Machine learning algorithms for applications in geotechnical engineering

Pouyan Pirnia, François Duhaime & Javad Manashti
Department of Construction Engineering – École de technologie supérieure, Montréal, Québec, Canada

ABSTRACT
Artificial neural networks (ANNs) applications are increasingly common in all fields of engineering. One of the main obstacles to the development of ANN applications in geotechnical engineering is the need for large datasets. This paper presents two application examples in which numerical methods were used to generate large datasets. The first example involves the determination of grain size distributions from soil pictures based on a dataset that includes 53130 synthetic soil images corresponding to different particle size distributions. The pictures were generated using YADE, a discrete element code. The second application example involves the use of the finite element method to generate a dataset. COMSOL's MATLAB programming interface was used to generate a large number of finite element simulations to predict the water level in the reservoir of a hypothetical dam based on pore pressure measurements obtained using an array of 15 piezometers.

RÉSUMÉ
L'utilisation de réseaux de neurones artificiels (RNA) est de plus en plus commune dans tous les domaines du génie. Un des principaux obstacles qui freine l'utilisation de RNA en géotechnique est la disponibilité de grands ensembles de données. Cet article présente deux exemples d'applications pour lesquels des méthodes numériques ont été utilisées pour produire les ensembles de données. Le premier exemple concerne la détermination de distributions granulométriques à partir d'une banque de 55 000 images synthétiques de sol qui correspondent à différentes distributions granulométriques. Les images ont été générées en utilisant YADE, un code d'éléments discrets. Le deuxième exemple d'application implique l'utilisation de la méthode des éléments finis pour générer un ensemble de données. L'interface de programmation MATLAB de COMSOL a été utilisée pour générer un grand nombre de simulations par éléments finis pour prédire le niveau d'eau dans le réservoir d'un barrage hypothétique en se basant sur des mesures de pression interstitielle obtenues en utilisant un réseau de 15 piézomètres.

1 INTRODUCTION

Machine learning (ML) is an empirical approach where a computer program learns from a dataset without the need to code the problem and a procedure to solve it. Artificial Neural Networks (ANN) are one example of machine learning models. The number of ML applications is growing rapidly in all fields of engineering.

Several ML applications can already be found in the geotechnical engineering literature. For example, Mustafa et al. (2013) evaluated the performance of four ANN training algorithms for modelling the dynamics of soil pore water pressure (PWP) in response to rainfall variations using multilayer perceptron (MLP) ANN. The results showed that the network performance (training time and prediction accuracy) is related to the type of training algorithm.

Taormina et al. (2012) assessed the ability of a feed forward neural network (FFNN) to predict hourly groundwater levels in a coastal unconfined aquifer. The network was first trained with observed groundwater elevations and external inputs such as rainfall and evapotranspiration in case of one-hour ahead predictions. Then, to simulate longer periods, the FFNN was fed back its own output instead of past observed water level data. The algorithm could accurately reproduce water level variations for several months. It was suggested as a reliable tool for modelling aquifer responses or reconstructing missing data.

Gordan et al. (2015) used artificial neural network (ANN) and particle swarm optimization (PSO)–ANN models to predict the factor of safety (FOS) of homogenous slopes during earthquakes. A dataset including 699 FOS values for different slope geometries, soil properties and peak ground accelerations was obtained using the Geostudio software package. The PSO–ANN technique showed a higher accuracy (R^2 = 0.97) compared to the ANN technique (R^2 = 0.91).

Obtaining large datasets is often the main difficulty in developing ML applications. Numerical methods can help to generate these datasets. The Finite Element Method (FEM) has been used for more than 50 years to study seepage, stresses and deformations in embankment dams and other earth structures (Day et al., 1998; Hnang, 1996; Ng & Small, 1999; Sharif et al., 2001; Sherard, 1992; Zhang & Du, 1997). On the other hand, the Discrete Element Method (DEM) is used to simulate the discrete nature of soil and its microscale behaviour. The DEM considers each particle explicitly in a granular material. Hence it can simulate finite displacements and rotations of particles (Cundall & Hart, 1993). Both FEM and DEM can be used to create large databases for the training of ML algorithms.

This paper presents two application examples in which FEM and DEM were used to generate large datasets to train ANNs.

The first example involves the determination of grain size distributions from soil pictures. The most common

geotechnical tests for granular soils, like sand and gravel, concern the determination of the particle size distribution (PSD). Sedimentation and sieving, the most common methods, date from the first half of the 20th century. These methods are time-consuming and novel techniques, like laser diffraction and image processing methods, are faster. Commercial laboratories would benefit from more rapid and precise methods to determine the grain size distribution. In the first example, the applicability of Convolutional Neural Networks (ConvNet) (LeCun et al., 2015; Krizhevsky et al., 2012) to evaluate the size of particles from soil photographs was verified. A dataset composed of synthetic soil pictures was generated using YADE (Šmilauer et al. 2010), a discrete element code.

The second application example involves the use of FEM to generate a dataset for the training of an ANN. COMSOL's MATLAB programming interface (COMSOL, 2013) was used to generate a large number of FEM simulations in which pore pressure time series were modelled for a series of virtual piezometers installed in the core of a dam. The upstream boundary condition was based on different water level time series for each simulation. An ANN was trained to predict the upstream water level during previous weeks based on the current piezometer measurements. The objective of this application example was to evaluate the capability of ML algorithms for data recovery or to generate alarm levels for instruments that take into consideration external factors or measurements from other instruments.

2 METHODOLOGY

2.1 Grain size distribution example

Convolutional Neural Networks require thousands of images for model training and parameter tuning. The discrete element code YADE was used to generate particles in five size categories corresponding to sieves with openings of 75, 106, 150, 250, 425 and 710 µm. A MATLAB script generated a list of 53130 different PSD based on the five size categories. A java interface read the data and sent the different PSD to the YADE python interface. A loose cloud of particles in a box was then generated for each PSD (Pirnia et al., 2016, 2017). After the particles settled at the bottom of the box, two pictures were taken from top and bottom viewpoints (Figure 1).

The particles were shaded using random grey levels for each image. The dataset involved 53130 pairs of top and bottom images with a resolution of 128x128 pixels. Image pairs were merged together in 256x128 pixels images as the input of the model (Figure 1)..

The Microsoft Cognitive Toolkit (CNTK) (Yu et al., 2014) version 2.2 was used to analyze the image pairs and to determine the percentage passing for five sieves (106, 150, 250, 425 and 710 µm).

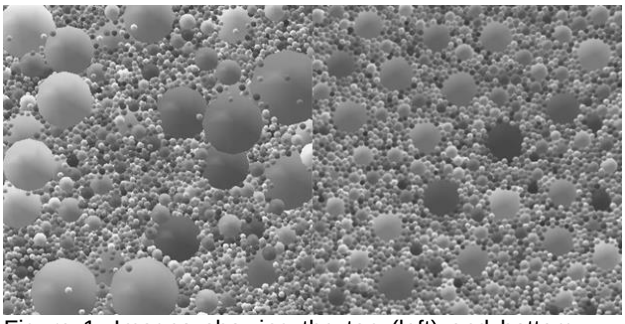

Figure 1. Images showing the top (left) and bottom (right) views of the synthetic soil particles in the virtual transparent box

The architecture of the convolutional neural network that provided the best results is displayed in Figure 2. Pooling layers were inserted after two convolutional layers to down-sample the spatial dimension of the input image (Krizhevsky et al., 2012). The Max pooling layer uses 5x5 receptive fields with a stride of 2 along the spatial dimension (width and height). Rectified Linear Units (ReLU) Layers was applied after each Convolutional layer. This layer intensifies the model nonlinearity. Rectified linear translates the negative value input to zero and when the input is above a certain quantity thresholds it at zero. The same combination of Conv-Pooling layers was repeated again to reduce the image size.

Three fully connected (Dens) layers followed by ReLU layer and Dropout. Fully connected layers have connection to all activation of the last layer. The Drop out layers avoid overfitting to the training data and improves the generalization. The last layer in the model is linear layer holding the output.

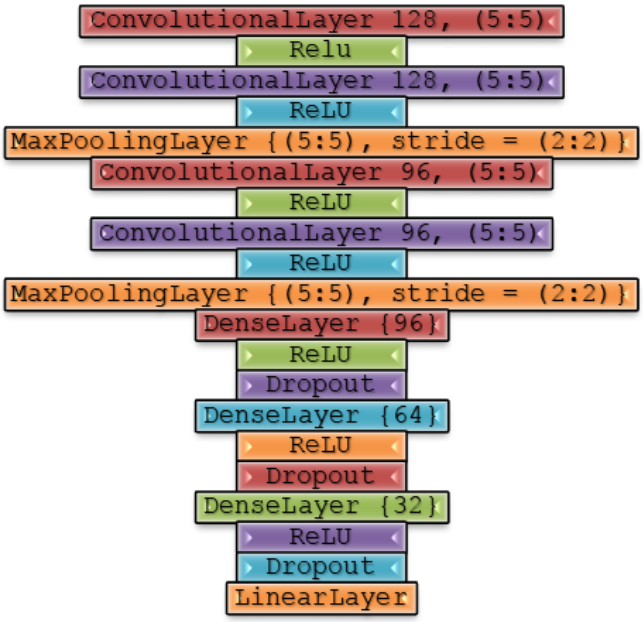

Figure 2. ConvNet model structure

2.2 Pore pressure monitoring example

A 2-D COMSOL model of transient and unsaturated seepage was built to simulate the pore water pressure at 15 points (virtual piezometers) aligned on three vertical lines in the core of a hypothetical dam in response to water level changes in the upstream reservoir (Figure 3). The pore water pressure in the dam was directly influenced by the water level time series during the weeks that preceded the pore pressure measurements.

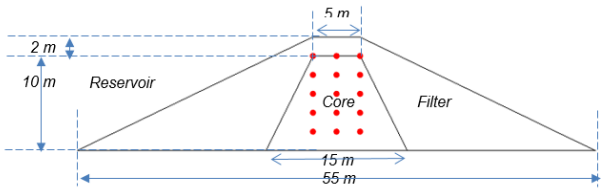

Figure 3. Geometry of the COMSOL model. The red points display the piezometer positions

The simulations were conducted with different time series of water level in the upstream reservoir for the five weeks that preceded the pore pressure measurements. These time series were constructed from five water levels at one week interval from t = 0 to t = 4 weeks. The water level was kept constant during the fifth week. Six discrete water level values were chosen for the first four water levels between t = 0 to t = 3 weeks (from 6 to 8.5 m in 0.5 m increments). Eleven discrete water levels were used for the last week (from 6 to 8.5 m in 0.25 m increments). Together, these discrete water levels can be combined to form a database of 14 256 water level time series. A MATLAB script was written to produce all possible water level conditions. The water level conditions were combined in COMSOL using a linear interpolation function.

The model solved the water conservation equation under unsaturated conditions (Richard's Equation). The unsaturated soil properties of both the filter and core materials (Table 1) were based on the van Genuchten (1980) model and the parameter values used by Dumberry et al. (2015).

The MATLAB LiveLink interface was used to access and modify the COMSOL model created through the graphical user interface. Through the MATLAB LiveLink interface, command lines were sent to COMSOL to change the initial conditions (water levels within five weeks) for each test, and to run the simulation. The results were written in an Excel file. The total simulation time for the COMSOL model was five weeks. The water level condition from five weeks ago was set as the initial condition using a steady state simulation. The outputs of the COMSOL model were the PWP for each piezometer at the end of the simulation.

The input dataset for the neural networks consisted of the PWP for the 15 points in the core of the dam at the end of the COMSOL simulation (Figure 3). Three algorithms were used to train the neural networks and to predict the water level in the reservoir for the 5 weeks that preceded the pore pressure measurements: Feedforward neural network, LSTM unit and GRU.

Table 1. Seepage properties used in the COMSOL simulations

Material	Air entry suction parameter α (m^{-1})	Pore-size distribution parameter n	Residual water content θ_r	Saturated water content θ_s	Permeability k (m/s)
Core	1.27	1.12	0.05	0.15	4.5E-6
Filter	2.64	2.22	0.01	0.40	1E-4

In general, training, validation and testing are the three stages of preparing a machine learning algorithm. The training and validation process can often be combined in a single training session, especially in the case of small datasets. Both parameter estimation and model structure selection happen during training. The Mean Square Error (MSE) was used to evaluate the performance of the training algorithms. For testing, a set of data was used to assess the predictive performance of the model.

The original data containing 14256 simulations was split into training, validation and testing sets (Figure 5). Most of the dataset (70 percent chosen randomly) was devoted for training and validation of the network. The rest was used for testing. Twenty percent of the training dataset was held aside as validation data. The testing dataset contains data which are not introduced to the network during training. The testing dataset shows how the model can generalize.

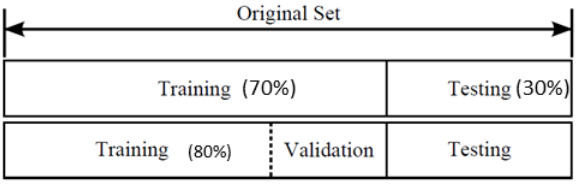

Figure 4. Training, validation and testing split

The activation function chosen for input and hidden layers was an Hyperbolic tangent sigmoid (tanH) with range (-1,1) because the PWP values ranged from positive to negative. The input data were normalized between (-1,1) to match the activation function (tanH). Moreover, normalization of data makes the training faster and minimizes the error (Rojas 1996). A Rectified Linear Unit (ReLU) layer was set as the output layer to extrapolate in the desired range of values (6 to 8.5 m).

Hyperparameters are variables that determine the model structure and training algorithm (e.g., learning rate). These values are set for the network before the training process begins. An optimization of hyperparameters is needed for the three proposed algorithms to obtain the best results. This optimization involves changing the number of

neurons in input and hidden layers, the number of hidden layers, the learning rate, and the batch size. Comparison of the best results for each algorithm and hyperparameter combination determines the appropriate algorithm for this problem. Keras (Chollet, 2015), an open source high-level neural network API written in python, was employed for modelling the three algorithms. Keras was used with the TensorFlow (Abadi et al., 2016) backend.

3 RESULTS AND DISCUSSIONS

3.1 First application example

The goodness of fit of the estimated particle size distributions to the actual results (sieve sizing) was evaluated based on the smallest Root mean squared error (RMSE) and the highest value of coefficient of determination (R-squared), which defines the fit integrity of the experimental data.

The Mean RMSE of all sieves is 6.94 % passing and R-Squared is 0.95. As shown in Figure 5, the bigger particles hide behind the smaller one and some of the big particles can only be seen in the top view (left part of Fig. 1). It is hypothesized that this issue increases the error for predictions of percent passing for bigger particles rather than smaller ones. Consequently, the smallest sieve has the lowest error (RMSE=4.2). Moreover, the RMSE gradually increases to 9.1 % passing for largest mesh sizes.

To be applicable to real soil, the determination of PSD from ConvNet needs to be based on real or at least more realistic soil images. Another dataset is under developement using the Unity game engine to give more realistic features (size, shape and texture) to the particles. This method could eventually provide rapid, precise and economic online PSD in the laboratory or in the field.

3.2 Second application example

The main parameters for the trained algorithms are presented in Table 2. The training was stopped when the MSE of the testing (prediction) results for the five weeks reached a minimum value. The training and validation errors showed a relatively high error due to errors from the first three weeks.

As can be seen in Figure 6, the GRU algorithm could converge fast and passed the underfitting region within 500 epochs. It could minimize the validation error in 5000 epochs (Table 2). The GRU showed the lowest error during training compared to the three other algorithms. The MSE reached 0.5 (m^2) after only 50 epochs. The Dense algorithm (FFNN) required more iteration (2500 epochs) to minimize both MSE to less than 0.45 (m^2). It shows more training error but less oscillation compared to the LSTM and GRU algorithms, which show more oscillations.

The prediction errors for the different algorithms presented in table 3 suggest that all three algorithms could be used to predict the water level in the reservoir during last week with high accuracy. The R-Squared for three algorithms is equal to 1 (Figure 7). The algorithms were also able to predict the water level two weeks before the PWP measurements with a reasonable accuracy (Figure 8).

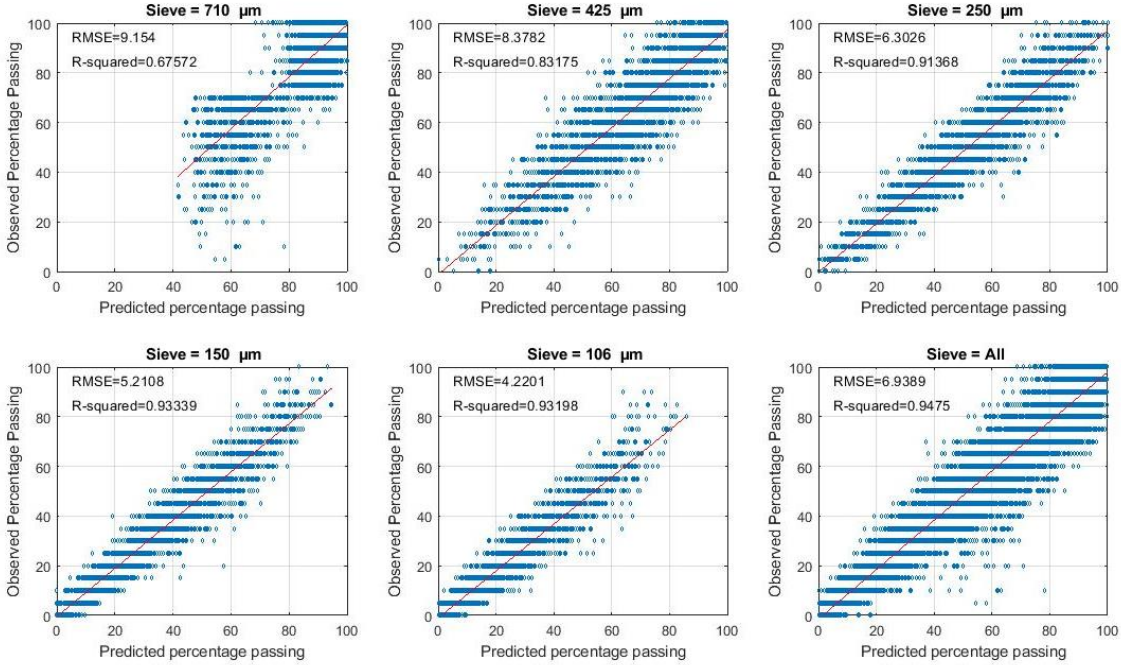

Figure 5. Comparison between real and predicted percentages passing for each sieve

Table 2. Hyperparameters studied for the optimization process.

Algorithm	number of neurons		Learning rate	Batch-size	Epoch	Training Error MSE (m2)	Validation Error MSE (m2)
	input layer	1st hidden layer					
LSTM	50	-	0.0001	600	5000	0.3822	0.378
GRU	50	-	0.001	600	5000	0.3869	0.3896
Dense	50	15	0.1	600	5000	0.3935	0.3878

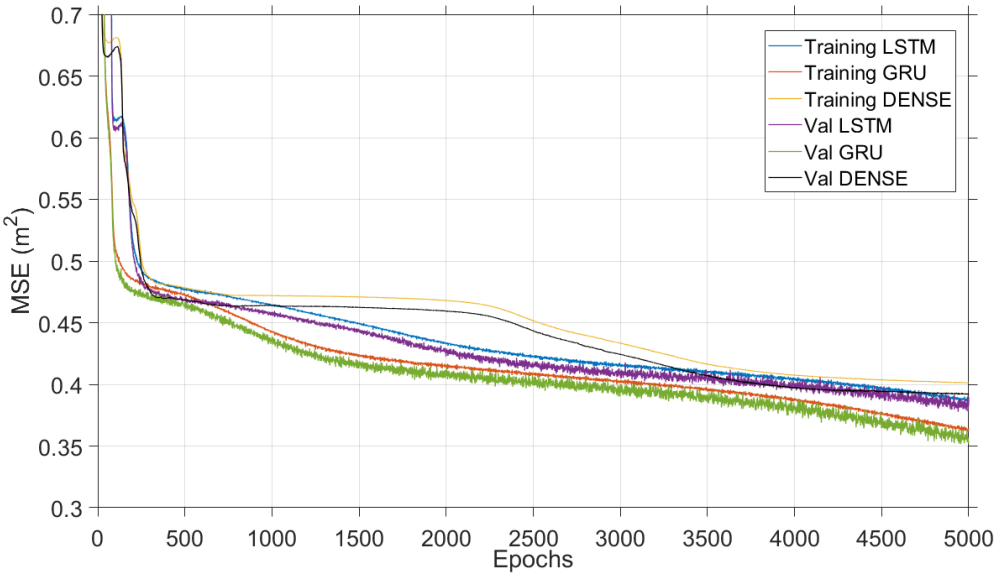

Figure 6. Training and validation curves per epoch of three algorithms

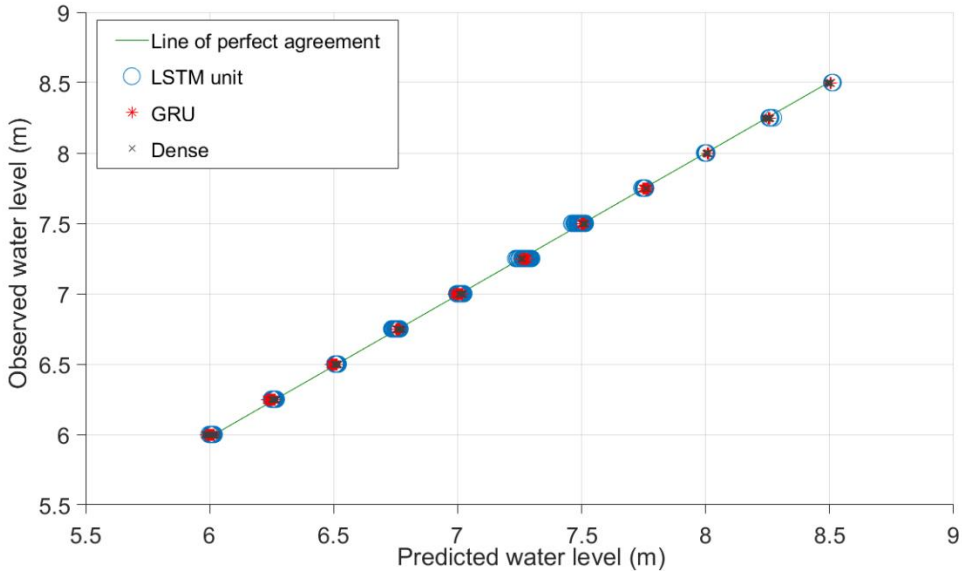

Figure 7. Comparison between predicted and observed water levels at the time of PWP measurements using different algorithms

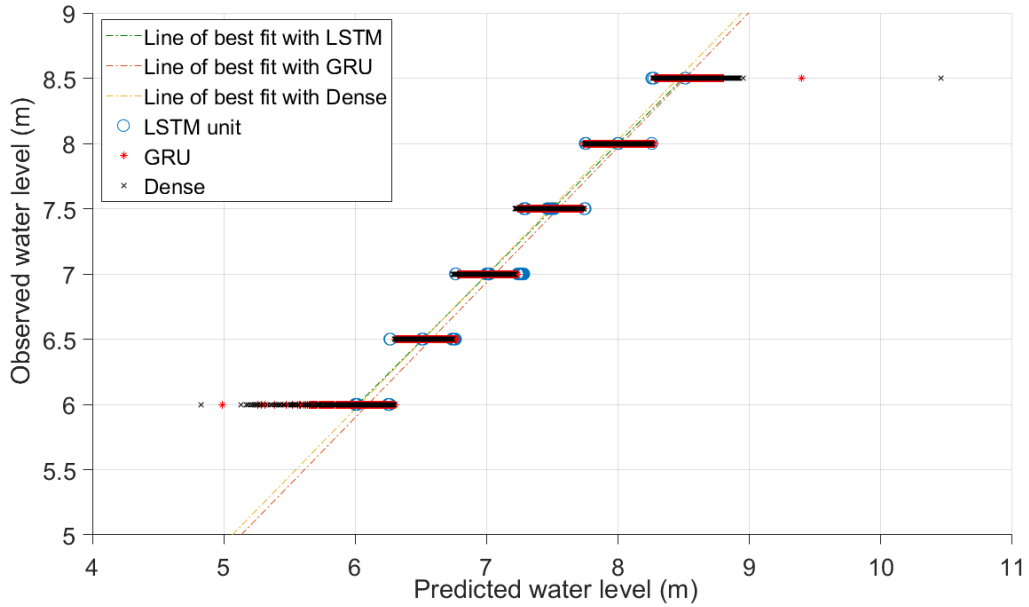

Figure 8. Comparison between predicted and observed water levels two weeks before the PPW measurements using different algorithms

Table 3. Performance statistics of different algorithms in testing

Algorithm		Prediction				
	Error	1 week ago	2 weeks ago	3 weeks ago	4 weeks ago	5 weeks ago
LSTM	RMSE (m)	0.0138	0.1878	1.1	0.9362	0.8352
	R-Squared	1	0.9531	0.2287	0.0285	0.0387
	MSE (m^2)	0.0002	0.0352	1.2099	0.8765	0.6975
GRU	RMSE (m)	0.0108	0.2170	0.9217	0.8863	0.9384
	R-Squared	1	0.9369	0.2423	0.0407	0.0663
	MSE (m^2)	7.66E-05	0.0414	1.2047	0.8328	0.6736
Dense	RMSE (m)	0.0147	0.2035	1.0159	0.9221	0.8814
	R-Squared	1	0.9436	0.1982	0.0236	0.0072
	MSE (m^2)	7.11E-05	0.0672	0.9454	0.8581	0.7783

The simulation results in Table 3 indicate that from PWP data inside the core of dams, the water level in the upstream reservoir could be estimated for the previous two weeks. An interesting observation that can be made from this result is that once the water level changes in the reservoir, it takes approximately two weeks for PWP inside the core to reach equilibrium with the new boundary condition for the hypothetical dam that was modelled.

Several applications could be developed based on this type of ANN. It could for instance be used to set “intelligent” alarm levels for geotechnical instruments based on external factors such as temperatures, upstream water levels or readings by other instruments. It could also be used to detect impaired instruments inside a real dam. For each instrument, an ANN could be trained to predict a reading based on previous reservoir level measurements (or some other external parameter), and on measurements from other instruments. The neural network would be able to identify measurements that are anomalous.

4 CONCLUSION

This study was centred on generating large datasets using numerical models for machine learning applications in geotechnical engineering. For the first application, YADE, a DEM code, was used to produce assemblages of spherical particles. A dataset involving 53130 synthetic soil pictures corresponding to different particle size distributions was generated. The Microsoft Cognitive Toolkit (CNTK) was used to analyze the images and the percentages passing for five sieves. The results show that Convolutional Neural Networks (ConvNet) can predict the PSD with a Root Mean Square Error (RMSE) of around 4 %.

For the second application, a dataset involving 14256 simulations was generated with COMSOL, an FEM engine. The model evaluated the PWP inside the core of an embankment dam for different water level time series in the reservoir during the five previous weeks. Three different ML algorithms could accurately predict the upstream water levels for the two weeks that preceded the PWP measurements in the core of the dam.

ACKNOWLEDGMENTS

The authors would like to acknowledge the funding of NSERC for this project.

5 REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., . . . Isard, M. 2016. TensorFlow: A System for Large-Scale Machine Learning. *Paper presented at the OSDI*.

Chollet, F. 2015. *Keras*. Retrieved from https://keras.io/

COMSOL. 2013. *COMSOL MULTIPHYSICS*. Version 5.3. Retrieved from http://www.comsol.com/

Dumberry, K., Duhaime, F., and Éthier, Y. A. 2005. Experimental study of contact erosion during core overtopping. *Paper presented at the annual conference of the Canadian Dam Association (CDA)*, Mississauga, ON, Canada.

Day, R., Hight, D., and Potts, D. 1998. Finite element analysis of construction stability of Thika Dam. *Computers and Geotechnics*, 23(4), 205-219.

Gordan, B., Armaghani, D. J., Hajihassani, M., and Monjezi, M. 2016. Prediction of seismic slope stability through combination of particle swarm optimization and neural network. *Engineering with Computers*, 32(1), 85-97.

Hnang, T.-K. 1996. Stability analysis of an earth dam under steady state seepage. *Computers & structures*, 58(6), 1075-1082.

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing systems*, 1097-1105.

LeCun, Y., Bengio, Y., and Hinton, G. 2015. Deep learning. *Nature*, 521(7553), 436.

Mustafa, M., Rezaur, R., Saiedi, S., Rahardjo, H., and Isa, M. 2012. Evaluation of MLP-ANN training algorithms for modeling soil pore-water pressure responses to rainfall. *Journal of Hydrologic Engineering*, 18(1), 50-57.

Ng, A. K., and Small, J. C. 1999. A case study of hydraulic fracturing using finite element methods. *Canadian Geotechnical Journal*, 36(5), 861-875.

Pirnia, P., Duhaime, F., Ethier, Y., and Dubé, J.-S. 2016. Development of a multiscale numerical modelling tool for granular materials. *Paper presented at the 69th Canadian Geotechnical Conference*, Vancouver, BC, Canada.

Pirnia, P., Duhaime, F., Ethier, Y., and Dubé, J-S. 2017. Multiscale numerical modelling of internal erosion with discrete and finite elements, *Paper presented at the 70th Canadian Geotechnical Conference*, Ottawa, ON, Canada.

Rojas, R. 1996. Neural networks: a systematic introduction. *Springer Science & Business Media.*

Sharif, N. H., Wiberg, N.-E., and Levenstam, M. 2001. Free surface flow through rock-fill dams analyzed by FEM with level set approach. *Computational mechanics*, 27(3), 233-243.

Sherard, J. L. 1986. Hydraulic fracturing in embankment dams. *Journal of Geotechnical Engineering,* 112(10), 905-927.

Šmilauer, V., Catalano, E., Chareyre, B., Dorofeenko, S., Duriez, J., Gladky, A., . . . Sibille, L. 2017. Yade reference documentation. Yade Documentation, 474 pp..

Taormina, R., Chau, K.-W., and Sethi, R. 2012. Artificial neural network simulation of hourly groundwater levels in a coastal aquifer system of the Venice lagoon. *Engineering Applications of Artificial Intelligence*, 25(8), 1670-1676.

Van Genuchten, M. T. 1980. A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil science society of America journal*, 44(5), 892-898.

Yu, D., Eversole, A., Seltzer, M., Yao, K., Huang, Z., Guenter, B., . . . Wang, H. 2014. An introduction to computational networks and the computational network toolkit. *Microsoft Technical Report MSR-TR-2014–112*.

Zhang, L., and Du, J. 1997. Effects of abutment slopes on the performance of high rockfill dams. *Canadian Geotechnical Journal*, 34(4), 489-497.