Advanced Engineering Informatics 44 (2020) 101058

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Full length article

Deep neural network based framework for complex correlations in engineering metrics

Vahid Asghari, Yat Fai Leung, Shu-Chien Hsu*

Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hong Kong

ARTICLE INFO

Keywords:
Deep neural networks
Machine learning
Regression modelling
Soil shear strength
Index properties of soil

ABSTRACT

Linear or polynomial regression and artificial neural networks are often adopted to obtain correlation models between various attributes in engineering fields. Although these are straightforward, they may not perform well for datasets that involve complex correlations among multiple attributes, and overfitting can occur when high-order polynomials are used to match the data from one scenario but fail to produce accurate predictions elsewhere. This paper presents a Deep Neural Networks (DNN) based framework for obtaining complex correlations in engineering metrics and provides guidelines to assess data adequacy, remove outliers and to identify and resolve overfitting problems. Moreover, a clear and concise set of procedures for tuning hyperparameters of DNN is discussed. As an illustration, a DNN model was trained to predict the undrained shear strength of clays based on liquid limit, plastic limit, water content, vertical effective stress, and preconsolidation stress. This analysis is conducted with 1101 samples gathered from different sites all over the world. Prediction of soil strengths often involved significant uncertainties due to the natural variations in earth materials and site conditions, contributing to complex relationships among various material properties. Our results show that the proposed framework performs better than conventional correlation models established from previous studies. The developed framework and accompanying Python script can be readily applied to the prediction of clay properties at other sites, and also to other types of engineering metrics.

1. Introduction

Establishment of correlation relationships among different quantities in a dataset, which may involve engineering material properties, engineering system features and/or observed system response, represents a classical problem for various engineering disciplines. In the past few decades, several algorithms have been developed for different applications, involving techniques such as linear regression, artificial neural networks (ANN), etc. For example, in construction management and real estate, Bailey et al. [1] proposed a regression model for the prediction of real estate price index in St. Louis, Missouri, United States, while Cheng et al. [2] used neural networks for overall cost estimation of projects. By using the same method, Chen et al. [3] analyzed tax reports of construction companies in northern Taiwan. Chau [4] made predictions for the outcome of construction claims, while Hola and Schabowicz [5] tried to estimate the time and cost of earthworks by excavators and haulers. Graham et al. [6] created a model for concrete delivery systems, and Gao et al. [7] developed a model to predict the air-gasification of fuels. In hydraulic engineering, Stol [8] obtained correlations for rainfall measured at different gage locations of a station. Later, Adamowski and Chan [9] used ANN for groundwater level forecasting. In other disciplines such as material engineering and structural engineering, Bazant et al. [10] fitted non-linear equations to create a model for creep deformation of concretes. Moreover, Bal and Buyle-Bodin [11] applied ANN to predict the variations in dimensions of concrete elements due to shrinkage, and Kewalramani and Gupta [12] applied the same method to predict concrete compressive strength. Li et al. [13] also used neural networks to predict the structural response of a building involving a high degree of nonlinearity. In the field of geotechnical engineering, which is known to involve significant uncertainties in the properties of natural soil and rock materials, Hansbo [14] and D'Ignazio et al. [15] have developed several correlation formulas among parameters of clays, in order to predict their undrained shear strength. Yilmaz and Kaynar [16] applied both regression and ANN to predict the swell potential of soils. Meanwhile, Shahin et al. [17] used neural networks for making a prediction model in settlements in shallow foundations. More recently, Momeni et al. [18] also applied ANN to predict foundation pile capacities. Other machine learning methods have also come into the attention of researchers recently to predict soil strength. For example, Das et al. [19] used support vector

* Corresponding author.
E-mail address: mark.hsu@polyu.edu.hk (S.-C. Hsu).

https://doi.org/10.1016/j.aei.2020.101058
Received 14 June 2019; Received in revised form 16 January 2020; Accepted 6 February 2020

regressions (SVR) and ANN to predict residual strength of clays. Chou et al. [20] applied SVR, classification and regression trees (CART), ANN, and ensemble methods to predict peak shear strength of fiber-reinforced soils, while Pohjankukka et al. [21] used neural networks to predict soil bearing capacity of boreal forest soils, and Pham et al. [22] employed similar techniques (SVR and ANN) to predict shear strength of soft soil.

Depending on the fields of applications and the nature of the specific problems, the abovementioned approaches are associated with different strengths and limitations. Linear regression modeling is intuitive, easily understood and performed by linear algebra, producing simple and concise formulas. However, linear regression may not be suitable for datasets with high degrees of nonlinearity, and most commercially-available software does not detect or resolve possible overfitting issues in linear regression models. On the contrary, ANN does not produce simple correlation formulas, but these models usually work better with nonlinearity in datasets, compared to linear regression. Also, user-friendly ANN toolboxes are now readily available in many programming or scripting languages such as MATLAB. However, like linear regression models, overfitting problems can exist and are usually not carefully considered. In addition, the performance of ANN also drops with an increasing degree of nonlinearity [23]. Therefore, the development and application of learning algorithms have come to the attention of researchers in various fields [24].

Deep neural networks (DNN) model, as a branch of deep learning, is a powerful tool that is being developed and applied mostly to computer vision, voice recognition, pattern recognition, natural language processing, and financial markets. DNN is an extension of ANN, and like other neural networks usually consists of three parts in its core: an input layer, hidden layer(s) and the output layer. The main differences between DNN and ANN are the number and type of hidden layers [23,25], where the output of each hidden layer becomes the input for the next layer. Long short-term memory (LSTM) and convolutional neural networks (CNN) are other examples of DNN structures, which are used in time-series analysis and computer vision [26,27]. DNN has also come into the attention of engineering researchers recently. Zhou and Gong [28] applied concepts of computer vision by DNN to detect building objects in airborne images. In the sustainability discipline, Singaravel et al. [29] used DNN in Building Performance Simulation to predict heating and cooling energy consumption. Fang et al. [30] used convolutional neural networks to analyze the unsafe behavior of workers in construction sites. Zhong et al. [31] used a classification system based on DNN and text mining approaches to classify building quality problem reports in China. DNN models are capable of handling datasets with high degrees of nonlinearity between the input and output variables, potentially providing more accurate results in comparison to linear regression and ANN, meanwhile tackling the issues of overfitting through various approaches. All in all, there are currently limited studies in the literature on developing a holistic framework to help engineers facilitate DNN modeling in the most rational form, and to reach the most accurate results without overfitting problem.

The full potentials of DNN models in engineering applications have not been realized, partly due to the absence of a systematic approach and clear guidelines for the construction of such networks. In fact, the flexibility and a large number of options in DNN could become a hindrance to engineering researchers or analysts, who are usually more familiar with the features of engineering datasets, but not equally knowledgeable regarding the most appropriate DNN structures for their specific analyses. Therefore, the main purpose of this study is to introduce a set of straightforward procedures to construct DNN models for regression problems in engineering material properties, meanwhile elucidating the relevant issues associated with the modeling of engineering datasets, and presenting a Python code that facilitates DNN modeling for regression problems. The Python code and the dataset used in this study are accessible online[1]. These proposed procedures enable a rational assessment of the sufficiency of data, i.e. whether additional data is required for building and training of the DNN model, and whether a correlation/ relationship can be established between the input and output variables. The DNN model created for this study is designed, trained, and tested in Python3.6 with KERAS [32], TensorFlow™ [33], and scikit-learn [34] libraries. Engineering studies which applied DNN modeling heretofore have concentrated on the application of neural networks, rather than discussing the special features of this complex model and unique precautions for its implementations in the engineering field. The main contribution of this study and framework is, therefore, to provide clear guidelines and pave the way for engineers, especially those who are not familiar with DNN and programming, to optimize DNN modeling and training. Thereby, the most accurate results with the smallest possible prediction error and no overfitting problem by DNN can be accessible for engineers in various domains. This will eventually lead to better decisions and optimized strategies which would render engineering designs and solutions more efficient and profitable. Moreover, the notion of deep learning models as black-boxes is discussed and proper solution is provided to extract knowledge from a DNN model.

The model and procedures proposed in this study are illustrated through a geotechnical problem, using various index properties of soil and stress conditions to predict the undrained shear strengths of clays. While this is the first time that DNN is applied to such a fundamental geotechnical problem, the proposed framework can be applied to other engineering problems. The structure of this study is as follows: the specific features of the geotechnical dataset used in this study are discussed in Section 2. The methodology section, Section 3, presents a short summary of DNN structure and the core optimization (learning) behind it. Design of appropriate DNN structure, data preprocessing, training and analysis algorithms are described in detail in Section 4 (Model development section). Finally, the results of the proposed framework are presented in Section 5. While the geotechnical properties of in situ soil materials are known to involve significant uncertainties due to heterogeneity and randomness in geological processes, this study demonstrates that the DNN models can outperform traditional approaches in the prediction of their properties. The program code compiled for this study can also be applied to similar regression modeling analyses and is included as supplemental information to this paper.

2. Geotechnical dataset

Geotechnical engineering is a discipline associated with significant uncertainty, primarily arising from natural variations in soil and rock materials. The geological history at a site, the sizes and spatial arrangements of soil particles, and their response under various hydrogeological and loading conditions all contribute to the variability and uncertainty in material properties. Meanwhile, sampling and testing of subsurface soil materials are costly, and therefore engineering judgment is usually based on limited information at the project site. Considering these difficulties, it is common for geotechnical practitioners and researchers to adopt 'correlation models' (also known as transformation models), which are empirical relationships between various material indices and properties, developed through laboratory and field test results. Conceptually, parts of these correlations can be explained through the physical phenomena in soil response. For example, the physical appearance and behavior of soils under various moisture contents – normally expressed by index properties of liquid limit (LL) and plastic limit (PL) – affect the interactions between water and soil particles under loading, and should, therefore, be related to the undrained shear strength (s_u) of soils. Numerous correlation

[1] https://github.com/vd1371/Developing-a-Deep-Neural-Network-based-Framework-for-Complex-Correlations-in-Engineering-Metrics.

relationships have therefore been developed between LL and/or PL with s_u [14,15,35,36,37,38]. In addition, since the determination of s_u requires particularly high-quality soil sampling and expensive testing procedures, it is preferable to predict s_u through other soil variables (e.g., LL and PL) that are cheaper and easier to obtain. This also gives rise to the need for such correlation formulas in the geotechnical engineering profession. The exact functional forms of these relationships are typically determined through regression analysis, and should be site-specific, which means correlations established using the dataset of a certain soil type at a certain location do not necessarily apply to other soils at other project sites, although this aspect is rarely given due consideration because of the abovementioned difficulties in sampling and characterizing soil properties.

This study proposes a model for estimation of undrained shear strength of clays using DNN, as an illustration of the applicability and potentials of deep learning methods in engineering. Performance of other models such as linear regression, ANN, and results from previous studies are also compared with DNN. Based on the proposed procedures for neural network formulation, soil data from various parts of the world can be analyzed effectively. The dataset used in this study is a combination of three separate datasets referred to as Clay/10/7490, S-Clay/7/168, and F-Clay/7/216, made publicly available at the website of the Technical Committee TC304 (Engineering Practice of Risk Assessment and Management[2]) of the International Society for Soil Mechanics and Geotechnical Engineering. The S-Clay/7/168 dataset, consisting of 168 samples of field vane test results for s_u, was gathered from 12 different sites in Sweden and 7 sites in Norway [15]. The F-Clay/7/216 dataset, with 216 test results of the field vane test, was compiled from 24 different sites in Finland [15]. Clay/10/7490 is a much larger dataset with over 7000 test results, consisting of field vane test, piezocone test, and standard penetration test, from different parts of the world [38]. Table 1 shows the available information from these datasets, including s_u, LL, PL and stress conditions such as the vertical effective stress ($\sigma_v^{'}$) experienced by the soil in situ.

Of the data in Clay/10/7490, only about 10% of the samples were complete without any missing values or included the attributes measured and collected in the other two datasets. Only these complete samples were used in this study. For example, Fig. 1 shows the liquid limit and pre-consolidation stress data from the three databases to illustrate the distribution of the data.

3. Methodology

Deep neural network models consist of three main parts: (1) input layer, (2) hidden layers, and (3) output layer. Fig. 2 depicts a deep neural network with three hidden layers. Each hidden layer consists of a number of nodes and an activation function. It is common to use one activation function for all the neurons within a layer, although it is possible to use different activation functions.

The output of each layer becomes the input of the next layer, and is calculated by Eq. (1), where $h^0 = x$ is the input of the network [25].

$$h^k = \sigma^k(b^k + W^k h^{k-1}) \tag{1}$$

where k is the layer number, σ^k is the activation function of the layer k, h^k is the output array of the layer k, b^k is the array of bias values in layer k, W^k is the matrix of weights of the layer k. The output of the final layer, $\hat{y}$, is the prediction of the output variable y.

Activation functions are used to add non-linearity to the relations between the input and output variables. If linear activation functions are adopted in all layers, the prediction of DNN will be a linear combination of input variables regardless of the number of layers in the network [39]. Hornik et al. [40] showed that by using "squashing functions" and sufficient hidden layers, any function with any degree of non-linearity could be approximated. Squashing functions are non-linear functions that map the input variables to an interval such as $[-1,1]$. Also, during the past decade, rectifiers such as *relu* [41] were proposed and used especially in hidden layers. Not only is this activation function non-linear, but it also does not activate all the neurons all the time [41]. Table 2 shows some of the most common and well-known activation functions. Activation functions must be non-linear and differentiable [42], in order to facilitate the analysis by optimization algorithms.

Training a DNN means minimization of dissimilarity between the actual output variables and the predictions. A cost function representing the aforementioned dissimilarity should be defined for this purpose. Table 3 shows some of the most common cost functions for training a DNN on a dataset with m samples.

Weights in all layers of the deep neural networks are initialized at the beginning of the training, and values of the bias vector are considered to be constant (usually 1). An initial prediction will be made by the DNN model with randomized weights. The weights of DNN will then be updated in an iterative manner by optimization algorithms to minimize the cost function. This optimization is usually called and done on the training set.

Optimization algorithms used in training a DNN are mostly extensions of the batch gradient descent (BGD) algorithm. According to Ruder [43], gradient descent [44] is an approach to minimize an objective function (or cost function) $J(\theta)$ by updating the model parameters ($\theta \in \mathbb{R}^d$) in the opposite direction of the gradient of the objective function, i.e. $\nabla_\theta J(\theta)$. In this process, the learning rate η determines the size of the steps taken to approach the (local) minimum. Eq. (2) shows the BGD formulations:

$$\theta_{\text{updated}} = \theta_{\text{old}} - \eta.\ \nabla_\theta J(\theta; x^m; y^m) \tag{2}$$

where m is the samples size, θ is a neuron's (connection between two nodes) weight in the DNN matrix of weights of each layer, η is the learning rate, x^m and y^m are the set of input and output variables respectively, and $\nabla_\theta J(\theta)$ is the average of the partial derivative of the cost function with respect to θ:

$$\nabla_\theta J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial\theta}J(\theta; x^i; y^i) \tag{3}$$

where x^i and $y^i$are the corresponding input and output variables of sample i. BGD updates the θ parameters utilizing all samples in the entire dataset. Therefore, converging to the optimum point would be too slow when the dataset is large. In addition, BGD cannot be used in online projects (where the datasets grow in time). Stochastic gradient descent (SGD) [45,46], as shown in Eq. (3), update the parameters for each training sample, one at a time. This is repeated for all samples on the dataset and each of these iterations is called an epoch.

$$\theta_{\text{updated}} = \theta_{\text{old}} - \eta.\ \frac{\partial}{\partial\theta}J(\theta; x^i; y^i) \tag{4}$$

SGD converges to the minimum point much faster than BGD and can be used in online projects. However, it fluctuates redundantly to reach the minimum point and might continue updating after reaching the minimum point since it has to run over all of the samples in the training dataset [43]. To overcome this problem, mini-batch gradient descent was proposed, which updates the parameters on mini-batches with n training examples randomly chosen from the whole dataset:

$$\theta_{\text{updated}} = \theta_{\text{old}} - \eta.\ \nabla_\theta J(\theta; x^{[i:i+n)}; y^{[i:i+n)}) \tag{5}$$

where $[i{:}\ i+n)$ is a subset of input and output variables with the size of n. The main problem of mini-batch gradient descent is that the learning rate (η) is the same for updating all parameters, and choosing a learning rate is difficult [43]. Also, Dauphin et al. [47] discussed the limitations of SGD with saddle points, where the slopes of the target function in two dimensions have opposite trends. Many algorithms have been

[2] http://140.112.12.21/issmge/tc304.htm.

Table 1
Datasets used in this study and the available attributes of clays.

Dataset	Locations	Attributes	Number of Samples
S-Clay/7/168	12 sites in Sweden and 7 sites in Norway	LL, PL, w, $\sigma_v^{'}$, $\sigma_p^{'}$, s_u	168
F-Clay/7/216	24 sites in Finland	LL, PL, w, $\sigma_v^{'}$, $\sigma_p^{'}$, s_u	216
Clay/10/7490	various countries	LL, PL, w, $\frac{\sigma_v^{'}}{Pa}$, $\frac{s_u}{\sigma_v^{'}}$, S_t, $\frac{q_t-\sigma_v}{\sigma v^{'}}$, $\frac{q_t-u_2}{\sigma_v^{'}}$, B_q	7490

Note: LL: Liquid limit(%), PL: Plastic limit(%), w: Water content(%), , $\sigma_v^{'}$: Vertical effective stress(kPa), $\sigma_p^{'}$ Preconsolidation stress(kPa), s_u: Undrained shear strength (kPa), PI: Plasticity index (%), P_a: Atmospheric pressure, S_t: Sensitivity, $\frac{q_t-\sigma_v}{\sigma v^{'}}$: Normalized cone tip resistance, $\frac{q_t-u_2}{\sigma_v^{'}}$: Effective cone tip resistance, B_q: Pore pressure ratio.

proposed to solve these problems, such as Nesterov Accelerated Gradient [48], Momentum [49], Adagrad [50], Adadelta [51], RMSprop [52], and Adaptive Moment Estimation (Adam) [53] algorithms. Adadelta, RMSprop, and Adam are conceptually similar, i.e. they update the initial learning rate for different weights (W_{ij}) during optimization, but Kingma and Ba [53] showed that Adam optimizer performs slightly better than the others. The Adam algorithm modifies the learning rate of step t by the first moment m_t, and the second moment, v_t, of past gradients, as shown in Eq. (6) and Eq. (7) :

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta) \tag{6}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta J(\theta))^2 \tag{7}$$

where β_1 and β_2 are user-defined constants. Since m_0 and v_0 are normally initialized as zero vectors, the values of m_t and v_t tend to remain close to zero in subsequent iterations. To resolve this issue, Kingma and Ba [53] proposed the use of bias-corrected formulas:

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t} \tag{8}$$

$$\widehat{v_t} = \frac{v_t}{1-\beta_2^t} \tag{9}$$

Finally, accordingly to the Adam algorithm, the parameters are updated by the following:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v_t}} + \epsilon}\widehat{m_t} \tag{10}$$

They proposed $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. These parameters are inherent features of the algorithm and usually are not changed by data analysts. These are adopted in the optimizer algorithm in this study, while other optimizers such as AdaMax [53] and Nadam [54] may be assessed in future research and is not in the scope of this study. It should be noted that reaching the global minimum is not always favorable, because this may be associated with the problem of overfitting [55], which will be discussed in more details in later sections.

4. Model development

The creation and assessment of DNN models consist of six main steps, as shown in Fig. 3. In the first step, some hyperparameters of the DNN structure are determined based on the nature of the problem. Then in the second step, the dataset is cleaned with outliers removed, and the data is scaled so the DNN can be trained more effectively. The third step involves obtaining information on the sufficiency of data and adequacy of input variables, as well as finding an initial guess for the design of the DNN model. Given the starting point for the design of DNN, a few potential network structures are assessed with one final structure chosen in the fourth step. Subsequently, the issue of model overfitting should be checked and resolved, if necessary, in the fifth step. The last step involves assessment of the performance and validity of the model.

A prediction model must perform better than the base model to be considered viable. The average of the output variable in regression models or random labeling in classifiers could be considered as the base model. The variance error, i.e. prediction error, of an acceptable model must be smaller than the error of the base model, referred to as baseline error (BLE) in this study. With the simplest prediction model, the mean of squared error (MSE) of the prediction for the output variable will be equal to its variance:

$$MSE = \frac{1}{m}\sum_{i}^{m}(\widehat{y_i} - y_i)^2 = \frac{1}{m}\sum_{i}^{m}(\mu_y - y_i) = \sigma^2 \tag{11}$$

Therefore, the MSE of a viable prediction model must be smaller than the variance of the output variable.

Prediction models can usually have two problems in making predictions: (1) Variance problem, (2) Overfitting problem (Bias problem). The variance problem refers to the situations where the model performs merely slightly better than the base model. For example, a binary classifier with classification accuracy similar to tossing a fair coin. A model suffers from overfitting when it performs well on the training set (i.e. the subset of data for fitting and training) but poorly on a dataset different from the training set (e.g. the subset for cross-validation). The difference between prediction errors on the training set and cross-validation set is called the bias error. In fact, with a sufficient number of hidden layers and nodes, it is possible to create a model with almost zero error on the training set, but it will not perform well on any other datasets apart from the training set itself, which defeats the very first purpose of formulating the DNN. To detect and solve the overfitting problem, the dataset is usually sliced into two portions (such as 70:30 splits). The bigger slice is called the training set and the other one is the test set. However, since the creation of an optimal DNN model requires testing and comparing numerous models before reporting the model performance, it is common to slice the dataset into three parts, with 60:20:20 portions. The middle slice is called the cross-validation set and is used for comparison of performance between different candidate models. Finally, the performance of the chosen optimal model on the test set will be reported. A properly constructed model must have low bias error and low variance error.

It should be noted that modeling of datasets with trends (e.g. prediction of energy price) or datasets with inherent clusters (e.g. those gathered from various countries) can be problematic if the data is not shuffled prior to modeling [39]. This is mainly because the DNN can potentially adapt to a section of the trend or certain clusters (data from certain countries) in the entire set, which can compromise its capability of predicting the remaining parts of the dataset.

4.1. Preliminary DNN structure setup

DNN models can be repeatable: they can be built and used by modelers, and then re-built by others to achieve the same accuracy with the implementation of the same hyperparameters. These hyperparameters normally include types of layers, number of layers, number of nodes in each layer, activation functions for each layer, cost function, early stop value and patience number, batch size, and the optimization algorithm, and the learning rate. Of the mentioned hyperparameters,

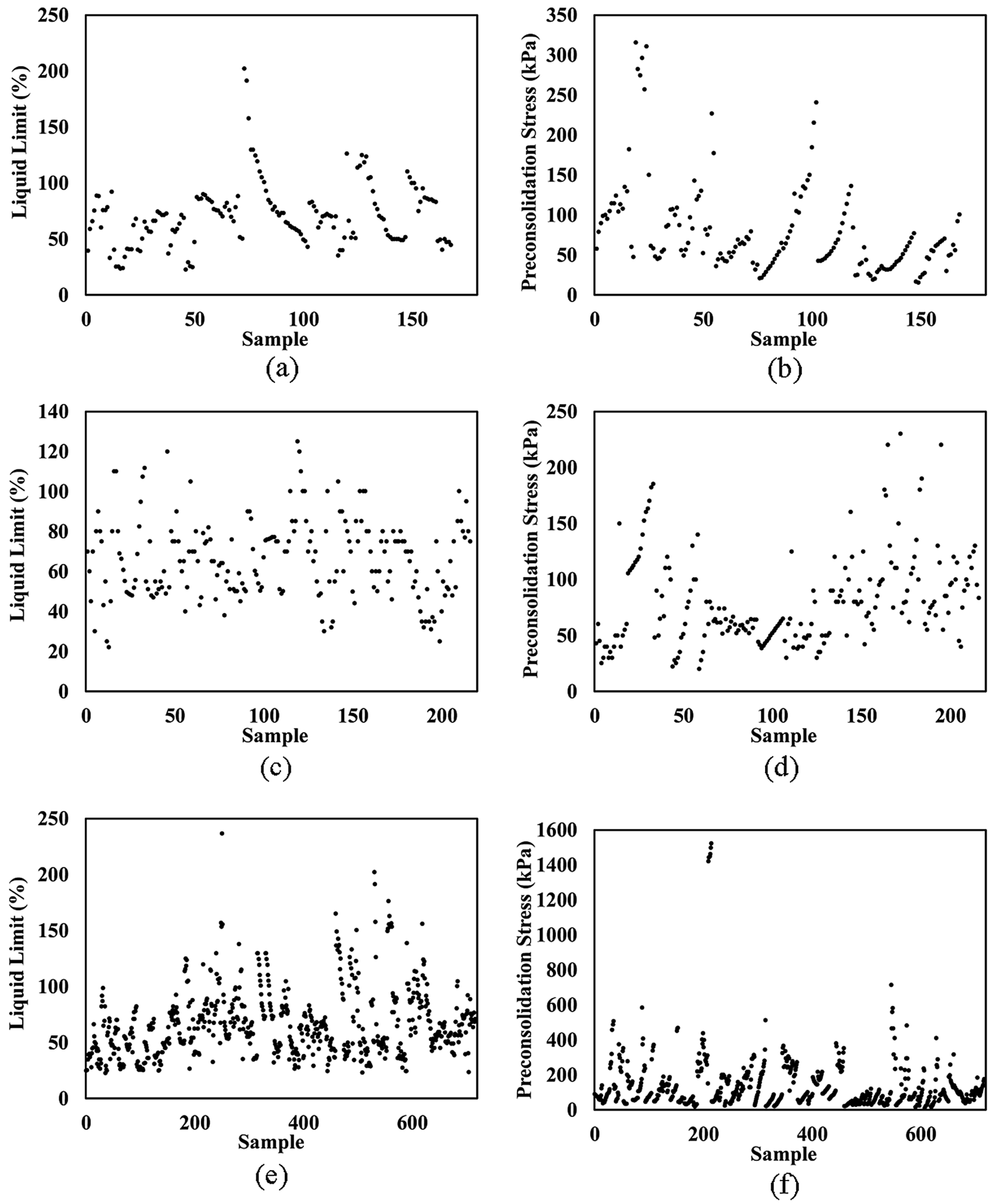


Fig. 1. (a) Liquid limit of S-CLAY/7/168, (b) Preconsolidation stress of S-CLAY/7/168, (c) Liquid limit of F-CLAY/7/216, (d) Preconsolidation stress of F-CLAY/7/216, (e) Liquid limit of CLAY/10/7490, (f) Preconsolidation stress of Clay/10/7490.

some should be determined based on the nature of the problem prior to conducting the model analysis, and others should be tested or identified by the procedures to be elaborated in the following sections. There is no universal rule for the choice of some of these hyperparameters of DNN, and these are mostly chosen based on the experience of the modeler. However, Ng [39] provided some guidelines about choosing the number of layers and data regularization methods.

Table 4 summarizes the hyperparameters of a typical DNN and the corresponding values adopted in this study. The final activation layer must be linear since the ultimate goal of this study is to predict the undrained shear strength of the clays as a regression problem. Tanh and Relu activation functions were used as the activation function for the first layer and hidden layers, respectively. Although convolutional layers, max-pooling layers, fully connected layers, etc. are usually used in DNN networks, especially for computer vision problems, fully connected layers are more applicable to this study. Discussion about other layers and their use is not in the scope of this study. Mean of squared error of the predictions has been used, as the cost function. The Adam

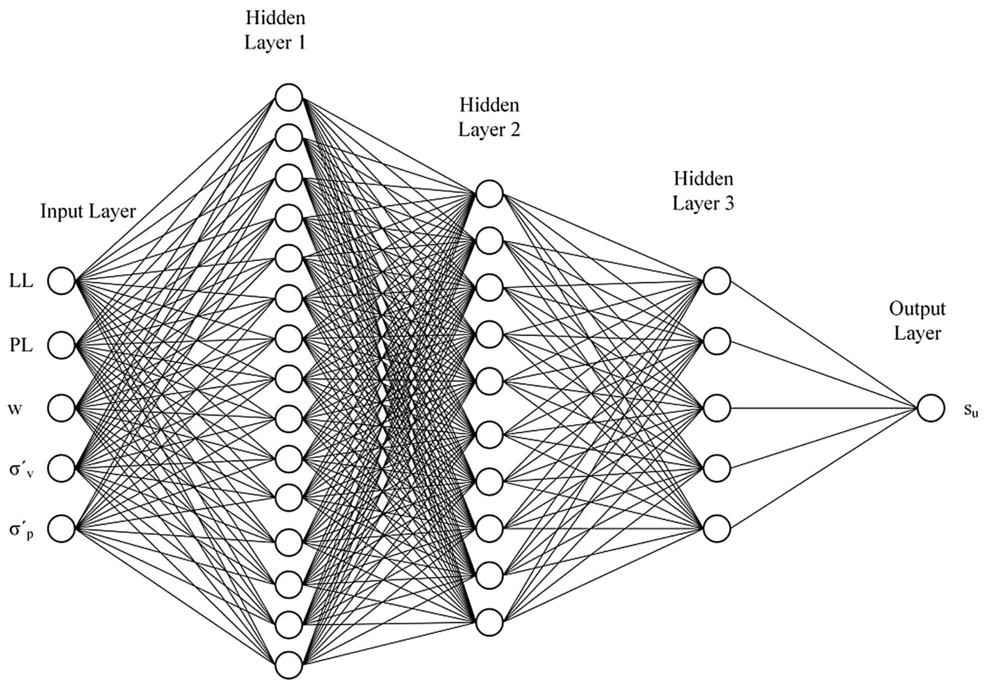


Fig. 2. An example of DNN with three hidden layers.

Table 2
Some of the most common activation functions.

Activation Function	Formula
sigmoid	$s(x) = \frac{1}{1+e^{-x}}$
tanh	$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
relu	$y(x) = \max(0, x)$
leaky-relu	$y(x) = \max(\alpha x, x)$: $0 < \alpha \ll 1$
linear	$y(x) = x$

Table 3
Common cost functions used in deep neural network modeling.

Cost function name	Abbreviation	Formula
Mean of squared error	MSE	$J(y,\hat{y}) = \frac{1}{m}\sum_i^m\left(\hat{y}_i - y_i\right)^2$
Mean of absolute error	MAE	$J(y,\hat{y}) = \frac{1}{m}\sum_i^m\left(\lvert\hat{y}_i - y_i\rvert\right)$
Mean of absolute percent error	MAPE	$J(y,\hat{y}) = \frac{1}{m}\sum_i^m\left\lvert\frac{\hat{y}_i - y_i}{y_i}\right\rvert$
Mean of squared logarithmic error	MSLE	$J(y,\hat{y}) = \frac{1}{m}\sum_i^m\left(\log(\hat{y}_i + 1) - \log(y_i + 1)\right)^2$

optimizer was also used as the optimization algorithm, and the size of mini-batches was selected to be 32 [56]. Optimizers iterate a certain number of epochs over the whole dataset, or until the reduction of cost function within a period of epochs are less than a certain tolerance. This method also resolves the overfitting problem by preventing the model from further training on the training set. The period is called patience and the values are called minimum delta in the KERAS library. In this study, the total number of epochs is designated to be a large number so that it does not control the termination of the training process. When the reduction of the cost is less than 1% of the variance of the output variables in the previous 50 steps [57], the optimization will be stopped.

4.2. Data preprocessing

Data preprocessing is the first step in creating any statistical, machine learning, or data mining models. With a dataset full of errors and outliers, even the most powerful statistical or machine learning algorithms will not yield better predictions than average of the output. Missing data, outliers, and differences in dimension or order of input variables are common problems in engineering datasets, and more specifically those associated with underground soil sampling in geotechnical engineering. In addition, engineering datasets have specific characteristics in comparison to datasets being used in computer science such as computer vision. One of the key differences lies in the availability of data. For example, through some simple image processing techniques such as mirroring or cropping, it is possible to expand a dataset of images. In civil engineering, however, gathering more information or soil samples at a site usually carries significant cost implications. In addition, pixels of an image are in a specific range (i.e. from 0 to 255), while the distribution of the collected data in engineering datasets is not limited. This issue affects the training of DNN and its performance. Due to these unique features, engineering datasets require special data cleaning and outlier detection considerations.

Data cleaning is the first step in data preprocessing, which ensures a dataset is ready for the subsequent modeling steps. Missing values, typos by the operators, or other human errors in data collection stages may cause various inconsistencies and flaws. In addition, the limitations of the test machines can lead to similar problems. Some datasets, also, contain attributes that are not important for certain purposes or intended output variables. For example, Clay/10/7490 consists of many attributes such as site ID or country which may not affect the physical behavior of the soil material, and are therefore not reported in the two other datasets (F-CLAY/7/216 and S-CLAY/7/168). It is also possible that some values are reported incorrectly due to human error which should be eliminated. An example of such problems would be negative values for water content or shear strength parameters, although such errors have not been identified in the three datasets used in this study. It should be noted that wrong values should not be mistaken with outliers, which affect data modeling in a different way and require a different type of consideration.

Outliers alter the statistical characteristics of the dataset,

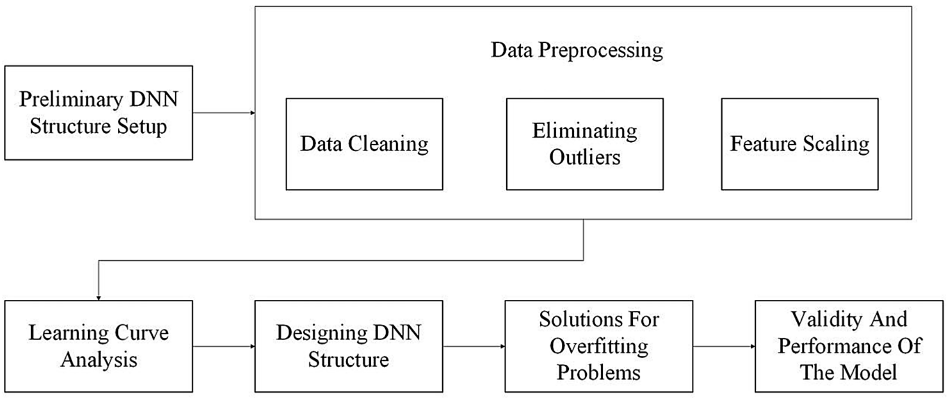


Fig. 3. Steps toward conducting a DNN model.

Table 4
Hyperparameters of a DNN structure.

Hyperparameters	Status
Type of layers	Fully connected layers
Input layer activation function	*tanh*
Hidden layers activation function	*relu*
Output layer activation function	*linear*
Optimizer	Adam
Cost function	MSE (Mean of Squared Error)
Minimum delta, and patience	$0.01\sigma^2$, 50
epoch	50,000
Batch size	32
Slicing proportions	60:20:20
Number of hidden layers and nodes	To be decided in the following steps
Weight regularization parameter and type	To be decided in the following steps
Learning rate	0.0001 (The default value of KERAS)

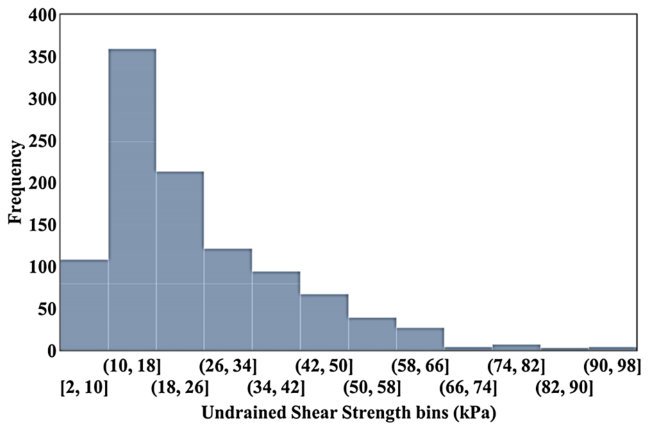


Fig. 4. Histogram of undrained shear strength.

performance of any prediction model, and affect the understanding of the relationship between various variables of the problem. The outliers should be found and eliminated from the dataset prior to fitting any prediction model. The detection and elimination of outliers can be performed based on probability theories, expert judgment, and/or domain knowledge of the data analyst. Outliers are not necessarily due to human errors by the operator, or faults in the test equipment, but are usually related to the nature of the problem. For example, they may arise due to the existence of a particularly strong or weak soil layer at the site. Although they are parts of the real site condition, they will alter the statistical measures of the model and the ensuing training of DNN can be biased and problematic. Numerical techniques such as multivariate outlier detection [58] can be used to eliminate the outlier. For example, Garrette [59] used this approach in geochemical data analysis to eliminate outliers in a dataset with the chi-squared distribution. The algorithm for outlier detection also depends on the pattern or distribution of the data (e.g. Gaussian distribution or lognormal distribution) itself. The input and output variables in this study are shown to be lognormally distributed. Fig. 4 shows the histogram of s_u in this dataset.

Since various aspects of soil behavior are generally correlated, the multivariate log-normal distribution can be used to identify outliers:

$$f_X(x_1,\cdots,x_i,\cdots,x_n)=\frac{\exp\left(-\frac{1}{2}(\ln(X)-\mu))^T\Sigma^{-1}(\ln(X)-\mu)\right)}{\sqrt{(2\pi)^n|\Sigma|}}\prod_{i=1}^{n}x_i^{-1} \tag{12}$$

where f is the multivariate log-normal distribution density function, X is the vector of variables, μ is the mean vector, and Σ is the covariance matrix of variables. The probability density of each sample in accordance with the multivariate normal distribution has been calculated. It is assumed that 5% of data are among outliers in the dataset. Then, the samples with relative likelihood of less than 95% of all samples were eliminated. Notably, this method is not bounded to DNN models and has also been applied to other statistical or machine learning approaches. Table 5 shows some statistical characteristics of the dataset before and after removing outliers.

Based on the results of Table 5, the variance and covariance of the variables reduced drastically by eliminating 5% of the data. For example, variance and covariance of $\sigma_p^{'}$ has reduced by around 66% and 30% respectively. To illustrate these effects Fig. 5 shows the undrained shear strength of the samples before and after removing outliers. The variance of the output variable, i.e. undrained shear strength, is about 250. Therefore, the BLE is 250 and the minimum delta will be 2.5.

Datasets usually consist of input variables with different orders and ranges. This may lower the efficiency of the optimizer and lead to premature termination of the optimization process [39]. To resolve this issue, feature scaling is a vital step in training a DNN. Standardization transforms the data so the average and standard deviation will be 0 and 1, respectively:

Table 5
Statistical characteristics of variables before and after eliminating outliers.

Parameter	Average		Var		CoV	
	Before	*After*	*Before*	*After*	*Before*	*After*
LL	66.89	66.02	789.31	663.56	0.42	0.39
PL	28.04	27.71	92.39	75.65	0.34	0.31
w	71.64	71.66	753.13	645.45	0.38	0.35
$\sigma_v^{'}$	65.89	61.16	3344.18	2198.07	0.88	0.77
$\sigma_p^{'}$	116.31	102.05	18155.90	5916.11	1.16	0.75
s_u	27.65	24.69	608.11	249.20	0.89	0.64

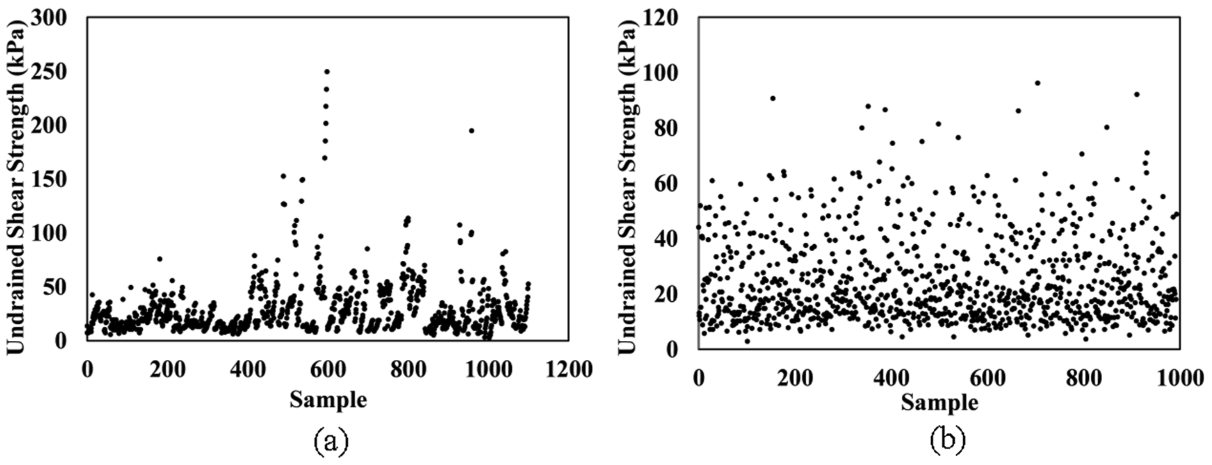


Fig. 5. Undrained shear strength of clays (a) before eliminating outliers, (b) after eliminating outliers.

$$y(x_i) = \frac{x_i - \mu_x}{\sigma_x} \tag{13}$$

where μ is the average of the input variable x, and σ is the standard deviation. It should be used when the data follows the Gaussian distribution. Normalization, on the other hand, should be used when the data is uniformly distributed. It maps the data to an interval such as [0,1]:

$$y(x_i) = \frac{x_i - \min_x}{\max_x - \min_x} \tag{14}$$

4.3. Learning curve analysis

Learning curve analysis provides valuable information about the sufficiency of the dataset and the potential of training a useful model. It can also serve as a starting point for the further design and modification of the DNN model structure. Learning curve analysis can be conducted in an arbitrary number of steps. The training dataset is usually sliced into portions with respect to the step number: in the first step, a small portion of the training set is used for learning curve analysis. This portion gradually grows with the step number according to Eq. (14), and finally, the entire training set will be used in the last step:

$$X_i = X^{\left(0:\left(\frac{i}{n}\right)\times l\right)} \tag{15}$$

where i is the step number, n is the total number of steps, and l is the length of the training set. The training error and cross-validation error, depending on the cost function, will be plotted versus the step number. Results of the learning curve analysis should satisfy the following: (1) the sample size is large enough, and no further data collection is needed, (2) a model with low prediction error could be built and trained in the next steps. An initial neural network structure should be designed at this point, though it will be further modified in the final modeling step. Considering the number of input variables in this study, i.e. five attributes, the initial structure of the neural network is suggested to involve two layers, with 10 neurons in each layer. Other parts of the neural network model, such as activation functions and optimizer algorithms, have been described in the previous sections.

Learning curve plots may take various forms, each of which has its own interpretation. Fig. 6 depicts four of the most common output of learning curve plots.

(1) Training error and cross-validation error have converged after the 6th step, and the converged error is less than the variance of the model. This indicates that the adopted model structure is appropriate, and the number of attributes and samples is sufficient.
(2) The training error and cross-validation error are both smaller than the variance, but the two have not converged and remained parallel to each other after the 7th step. This means adding new samples to the dataset will not be helpful in creating better models. Since there is a gap between the two, the model suffers from overfitting. The solution for this will be discussed in later sections.
(3) The training error and cross-validation error are approaching, but have not yet reached convergence, within all of the steps. Fig. 6(c) shows the aforementioned situation and indicates that better results can be achieved with more data samples. Since the training and cross-validation errors are about to converge to a smaller error than the sample variance, the prospective model created with further data should be valid.
(4) In this figure, cross-validation error and training error does not converge, and the models suffer from bias error. This usually means that either the DNN is overly complex, or no relationship exists between the input and output variables. These issues can be solved by simplifying the neural network structure, identifying and including other input variables, or feature selection, respectively. Meanwhile, the separation distance between the cross-validation error and variance error can also provide new insights. If the cross-validation error line is close to the BLE line, the attributes are not useful and feature selection will not be helpful. On the other hand, if the aforementioned two lines have a reasonable distance from each other, simplification of the structure or feature selection should be considered as a solution.

A learning curve analysis has been performed with 10 steps for the datasets in this study. Fig. 7 depicts the corresponding learning curve plot. From this figure, the following conclusions will be deduced:

1. The size of the dataset is sufficient as the cross-validation error line and training-error line converged.
2. A useful model can be built with the current attributes since the convergence line is much lower than the BLE.
3. Two hidden layers with 10 nodes in each layer is a good starting point as the error is relatively low in comparison to BLE.

4.4. Designing deep neural network structure

Tuning and choosing the hyperparameters of deep neural networks determine the effectiveness of the model. Numerous structured and designed models have been used for machine learning problems and have been proposed in the literature. For example, computer scientists have proposed specific structures for computer vision problems, and similar structures have been used in other disciplines, such as VG16 [26]. Engineering problems have unique characteristics and a different

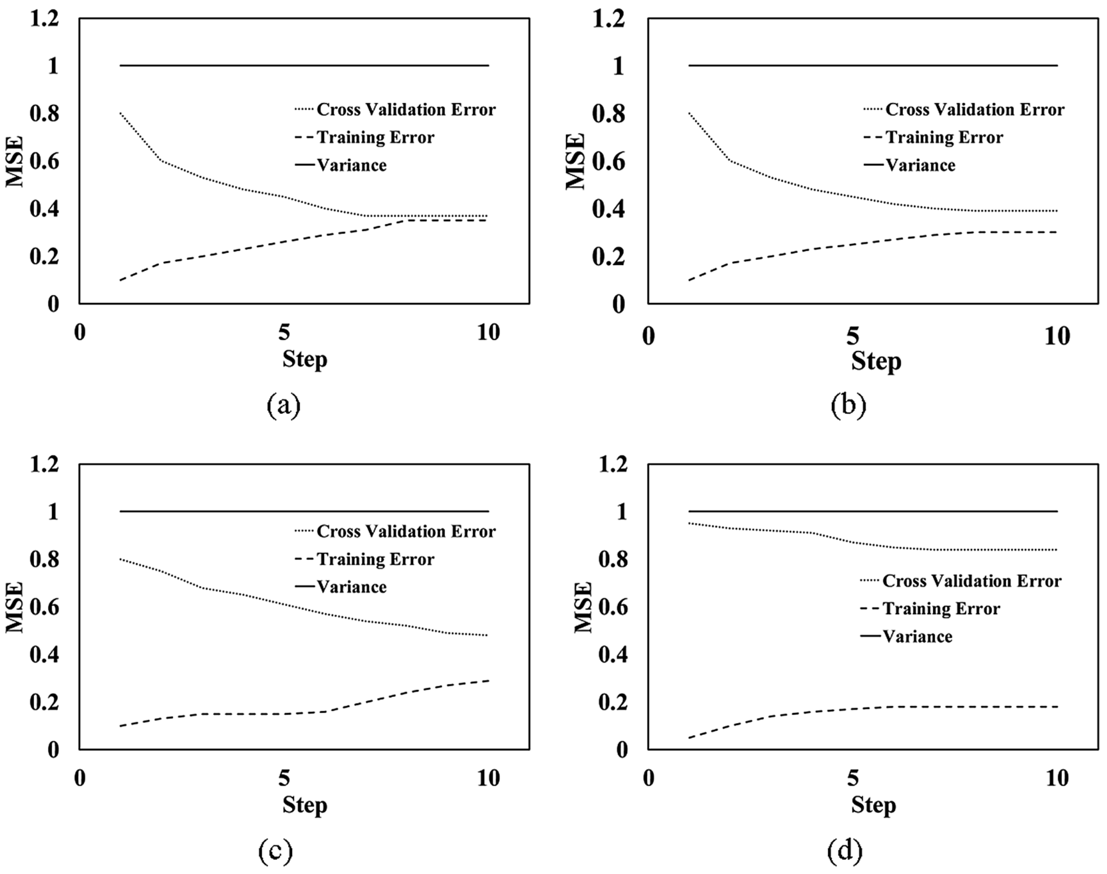


Fig. 6. Possible outcomes of learning curve analysis.

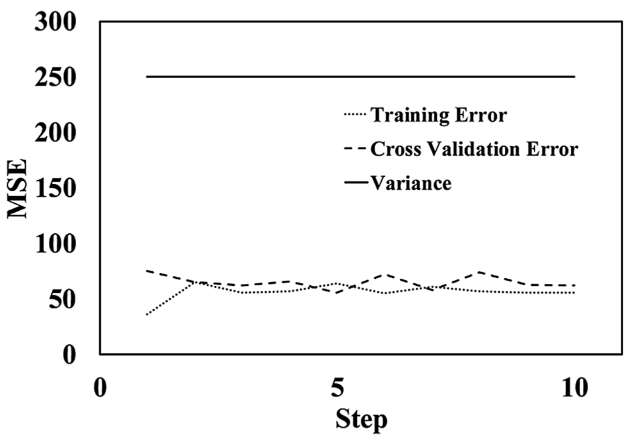


Fig. 7. Learning curve plot of the initially constructed DNN model.

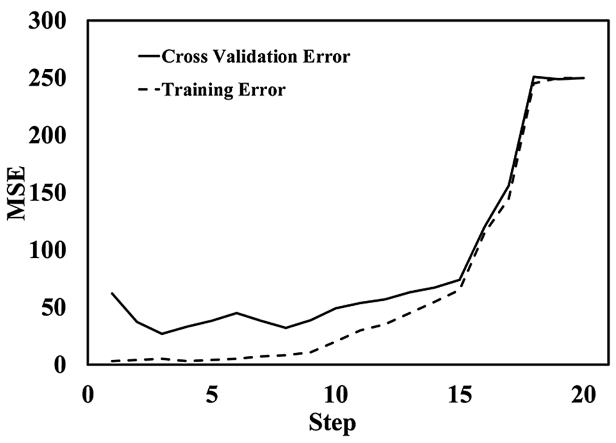


Fig. 8. Assessing training and cross-validation error in 20 steps for different regularization parameters.

Table 6
Trial and Error results for finding a suitable DNN structure.

Model No.	Hidden layers and nodes	Training error	Cross-validation error	Comments
1	10	92	130	Variance error and bias error
2	10,10	66	71	Starting point
3	15,15,15	41	68	Bias error
4	15,10,5	55	69	Acceptable
5	20, 20, 20, 20	28	72	Bias error

neural network structure needs to be created for each problem. Unfortunately, there is no universal rule to determine the optimal number of hidden layers and nodes. As discussed earlier, a more complex model structure tends to result in a smaller error in the training set but can suffer from overfitting problems. On the other hand, by reducing the complexity of the model, it may not perform well on datasets with high degrees of non-linearity. The results of learning curve analysis could be the starting point for the trial-and-error process. As shown in Fig. 7, the DNN model with two hidden layers, each containing 10 nodes, results in training error converging with the cross-validation error at around 60, which is much smaller than the BLE of 250.

The final structure of the DNN is usually fine-tuned by trial and error, which is very common for computer science problems with

Table 7
Model performance on the data with *L1* and *L2* regularizers.

Training Session	Training Error	Cross-Validation Error	Test Error	MAE	MAPE	R^2
DNN with L1 regularizer	64.56	57.30	61.74	5.48	23.40	0.74
DNN with L2 regularizer	66.39	71.29	65.89	5.51	25.41	0.71

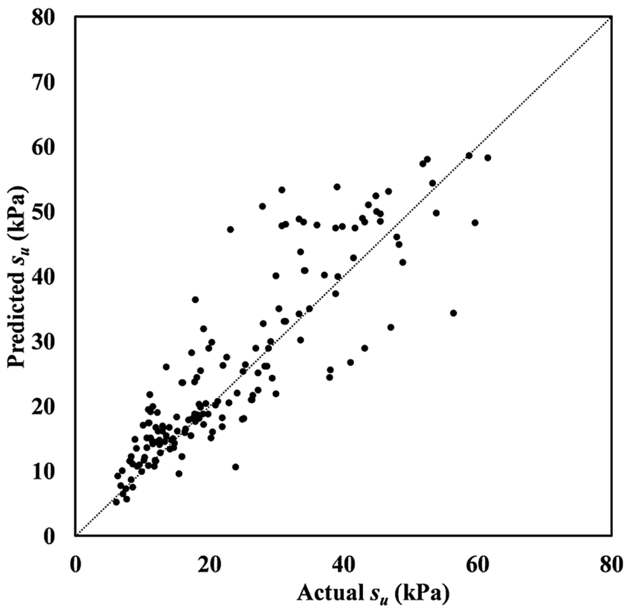


Fig. 9. Predicted and actual values of undrained shear strength on the test set.

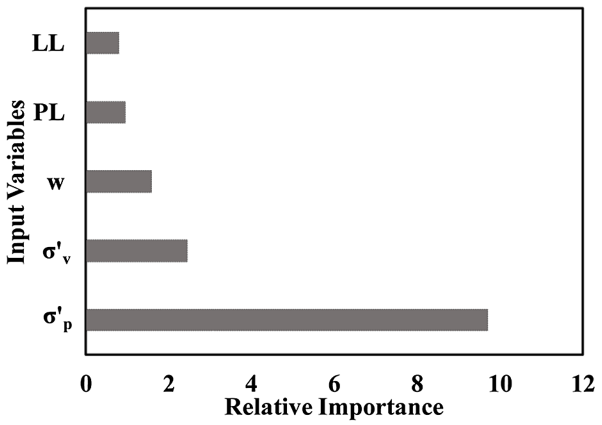


Fig. 11. The relative importance of input variables based on SHAP values for the proposed model.

hundreds or thousands of inputs. For engineering problems that involve smaller datasets, a structure with sufficient complexity should be chosen, and the overfitting problem should be solved by a regularization method or using dropout layers. The DNN model structure should normally be proportional to the sample size and number of input variables. For example, a structure with 10 hidden layers and 20 nodes in each layer in a DNN model is too large for the dataset used in this study, which consists of almost 1000 samples and 5 input variables. It is expected that variance error increases by using simpler structures, i.e. less hidden neurons and layers, since an insufficient number of nodes might not be able to represent the linear or non-linear relationship between the input variables and output variable. On the other hand, bias error increases by using more complex structures, i.e. more hidden layers and neurons. Table 6 shows the training error and cross-validation error for 5 different DNN structures with different levels of complexity. Although more complex structures of DNN models could possibly provide more accurate results, small dataset size may hinder the model to be trained well.

Based on the results of learning curve analysis, two hidden layers with 10 nodes is a good starting point, i.e. Model 2. According to Table 6, by simplifying the structure, a simple ANN model will be derived and the variance error increases (Model 1). A more complex structure would be Model 3 with three hidden layers and 15 nodes in each layer. However, this model suffers from bias error and therefore a slightly simpler model (which should be more complex than Model 2) should be tested. Model 4 with three hidden layers with 15, 10, 5 layers provided acceptable results. The difference between the training error and the cross-validation error in Model 4 is acceptable. Therefore, Model 4 will be chosen as the candidate for the final structure of the model. However, this model suffers from minor overfitting problem which needs to be addressed and solved. A more complex model (Model 5) has encountered bias error and there is no need for further investigation and assessment.

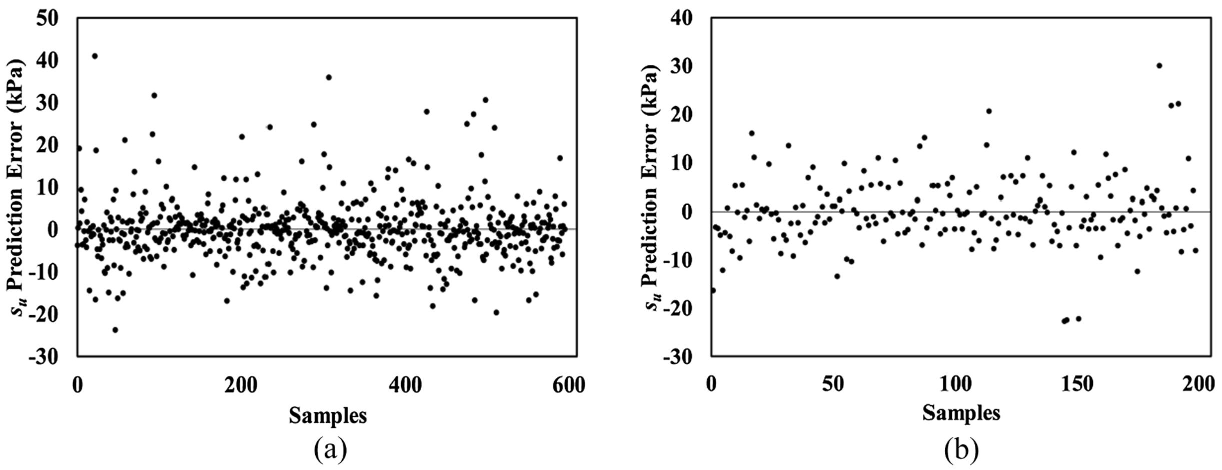


Fig. 10. Predictions error on (a) train set, (b) test set.

Table 8
Comparison between previous correlation formulas, linear regression and DNN for prediction of s_u.

	Model/Formula	MSE	MAE	MAPE	R^2	Comments	Corr. Formula	Pred. Vs. Act.
Literature	$\frac{s_u}{\sigma'_p} = 0.45LL$	240.1	9.68	41.12%	0.51	Hansbo [14]	Fig. 12(a)	Fig. 12(b)
	$\frac{s_u}{\sigma'_p} = 0.08 + 0.0055PI$	192.5	8.77	36.76%	0.53	Larsson [35]	Fig. 12(c)	Fig. 12(d)
	$\frac{s_u}{\sigma'_p} = 0.11 + 0.0037PI$	142.7	7.71	30.89%	0.58	Chandler [36]	Fig. 12(e)	Fig. 12(f)
	$\ln\left(\frac{s_u}{\sigma'_v}\right) = -0.87 + 0.24\mathrm{LI}$	460.1	14.87	69.2%	0.36	Ching and Phoon [38]	Fig. 12(g)	Fig. 12(h)
	$\ln\left(\frac{s_u}{\sigma'_v}\right) = -1.47 + \ln(\mathrm{OCR})$	121.4	7.44	29.57%	0.63	Ching and Phoon [38]	Fig. 12(i)	Fig. 12(j)
	$\frac{s_u}{\sigma'_v} = 0.23OCR^{0.8}$	113.3	7.41	41.59%	0.63	Jamiolkowski [64]	Fig. 12(k)	Fig. 12(l)
This study	Regression on the training set	86.8	6.56	31.36%	0.65	Overfitting problem		
	Regression on the test set	92.0	6.70	31.70%	0.63			Fig. 12(m)
	Shallow Neural Network	85.9	6.33	28.97%	0.64	No overfitting problem		Fig. 12(n)
	DNN/ *L2* regularizer	65.9	5.51	25.41%	0.71	No overfitting problem		
	DNN/ *L1* regularizer	61.7	5.48	23.40%	0.74	No overfitting problem		Fig. 12(o)

Note: **Corr. Formula**: correlation formula with the dataset used in this study, **Pred. Vs. Act.**: prediction versus actual values of the test set, LI: liquidity index, which is $(w - PL)/(LL - PL)$, and OCR: over-consolidation ration, which is σ'_p/σ'_v.

4.5. Solutions for overfitting problem

Dropout layers and weight regularization are two main tools for solving the overfitting problem in computer science [60,61]. Weight regularization sets a penalty value for weight coefficients in hidden nodes. By optimizing the cost function, the weights of nodes with less importance for the prediction of the output variable will be reduced compared to those of the important ones. *L1* and *L2* regularization let the optimizer minimize the sum of the absolute and squares of coefficients, respectively. A detailed explanation of the weight regularization process can be found in Ng [60]. Regularization can be best used in a dataset with limited input variables and a simple DNN model structure. Dropout is another technique for regularization and solving the overfitting problem. This algorithm ignores parts of nodes and links of a DNN model in each training epoch with a pre-determined probability, usually equal to 0.5. Further information and details can be found in Srivastava et al. [61]. It is mainly used in big DNN structures such as computer vision since not all of the nodes and links take part in the prediction. Implementation of dropout layers in engineering datasets (or similar ones) with a limited number of input variables and simple structures would usually lead to larger errors in comparison to weight regularization. This is because the input variables chosen by modelers in engineering disciplines are usually relevant (e.g. physically related) to the output variable, and dropout or ignoring some of the input variables may lead to less accurate results.

Adjustments of weight regularization parameters determine the final performance of the model while alleviating the overfitting problem. To this end, the training set error and cross-validation set error will be calculated for a range of values of *L2*-regularization weight. Small values of weight regularization parameter would let the model overfit on the dataset, while large values diminish the effect of input parameters on the output variable, so the average of output variable would be the prediction of the model. In that case, the variance of the output variable will be the model's error. Fig. 8 shows the regularization parameter analysis, in which the training error and cross-validation error are depicted. In each step, the regularization parameter grows by a factor of 3 with an initial value equal to 10^{-7}.

Around the point where the cross-validation set error is at its minimum (i.e. step 3), the optimal value for the neural network can be obtained. In this study, 10^{-6} has been chosen as the weight regularization parameter. The same procedures for *L1*-regularization lead the regularizer parameter to be approximately equal to 10^{-5}. Determination of the weight regularization parameter is the final step of constructing a DNN model structure. Finally, the only unknown parameter for creating a DNN model of Table 4 is found.

5. Results and discussion

The proposed framework was applied on a geotechnical dataset and its results are compared with related correlation formulas in the literature. The processing time of training machine learning models is one of the most crucial parts which needs to be addressed. Each training session with a Core i7-8700 T and 8.00 GB of RAM took 9.5 ± 0.3 seconds. However, this could vary drastically with larger datasets or more complex DNN structures. Table 7 summarizes the results of the proposed structure for two different types of regularization. Based on the results of Table 7:

1. Error (MSE) of the model with *L1* regularization is less than that for *L2* regularization; therefore, *L1* is chosen as the regularizer.
2. There is no evidence of overfitting in both cases as the error of the cross-validation set and training set are very similar. Therefore, the model does not have a bias error.
3. The prediction error of the model is less than the variance of the output variable which is approximately equal to 250; hence, the model does not suffer from variance problem.
4. Correlation between the prediction and actual values of the output variable is acceptable. Fig. 9 depicts the prediction versus the actual values of the output variable in the test set.
5. Fig. 10 also demonstrates the prediction error, based on which no trend or heteroscedasticity exists in the errors.
6. Due to the convergence of training and cross-validation error lines in Fig. 7, collecting more data would not lead to more accurate prediction, though adding other input variables might help in this regard.

Although DNN has been known as black-boxes, researchers [62,63] have proposed methods for better understanding of machine learning models including DNN models. Lunderberg and Lee [63] applied the game theory on machine learning models and proposed the indicator known as SHapley Additive exPlanations (SHAP). They considered each input variable as a player and the prediction as a game and tried to show the role of each input variable in every prediction. In other words, the SHAP values represent the importance of each input variable in the DNN regression models. Fig. 11 depicts the average of absolute SHAP values of the input variables which are calculated for all predictions.

Correlation formulas have been proposed and commonly adopted by the geotechnical engineering community to predict the undrained shear strength of clays, or obtain correlations between various aspects of soil properties. Some of these formulas are summarized in Table 8 and their comparisons are depicted in Fig. 12 which show their performance in predicting s_u in the test section of the current datasets.

It should be noted that many of these formulas were established using data from a small number of sites. The applicability or accuracy of

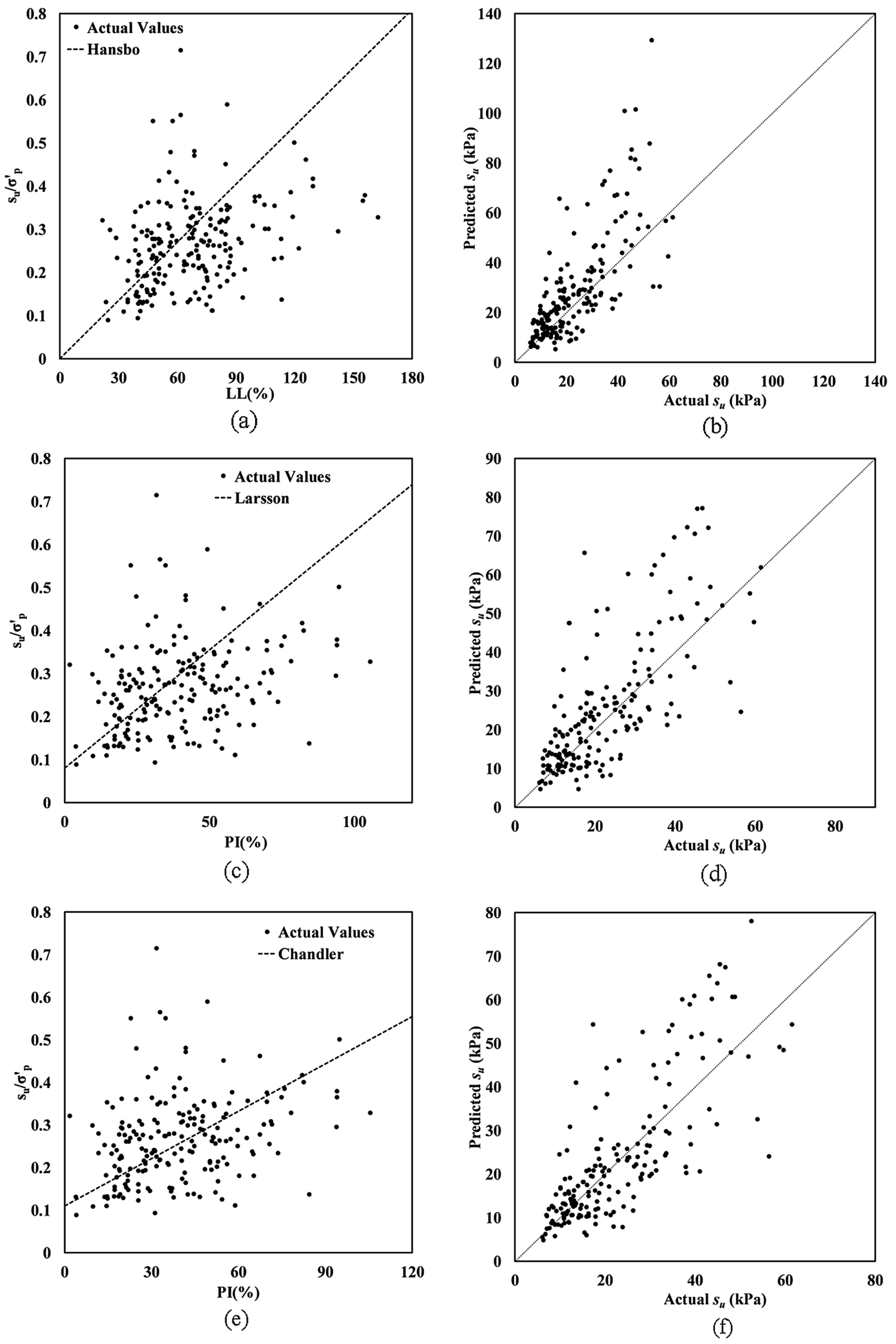
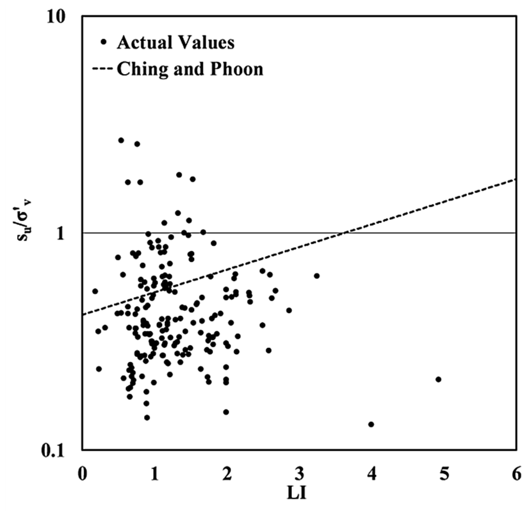
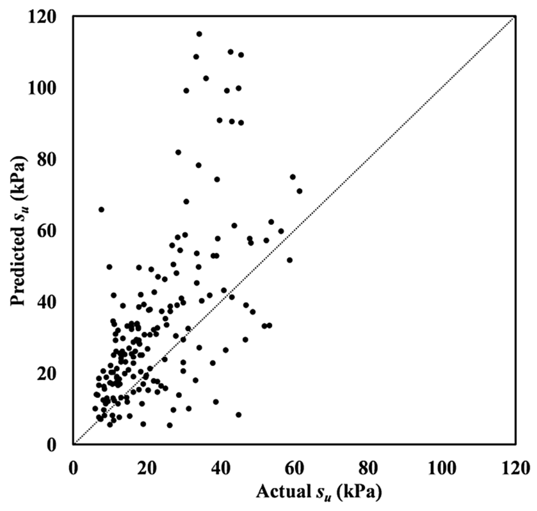


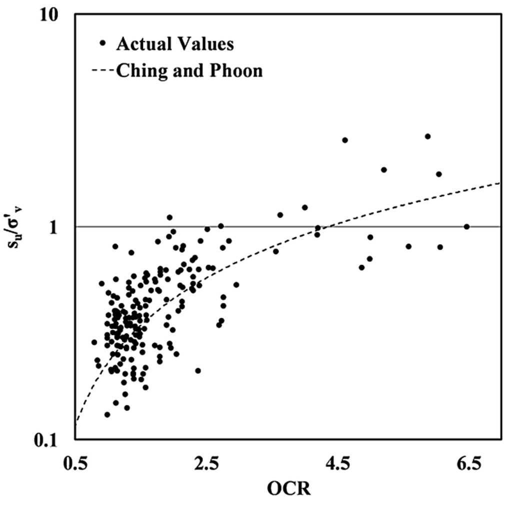
Fig. 12. Summary of the performance of correlation formulas and proposed models on the test set.



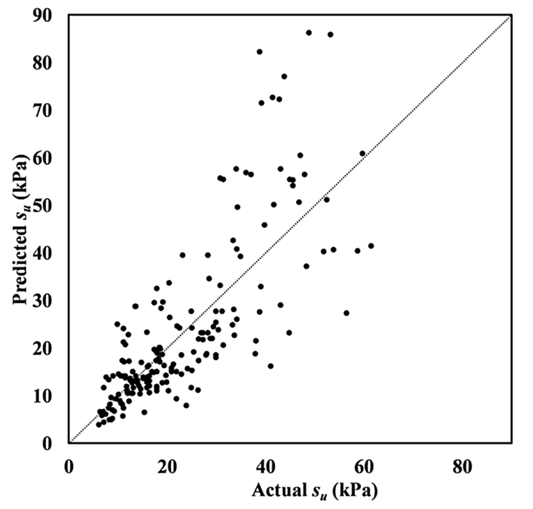
(g)



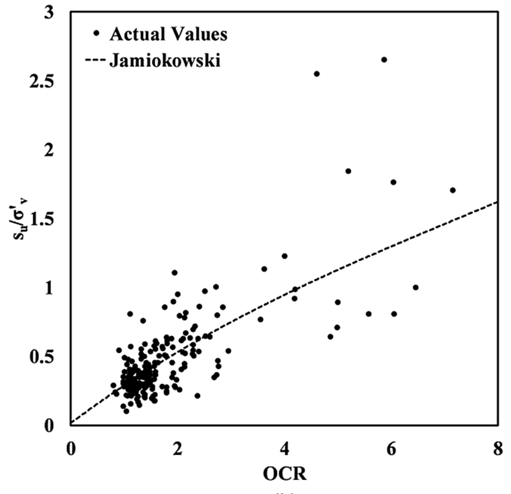
(h)



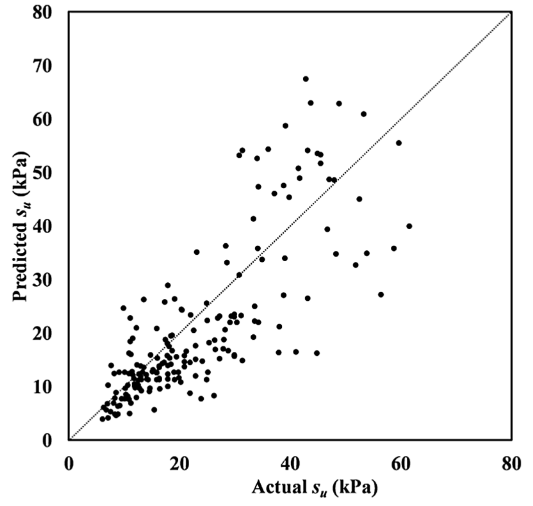
(i)



(j)



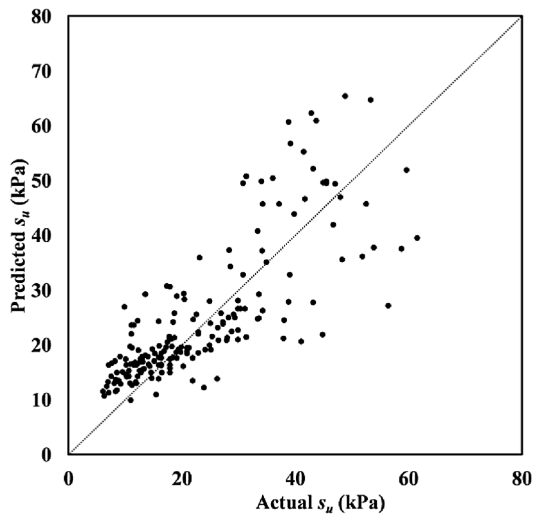
(k)



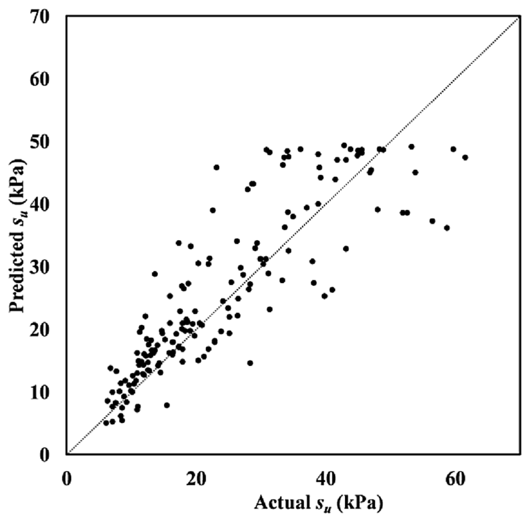
(l)

Fig. 12. (*continued*)

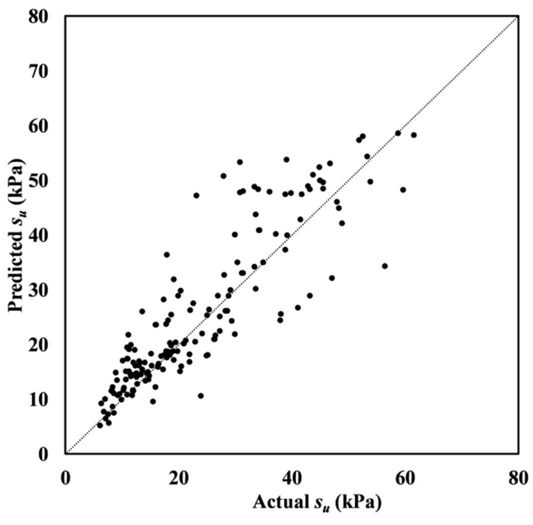
these formulas for other sites depends largely on site-specific features or the level of similarity between the new site and those associated with the establishment of the formulas. Notably, these transformation formulas consider a portion of input variables, e.g. Jamiolkowski [64] used $\sigma_p^{'}$, $\sigma_v^{'}$ for making a model for s_u. Conversely, machine learning/statistical models can utilize all input variables in developing models. This idea, if correctly implemented, would result in more accurate predictions. Table 8 and Fig. 12 show that the proposed DNN model



(m)



(n)



(o)

Fig. 12. (*continued*)

with *L1* regularizer is more reliable in making predictions for the undrained shear strength of clays considering its lower errors. It generally outperforms the conventional correlation models. In addition to its higher level of accuracy, the overfitting problem has not occurred in this model. Since the presented DNN model structure and hyperparameters are determined using three global datasets, they should, on average, also perform better than the correlation formulas for soils at other sites, even when the same model structure and hyperparameters are adopted (i.e. without additional training). In addition, the accuracy of DNN predictions at a site can be further improved if site-specific information is available for additional fine-tuning and training of the model. The various features of DNN, which include hyperparameter tuning, data preprocessing, and feature analysis can provide engineers and practitioners with accurate and reliable results if properly implemented. However, such flexibility would lead to inaccurate results if the features are not being meticulously and carefully applied. With the proposed framework and provided code, engineers and researchers, especially those who are not familiar with programming, could easily and correctly train DNN models. These efforts are made so engineers could obtain better decisions and a deeper understanding of their problems.

6. Conclusion

This study introduced a straight-forward approach toward creating DNN models for engineering datasets and conducted a DNN model for prediction of undrained shear strength in clays. Within this approach, the preliminary setup of the DNN models was discussed, and data preprocessing for engineering datasets in DNN models were pointed out. Next, issues regarding the sufficiency of the collected data for making a prediction model were addressed by conducting a learning curve analysis. After designing the DNN model, solutions for overfitting problem in DNN models were provided. Finally, the validity and performance of the proposed model were assessed as the final step of creating DNN models. The DNN models were shown to be superior to previous correlation models in some aspects which are presented as follows:

1. DNN modeling by the proposed procedures is straight-forward and robust in comparison to finding non-linear regression models which are usually done by trial and error.
2. DNN model can be trained on datasets with any degree of complexity [40].
3. The DNN model is shown to be more accurate than the previously

proposed correlation formulas for the prediction of undrained shear strength.

4. Methods for solving the overfitting problem in DNN models were discussed, while the assessment and solution of this problem in other models (e.g. non-linear regression models) are quite difficult.

The main limitation of this study is the limited attributes of the collected dataset. Based on the results of Section 4.3 (learning curve analysis), collecting more data will not improve the model accuracy. However, more accurate results could possibly be achieved by adding other characteristics of the clays which were not presented. Civil engineering is an area with great opportunity to apply DNN modeling in its different aspects such as correlations between material properties. However, DNN models are not bounded to civil engineering datasets and can be used as a replacement of linear regression or mathematical formulations in future research. DNN models are also used for classification in different areas of engineering where further investigations are required. Furthermore, by collecting more attributes of the clays, it is possible to achieve more accurate results with DNN than those provided in this study.

The flexibility and robustness of the proposed framework in dealing with highly complex datasets provide the opportunity for practitioners of various fields of engineering to analyze and assess their datasets of interests. The proposed framework can be applied to any dataset for making predictions of an output variable based on some input variables, as well as the extraction of knowledge about the importance of those input variables.

Declaration of Competing 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.

Acknowledgements

The authors would like to thank the reviewers whose comments greatly enhanced the quality of this paper. The authors would also like to thank the members of the TC304 Committee on Engineering Practice of Risk Assessment & Management of the International Society of Soil Mechanics and Geotechnical Engineering for developing the database 304 dB used in this study and making it available for scientific inquiry. We also wish to thank M. D'Ignazio, T.T. Länsivaara, J. Ching and K.K. Phoon for contributing this database to the TC304 compendium of databases. Financial support by the Chinese National Engineering Reseasrch Centre for Steel Construction (Hong Kong Branch) (Project No. P0013870) is gratefully acknowledged.

References

[1] M.J. Bailey, R.F. Muth, H.O. Nourse, A regression method for real rstate price index construction, J. Am. Stat. Assoc. 58 (1963) 933–942, https://doi.org/10.1080/01621459.1963.10480679.
[2] M.Y. Cheng, H.C. Tsai, E. Sudjono, Conceptual cost estimates using evolutionary fuzzy hybrid neural network for projects in construction industry, Expert Syst. Appl. 37 (2010) 4224–4231, https://doi.org/10.1016/j.eswa.2009.11.080.
[3] J.H. Chen, M.C. Su, C.Y. Chen, F.H. Hsu, C.C. Wu, Application of neural networks for detecting erroneous tax reports from construction companies, Autom. Constr. 20 (2011) 935–939, https://doi.org/10.1016/j.autcon.2011.03.011.
[4] K.W. Chau, Application of a PSO-based neural network in analysis of outcomes of construction claims, Autom. Constr. 16 (2007) 642–646, https://doi.org/10.1016/j.autcon.2006.11.008.
[5] B. Hola, K. Schabowicz, Estimation of earthworks execution time cost by means of artificial neural networks, Autom. Constr. 19 (2010) 570–579, https://doi.org/10.1016/j.autcon.2010.02.004.
[6] L.D. Graham, D.R. Forbes, S.D. Smith, Modeling the ready mixed concrete delivery system with neural networks, Autom. Constr. 15 (2006) 656–663, https://doi.org/10.1016/j.autcon.2005.08.003.
[7] W. Gao, A. Aslam, F. Li, Effect of equivalence ratio on gas distribution and performance parameters in air-gasification of asphaltene: A model based on Artificial Neural Network (ANN), Pet. Sci. Technol. 37 (2019) 202–207, https://doi.org/10.1080/10916466.2018.1533864.
[8] P.T. Stol, Rainfall interstation correlation functions, an analytic approach, J. Hydrol. 50 (1981) 45–71, https://doi.org/10.1016/0022-1694(81)90061-5.
[9] J. Adamowski, H.F. Chan, A wavelet neural network conjunction model for groundwater level forecasting, J. Hydrol. 407 (2011) 28–40, https://doi.org/10.1016/j.jhydrol.2011.06.013.
[10] Z.P. Bazant, S. Prasannan, Solidification theory for concrete creep. II : verification and application, J. Eng. Mech. 115 (1989) 1704–1725.
[11] L. Bal, F. Buyle-Bodin, Artificial neural network for predicting drying shrinkage of concrete, Constr. Build. Mater. 38 (2013) 248–254, https://doi.org/10.1016/j.conbuildmat.2012.08.043.
[12] M.A. Kewalramani, R. Gupta, Concrete compressive strength prediction using ultrasonic pulse velocity through artificial neural networks, Autom. Constr. 15 (2006) 374–379, https://doi.org/10.1016/j.autcon.2005.07.003.
[13] L. Li, G. Song, J. Ou, DNN based fault tolerant control of nonlinear structural vibration with actuator faults, Adv. Struct. Eng. 14 (2011) 871–879, https://doi.org/10.1260/1369-4332.14.5.871.
[14] S. Hansbo, A new approach to the determinatio n of the shear strength of clay by the fall cone test, R. Swedish Geotech. Inst. Proc. (1957) 5–47.
[15] M. D'Ignazio, K.-K. Phoon, S.A. Tan, T.T. Länsivaara, Correlations for undrained shear strength of Finnish soft clays, Can. Geotech. J. 53 (2016) 1628–1645, https://doi.org/10.1139/cgj-2016-0037.
[16] I. Yilmaz, O. Kaynar, Multiple regression, ANN (RBF, MLP) and ANFIS models for prediction of swell potential of clayey soils, Expert Syst. Appl. 38 (2011) 5958–5966, https://doi.org/10.1016/j.eswa.2010.11.027.
[17] M.A. Shahin, H.R. Maier, M.B. Jaksa, Predicting settlement of shallow foundations using neural networks, J. Geotech. Geoenviron. Eng. 128 (2002) 785–793, https://doi.org/10.1061/(asce)1090-0241(2002) 128:9(785).
[18] E. Momeni, R. Nazir, D.J. Armaghani, H. Maizir, Application of artificial neural network for predicting shaft and tip resistances of concrete piles, Earth Sci. Res. J. 19 (2015) 85–93, https://doi.org/10.15446/esrj.v19n1.38712.
[19] S.K. Das, P. Samui, S.Z. Khan, N. Sivakugan, Machine learning techniques applied to prediction of residual strength of clay, Open Geosci. 3 (2011) 449–461, https://doi.org/10.2478/s13533-011-0043-1.
[20] J.S. Chou, K.H. Yang, J.Y. Lin, Peak shear strength of discrete fiber-reinforced soils computed by machine learning and metaensemble methods, J. Comput. Civ. Eng. 30 (2016), https://doi.org/10.1061/(ASCE)CP.1943-5487.0000595.
[21] J. Pohjankukka, H. Riihimäki, P. Nevalainen, T. Pahikkala, J. Ala-Ilomäki, E. Hyvönen, J. Varjo, J. Heikkonen, Predictability of boreal forest soil bearing capacity by machine learning, J. Terramechanics. 68 (2016) 1–8, https://doi.org/10.1016/j.jterra.2016.09.001.
[22] Binh Thai Pham, Le Hoang Son, Tuan-Anh Hoang, Duc-Manh Nguyen, Dieu Tien Bui, Prediction of shear strength of soft soil using machine learning methods, CATENA 166 (2018) 181–191 https://linkinghub.elsevier.com/retrieve/pii/S034181621830119Xhttps://doi.org/10.1016/j.catena.2018.04.004.
[23] I. Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learn, MIT Press, 2016.
[24] W. Gao, J.L.G. Guirao, B. Basavanagoud, J. Wu, Partial multi-dividing ontology learning algorithm, Inf. Sci. (Ny) 467 (2018) 35–58, https://doi.org/10.1016/j.ins.2018.07.049.
[25] Y. Bengio, Learning deep architectures for AI, FNT Mach. Learn. 2 (1) (2009) 1–127 http://www.nowpublishers.com/article/Details/MAL-006https://doi.org/10.1561/2200000006.
[26] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, ArXiv Prepr. ArXiv1409.1556, 2014. http://arxiv.org/abs/1409.1556.
[27] A. Zhang, K.C.P. Wang, B. Li, E. Yang, X. Dai, Y. Peng, Y. Fei, Y. Liu, J.Q. Li, C. Chen, Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network, Comput. Civ. Infrastruct. Eng. 32 (2017) 805–819, https://doi.org/10.1111/mice.12297.
[28] Z. Zhou, J. Gong, Automated residential building detection from airborne LiDAR data with deep neural networks, Adv. Eng. Inform. 36 (2018) 229–241, https://doi.org/10.1016/j.aei.2018.04.002.
[29] S. Singaravel, J. Suykens, P. Geyer, Deep-learning neural-network architectures and methods: using component- based models in building-design energy prediction, Adv. Eng. Inform. 38 (2018) 81–90, https://doi.org/10.1016/j.aei.2018.06.004.
[30] W. Fang, B. Zhong, N. Zhao, P.E.D. Love, H. Luo, J. Xue, A deep learning-based approach for mitigating falls from height with computer vision: convolutional neural network, Adv. Eng. Inform. 39 (2019) 170–177, https://doi.org/10.1016/j.aei.2018.12.005.
[31] B. Zhong, X. Xing, P. Love, X. Wang, H. Luo, Convolutional neural network: deep learning-based classification of building quality problems, Adv. Eng. Inform. 40 (2019) 46–57, https://doi.org/10.1016/j.aei.2019.02.009.
[32] F. Chollet, KERAS, GitHub Repos, 2015. https://github.com/keras-team/keras.
[33] M. Abadi, A. Agarwal, E.B. Paul Barham, A.D. Zhifeng Chen, Craig Citro, Greg S. Corrado, I.G. Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Y.J. Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, M.S. Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, J.S. Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, P.T. Benoit Steiner, Ilya Sutskever, Kunal Talwar, F.V. Vincent Vanhoucke, Vijay Vasudevan, M.W. Oriol Vinyals, Pete Warden, Martin Wattenberg, Yuan Yu, X. Zheng., TensorFlow: Large-scale machine learning on heterogeneous systems, ArXiv Prepr. ArXiv1603.04467, 2016. https://doi.org/10.1016/0076-6879(83)01039-3.
[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel OLIVIERGRISEL, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, D. Cournapeau, O. Grisel, A. Passos, M. Brucher, M. Perrot and Édouardand, A. Duchesnay, Fré. Duchesnay EDOUARDDUCHESNAY, Scikit-learn: machine learning

in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830. http://scikit-learn.sourceforge.net.

[35] R. Larson, Undrained shear strength in stability calculation of of embankments and foundations on Soft Clays, Candian Geotech. J. 17 (1980) 591–602.

[36] R.J. Chandler, The in-situ measurements of the undrained shear strength of clays using the field vane, Vane Shear Strength Test. Soils F. Lab. Stud. ASTM Int., 1988.

[37] J. Ching, K.-K. Phoon, Correlations among some clay parameters — the multivariate distribution, Can. Geotech. J. 51 (2014) 686–704, https://doi.org/10.1139/cgj-2013-0353.

[38] J. Ching, K.-K. Phoon, Transformations and correlations among some clay parameters — the global database, Can. Geotech. J. 51 (2014) 663–685, https://doi.org/10.1139/cgj-2013-0262.

[39] A. Ng, Machine Learning Yearning, 2016. https://www.mlyearning.org/.

[40] Kurt Hornik, Maxwell Stinchcombe, Halbert White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366, https://doi.org/10.1016/0893-6080(89)90020-8.

[41] V. Nair, G. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proc. 27th Int. Conf. Mach. Learn., 2010. https://doi.org/10.1.1.165.6419.

[42] C. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall, Activation functions: comparison of trends in practice and research for deep learning, ArXiv Prepr. ArXiv1811.03378, 2018. http://arxiv.org/abs/1811.03378.

[43] S. Ruder, An overview of gradient descent optimization algorithms, ArXiv Prepr. ArXiv1609.04747, 2016. http://arxiv.org/abs/1609.04747.

[44] A.-L. Cauchy, Méthode générale pour la résolution des systèmes d'équations simultanées, Comptes Rendus Hebd, Seances Acad. Sci. Paris. 25 (1847).

[45] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. (1951) 400–407.

[46] S. Kullback, R.A. Leibler, Stochastic estimation of the maximum of a regression function, Ann. Math. Stat. 23 (1951) 462–466, https://doi.org/10.1214/aoms/1177729392.

[47] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, Y. Bengio, Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, Adv. Neural Inf. Process. Syst. (2014) 1–14, https://doi.org/10.1016/j.memsci.2004.03.009.

[48] Yurii Nesterov, A method for solving a convex programming problem with rate of convergence rate O(1/k2), Dokl. Akad. Nauk SSSR. 269 (1983) 543–547 https://scholar.google.com/citations?user=DJ8Ep8YAAAAJ&hl=en.

[49] N. Qian, On the momentum term in gradient descent learning algorithms, Neural Netw. 12 (1999) 145–151, https://doi.org/10.1016/S0893-6080(98)00116-6.

[50] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (2011) 2121–2159, https://doi.org/10.1109/CDC.2012.6426698.

[51] M.D. Zeiler, ADADELTA: An adaptive learning rate method, ArXiv Prepr. ArXiv1212.5701, 2012. http://arxiv.org/abs/1212.5701.

[52] G. Hinton, RMSProp, Lecture 6e Coursera Class, 2013. http://www.cs.toronto.edu/%7B~%7Dtijmen/csc321/slides/lecture_slides_lec6.pdf.

[53] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, ArXiv Prepr. ArXiv1412.6980, 2014. http://arxiv.org/abs/1412.6980.

[54] T. Dozat, Incorporating nesterov momentum into Adam, ICLR Work, 2016.

[55] A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, Y. LeCun, The loss surfaces of multilayer networks, Artif. Intell. Stat. (2015) 192–204 http://arxiv.org/abs/1412.0233.

[56] N.S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P.T.P. Tang, On large-batch training for deep learning: Generalization gap and sharp minima, ArXiv Prepr. ArXiv1609.04836, 2016, pp. 1–16.

[57] J. Brownlee, Better Deep Learning Train Faster, Reduce Overfitting, and Make Better Predictions, 2018. https://machinelearningmastery.com/better-deep-learning/.

[58] M. Hubert, P.J. Rousseeuw, S. Van Aelst, Multivariate outlier detection and robustness, Handb. Stat. 24 (2005) 263–302, https://doi.org/10.1016/S0169-7161(04)24010-X.

[59] R.G. Garrett, The chi-square plot: a tool for multivariate outlier recognition, J. Geochemical Explor. 32 (1989) 319–341, https://doi.org/10.1016/0375-6742(89)90071-X.

[60] A. Ng, Feature selection, L 1 vs. L 2 regularization, and rotational invariance, in: Proc. Twenty-First Int. Conf. Mach. Learn. - ACM, 2004, pp. 78. https://doi.org/10.1145/1015330.1015435.

[61] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958, https://doi.org/10.1109/ICAEES.2016.7888100.

[62] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?": Explaining the predictions of any classifier, in: Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., ACM, 2016, pp. 1135–1144. http://arxiv.org/abs/1602.04938.

[63] S.M. Lundberg, Lee Su-In, A unifed approach to interpreting model predictions, Adv. Neural Inf. Process. Syst. (2017) 4765–4774 http://ovidsp.ovid.com/ovidweb.cgi?T=JS&PAGE=reference&D=ovftm&NEWS=N&AN=01300529-201201150-00010.

[64] M. Jamiolkowski, New developments in field and laboratory testing of soils, State of the Art Report, in: 11th Int. Conf. SMFE, 1985, pp. 1985.