlo-style stochastic analysis, the influencing factors are considered as inputs, the quantities of interest are considered as outputs, and cheaper-to-evaluate surrogates models are built from a small amount of sample data and are used for the full Monte-Carlo analysis. This paper presents a framework with three innovations: (1) we build surrogate models for a particular problem that covers any possible material properties or boundary conditions commonly encountered in practice, so the models are ready to use, and do not require new data or training anymore. (2) The inputs and outputs to the problem are both spatially variable. Even after discretization, the input and output sizes are in the order of tens of thousands, which is challenging for traditional machine-learning algorithms. We take the footing failure mechanism as an example. Two types of neural networks are examined, fully connected networks and deep neural networks with complicated non-sequential structures (a modified U-Net). (3) This study is also the first attempt to use U-Nets as surrogate models for geotechnical problems. Results show that fully connected networks can fit well simple problems with a small input and output size, but fail for complex problems. Deep neural networks that account for the data structure give better results.

Keywords Deep learning · Deep neural network · Footing failure mechanism · U-Net

Abbreviations

B Strip footing width
c Cohesion
c' Normalised cohesion, $c' = \frac{c}{c_r}$
$\mathbf{C}'$ Representation of the normalised cohesion field as a 2D array
c_r Reference strength
l_x Scale of fluctuation (horizontal)
l_y Scale of fluctuation (vertical)
q_0 Overburden load
q_0' Normalised overburden load $\frac{q_0}{c_r}$
q_u' Normalised bearing capacity $\frac{q_u}{c_r}$
s_u Undrained strength
$\mathbf{U}_\mathbf{u}'$ Representation of the normalised failure velocity field (horizontal) as a 2D array
$\boldsymbol{v}_u$ Failure velocity, $\boldsymbol{v_u} = (u_u, v_u)$
$\boldsymbol{v}_u'$ Normalised failure velocity
$\mathbf{V}_u'$ Representation of the normalised failure velocity field (vertical) as a 2D array
w Spatially variable weight to calculate errors
$\boldsymbol{x}$ Coordinate vector
$\boldsymbol{x}'$ Normalised coordinate vector, $\boldsymbol{x}' = \frac{\boldsymbol{x}}{B}$
γ Unit weight
γ' Normalised unit weight, $\gamma' = \frac{\gamma B}{c_r}$
$\boldsymbol{\Gamma}'$ Representation of the normalised unit weight field as a 2D array
$\boldsymbol{\sigma}_u$ Failure stress
$\boldsymbol{\sigma}_u'$ Normalised failure stress, $\boldsymbol{\sigma}_u' = \frac{\boldsymbol{\sigma}_u}{c_r}$
ϕ Friction angle
$\boldsymbol{\Phi}$ Representation of the normalised friction angle field as a 2D array

List of symbols

COV Coefficient of variation
CNN Convolutional neural network
FCN Fully connected network

✉ Xuzhen He
xuzhen.he@uts.edu.au

Haoding Xu
haoding.xu@student.uts.edu.au

Daichao Sheng
daichao.sheng@uts.edu.au

[1] School of Civil and Environmental Engineering, University of Technology Sydney, Ultimo, NSW 2007, Australia

1 Introduction

Numerical modelling and simulations of geotechnical problems have seen significant advancements in recent decades. Extensive new models have been made regarding the discretisation of governing equations, constitutive modelling, and boundary treatments (Fig. 1), which all contribute to more reliable predictions, and even make some previously impossible tasks possible. For example, to tackle the issue of mesh distortion and grid tangling in large deformation problems, meshless methods are developed, including but not limited to the particle finite element method [16], the material point method [26], and the smooth particle hydrodynamics [11]. Various coupled numerical models are also developed to model multi-physics problems such as soil–water interaction [4] and solute transport with porous fluid [2]. Many soil models are developed for the various kinds of soils in nature and the distinctive soil behaviours, including but not limited to the hypoplastic models [3], the bounding surface models for sands [1], and the unsaturated soil models [20, 21].

Despite all improvements and advantages of advanced numerical models, they are hardly used in real-world projects, where practitioners still prefer well-established models like the linear elastic Mohr–Coulomb model and tools like slip-line methods. The application of these advanced models is often restricted to validations against well-controlled experiments or extensively studied cases. Some factors that may hinder their accessibility in practice are:

(1) Research outcomes are often implemented in in-house codes that are not available to practitioners.
(2) Good user interface and documentation are often lacking such that the very steep learning curve drives practitioners away.
(3) Some models are implemented in general-purpose software packages. The operation of these models is often cumbersome compared with dedicated tools like limit analysis for slopes. Defining a single simulation involves several components, including domains, materials, boundaries and mesh. Incorrect operation in any component would result in unpredictable results.
(4) The computational time for coupled multi-physics problems is time-consuming, which makes stochastic analysis and optimisation practically impossible because these tasks all require many such analyses [32].

One POSSIBLE solution is to use surrogate models, which are mostly based on machine learning in the present day. These surrogate models only solve a particular problem, e.g. the stability of slopes, the bearing capacity of foundations, or the ground settlement due to excavation. The influencing factors for these problems (e.g. material parameters and boundary conditions in Fig. 1) are considered as inputs. And the quantities of interest are considered as outputs. Surrogate models are then built based on sample data—results from advanced models. Therefore, surrogate models are equivalent to the advanced models for a particular problem, but they have the advantage of being easy to operate and getting results in a shorter time.

Surrogate models are not a new topic and are extensively used in stochastic analysis [14] and optimisation [29]. In Monte-Carlo-style stochastic analysis, the influencing factors are usually random variables/fields of material properties, and the quantity of interest is often safety or reliability-related. Cheaper-to-evaluate surrogate models are built from a small amount of sample data and are used for the full Monte-Carlo analysis. In optimisation, surrogate models are used as the objective functions. However, most applications of surrogate models are only for some specific cases. When a new case is studied, the surrogate model requires new data and re-training, which is an ISSUE. This paper presents some techniques to solve this problem with three innovations: (1) we will build surrogate models for a particular problem that covers any possible material properties or boundary conditions commonly encountered in practice, so the models are ready to use, and do not require new data or training anymore. (2) On this occasion, the relationship between inputs and outputs would be extremely complicated, particularly when the inputs and outputs are both spatially variable. Even after discretization, the input and output sizes are in the order of tens of thousands, and regular machine-learning tools cannot give reasonable results. We will explore the use of deep neural networks that account for the data structure. (3) This study is also the first attempt to use U-Nets as surrogate models for geotechnical problems. With these new techniques, the use of deep-learning surrogate models to facilitate the application of advanced numerical models becomes really POSSIBLE.

Data-driven intelligent surrogate models have seen many successful applications [6, 9, 10, 17, 23]. Regarding surrogate models for stochastic analysis with spatial variability, we presented a deep-learning surrogate model to predict the bearing capacity of strip footings in our previous study [33]. The deep neural network is stacks of convolutional layers, average pooling layers, locally connected layers, and fully connected layers. It is trained with a big dataset covering any soil properties, spatial variabilities, or boundary conditions encountered in practice. The model is very accurate for unseen testing data (3.3% error). If equipped with a good user interface, this deep-learning model is a perfect tool for practitioners. It is also shown

that regular machine-learning algorithms cannot reproduce reliable models (more than 20% error) for such a complicated problem. Although no advanced models are used to generate the sample data in the study, the framework is generic and can be used on other occasions.

In our previous study [33], of the four inputs, three are spatial fields of material parameters, and one is a scalar measuring loading condition. Although the inputs are complex, the output is only a scalar—the bearing capacity. For some problems, the quantities of interest can be spatially variable (Fig. 1, e.g. the failure mechanism of slopes, the displacement field of the ground, and the concentration field of hazardous materials in soils) or temporally variable (Fig. 1, e.g. the settlement of embankment with time and the creeping of landslides). These problems are relatively more complicated, and a deep-learning model would require a different structure from the one used in He et al. [33]. In this study, we present how to build a deep-learning surrogate model for problems where the output is also a field. The prediction of footing failure mechanism is taken as an example, but this generic framework easily transcends to other problems (e.g. slope failure mechanism [18], landslide runout [31], and seepage in unsaturated soil [12]). In Sect. 2, we explain the inputs and outputs for the problem of footing failure mechanism, the setup of finite element simulations, and how to evaluate the failure

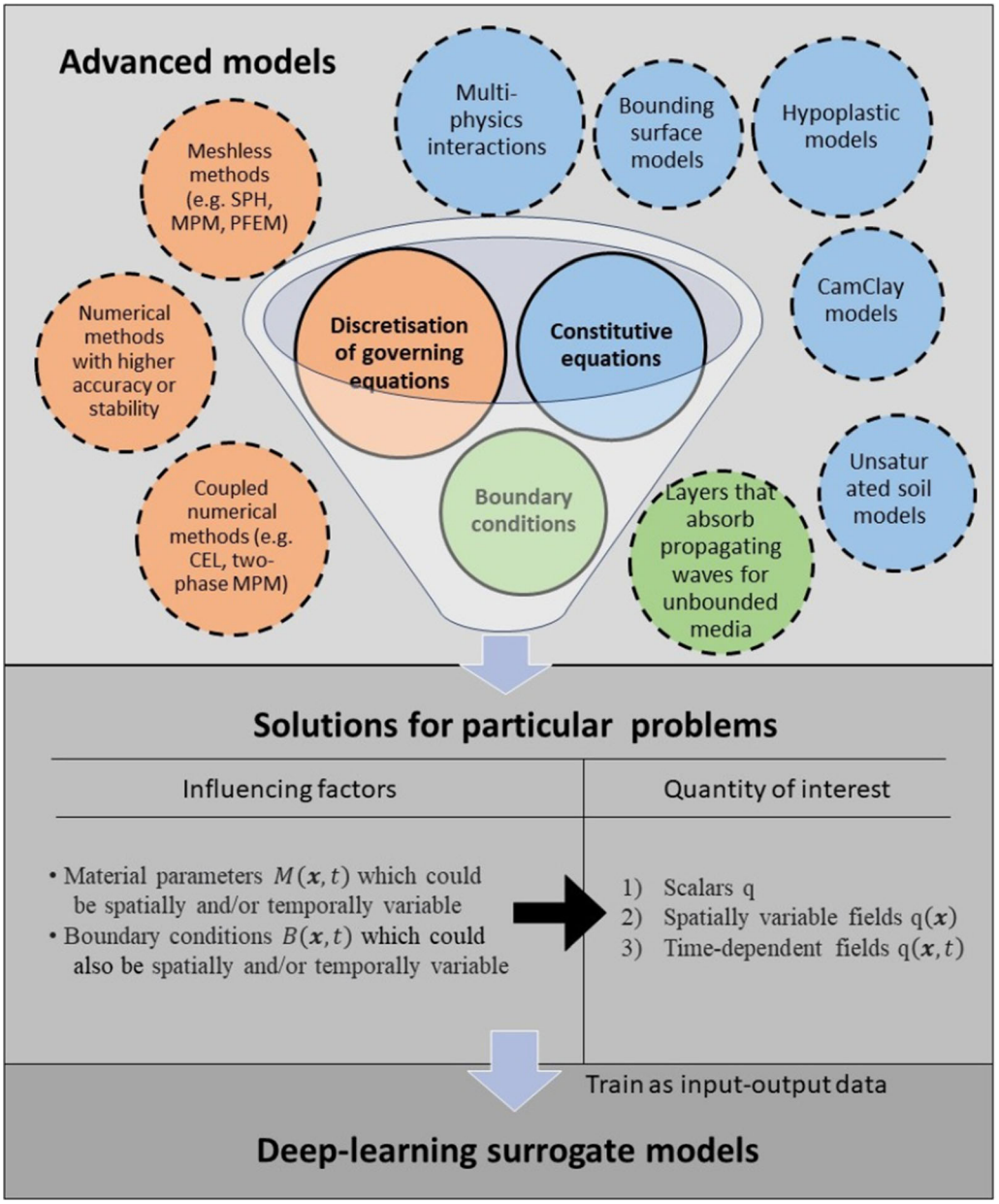


Fig. 1 Framework to make advanced models more accessible using deep-learning surrogate models

mechanism. The dataset is mentioned in Sect. 3 with imposed limits for the material parameters and loading conditions. In Sect. 4, deep neural networks, some layer types, and the structure of our deep-learning model are explained. A weighted error is introduced in Sect. 5 as the loss function. Then we start to build models for a reduced problem with a smaller size of inputs and progressively increase the size until the full problem is examined.

2 Footing failure mechanism

If the soil under a strip footing is modelled by perfect plasticity, after reaching the ultimate failure condition, the soil will undergo unrestricted plastic flow (like steady-state fluid flow) with a constant stress field. The Mohr–Coulomb yield surface is adopted for soils in this study, with c denoting the cohesion and ϕ denoting the friction angle. The failure stress and velocity fields depend on the soil properties and boundary (loading) conditions. Considering spatial variability of soils (e.g. material parameters are fields), we can write a relationship as

$$[\boldsymbol{\sigma}_u(\boldsymbol{x}), \boldsymbol{v}_u(\boldsymbol{x})] = f[B, c(\boldsymbol{x}), \phi(\boldsymbol{x}), \gamma(\boldsymbol{x}), q_0] \tag{1}$$

where $\boldsymbol{x} = (x, y)$ is the coordinate vector, $\boldsymbol{\sigma}_u$ is the failure stress, $\boldsymbol{v_u} = (u_u, v_u)$ is the failure velocity, B is the width of footings, γ is the unit weight, and q_0 is the overburden load (Fig. 2a).

As illustrated in our previous study [33], we can conduct dimensional analysis to have:

$$\left[\boldsymbol{\sigma}_u'(\boldsymbol{x}'), \boldsymbol{v}_u'(\boldsymbol{x}')\right] = f\left[c'(\boldsymbol{x}'), \phi(\boldsymbol{x}'), \gamma'(\boldsymbol{x}'), q_0'\right] \tag{2}$$

With the mean of the cohesion as a reference strength c_r, all quantities in the equation are normalised. $\boldsymbol{x}' = \frac{\boldsymbol{x}}{B}$ is the normalised coordinate vector. The normalised unit weight ($\gamma' = \frac{\gamma B}{c_r}$) quantifies the relative significance of gravity and strength, and the normalised overburden load ($q_0' = \frac{q_0}{c_r}$) quantifies the relative significance of overburden and strength. Normalised cohesion is $c' = \frac{c}{c_r}$. The failure stress is normalised as $\boldsymbol{\sigma}_u' = \frac{\boldsymbol{\sigma}_u}{c_r}$, and bearing capacity is its integral along the footing width $q_u' = \int_L \boldsymbol{\sigma}_u' \boldsymbol{n} dA$. The failure velocity field is normalised as $\boldsymbol{v}_u' = \frac{\boldsymbol{v}_u}{v_0}$ where v_0 is the vertical velocity of the footing. So, the failure velocity of footing is always $\boldsymbol{v}_u' = (0, 1)$.

The failure stress and velocity fields are largely influenced by soil properties near the footing, the properties far away are often insignificant. Thus, the spatial dependence is limited to a small region ($-6.4 \le x' \le 6.4$, $0 \le y' \le 6.4$).

Although the failure stress and velocity depend on continuous fields of material parameters, a finite resolution is often adopted in numerical analysis. The scale of fluctuation for soils is larger than 0.2 m [22], so with B = 0.2–1.0 m, normalised scale of fluctuation is at least 0.2, and a mesh size of $\frac{\Delta l}{B} = 0.1$ is fine enough. The final mesh has 64 × 128 elements (Fig. 2a), each with material parameters evaluated from the continuous field with a mid-point method. There are 65 × 129 nodes, and the displacement and velocity are solved on nodes.

We thus have the following equation.

$$\left[q_u', \mathbf{U}_\mathbf{u}', \mathbf{V}_\mathrm{u}'\right] = f\left[\mathbf{C}', \mathbf{\Phi}, \mathbf{\Gamma}', q_0'\right] \tag{3}$$

Because of the limited-region dependence and the finite resolution, $\mathbf{C}'$, $\mathbf{\Phi}$, and $\mathbf{\Gamma}'$ are the soil properties represented as 2D arrays of size 64 × 128. $\mathbf{U}_{\boldsymbol{u}}'$ and $\mathbf{V}_u'$ are horizontal and vertical failure velocities, respectively, which are represented by 2D arrays of 65 × 129.

Our previous study focuses on building a deep-learning model for the bearing capacity, i.e.

$$q_u' = f_q\left[\mathbf{C}', \mathbf{\Phi}, \mathbf{\Gamma}', q_0'\right] \tag{4}$$

where the output is a scalar quantity. This study aims to build a deep-learning model for the failure mechanism, i.e.

$$\left[\mathbf{U}_\mathbf{u}', \mathbf{V}_u'\right] = f_v\left[\mathbf{C}', \mathbf{\Phi}, \mathbf{\Gamma}', q_0'\right] \tag{5}$$

where the outputs are spatially variable fields.

In our previous study, for each input $(\mathbf{C}', \mathbf{\Phi}, \mathbf{\Gamma}', q_0')$, three different numerical methods are used to evaluate the bearing capacity q_u' (for cross-validation). But the results from the finite element simulations with an implicit dynamic scheme are mainly used for training. In this study, failure velocity fields are also evaluated by this finite element model. In all simulations (Fig. 2a), the footing is 1 m wide, and the height and width of the domain are 6.4 and 12.8 m, respectively. The domain is made of 64 × 128 quadrilateral elements. The bottom is fixed and the left and right are roller boundaries.

A large elastic modulus is used, so ultimate failure is reached when the footing settlement is between 0.003 B (cases with relatively small q_u') and 0.08 B (cases with relatively large q_u'). After each displacement increment, we think the load increases only if it is greater than 1.005 of the current maximum, and we stop the simulation for a further displacement of 0.005 B if without load increase. Figure 2b shows the load/displacement curve for a homogenous soil with ϕ = 0 and γ = 0. The load does not increase from around 0.003 B and the simulation stops at around 0.008 B. The predicted bearing capacity is very close to the analytical value (5.12 from the limit equilibrium analysis [25]).

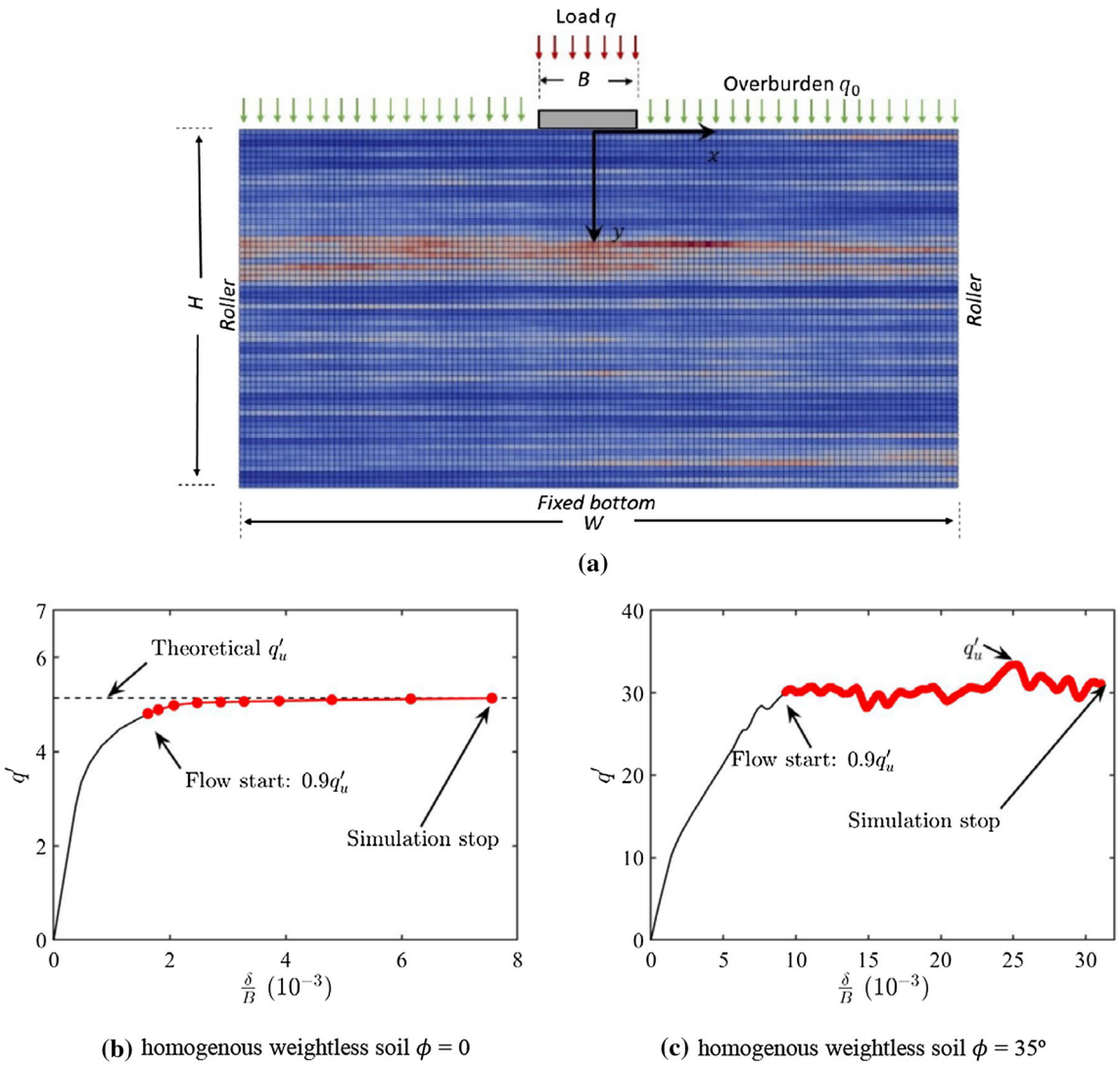
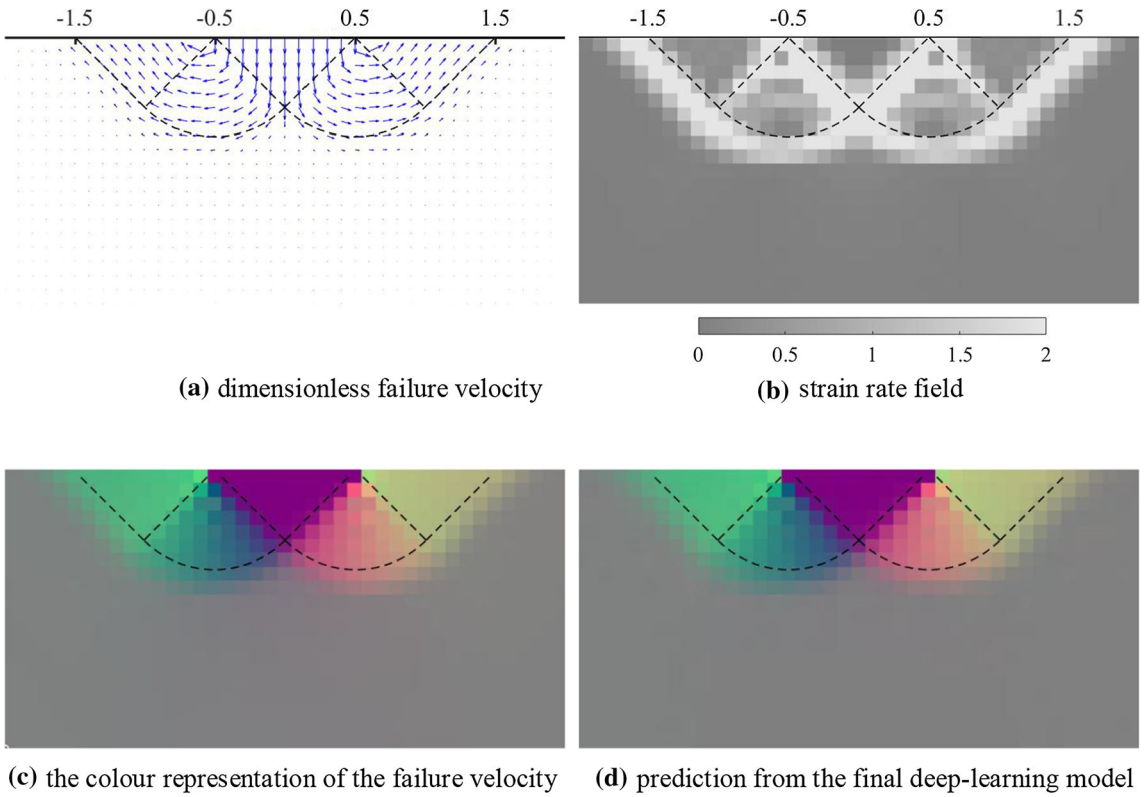


Fig. 2 Illustration of the simulation setup and load/displacement curves

The dimensionless failure velocity can be evaluated from the displacement increment field as $\boldsymbol{v}'_u = \frac{\boldsymbol{v}_u}{v_0} = \frac{d\boldsymbol{x}_u/dt}{dy_0/dt} = \frac{d\boldsymbol{x}_u}{dy_0}$. The displacement increment field $d\boldsymbol{x}_u$ and the footing settlement dy_0 are both calculated after failure and in the same time interval dt. A larger interval will lead to a more accurate evaluation, which is particularly true for simulations where the friction angle is large (relatively large q'_u and simulations are unstable). In this study, the start time of the time interval is selected as when the load reaches 90% of the bearing capacity (Fig. 2b), and the end time is when the simulation stops.

Figure 3a shows the evaluated normalised failure velocity for the homogenous soil with $\phi = 0$ and $\gamma = 0$. The dash lines are slip lines from the limit equilibrium analysis [25]. Agree with analytical analysis, a triangular region is subsiding vertically with the footing, and the extent of the failure zone is from -1.5 to 1.5. From this velocity field, we can also estimate a strain rate field as shown in Fig. 3b (a contour plot of the Euclidian norm of the strain rate tensor). Again, it is shown that shearing happens mostly on the analytical slip lines.

It is difficult to visually compare two velocity fields from quiver plots like Fig. 3a. So, we use a colour representation like Fig. 3c, in which each position is filled with a colour based on the velocity. The RGB value is calculated as Red = $\frac{1+u'_u}{2}$, Green = $\frac{1-v'_u}{2}$, and Blue = 0.5. Each component of the failure velocity has $-1 \leq u'_u \leq 1$ and $-1 \leq v'_u \leq 1$, which guarantees that the red and green values are between 0 and 1. The outer region has a velocity of zero, so the corresponding RGB value is (0.5, 0.5, 0.5) and the colour is grey. The triangular region under the footing has a velocity of (0, 1), so the corresponding RGB value is (0.5, 0.0, 0.5) and the colour is magenta. Figure 3d shows



(a) dimensionless failure velocity (b) strain rate field

(c) the colour representation of the failure velocity (d) prediction from the final deep-learning model

Fig. 3 Determination and representation of the failure velocity (dash lines are analytical slip lines)

the prediction from the final deep-learning model, which closely matches the failure mechanism from finite element simulations and from analytical slip lines.

Figure 2c shows the load/displacement curve for a homogenous soil with $\phi = 35°$ and $\gamma = 0$. The load does not increase at around 0.025 B and the simulation stops at around 0.03 B. The analytical bearing capacity is 46.1 (readers can refer to He et al. [33] for the equation), but the finite element model gives a $q_u^{'}$ of only 30.7. As explained in our previous study, the finite element model gives accurate bearing capacity when $\phi \leq 25°$ and give conservative results when $\phi > 25°$. This is because the extent of analytical slip lines increases exponentially with the friction angle and will be cut by the left and right boundaries in numerical models when $\phi > 25°$. In practical geotechnical design, when the friction angle is large, a safety factor is often introduced [25] to avoid over-optimistic predictions. So, these conservative predictions are desired in practice.

The left column of Fig. 4 shows the dimensionless failure velocity for homogenous weightless soils with various friction angles (from top to bottom are 5°, 15°, 25°, and 35°). The top row is results with $\phi = 5°$ and the failure mechanism agrees with the analytical results. From the second row when $\phi = 15°$, the predicted failure mechanisms do not agree with analytical results. From the third row when $\phi = 25°$, the analytical slip lines extend further than $-6.4 \leq x' \leq 6.4$. In all four plots, the size of triangular subsiding regions is well captured in finite element simulations. In practice, it is very rare that there are no other nearby structures for a target footing, so the assumption of an unbounded soil domain in the limit equilibrium analysis is often unreasonable, and the results from finite element simulations are closer to reality. The right column of Fig. 4 shows the corresponding failure velocity from the final deep-learning model, which proves its high accuracy.

3 Dataset

As discussed in the introduction section, to make a ready-to-use surrogate, the model needs to cover any conditions commonly encountered in practice. Phoon and Kulhawy [22] reported typical soil parameters (Table 1) based on an extensive literature review.

The parameters for the normalised inputs are listed in Table 2. The reference strength c_r is the mean of cohesion, so the mean of normalised cohesion is 1. The width of strip footings is often 0.2–1.0 m, so the mean of normalised unit weight ($\gamma' = \frac{\gamma B}{c_r}$) is 0–2. Shallow foundations are buried below the surface with $D_f < 2.5B$, which causes the overburden load, so the mean of normalised overburden load $q_0^{'} = \frac{q_0}{c_r} = \frac{\gamma D_f}{c_r}$ is 0–6.

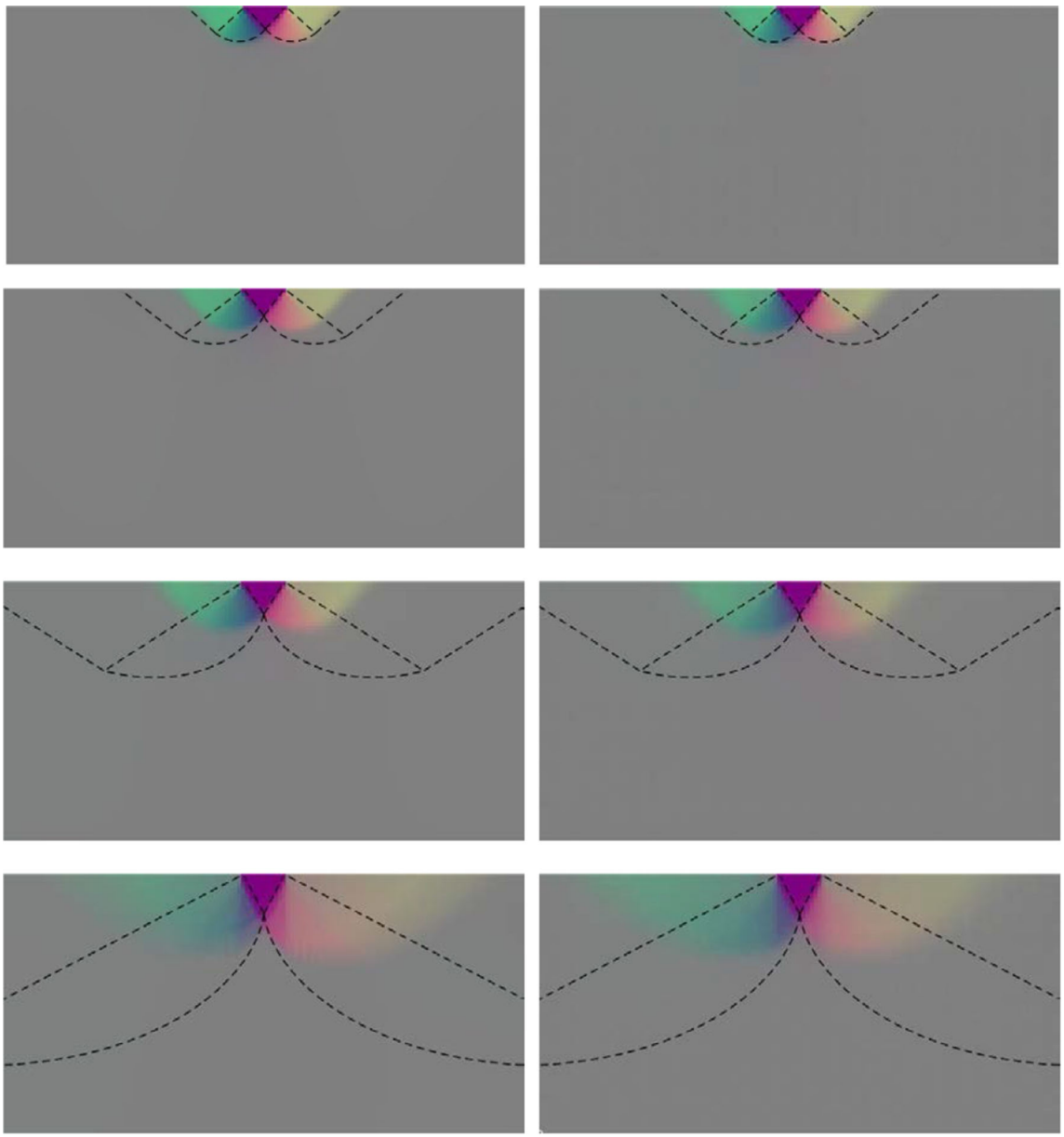


Fig. 4 Failure velocity of homogenous weightless soil (left column: results from finite element simulations; right column: results from the final deep-learning model; dash lines: analytical slip lines; from top to bottom, friction angles are 5°, 15°, 25°, and 35°)

Table 1 Soil parameters reported by Phoon and Kulhawy

Parameters	Mean	COV	Vertical scale of fluctuation l_y	Horizontal scale of fluctuation l_x
c or s_u	10–700 kPa	0.1–0.55	0.2–6.2 m	23–66 m
ϕ	20–40°	0.05–0.15		
γ	13–20 kN/m^3	0–0.1		

Table 2 Parameters for the normalised parameters

Parameters	Mean	COV	l'_y	l'_x
c'	1	0–0.55	0.2–$\propto$	23–$\propto$
ϕ	0–40°	0–0.15		
γ'	0–2	0–0.1		
q'_0	0–6			

As for the coefficient of variation (COV) for these normalised inputs. We choose larger ranges than those in Table 1 to include smaller values. This is because homogenous fields have COV = 0, small COVs correspond to near-homogenous fields, and these conditions are included. Therefore, the COVs for the normalised cohesion, the friction angle, and the normalised unit weight are 0–0.55, 0–0.15, and 0–0.1, respectively. Similarly, homogenous or near-homogenous fields have a very large scale of fluctuation, so the limit for the normalised vertical and horizontal scales of fluctuation are 0.2–∞ and 23–∞, respectively.

A total of 12,332 data are prepared for training. They belong to five subsets (Table 3). Some subsets are reduced problems of the full problem. Subsets 1 and 2 contain data for homogenous soils. Subsets 3 and 4 are data for undrained soils with the undrained strength as a random field. For Subset 5, c', ϕ, and γ' are all random fields. Detailed information about the generation of samples can also be found in our previous study [33]. The quasi-random sequence is extensively used in sampling a parameter space, which has the advantage of uniformly covering the space. Samples of the random fields are generated using GSTools (an open-source software [13]) with exponential autocorrelation functions and lognormal distributions. It is shown that even if training data are from lognormal distributions, the obtained model is tested and valid for data from other distribution types [33].

Table 3 Dataset information

Subsets	Description	No. of data	Features of inputs
1	Homogenous weightless soil	25	c' and ϕ are independent of $\boldsymbol{x}'$; $\gamma' = 0$
2	Homogenous soil	306	c', ϕ, and γ' are independent of $\boldsymbol{x}'$
3	Weightless undrained soil with no overburden	3000	c' is random field; $\phi = 0$; $\gamma' = 0$; $q_0' = 0$
4	Undrained soil	3000	c' and γ' are random fields; $\phi = 0$
5	The full problem	6000	c', ϕ, and γ' are random fields
Total		12,332	

4 Deep neural networks

Artificial neural networks are computing models with elements and structures very similar to biological neural networks [27]. The most basic elements are artificial neurons

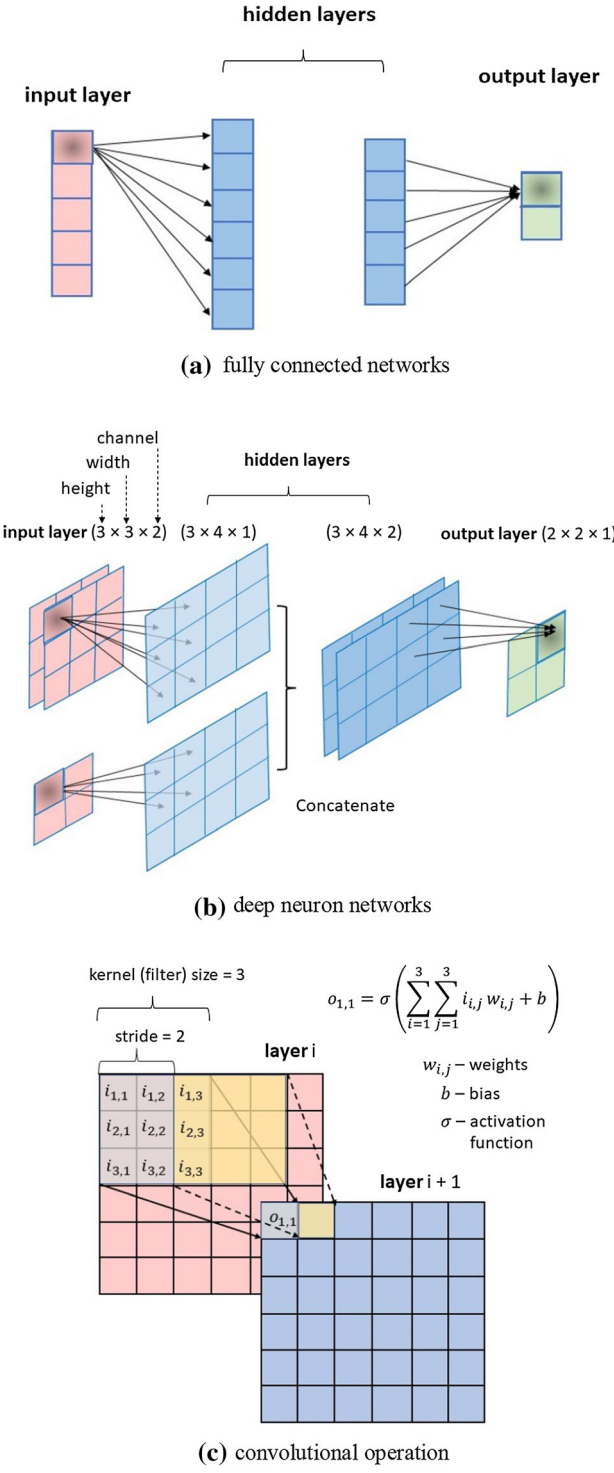


Fig. 5 Illustration of neurons, layers, connections, and operations in deep neural networks

or simply neurons (illustrated as squares in Fig. 5), and many such neurons form a layer. Neurons are connected with "edges" (arrows in Fig. 5). A given neuron can have multiple input connections (the marked green neuron in Fig. 5a) and multiple output connections (the marked red neuron in Fig. 5a).

The first network invented is the fully connected network (FCN), which consists of one input layer, several hidden layers, and one output layer. All layers in this network are often arranged in lines as Fig. 5a, and each neuron is connected with all neurons in its preceding layer (where the name fully connected comes from). From the perspective of computing, each neuron is a real number, and a layer is then a vector, and the connections between two layers can be seen as an operation on an input vector to produce an output vector. This operation is often in two steps: weights (as a matrix) are firstly multiplied by the input vector to produce a vector (a bias vector can be optionally added onto this vector), on which an element-wise nonlinear function is then applied. The weights and biases are trainable parameters, which are adjusted by associated learning strategies to minimise prediction errors. The nonlinear function is often called the activation function.

From universal approximation theorems [8], one-hidden-layer FCNs can approximate any complex function if the number of neurons in the hidden layer is unbounded. An FCN for the present problem [Eq. (5)] can be simply built: (a) each of the inputs $\mathbf{C}', \mathbf{\Phi}$, and $\mathbf{\Gamma}'$ has a shape of 64×128 and $q_0^{'}$ is a scalar. These inputs can be flattened and concatenated as a single input with 24,577 neurons; (b) one hidden layer has p neurons where p is a tuneable hyperparameter; (c) each of the outputs $\mathbf{U}_{\boldsymbol{u}}^{'}$ and $\mathbf{V}_{u}^{'}$ has a shape of 65×129. Similarly, a flattened output is with 16,770 neurons. Due to the dense connectivity between layers, FCNs involve many trainable parameters even for a small hidden layer size (p). For example, if the hidden layer size is chosen as 50 for the present problem, a very small number, there are 2.08 million trainable parameters. If the size rises to 2000, the number of trainable parameters goes up to 82.7 million.

FCNs do not account for the data structure of the specific problem, and existing training strategies cannot guarantee the convergence of approximations (e.g. local minima, or the convergence rate is slow). Additionally, deep FCNs often suffer from the problem of overfitting. Therefore, some dedicated layers and operations have been created to accommodate different needs and data structures. For example, the field inputs and outputs in this study extend in two directions, neurons in each layer can thus be arranged in 3 dimensions (height, width, and channels) like in Fig. 5b. This is very similar to the deep-learning models for analysing images, where the height and width correspond to image pixel sizes, and the channel corresponds to the colour channels like the RGB values (different material properties for the present study). We use a tuple (height $\times$ width $\times$ channels) to denote the size of each layer. The structure of networks can also be complex (like in Fig. 5b) rather than sequential. Some dedicated layer types used in this study are:

Convolution layer: When the data are arranged like Fig. 5b, the spatial relationship is then naturally embedded into the neighbouring of neurons. In principle, the output at a position should be mainly determined by inputs at neighbouring potions. Inspired by this, convolution layers are invented, in which a neuron is, rather than fully connected to all neurons in its preceding layer, but locally connected to a cuboid of neurons. This cuboid is often called kernels or filters (Fig. 5c). The horizontal and vertical sizes of the cuboid are the filter size and are represented by a tuple (height $\times$ width). The channel direction of this cuboid is usually equal to the channel size of the preceding layer. When neurons slide by one, the corresponding cuboids also slide by a distance (Fig. 5c), which is defined as the stride of filters and represented by a tuple (height $\times$ width), too. Padding is a process of adding layers of zeros to the preceding layer to control the shape of the current layer. The number of filters determines the channel size of the current layer. The height and width of the current layer are determined by the height and width of the preceding layer, filter size, stride, and padding.

In addition to this local connectivity, parameter sharing is also used to further reduce the number of trainable parameters, which is based on the rationale that filters serve to conduct high-level abstractions (e.g. finding curves, corners, soldiers, and flags in images), and they should be independent of the spatial position, so all filters in the same channel have the same trainable parameters. The operation between a convolution layer and its preceding layer is then computed as a convolution of the filter's weights with the input cuboid (where the name convolution comes from).

Transposed convolution layer: The data size (height and width) reduces continuously after passing through convolution layers and abstraction of the input is created. For problems like the present one where the output has a data size similar to the input, some layers are needed to increase the size (Upsampling). The connection between a transposed convolution layer and its preceding layer is like the convolution layer but exchanges the order of input and output. A transposed convolution layer is also characterised by the filter size, stride, number of channels, and padding.

Batch normalisation: In machine learning, the trainable parameters are updated only after a batch of samples are trained, not after each sample. The training of deep neural networks is challenging because the distribution of input

data to layers deep in the network may change after each batch. So, the learning algorithm is chasing a moving target. Batch normalisation is to standardise the data for each batch (maintains the mean of output close to 0 and the standard deviation close to 1), which can stabilise the learning process and reduce the number of training epochs.

Dropout layer: Deep neural networks with a huge number of trainable parameters tend to co-adapt to training data without generalisation and improving accuracy on testing data, which is called overfitting. The dropout is a regularisation technique to prevent this, by randomly setting neurons in the preceding layer to 0 at a frequency of dropout rate.

ReLU layer: This layer returns an element-wise operation of its preceding layer, which is max(x, 0), and is called Rectified Linear Unit in machine learning.

LeakyReLU layer: Like ReLU layer, but with a Leaky version of the ReLU operation ($f(x) = x$ when $x \geq 0$, and $f(x) = \alpha x$ when $x < 0$). α defaults to 0.3 in this study.

Deep neural networks with stacks of these dedicated layers are trained with big data and have been successfully used in very challenging tasks. For example, in the game Go, the number of legal board positions is approximately 2.1×10^{170}, a computer programme must make an optimal decision with so many possibilities. Recently, the programme AlphaGo [7] has won human experts and constantly ranks 1st worldwide, which is based on a deep neural network mainly consisting of convolution layers. For our footing failure mechanism problem, we will build a modified U-Net, which was initially developed for biomedical image segmentation [5]. It is also successfully used in medical image reconstruction [19], pixel-wise regression for pansharpening [30], and image-to-image translation for fluorescent stain estimation [28]. A U-Net is made from downsamplers and upsamplers (Fig. 6), which are further made from stacks of the layers mentioned. For a downsampler, the data pass through a convolution layer, a batch normalisation layer (optional), and a LeakyReLU layer. If a stride of (2, 2) is used with padding for the convolution layer, the input size (height and width) is halved after passing through a downsampler (Fig. 6). The channel number of the output is determined by the convolution layer. For an upsampler, the data pass through a transposed convolution layer, a batch normalisation layer, a dropout layer (optional and default rate = 0.4) and a ReLU layer. Similarly, a stride of (2, 2) with padding will double the height and width. In this study, the filter size is fixed as (4, 4) for downsamplers and upsamplers.

With the building blocks, our final deep-learning model is illustrated in Fig. 7. The inputs ($\mathbf{C}', \mathbf{\Phi}$, and $\mathbf{\Gamma}'$) are stacked together as a matrix input of size (64, 128, p_0). Here, p_0 depends on the problem to solve. For the full problem, p_0 is 3. If the model is for weightless undrained soil ($\mathbf{\Phi} = \mathbf{0}$, $\mathbf{\Gamma}' = 0$), p_0 is 1. The matrix input and the scalar input (q_0') do not enter calculation at the same position. The matrix input firstly passes a downsampler and the output data has a shape of (32, 64, p_1), which will further flow in two directions (arrows in Fig. 7). In one direction, the data pass another downsampler—further reducing the size and achieving abstraction. In the other direction, the data will be concatenated with the output data (size = (32, 64, p_{11})) of the last upsampler to have a new output (size = (32, 64, $p_1 + p_{11}$)), which then passes a transposed convolution layer and produces the output (failure mechanism; size = (65, 129, 2)). So, this model is very complex with a highly nonlinear structure. In the upper part, the matrix input passes through six downsamplers; higher degrees of abstraction are achieved; the data size is progressively reduced until finally becomes (1, 2, p_6). Meanwhile, the scalar input (q_0') is extend to shape (1, 2, s_0). Here, s_0 also depends on the problem to solve. These data are then concatenated as data of shape (1, 2, $p_6 + s_0$). In the bottom part, the data pass through five upsamplers.

There are 11 tuneable hyperparameters (from p_1 to p_{11}) for the model. When the data pass through the downsamplers, the channel size should increase ($p_1 < p_2 < \cdots < p_5 < p_6$) to accommodate the higher degrees of abstraction. When the data pass through the upsamplers, the channel size should decrease ($p_6 > p_7 > \cdots > p_{10} > p_{11}$) towards the final channel size of the output. For simplicity, we only test three sets of hyperparameters (labelled as small, medium, and large models, respectively in Table 4). For the full problem ($p_0 = 3$ and $s_0 = 1$), a small model has 1.83 million trainable parameters, and it rises to 29.25 million for the large model.

5 Training and testing

The full problem of footing failure mechanism is complicated with 24,577 input features and 16,770 output features [Eq. (5)], so we firstly start with a reduced problem with a smaller size of input features, and progressively increase the size until the full problem is examined.

In machine learning, a dataset is often separated as training, validation, and testing datasets, where models are trained with learning strategies based on the training dataset. The validation dataset is used to avoid overfitting—stop training when the error on the validation dataset does not improve. The testing dataset is used to provide an unbiased evaluation of the final model. For each problem, whether reduced or full, 70% of available data are used for

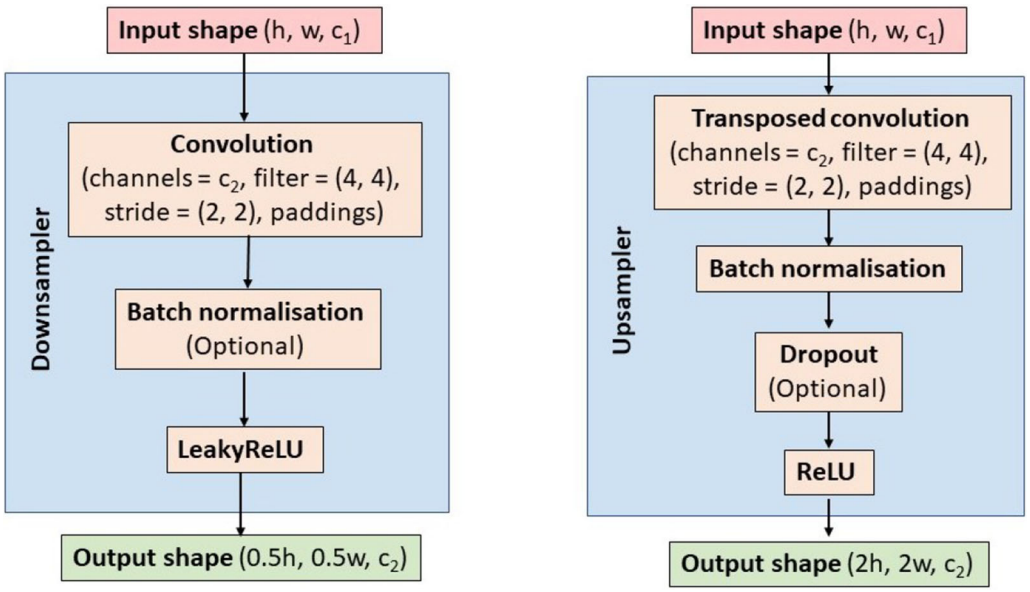


Fig. 6 Downsampler and upsampler

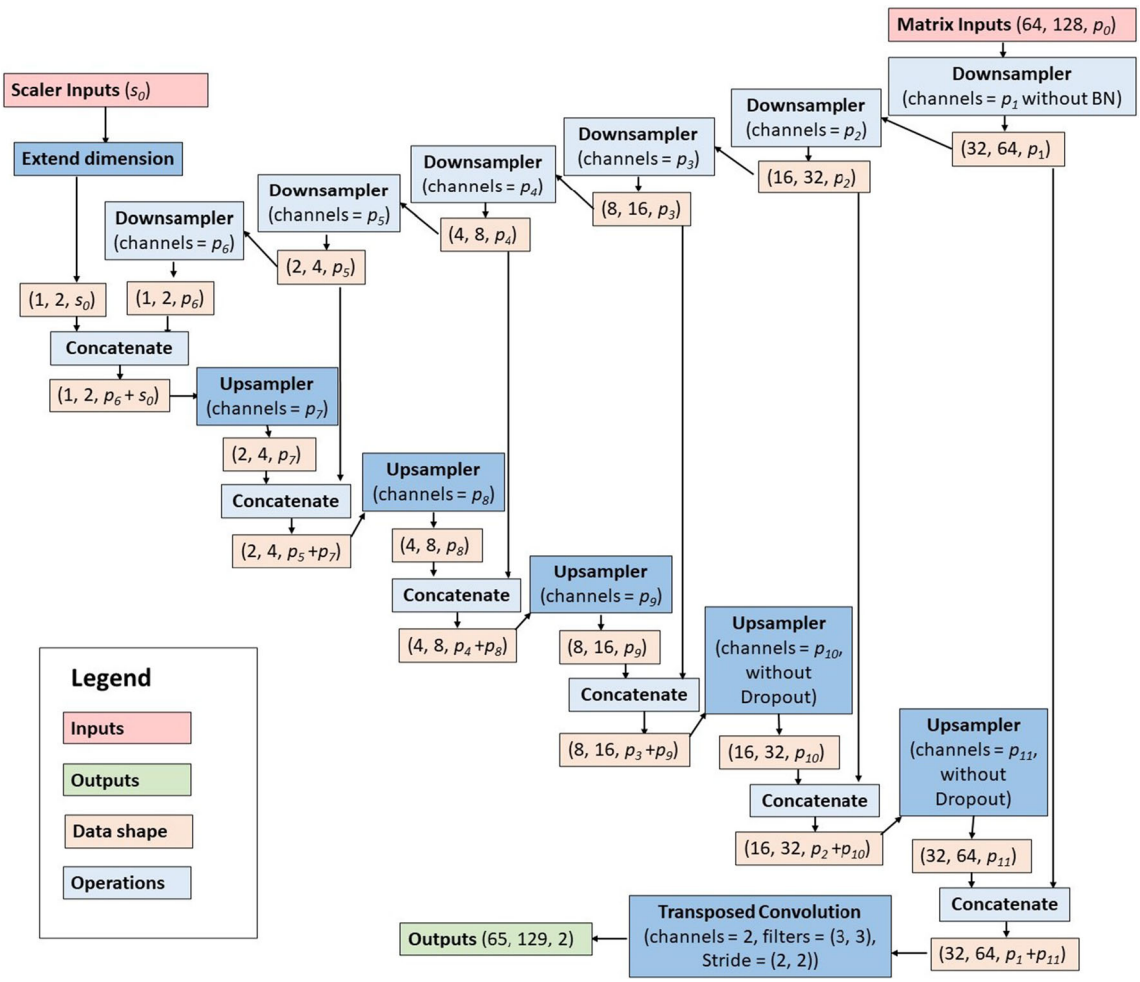


Fig. 7 Structure of the modified U-Net

Table 4 Hyperparameters for the deep-learning model

Model size	No. of trainable parameters for the full problem	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}
Small	1.83 million	16	32	64	128	128	128	128	128	64	32	16
Medium	7.32 million	32	64	128	256	256	256	256	256	128	64	32
Large	29.25 million	64	128	256	512	512	512	512	512	256	128	64

training, 15% for validation, and the remaining 15% for testing.

In training, a loss function must be defined such that trainable parameters are adjusted to minimise it. For a sample, if the "true" failure velocity is $\boldsymbol{v}_{\mathbf{u}}^{'}$, the predicted failure velocity is $\boldsymbol{v}_{\mathbf{u}}^{p'}$, a simple selection of loss function would be the sum of error length over all positions, i.e. $\sum\left|\boldsymbol{v}_{\mathbf{u}}^{'}-\boldsymbol{v}_{\mathbf{u}}^{p'}\right|$. But we want the predicted failure velocity to be more accurate close to the footing, so a weighted error is introduced ($\sum w\left|\boldsymbol{v}_{\mathbf{u}}^{'}-\boldsymbol{v}_{\mathbf{u}}^{p'}\right|$), where the weight is

$$w(\boldsymbol{x}')=\frac{1}{|\boldsymbol{x}'|+3} \tag{6}$$

It is the maximum (0.33) at the centre of footing, decreases continuous with the distance, and is halved at position $|\boldsymbol{x}'|=3$.

5.1 Weightless undrained soil with no overburden

This is a reduced problem with $\boldsymbol{\Phi}, \boldsymbol{\Gamma}'$, and $q_0^{'}$ all equal to zero, and the corresponding dataset is Subset 3 with 3000 data. The input is solely $\mathbf{C}'$ (or $\mathbf{S}_u^{'}$).

Firstly, an FCN with one hidden layer can be built: the flattened input has 8192 neurons, and the flattened output has 16,770 neurons. A dropout layer is added between the only hidden layer and the output layer, and randomly sets "signals" from the hidden layer to 0 with a frequency of dropout rate (between 0 and 1) during training time, which means that after passing the dropout layer, the "signals" have only a probability of (1 – dropout rate) to retain its original value. The dropout layer can help to avoid overfitting and the rate is often between 0.2 and 0.5 [24]. We set the dropout rate to 0.4 in this study, and its specific choice does not influence the conclusions of this study. The performance of trained models is often affected by how the trainable parameters are initialised [9]. In this study, for each combination of hyperparameters, we run the training 11 times—one run with trainable parameters initialised with zeros and ten runs with trainable parameters initialised according to Glorot normal distribution (also called Xavier normal initializer) [15]. We found that the average errors are only slightly influenced by the initialisation, and it does not affect the conclusions of this study. In the following presentation, among these 11 runs, the results with the lowest average error for the testing dataset are reported. For the FCN with one hidden layer, the only hyperparameter is then the number of hidden neurons. When it varies from 50 to 2000, the total number of trainable parameters changes from 1.26 million to 49.9 million, the average errors of training, validation, and testing dataset are presented in Fig. 8a as circles. With more hidden neurons and so more trainable parameters, the network is more complex and will lead to a more accurate model. But when the neuron size increases over 1200 (29.97 million parameters), no improvement is observed for the average errors of the validation and testing dataset (around 1.4), while the average error of the training dataset decreases slowly, indicating slightly overfitting. So, a robust model is with 1200 hidden neurons, and the error (around 1.4) is larger than that of the U-Net (around 1.0, diamonds in Fig. 8a).

We also test deep FCNs, particularly networks with two hidden layers that have the same number of neurons in each hidden layer (from 10 to 1000). A dropout layer is similarly added between the last hidden layer and the output layer. The total number of trainable parameters varies from 0.27 million to 26.0 million, and the average errors are presented in Fig. 8a as triangles. The performance of deep FCNs is better than that of one-hidden layer FCNs regarding the average error of the testing dataset (1.0 compared with 1.4), and is comparable to that of the U-Net (1.0, diamonds). However, stronger overfitting is observed. With more trainable parameters (i.e. more complex models), the error of the training dataset decreases continuously and even reaches a very low value of 0.33—a clear indication of overfitting. Deep FCNs with more hidden layers should get even stronger overfitting, which is a well-recognised issue of deep FCNs. That is why other deep neural networks such as CNNs, locally connected networks and U-Net are created—they are regularised versions of deep FCNs.

For undrained soils, failure is concentrated in a small region (Fig. 4) and is only determined by soil elements very close to the footing. So, we do not need an input $\mathbf{C}'$ as large as 64 × 128. If we only consider soil elements in a small region ($-2.5 \le x' \le 2.5$, $0 \le y' \le 2.0$), the input $\mathbf{C}'$ is

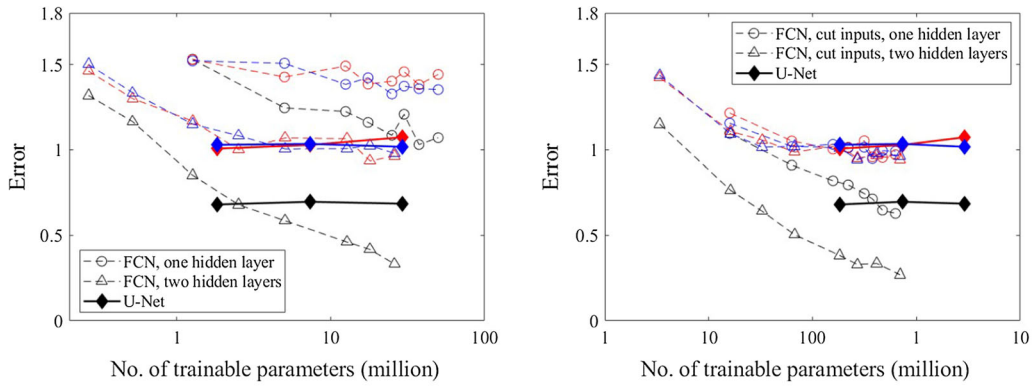


Fig. 8 Errors of different models for weightless undrained soil with no overburden (black = training; red = validation, blue = testing)

cut to 20 × 50, and the output ($\mathbf{U}_{\mathbf{u}}^{'}$ and $\mathbf{V}_{u}^{'}$) is cut to 21 × 51 × 2. An FCN can then be built with these cut inputs and outputs. A dropout layer is also used, and Fig. 8b gives the average errors (circles). Similarly, when the hidden neuron size is over 1200 (3.77 million trainable parameters), the average errors of the validation and testing dataset do not improve, and the average error of the training dataset decreases slowly. So, a robust model is with 1200 hidden neurons, achieving an average error of 0.95, 30% better than the FCN with one hidden layer. Additionally, there are only 3.77 million trainable parameters compared with the 29.97 million of the FCN with one hidden layer, which shows that more input features or a more complex model do not necessarily mean a better model. Similarly, we test FCNs with two hidden layers. For this case of cut inputs and outputs, the deep FCNs cannot improve the error of the testing dataset but only lead to stronger overfitting.

In the second FCN, we in fact apply a filter on the original inputs and outputs by filtering out insignificant features (soil properties and failure velocity far from the footing) based on our understanding of the physical mechanism (failure is concentrated in a small region) [33]. Deep learning is a powerful tool because abstractions (filters) are learnt from data automatically. For example, in computer vision, the high-level features (e.g. curves, corners, soldiers, and flags) can be automatically learnt from images with CNNs. For some tasks such as image-to-image translation, face ageing and generating realistic photographs, these operations can be learnt from images with generative adversarial networks (the U-Net is one type). Another benefit of these deep neural networks is that they are regularised versions of deep neural networks and can avoid overfitting to some extent. A deep-learning model is built based on the U-Net for this reduced problem. The matrix input is (64, 128, 1) and the scalar input ($q_0^{'}$) is not needed. Three models with different sizes of trainable parameters are trained and the results are presented in Fig. 8 as diamonds. For this simple task, even the smallest model (1.83 million trainable parameters) can give good results (the average error is about 1.0), and models with more trainable parameters will not improve. As shown in Fig. 8, the U-Net is as good as the FCN with cut inputs and performs much better than the FCN with one hidden layer. Most importantly, it will not easily fall into the overfitting problems as the FCNs with two hidden layers do.

5.2 Undrained soil

This second scenario (failure mechanism of undrained soil under footing) has comparatively more input features—two arrays ($\mathbf{C}' = \mathbf{S}_{\mathbf{u}}^{'}$, $\mathbf{\Gamma}'$) and a scalar $q_0^{'}$. The corresponding dataset is Subset 4 with 3000 data. The previous problem is a reduced problem of the present one with ($\mathbf{\Gamma}' = 0$ and $q_0^{'} = 0$), so there are in total 6000 data available for training.

Firstly, an FCN with one hidden layer can be built with a flattened input of 16,385 neurons. Following the same process, the optimal number of hidden neurons is 1500, which has 49.57 million trainable parameters and achieves an average error of about 1.37 for the testing dataset (circles in Fig. 9a). The FCNs with two hidden layers (triangles in Fig. 9a) again have overfitting problems. A robust FCN with cut inputs and outputs achieves a smaller error of 0.89. In the modified U-Net for this problem, the matrix input is (64, 128, 2) and the scalar input ($q_0^{'}$) enters the calculation as in Fig. 7. Without applying artificial filters and human intervention, the modified U-net with a medium and large number of trainable parameters can achieve a low average error (0.97), slightly better than the small U-Net model. This error is slightly larger than the FCN with cut inputs, but this modified U-Net should be considered the same accurate because, for the FCN with cut inputs and

outputs, the prediction of velocities outside the region ($-2.5 \le x' \le 2.5,\ 0 \le y' \le 2.0$) is missing and is assumed accurate by default.

5.3 The full problem

The first two scenarios have relatively small size input features, and failure is concentrated within a small region. So, the FCNs can still give reasonable results, but we need to understand the mechanism and process the data (ignoring insignificant features) before building the networks. In this last section, the problem with full inputs (three matrices $\mathbf{C}'$, $\mathbf{\Gamma}'$, and $\mathbf{\Phi}$ and a scalar q_0') is considered. The corresponding dataset is Subset 5 with 6000 data, but Subsets 1–4 are data of reduced problems, so a total of 12,332 data are available.

Firstly, an FCN with one hidden layer with a flattened input of 24,577 neurons is built. From Fig. 10a, the optimal structure has 1700 hidden neurons, which has a huge amount of training parameters—70.31 million. With such a complicated model and so much training data (> 12,000), the training process becomes very challenging, about one hour of computation is required on a GPU workstation (Nvidia Quadro GV100, processing power 14.8 TFLOPS, and bandwidth 870 GB/s) to train a single model. Yet, the obtained model has a very large average error (2.00). Using deep FCNs (two hidden layers) cannot achieve a smaller error comparable to that of the U-Net (diamonds) but only lead to overfitting.

We can also build an FCN with cut inputs. The matrix input (64, 128, 3) is firstly cut into (20, 50, 3), then flattened and combined with the scalar input q_0', resulting in the input of 3001 neurons. Because in some samples, the failure region extends further than ($-2.5 \le x' \le 2.5$, $0 \le y' \le 2.0$), we cannot cut the output, so the flattened output is 16770 neurons. Figure 10b shows that the robust model is with 1500 hidden neurons, which has 29.67 million trainable parameters and has an average error of 1.84. The modified U-Net with medium and large size trainable parameters is a more accurate model (the average error is 1.50).

It must be emphasised that the performance gap between the modified U-Net and the one-hidden-layer FCN with cut inputs is larger than these errors suggested (1.50 compared with 1.84). Table 5 shows the performance of the models on data of different datasets. The modified U-Net and the FCN with cut inputs perform equally well on Subsets 3 and 4 because, for these data, failure is concentrated in a small region. However, the FCN will have problems for Subsets 1, 2, and 5 (underline in Table 5), in which some samples have the failure extending very deeply and widely. The average errors of the modified U-Net are only 80% of those of the FCN (underline in Table 5). Therefore, deep neural networks that account for the data structure can be used for very complicated tasks with a huge number of input features and output features and have a competitive advantage over the fully connected networks in these tasks.

We have demonstrated the performance of the final modified U-Net model for homogeneous soils in Fig. 3d and Fig. 4. Figure 11 shows the first eight testing samples of Subset 5. In some samples (2, 6, and 7), the failure zone is shallow. Particularly, for the 7th sample, it is only a thin layer under the footing. For others (1, 3, 4, 5, and 8), failure develops deeply and widely into the soil. Because the soil properties vary spatially, failure can develop along weak zones asymmetrically: deeper on the right part for the 2nd and 4th samples, and deeper on the left for the 5th and 6th samples. In general, this deep-learning model is very accurate and can be used to predict footing failure mechanisms for any soil properties, spatial variabilities, or load conditions encountered in practice. If equipping with a good user interface, it is an ideal tool for practitioners.

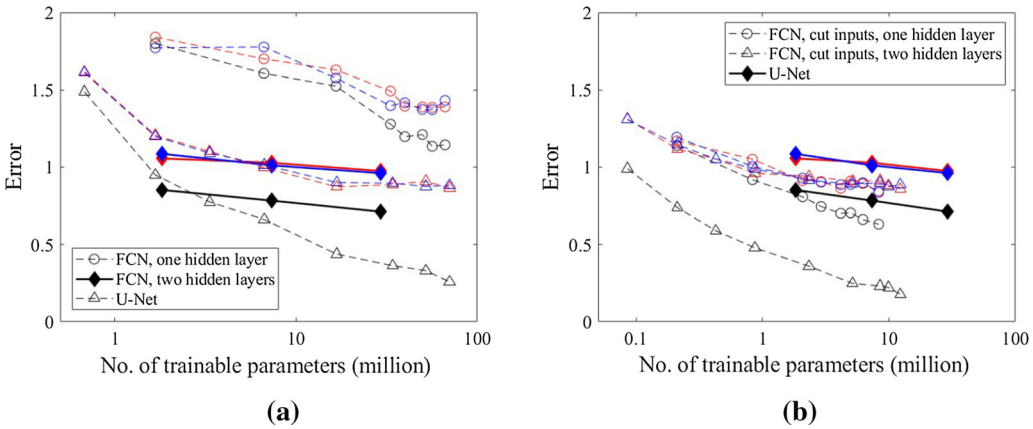


Fig. 9 Errors of different models for undrained soil (black = training; red = validation, blue = testing)

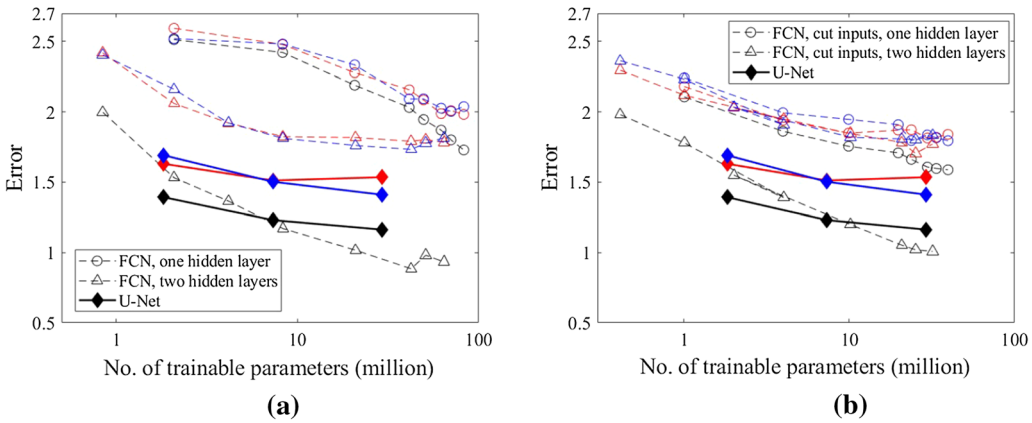


Fig. 10 Errors of different models for the full problem (black = training; red = validation, blue = testing)

Table 5 Performance of the models on data of various subsets

	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5
FCN with one hidden layer	2.116	1.947	1.476	1.495	2.346
FCN with cut inputs and one hidden layer	2.049	1.721	1.001	1.121	2.305
U-Net	1.436	1.376	0.876	1.068	1.855

6 Conclusion

This paper presents a framework to connect advanced models with practical designing tools using deep learning. For a particular problem, the influencing factors are considered as inputs, and the quantities of interest are considered as outputs. Deep-learning models are built based on results from advanced models. So, deep-learning models are equivalent to the advanced models for a particular designing problem but have the advantage of being easy to operate and getting results in a shorter time.

The designing tool must cover any conditions commonly encountered in practice, so the relationship between inputs and outputs would be extremely complicated (particularly when spatial variability is considered), and only the recently developed deep learning can cope with it.

The prediction of footing failure mechanisms is taken as an example. After dimensional analysis and assuming limited-region dependence and finite resolution, it is presented that for this problem, the inputs are the normalised overburden, and finite fields of the normalised cohesion, the normalised unit weight, and the friction angle (represented as 2D arrays). The outputs are finite fields of dimensionless failure velocity (also represented by matrices). The failure velocity is estimated from displacement increments after failure in finite element models.

Typical parameters of soil properties, spatial variability, or load condition are identified, and representative samples are generated to cover all the conditions with techniques like the quasi-random sequence and random field generator (with randomisation method) There are in total 12,332 data available for training and belong to 5 subsets. Some data belongs to reduced problems of the full problem.

Two types of neural networks are examined. One is the traditional fully connected network, and the other is a modified U-Net, which has a very complicated nonlinear structure. It is made up of building blocks like downsamplers and upsamplers, which are further made of layers such as convolution layers, transposed convolution layers, batch normalisation layers, dropout layers, ReLU layers, and LeakyReLU layers. A weighted average of error length is introduced as the loss function, which has a greater weight for positions close to the footing and decreases continuously with increasing distance from the footing.

We start with a reduced problem—the failure mechanism of weightless undrained soil with no overburden. 3000 samples are generated. Fully connected networks and a modified U-Net are built. The fully connected network with cut inputs and the modified U-Net perform equally well. However, a filter is artificially applied for the fully connected network based on our understanding of the underlying mechanism, while in the deep-learning model, filters are automatically "learned" from data. A relatively more complicated problem is then the failure mechanism of any undrained soils with overburden. In this case, the fully connected network with cut inputs and the modified U-Net also perform equally well.

At last, the full inputs are considered, and all the 12,322 data are used for training. Results show that fully connected networks with one hidden layer can fit well simple problems with a small size of inputs and outputs, but they

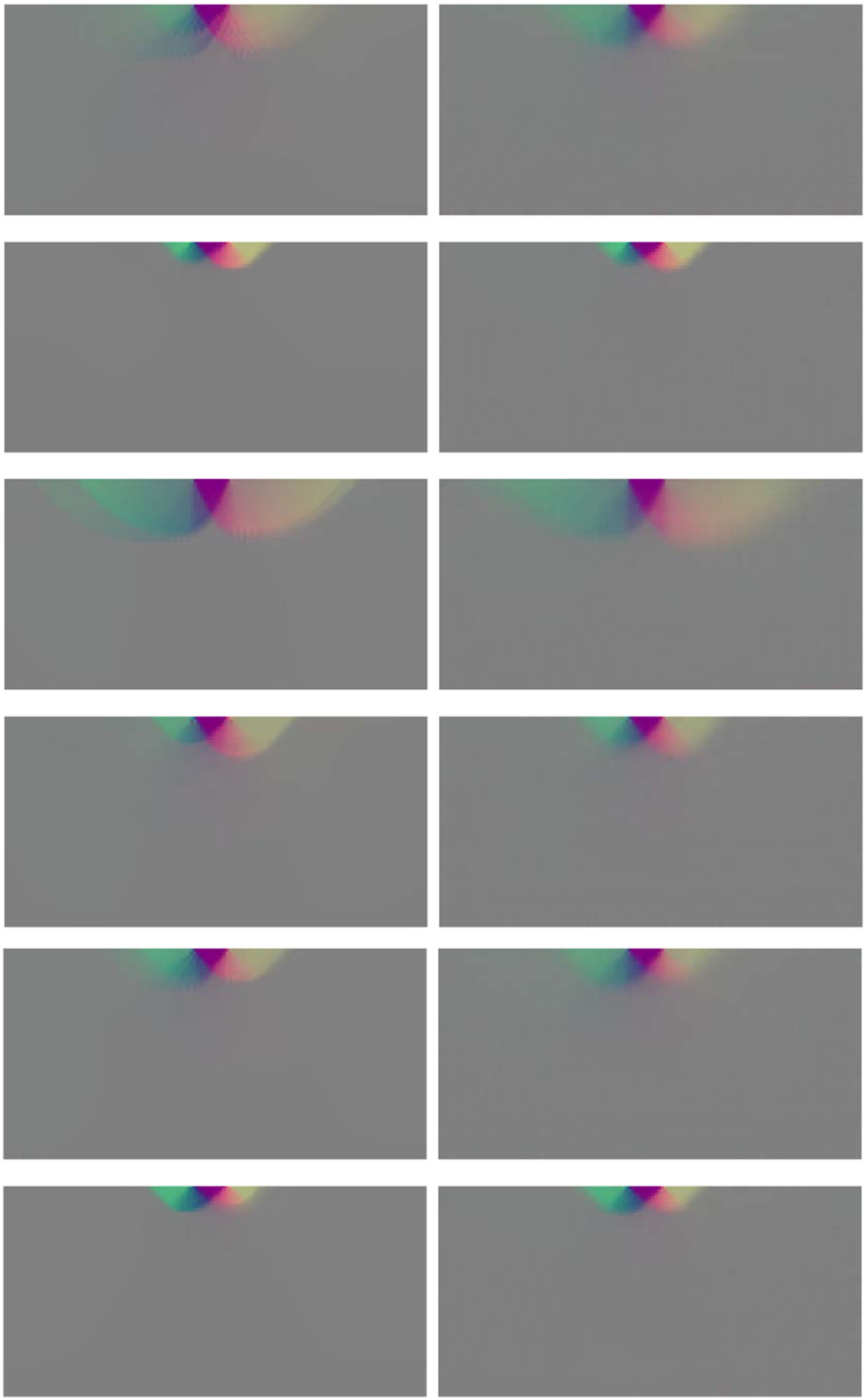


Fig. 11 Performance of the final deep-learning model (the first eight testing samples in Subset 5; left column: results from finite element simulations; right column: results from the final deep-learning model)

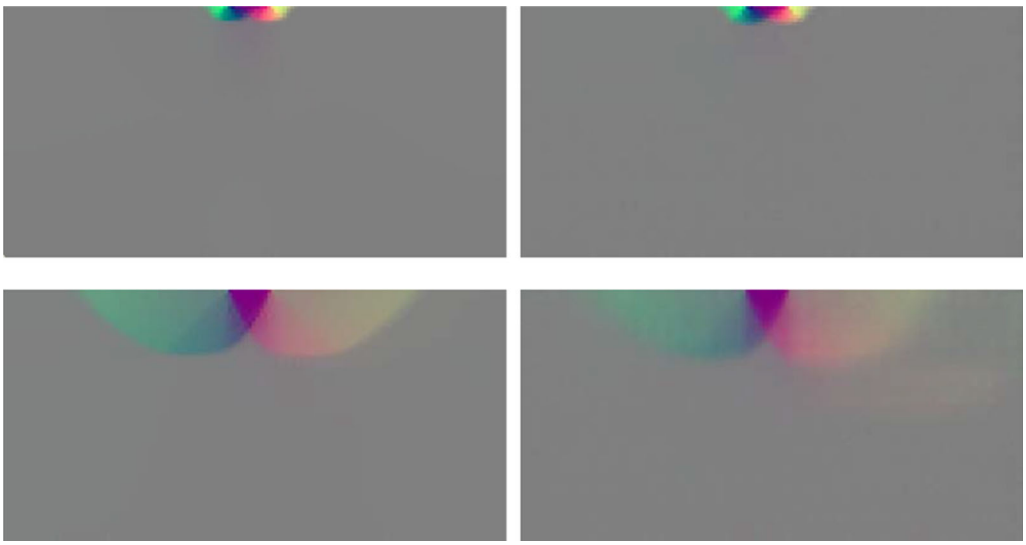


Fig. 11 continued

fail for complex problems. Fully connected networks with many hidden layers suffer from the problem of overfitting. The modified U-Net that accounts for the data structure of specific problems gives better models. The accuracy of the final deep-learning model is demonstrated with both the average error and some plots of sample failure mechanisms.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Data Availability Statement The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Alonso EE, Gens A, Josa A (1991) Discussion: a constitutive model for partially saturated soils. Géotechnique 41(2):273–275. https://doi.org/10.1680/geot.1991.41.2.273
2. Anderson JA (1995) An introduction to neural networks. MIT Press
3. Andersson J, Ahlström H, Kullberg J (2019) Separation of water and fat signal in whole-body gradient echo scans using convolutional neural networks. Magn Reson Med 82(3):1177–1186. https://doi.org/10.1002/mrm.27786
4. Bandara S, Soga K (2015) Coupling of soil deformation and pore fluid flow using material point method. Comput Geotech 65:302. https://doi.org/10.1016/j.compgeo.2014.12.007
5. Bolton M (1979) A guide to soil mechanics. Macmillan Education, London. https://doi.org/10.1007/978-1-349-16208-6
6. Chen CT, Gu GX (2020) Generative deep neural networks for inverse materials design using backpropagation and active learning support. Adv Sci. https://doi.org/10.1002/advs.201902607
7. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Math Control Signals Syst 2(4):303–314. https://doi.org/10.1007/BF02551274
8. Dafalias YF, Taiebat M (2016) SANISAND-Z: zero elastic range sand plasticity model. Géotechnique 66(12):999–1013. https://doi.org/10.1680/jgeot.15.P.271
9. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) ImageNet: a large-scale hierarchical image database, pp. 248–255. https://doi.org/10.1109/cvprw.2009.5206848
10. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural network. Proc Mach Learn Res 9:249–256. https://doi.org/10.1109/LGRS.2016.2565705
11. Griffiths DV, Lane PA (1999) Slope stability analysis by finite elements. Geotechnique 49(3):387–403. https://doi.org/10.1680/geot.1999.49.3.387
12. Guardiani C, Soranzo E, Wu W (2022) Time-dependent reliability analysis of unsaturated slopes under rapid drawdown with intelligent surrogate models. Acta Geotech 17(4):1071–1096. https://doi.org/10.1007/s11440-021-01364-w
13. He X, Liang D, Bolton MD (2018) Run-out of cut-slope landslides: mesh-free simulations. Géotechnique 68(1):50–63. https://doi.org/10.1680/jgeot.16.P.221

14. He X, Wang F, Li W, Sheng D (2021) Deep learning for efficient stochastic analysis with spatial variability. Acta Geotech. https://doi.org/10.1007/s11440-021-01335-1
15. He X, Xu H, Sabetamal H, Sheng D (2020) Machine learning aided stochastic reliability analysis of spatially variable slopes. Comput Geotech 126:103711. https://doi.org/10.1016/j.compgeo.2020.103711
16. Huang M, Jia CQ (2009) Strength reduction FEM in stability analysis of soil slopes subjected to transient unsaturated seepage. Comput Geotech 36(1–2):93–101. https://doi.org/10.1016/j.compgeo.2008.03.006
17. Kandel ME et al (2020) Phase imaging with computational specificity (PICS) for measuring dry mass changes in sub-cellular compartments. Nat Commun. https://doi.org/10.1038/s41467-020-20062-x
18. Kang F, Xu Q, Li J (2016) Slope reliability analysis using surrogate models via new support vector machines with swarm intelligence. Appl Math Model 40(11–12):6105–6120. https://doi.org/10.1016/j.apm.2016.01.050
19. Monaghan JJ, Gingold RA (1983) Shock simulation by the particle method SPH. J Comput Phys 52:374–389
20. Müller S. GSTools. https://geostat-framework.org/
21. Oñate E, Idelsohn SR, Del Pin F, Aubry R (2004) The particle finite element method—an overview. Int J Comput Methods 1(2):267–307. https://doi.org/10.1142/S0219876204000204
22. Phoon K-K, Kulhawy FH (1999) Characterization of geotechnical variability. Can Geotech J 36(4):612–624. https://doi.org/10.1139/t99-038
23. Ronneberger O, Fischer P, Brox T (2015) U-Net: convolutional networks for biomedical image segmentation. IEEE Access 9:16591–16603. https://doi.org/10.1109/ACCESS.2021.3053408
24. Sheng D, Fredlund DG, Gens A (2008) A new modelling approach for unsaturated soils using independent stress variables. Can Geotech J 45(4):511–534. https://doi.org/10.1139/T07-112
25. Sheng D, Smith DW (2002) 2D finite element analysis of multicomponent contaminant transport through soils. Int J Geomech 2(1):113–134. https://doi.org/10.1061/(ASCE)1532-3641(2002)2:1(113)
26. Silver D et al (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529(7585):484–489. https://doi.org/10.1038/nature16961
27. Sulsky D, Zhou SJ, Schreyer HL (1995) Application of a particle-in-cell method to solid mechanics. Comput Phys Commun 87(1–2):236–252. https://doi.org/10.1016/0010-4655(94)00170-7
28. Wang ZZ, Goh SH (2022) A maximum entropy method using fractional moments and deep learning for geotechnical reliability analysis. Acta Geotech 17(4):1147–1166. https://doi.org/10.1007/s11440-021-01326-2
29. Wu W, Bauer E, Kolymbas D (1996) Hypoplastic constitutive model with critical state for granular materials. Mech Mater 23(1):45–69. https://doi.org/10.1016/0167-6636(96)00006-3
30. Yao W, Zeng Z, Lian C, Tang H (2018) Pixel-wise regression using U-Net and its application on pansharpening. Neurocomputing 312:364–371. https://doi.org/10.1016/j.neucom.2018.05.103
31. Zhang P, Yin ZY, Jin YF, Chan THT (2020) A novel hybrid surrogate intelligent model for creep index prediction based on particle swarm optimization and random forest. Eng Geol 265:105328. https://doi.org/10.1016/j.enggeo.2019.105328
32. Zhang P, Yin ZY, Jin YF (2022) Machine learning-based modelling of soil properties for geotechnical design: review, tool development and comparison. Arch Comput Methods Eng 29(2):1229–1245. https://doi.org/10.1007/s11831-021-09615-5
33. Zhang P, Yin Z-Y, Jin Y-F (2022) Bayesian neural network-based uncertainty modelling: application to soil compressibility and undrained shear strength prediction. Can Geotech J 59(4):546–557. https://doi.org/10.1139/cgj-2020-0751