Modelling stochastic behaviour in simulation digital twins through neural nets

Sean Reed , Magnus Löfstrand & John Andrews

JOURNAL OF SIMULATION
https://doi.org/10.1080/17477778.2021.1874844

OPEN ACCESS

Modelling stochastic behaviour in simulation digital twins through neural nets

Sean Reed[a], Magnus Löfstrand[a] and John Andrews[b]

[a]School of Science and Technology, Örebro University, Örebro, Sweden; [b]Department of Mechanical, Materials and Manufacturing Engineering, University Park, University of Nottingham, Nottingham, UK

ABSTRACT
In discrete event simulation (DES) models, stochastic behaviour is modelled by sampling random variates from probability distributions to determine event outcomes. However, the distribution of outcomes for an event from a real system is often dynamic and dependent on the current system state. This paper proposes the use of artificial neural networks (ANN) in DES models to determine the current distribution of each event outcome, conditional on the current model state or input data, from which random variates can then be sampled. This enables more realistic and accurate modelling of stochastic behaviour. An application is in digital twin models that aim to closely mimic a real system by learning from its past behaviour and utilising current data to predict its future. The benefits of the approach introduced in this paper are demonstrated through a realistic DES model of load-haul-dump vehicle operations in a production area of a sublevel caving mine.

ARTICLE HISTORY
Received 16 March 2020
Accepted 6 January 2021

KEYWORDS
Discrete event simulation; mixture density network; digital twin; artificial neural network; industry 4.0

1. Introduction

Discrete event simulation (DES) is a popular modelling approach due to its ability to represent complex, dynamic systems with stochastic behaviour whilst requiring fewer simplifying assumptions compared to analytical models (Banks, 1998). Stochastic behaviour is incorporated in DES by using random variables to represent event outcomes, with outcome probabilities represented by a probability distribution function. Random variables in a DES may relate to the time until an event occurs (e.g., the next customer arrives) or the change in the model state when an event occurs (e.g., number of items the customer orders). Usually, the distribution associated with a random variable in a DES model is set at the model design stage, remaining fixed throughout a simulation. For example, the time to failure for successive machine failures may be represented by a Weibull distribution with specified shape and scale parameters. In many real systems and processes, however, the distribution of an event outcome is statistically dependent on the values of certain variables. For example, the rate of arrival of customers may change with the time of day or the rate of failure for a manufacturing machine may change with the production rate demand. In these cases, the realism of the model may be improved if, instead of modelling average behaviour throughout a simulated time period, the distributions are adapted according to the current model state. Two approaches to representing the influence of covariates on time to event distributions are the proportional hazards and accelerated life models (Leemis et al., 1990). In these models, the covariates are modelled as having a multiplicative effect on the hazard rate and event time, respectively. Leemis et al. (1990) showed how random variates can be generated in a simulation for accelerated life and proportional hazard models assuming a baseline parametric distribution function (Bender et al., 2005). Harden and Kropko (2019) introduced a method for generating random variates from a proportional hazard model without assuming a particular form for the baseline distribution function. The distributions used in a model are usually chosen through statistical analysis of relevant data (Boos & Stefanski, 2013) or estimation by experts (O'Hagan et al., 2006). However, the true relationship between the covariates and the distribution of the output variable may be extremely complex, interacting and non-linear. Furthermore, the number of covariates influencing the outcome distribution for an event may be large. Therefore, finding an accurate relationship between covariate values and the outcome distribution can be difficult using traditional statistical techniques.

Recent research has investigated the use of artificial neural networks (ANN) to predict outcomes or decisions within a DES model. ANN represents a machine learning approach that "learns" to predict an output for a given input by considering examples without explicit programming of rules. Bergmann et al. (2014) trained an ANN to make job scheduling decisions in a DES model of a manufacturing system. Chang and Chang (2018) utilised them to predict the treatment duration for patients within a DES model of a dental clinic based on their age and oral care habits

CONTACT Magnus Löfstrand magnus.lofstrand@oru.se School of Science and Technology, Örebro University, Örebro SE-701 82, Sweden

amongst other variables. De la Fuente et al. (2018) used ANNs in a DES model of a banking process to determine whether customer loan applications were accepted or rejected based on 20 variables such as the employment status of the customer. The existing literature on integrations of ANNs within DES, such as described above, have exclusively described its use to model deterministic behaviour, where the outcomes depend only on the inputs. This is desirable when the behaviour that the model is seeking to replicate is itself deterministic, such as the job scheduling in Bergmann et al. (2014). However in other cases, such as the patient treatment duration in a dental clinic from Chang and Chang (2018), the real behaviour features uncertainty or randomness that would be beneficial to include in the model. Studying the stochastic behaviour of complex systems or processes to gain insight into the likelihood of different outcomes is often a key reason for choosing to develop a DES model over alternative modelling approaches (Ross, 2012).

The prediction of conditional distributions with ANN has also been described in the literature through three main approaches: mixture density networks (MDNs) (Bishop, 1994), quantised Softmax networks (Van Den Oord et al., 2016) and kernel mixture networks (Ambrogioni et al., 2017). Compared to traditional methods that have been used with DES previously, such as the proportional hazards model (Leemis et al., 1990), the advantages include the ability to:

(1) Learn highly complex, non-linear relations between the distribution of a random variable and the values of covariates.
(2) Output arbitrary distributions that are not constrained to a particular form.
(3) Train efficiently on large data sets and update when new training data become available.

The main contribution of this paper is to introduce and demonstrate an approach to using these special types of ANN, such as MDN, for predicting the conditional distribution of outcomes of events within DES models. The advantage over the use of conventional regression ANNs within DES, that merely predict the most likely value, is that these conditional distributions can then be sampled from, embedding realistic modelling of stochastic behaviour and uncertainty. Since obtaining predictions from a trained ANN has very low computational expense, these advantages can be obtained without compromising the efficiency of a DES model. The paper also contributes towards the creation of realistic digital twin DES models that represent a real physical counterpart. This is achieved by training the ANN embedded in the DES on data from the real system received via sensors and IT systems, resulting in it mimicking the stochastic behaviour and adapting over time through continuous learning. Digital twin models (Grieves, 2014; Haag & Anderl, 2018; Negri et al., 2017; Tao et al., 2018) have a key role in the fourth industrial revolution in the manufacturing industry, known as Industry 4.0 (Lasi et al., 2014), that encompasses the trends of connectivity, intelligence and flexible automation.

The remainder of the paper is organised as follows: Section 2 introduces ANNs for conditional distribution prediction, Section 3 describes how to integrate them within DES models, Section 4 demonstrates the benefits of the approach in a DES model of operations within a sublevel caving mine, and Section 5 provides some conclusions and areas for future work.

2. ANNs for prediction of conditional distributions of random variables

An ANN (Aggarwal, 2018) is a type of computing system that seeks to learn the transformation from a set of real-valued input variables to a set of real-valued set of output variables by learning the association from example training data. They comprise of a collection of connected nodes, where each node receives a set of real values as input and performs a transformation on those inputs to produce a real-valued output. In a feedforward design (Francois Chollet, 2017), nodes are arranged in layers with each node receiving input from nodes in the previous layer and feeding its output to nodes in the subsequent layer. The architecture of an ANN, see Figure 1, consists of three types of layer:

- The first layer, known as the input layer, represents the input vector of received information (sometimes called features).
- Intermediate layers, known as hidden layers, process information received from the previous layer into output information.
- The last layer, known as the output layer, represents the output vector calculated from information received from the previous layer.

In addition to feedforward networks, other architectures exist, such as recurrent neural networks (Francois Chollet, 2017) where connections between nodes form a directed graph along a temporal sequence. When each node in a layer is connected to all the nodes in the previous layer, the ANN is known as fully connected or dense. Each input connection to a node is assigned a weight and the weighted sum of its inputs plus a possible bias term is computed. The computation performed by an ANN node is represented in Figure 2.

A non-linear activation function is then applied to the summation value to produce the output value, the purpose of which is to enable the ANN to represent complex non-linear functions. The rectified linear unit

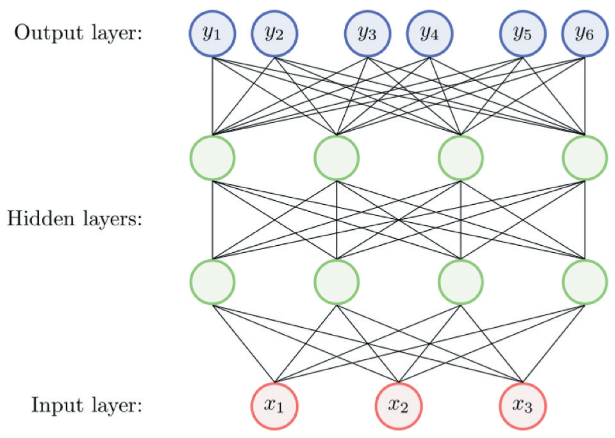

Figure 1. A dense ANN with three input nodes (representing input vector $x = (x_1, x_2, x_3)$), two hidden layers with four nodes each and an output layer with six nodes (representing output vector $y = (y_1, y_{2,}y_3, y_4, y_5, y_6)$).

(ReLU) and Softplus are two widely used activation functions that force the output to a non-negative value:

$$ReLU(s) = max(0, s) \qquad 1$$

$$Softplus(s) = ln(1 + e^s) \qquad 2$$

where $\ln(x)$ is the natural logarithm of x. Softmax is another widely used activation function that, when applied to a set of nodes, normalises the outputs such that each is from the interval 0 to 1 and they sum to 1, thus allowing them to be interpreted as probabilities:

$$Softmax(s_i) = \frac{e^{s_i}}{\sum_{j=1}^{N} e^{s_j}} \qquad 3$$

where N is the number of nodes to which the activation is applied and s_i is the summation value for node i.

The neural network is trained by estimating the error gradient with respect to the weights and bias in the ANN, where the error is computed through a loss function, for training examples comprising of input and output pairs and then adjusting the weights in the network to minimise the error through a process known as backpropagation (Hecht-Nielsen, 1992). An optimisation algorithm is used to determine how the weights are adjusted during training. A recent review of popular optimisation algorithms (Ruder, 2016) suggested Adam (Kingma & Ba, 2014) might be the best overall choice. A best practice, to speed up learning and improve convergence, is to perform feature scaling to normalise each input variable so that they share approximately the same distribution prior to training the ANN (Francois Chollet, 2017). A common approach is z-score normalisation to set the mean and standard deviation to 0 and 1, respectively, for each input variable:

$$x' = \frac{x - \tilde{x}}{\sigma_x} \qquad 4$$

where x is the value of the input variable prior to normalisation and $\tilde{x}$ and σ_x are the values for the mean and standard deviation of the input variable within the training data.

The batch size and number of epochs are user-definable hyperparameters that specify the number of training examples utilised per update iteration and

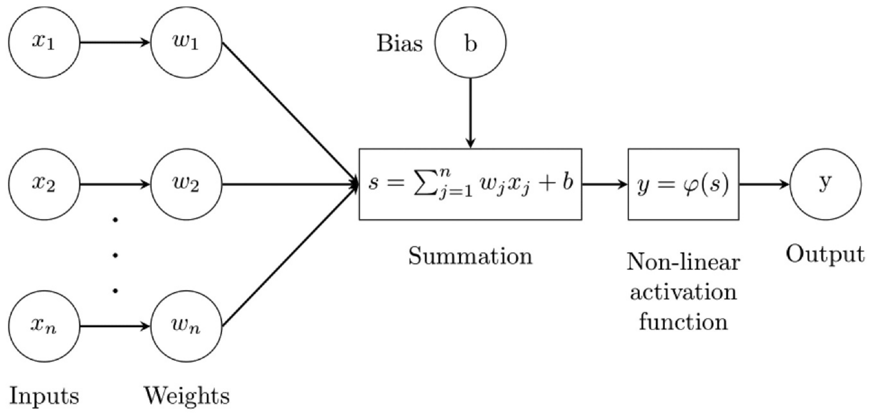

Figure 2. Computation performed by an ANN node with bias and non-linear activation function.

number of complete passes through the training data set, respectively. The goal of training is to enable the network to give an optimal prediction for the output when it is presented with an input value. The tendency for ANNs to perform better on training data compared to new data is known as overfitting (Francois Chollet, 2017). A common step to reduce overfitting is to split the training data into training and validation sets, where the latter is not used to train the ANN but instead test the performance of the ANN once trained. It was proven by Hornik (Hornik, 1991) that multilayer feedforward ANNs are universal approximators, theoretically capable of approximating any measurable function.

A conventional regression ANNs predicts a continuously valued scalar target value from the values of the input vector variables, using a loss function during training that gives a measure of the average difference between this output and the true value (e.g., root mean squared error (RMSE) or mean squared error (MSE)) (Francois Chollet, 2017). However, this results in a conditional average output that provides only a limited description of the properties of the target value (Bishop, 1994). It does not account for variance in the target value or the possibility of it taking one of multiple values for a given input. A solution that gives a complete description of the target value is to instead have the ANN predict its probability distribution conditional on the input vector. This can be achieved by configuring the output layer of the ANN to represent the density function of the distribution and setting the loss function to be minimised as the negative natural logarithm of the likelihood of the target values in the training data (Bishop, 1994). Three main strategies have been proposed in the literature: mixture density networks (MDNs) (Bishop, 1994), quantised softmax networks (Van Den Oord et al., 2016) and kernel mixture networks (Ambrogioni et al., 2017). For an MDN, the outputs represent the weights (non-negative and summing to 1) and parameters for the weighted sum of multiple parametric distributions, i.e. representing a mixture density model. MDNs can in theory represent arbitrary conditional probability distributions, similar to how a conventional ANN can represent arbitrary functions (Bishop, 1994). The number of components in the mixture and choice of parametric distribution for each are user-definable hyperparameters for the network. MDNs have been shown to accurately predict distributions in a number of applications, such as speech synthesis (Zen & Senior, 2014). An example showing the advantage of MDNs over regression ANNs for predicting output values from stochastic and multi-modal functions is given in Appendix A. For a quantised softmax network, the outputs represent the weights (where the weights are non-negative and sum to 1) for a mixture of a finite set of uniform distributions with non-overlapping ranges of equal length that together cover the range of the output values in the training data. The number of bins over which the range of the random variable is divided is a hyperparameter for the network. Compared to MDNs, they have the advantage of approximating arbitrarily complex conditional distributions without making any parametric assumptions (Ambrogioni et al., 2017). For a kernel mixture network, the outputs represent the non-negative weights, summing to 1, of a family of sets of distribution functions (referred to as kernel functions in this context), where the distributions in each set have different scale parameters (where those scale parameters are the same for each family of sets) and each family of sets is centred at output values from a subset of the training data. Typically, a normal distribution is used for the distribution function where the scale parameter corresponds to the standard deviation and the mean parameter is used to centre the distributions at the training data output values. The set of scale parameters to use for each family of distributions is a hyperparameter for the network. Ambrogioni et al. (2017) suggest that the subset of output values from the training data used as centre points are chosen by recursively removing each value that is closer than a specified constant to its predecessor. In their experiments, kernel mixture networks resulted in higher likelihoods on the test data and less overfitting compared to quantised softmax networks. In summary, the three strategies all essentially represent mixtures of distributions in different forms: the parameters and weights of parametric distributed components are predicted by an MDN, the weights of non-overlapping uniform distributed components are predicted by a quantised softmax network and the weights of components from families of distributions with different scale parameters centred at a subset of output values from the training data are predicted by a kernel mixture network. A weakness of ANNs is that they are “black box” models as, although they can approximate any function, they do not readily provide insight into the structure of the function being approximated.

3. Integration of ANNs within a DES to predict random variable distributions

The idea proposed by this paper is to use ANNs within a DES model to predict the distribution of random variables during simulation according to the values of covariates derived from the model state. This gives the advantage over traditional covariate distribution methods that complex relationships between covariate values and the random variable distribution can be accurately modelled. Compared to conventional regression ANNs which only predict the most likely value, it also has the advantage of retaining the stochastic behaviour in the model by predicting arbitrary distributions. An approach to integrating ANNs with a DES model is now given.

The initial step is to identify the random variables in the DES model for which the distribution will be predicted by an ANN and, for each of these, identify the covariates that will form the input vector. A requirement is that training data pairs of covariate value vectors and outcome variable values are available and that the input variables to an ANN can be derived from the DES model state or input whenever the generation of a random variate is required.

For each ANN, the relevant training data should then be divided into training and validation sets and normalised. The configuration for the ANN must then be chosen including the number of nodes per layer, number of hidden layers, number of mixture components, distribution function for each mixture component, loss function, optimisation algorithm, batch size and number of epochs. In most cases, the process of choosing an ANN configuration should be iterative, using feedback from performance on the validation data set during training. Once the ANN is trained and performing sufficiently well on the validation data set, it is ready for embedding within the DES model.

The trained ANNs are integrated into the DES model, along with the input normalisation function that was used during training and a data structure, such as an array, referencing the mixture distribution components of the density function predicted by the ANN. If the ANN is a quantised softmax network then the mixture components referenced in the data structure will be uniform distributions with minimum and maximum parameters set to each cover a portion of the output range. If the ANN is a kernel mixture network, then the mixture components referenced in the data structure will correspond to some parametric distribution (such as normal) with width parameters set to the values chosen as hyperparameters for the network and location parameters set to centre them at the subset of output values from the training data. Finally, if the ANN is an MDN then the mixture components referenced in the data structure will be types of parametric distribution (e.g., Weibull) but the parameter values will not be set since these are predicted by the MDN rather than fixed. The authors recommend that the DES model is constructed as a software code to ease this integration step and suggest the use of open-source DES modelling libraries, such as DESMO-J (Göbel et al., 2013).

To generate a random variate for a random variable that uses an ANN to predict its conditional distribution, several steps are performed. The input vector is formed from variable values from the current DES model state or input and then normalised using the scaling function associated with the ANN that was used for the training data. An output vector, representing the conditional distribution, is then obtained as a prediction from the ANN using the normalised input vector. A categorical distribution is then formed that is parametrised with the predicted weights for the mixture distribution components in the output vector. A random sample is taken from this categorical distribution to select a mixture distribution component according to the predicted weights. Where the ANN is an MDN, the parameter values for this distribution component are set according to the predicted values in the output vector (for the other cases, the parameter values are already fixed). Finally, the random variate is sampled from this mixture distribution component. The sampling of random variates from these distributions is performed in the same way as in any other DES model through the use of a pseudo-random number sampling algorithm (Andrews & Moss, 2002). The full process is illustrated by the flowchart in Figure 3.

4. Application to modelling load-haul-dump vehicle operations in a sublevel caving mine

To demonstrate the benefits of the new approach presented in this paper, a DES model representing load-haul-dump (LHD) vehicle operations in the production area of a sublevel caving mine was developed. LHD vehicles are used in these mines to load mined material, transport it to ore pass shaft entrances and dump it down the ore shafts. The efficiency with which they operate can have a significant impact on the overall mining production rate (Skawina et al., 2015). The goal of the analysis is to evaluate the impact on the production from different configurations in terms of the number of ore passes and the number of rock breaking machine operators. The modelling was implemented in the Python programming language, with MDNs developed and trained using the Keras (François Chollet, 2017) neural network library and Tensorflow (Abadi et al., 2015) machine learning library, and the DES model developed using the SimPy discrete event simulation library (Dagkakis & Heavey, 2016). The example is for demonstrative purposes only and does not represent a real mining site, although realistic assumptions were used.

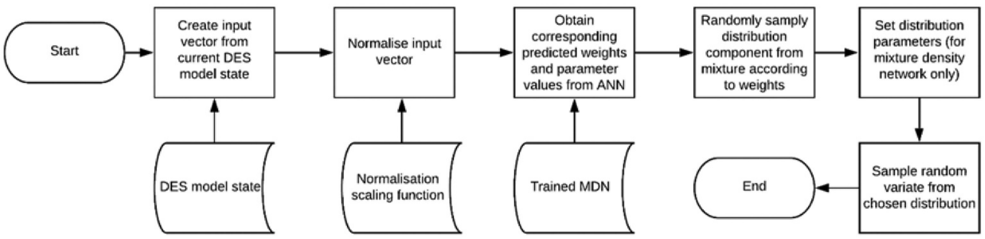
Start
Create input vector from current DES model state
Normalise input vector
Obtain corresponding predicted weights and parameter values from ANN
Randomly samply distribution component from mixture according to weights
Set distribution parameters (for mixture density network only)
DES model state
Normalisation scaling function
Trained MDN
End
Sample random variate from chosen distribution

Figure 3. Flowchart of the steps involved with sampling a random variate for a random variable for which the distribution is represented by an MDN.

4.1. Sublevel caving mining process overview

Figure 4 shows a typical sublevel caving mine. A production area is a subsection of the mine at a certain depth (known as a sublevel) and consists of:

- A footwall drift parallel to the ore body.
- A set of production drifts, with entrances spaced along the footwall drift, aligned perpendicular to the footwall drift that go through the ore body.
- A set of ore pass drifts, with entrances from the footwall drift, with an ore pass shaft entrance and rock breaking machine at the far end.

Sets of holes forming a fan pattern, known as a ring, are drilled upwards into the ore body within the production drifts at set intervals and loaded with explosive charges. During a process known as blasting, the charges in the rings at the far end of a production drift are detonated. This causes the drilled ore to fracture and cave into the vacant space below under the force of gravity (Kvapil, 2008). The caved material forms a drawpoint from which an LHD vehicle removes material through the following process:

(1) Travels to a drawpoint.
(2) If the shut-off criteria (described later) for the drawpoint are not met:
    a. Loads its bucket with caved material.
    b. Selects the ore pass where the material will be dumped.
    c. Transports the loaded material to the shaft entrance of the selected ore pass.
    d. Dumps the loaded material from the LHD vehicle bucket into the ore pass shaft.
(3) If the shut-off criteria are met, a catch-wall is built using the LHD vehicle bucket.

The adjacent ring in the drift is then be blasted and the process repeated for the new drawpoint until the drift is completely developed.

4.2. Impact of waste rock on the mining process

During caving, waste rock from above the ore body mixes with the ore, resulting in dilution at the drawpoint (Bull & Page, 2000). The extraction of material from a drawpoint causes further caving, replacing the

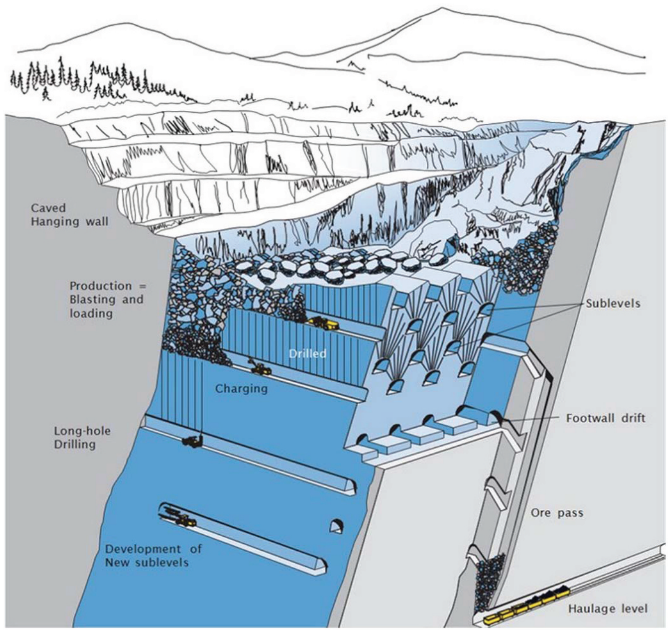

Figure 4. A typical sublevel caving mine (Courtesy of Atlas Copco AB).

removed material. The general trend is for the waste rock proportion to increase with the extraction ratio, the latter defined as the total weight of material removed from a drawpoint of a blasted ring divided by the planned tonnage calculated from the ring geometry (Shekhar, 2018). However, it can vary considerably between successive LHD bucket loads (Shekhar et al., 2019) and factors such as the ring geometry and composition of caved rock from the adjacent production area above. If waste rock dilution exceeds a certain level, the material becomes uneconomical to process. Mines, therefore, utilise a set of heuristics known as shut-off criteria to determine when to stop extracting from a drawpoint. Waste rock is considerably less dense than ore (e.g. reported averages of 2.8 and 4.8 tonnes per m^3, respectively, at the LKAB Kiruna mine (Shekhar et al., 2017)) and therefore the LHD is able to accelerate and decelerate faster when loaded with material at higher waste rock dilutions, reducing average travel times for a given route. Furthermore, the material dumped at an ore pass shaft entrance may contain boulders that result in a blockage that must be cleared. Until a blockage is cleared, an LHD cannot dump material there and must either wait or use another. A remotely operated rock breaking machine positioned at each shaft entrance is used to clear these blockages. On average, waste rock contains more boulders and each takes longer for the rock-breaking machine to destroy than ore boulders. Reported data from the LKAB Kiruna mine (Drakenberg, 2007) showed that the mean number of boulders per tonne of ore and waste rock was 0.063 boulders and 0.115, respectively, and the proportions of boulders of size classification minor, small, medium (classified as big in (Drakenberg, 2007)) and large was 22.5%, 52.5%, 18.75% and 6.25%, respectively. The same study found that whilst minor size boulders could pass through the ore shaft without a problem, the mean time to destroy a boulder of the other sizes by a rock breaker (excluding waiting time for an available operator) was as shown in Table 1.

The influence of waste rock on LHD vehicle activities within a sublevel caving mine should therefore be included in the model due to its significant impact. However, the distribution of the waste rock in material from a drawpoint is conditional on the extraction rate and properties of the blasted ring; the distribution for the travel time of an LHD is conditional on the waste rock proportion and volume of loaded material; and the distribution of the time to clear an ore pass of boulders is conditional on the weight and proportion of waste material in the dumped material. The functional relationships between these distributions and covariates are complex and non-linear. DES models of operations in the production areas of sublevel caving mines have been developed in the past by Vagenas (1996) and Skawina et al. (2015). However, these models did not consider the impact of waste rock.

4.3. Model description and analysis scenarios

The model was developed to simulate a single LHD vehicle operating within a production area of a sublevel caving mine with the basic layout depicted in Figure 5.

Three scenarios were analysed using the model, named A, B, and C. In Scenario A and Scenario C, it was assumed that there were two ore passes in the production area (see Figure 5), whilst an ore pass with its entrance located equidistantly between the other two was added for Scenario B. In Scenario A and Scenario B, the availability of a single operator of the rock breaking machines at any time was assumed, whilst this was increased to two operators for scenario C. Realistic assumptions were used in the model for ring geometries and spacings; LHD operator selection of drawpoints and ore passes; material volumes loaded in the LHD vehicle bucket; LHD vehicle loading, dumping and catch wall construction times; the

Table 1. Mean times, in seconds, for a rock breaker to destroy boulders of different sizes and material types (derived from a study at the LKAB Kiruna mine (Drakenberg, 2007)).

	Material	
Size	Waste Rock	Ore
Small	220	142
Medium	348	276
Large	750	408

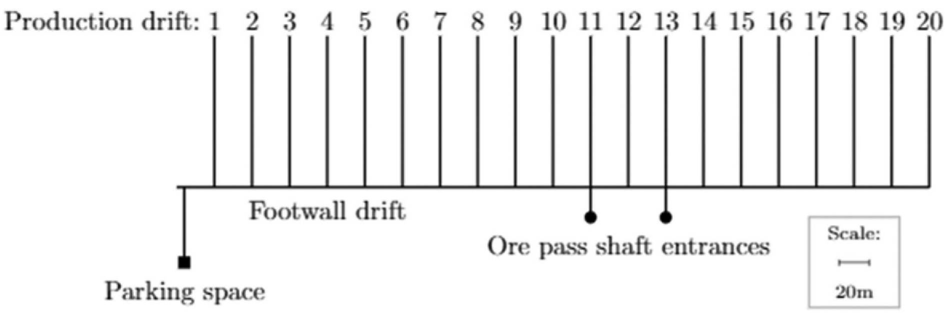
Production drift: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Footwall drift
Ore pass shaft entrances
Parking space
Scale:
20m

Figure 5. Layout of the drifts in the production area in which the LHD vehicle operates.

shut-off criteria for drawpoints; and the schedules for blasting and LHD vehicle operator shifts.

MDNs were used in the model to determine the distributions of three random variables during simulation: the proportion of waste rock in each bucket of material loaded by an LHD vehicle, the time taken for an LHD vehicle to travel between two points in the production area and the time taken for a rock-breaking machine to clear an ore pass shaft entrance of boulders after a bucket of material is dumped there.

4.3.1. Training data generation

For each of the MDNs, a training data set of input vector and output value pairs was generated from simulation models. For the vehicle travel time MDN, the training data were generated by sampling random routes and using a continuous simulation, with a small time step, of the vehicle kinematics with random variations in acceleration, braking and disturbances to replicate driver inputs and traffic. The data for training the waste rock MDN was generated by randomly sampling, for each drawpoint, a sigmoid curve and Gompertz curve for the mean and standard deviation, respectively, of the waste rock dilution against extraction ratio. These curves were chosen to mimic the dilution curves for sublevel caving mines suggested by Bull and Page (2000). The volume of material extracted in a bucket load from each drawpoint was simulated by sampling from a uniform distribution, which matches the distribution choice in the model by Vagenas (1996). Meanwhile, the waste rock proportion in the loaded material was sampled from a normal distribution, representing the variation between consecutive buckets as shown by Shekhar (2018), with the mean and standard deviation taken from the drawpoint curves that were computed from the simulated extraction ratio. For the rock breaking MDN, the training data were generated by randomly sampling dumped material weights and waste rock proportions from uniform distributions, where uniform distributions were chosen to follow the model from Vagenas (1996). The sampling of boulder breaking times utilised the results reported by Drakenberg (2007), summarised in Section 4.2, through the following procedure:

(1) The number of boulders in the waste and ore proportions were sampled from a Poisson distribution according to the reported rates.
(2) The boulder sizes were sampled from a categorical distribution according to the reported probabilities.
(3) The breaking times for the individual boulders were sampled from normal distributions with mean selected according to those reported for the boulder size and waste rock proportion and estimated standard deviations (since none were reported by Drakenberg (2007)).
(4) The individual boulder breaking times were summed to get the total time.

Note that in a practical setting, the training data might instead comprise observations of the real system or process allowing highly realistic modelling and the implementation of digital twins that learn over time. Each set of training data was split into two groups: training data (75%) and validation data (25%). The values for the input variables in the data were normalised using z-score normalisation (see Equation 4). A configuration for each MDN was found through trial and error that gave good predictive performance for the validation data.

4.3.2. Waste rock proportion in material loaded into bucket of LHD vehicle

An MDN was trained, using data generated from a simulation model, to predict the distribution of the

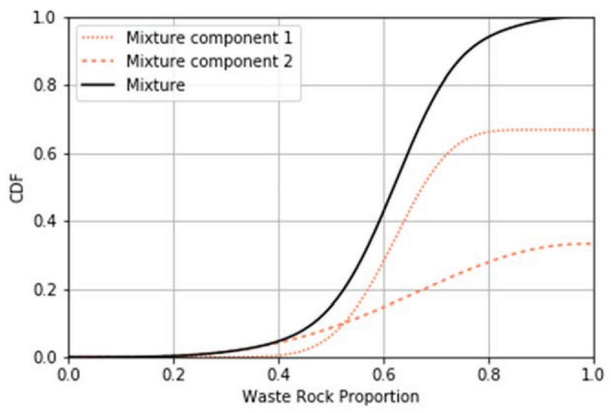

Figure 6. Cumulative distribution function for the waste rock proportion predicted by the trained MDN for an extraction ratio of 0.9677 and waste rock proportion by weight of the previous five loaded buckets of 0.71, 0.62, 0.76, 0.71 and 0.74. The contribution from each of the two mixture components is also shown.

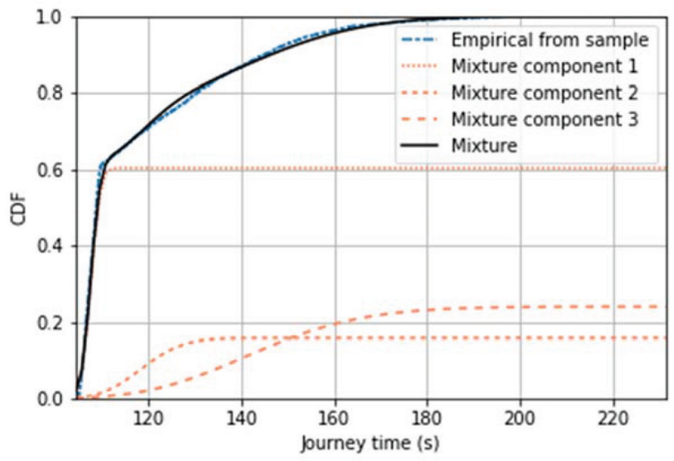

Figure 7. Cumulative distribution function for the journey time predicted by the trained MDN for a maximum load proportion of 0 and segment travel distances of 25 m, 400 m and 80 m. The contribution from each of the three mixture components is also shown along with the empirical cumulative distribution function derived from sample data generated from the training data simulation model for the same maximum load proportion and segment travel distances.

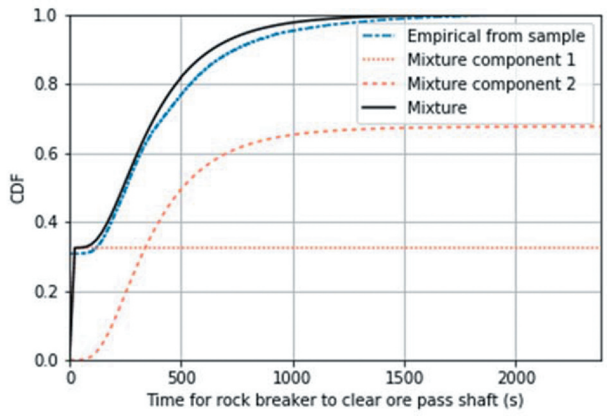

Figure 8. Cumulative distribution function for the time for the rock breaker to clear the ore pass shaft, as predicted by the trained MDN when the weight of material dumped is 17.3 tonnes and waste rock proportion is 0.58. The individual contribution to this from each of the two mixture components is shown separately. The empirical cumulative distribution function, generated from the training data simulation model for the same material weight and waste rock proportion, is also shown for comparison.

proportion by weight of waste rock in each bucket of material loaded by an LHD vehicle from a drawpoint based on the extraction ratio and the proportion of waste rock in previously loaded buckets of material. The MDN was configured to represent a mixture distribution with two Beta distributed components. The beta distribution was chosen since it is defined on the interval 0 to 1 matching the range of possible waste rock proportions, whilst the use of multiple components allows the MDN to predict more complex distributions, as discussed in Section 2 and demonstrated in Appendix A (e.g. comparing the predictions from a regression ANN, single component MDN and two component MDN shown in Figures A1–Figures A3, respectively, when trained on the same sample data). It was configured from dense layers: an input layer with six nodes, two hidden layers with 15 nodes each with ReLU activation and an output layer comprising two nodes with a Softmax activation, representing the weightings of the mixture components, and four nodes with a Softplus activation, representing the shape parameters of the mixture components. The loss function was set to the negative natural logarithm of the likelihood and the MDN was trained with a batch size of 50 over 25 epochs using the Adam optimisation algorithm. An example of a distribution predicted by the trained MDN is shown in Figure 6.

4.3.3. Travel time along a segment for an LHD vehicle

An MDN was trained, using data generated from a simulation model, to predict the distribution of the time for an LHD vehicle to travel between an origin and destination in the production area based on the proportion of the maximum load carried in its bucket and distances of segments of the journey. The MDN was configured to represent a mixture distribution with three lognormally distributed components. It was configured from dense layers: an input layer with four nodes, two hidden layers with 25 nodes each with ReLU activation and an output layer comprising three nodes with a Softmax activation, representing the weightings of the mixture components, three nodes representing the location parameters of the mixture components and three nodes with a Softplus activation, representing the scale parameters of the mixture distribution components. The loss function was set to the negative natural logarithm of the likelihood and the MDN was trained with a batch size of 50 over 25 epochs using the Adam optimisation algorithm. Figure 7 shows a distribution predicted by the MDN along with an empirical distribution derived from 10,000 samples generated from the training data simulation model, both corresponding to the same maximum load proportion and segment travel distances. The close agreement between the two distributions shows that the MDN accurately predicts the true distribution.

4.3.4. Rock breaker ore pass shaft entrance clearance time

An MDN was trained, using data generated from a simulation model, to predict the distribution of the time required by a rock-breaking machine to clear an ore pass of boulders in material dumped by an LHD vehicle based on the material weight and waste rock proportion. The MDN was configured to represent a mixture distribution with two lognormally distributed components. It was configured from dense layers: an input layer with two nodes, two hidden layers with 15 nodes each with ReLU activation and an output layer comprising two nodes with a Softmax activation, representing the weightings of the mixture components, three nodes representing the location parameters of the mixture components and three nodes with a Softplus activation, representing the scale parameters of the mixture distribution components. The loss function was set to the negative natural logarithm of the likelihood and the MDN was trained with a batch size of 50 over 25 epochs using the Adam optimisation algorithm. Note that since the probability of value 0 for a log-normally distributed variable is 0 and hence the log-likelihood undefined, a small value (1e-20) was added to each output value in the training data, thus allowing the log-normal mixture model to be fitted. Figure 8 shows a distribution predicted by the MDN along with an empirical distribution derived from 10,000 samples generated from the training data simulation model, both corresponding to the same material weight and waste rock proportion. The similarity between the empirical and predicted distributions shows that the MDN accurately predicts the true distribution.

4.3.5. Simulation of the LHD vehicle

A typical activity cycle for an LHD vehicle during a simulation will now be described. The drift from which it next loads material is selected and the travel time to that drawpoint then sampled. This sample is taken from the distribution predicted by the travel time MDN, based on the loaded material and the route from its current location. On arrival at the drawpoint, the volume of material loaded is sampled from a fixed distribution and the amount of waste rock in that material is sampled from a distribution predicted by the waste rock proportion MDN based on the current extraction ratio and waste rock proportions from previously loaded buckets. From the sampled load volume and waste rock proportion, the weight of loaded material is calculated. An ore pass is then selected to dump material, with priority to unblocked and closest, and the travel time sampled from the distribution predicted by the travel time MDN based on the loaded weight and route from the drawpoint. On arrival at the ore pass, if clear, the material is dumped otherwise the vehicle waits until it has been unblocked. The time to clear the ore pass of dumped material by the rock breaker is then sampled from the distribution predicted by the rock breaker MDN based on the weight of material dumped and waste rock proportion. In addition to this repeating activity cycle, other details including the scheduling of vehicle operator changeovers and breaks are also simulated.

5. Results

The model was simulated to determine the time taken to complete the development of the production area in the three scenarios under consideration, with the use of MDNs allowing the impact of waste rock on LHD vehicle operations to be simulated realistically. The time to develop the production area was simulated as 1529 days in Scenario A (shown in Figure 9 with progress in individual production drifts), 1427 days in Scenario B and 1266 days in Scenario C.

Figure 10 shows the amount of time the LHD vehicle spent performing different activities during the simulation of each of the three scenarios. It spent the largest proportion of time travelling between locations in the production area in all three scenarios and the time spent waiting to unload, due to boulders blocking the ore pass shaft entrance, was greatest in Scenario A and least in Scenario C. Therefore, increasing the number of rock breaking machine operators from one to two is more effective than increasing the number of ore passes from two to three for reducing waiting times to dump material and increasing the production rate.

6. Conclusions and future work

This paper showed how ANNs designed to predict the distribution of the output based on a given input, such as MDNs, can be incorporated within a DES model to enhance the realism of the modelled stochastic behaviour. The example model of LHD vehicle operation in a sublevel caving mine illustrated the significant impact that waste rock in mined material has on the production rate, something not captured in previous DES models of similar scenarios (e.g., Vagenas (1996) and Skawina et al. (2015)). The results show that improved modelling of complex stochastic behaviour can be achieved within a DES by embedding MDNs, or other types of distribution predicting ANN, using the approach shown in this paper. This represents an enhancement over embedding conventional regression ANNs into DES, as introduced previously (e.g., Bergmann et al. (2014)), as they can only predict expected outcomes in a deterministic way. A main application of the approach is in the development of digital twin models that model the true behaviour of real-world system, adapting to changes over time. An area for future work is the development of the example model into a digital twin where the MDNs are trained on data from a real mine, giving the potential to optimise the production plan using the efficient MDN-DES model.

Acknowledgments

The authors would like to thank the anonymous reviewers for critically reading the manuscript and providing suggestions and comments that helped to clarify and improve it. This work is, with regard to Dr. Sean Reed and Prof. M. Löfstrand, carried out within the project *A digital twin to support sustainable and available production as a service*, funded by Produktion2030, the Strategic innovation programme for sustainable production in Sweden, and is supported by the project *Production Centred Maintenance for real time predictive maintenance decision support to*

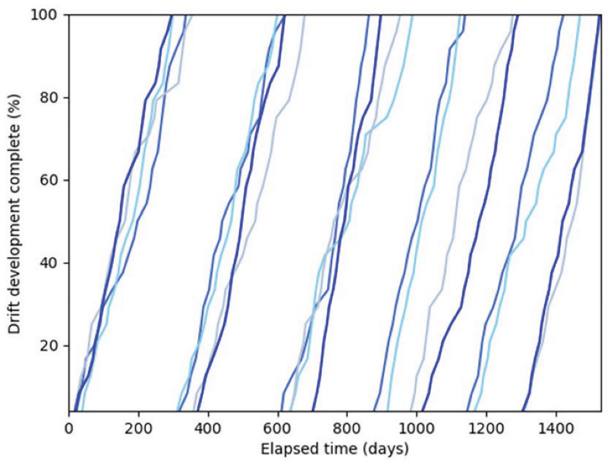

Figure 9. Development of production drifts over time during simulation of Scenario A, where each plot line represents an individual drift.

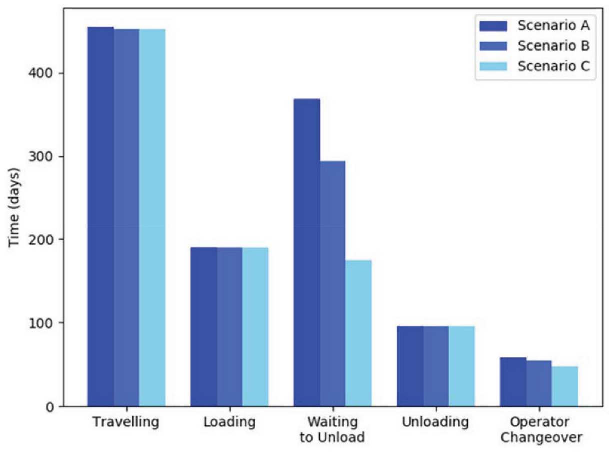

Figure 10. Time spent by LHD vehicle in each activity during simulated development of the mine production area in each scenario.

maximise production efficiency, funded by the Swedish Knowledge Foundation. We gratefully acknowledge the support and funding.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the VINNOVA.

ORCID

Sean Reed http://orcid.org/0000-0002-5698-6740
Magnus Löfstrand http://orcid.org/0000-0002-2014-1308
John Andrews http://orcid.org/0000-0002-2316-3959

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., & Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous distributed systems*. http://tensorflow.org/

Aggarwal, C. C. (2018). *Neural networks and deep learning: A textbook* (1st ed.). Springer.

Ambrogioni, L., Güçlü, U., van Gerven, M. A. J., & Maris, E. (2017). *The kernel mixture network: A nonparametric method for conditional density estimation of continuous random variables*. http://arxiv.org/abs/1705.07111

Andrews, J. D., & Moss, B. (2002). *Reliability and risk assessment* (2nd ed.). Wiley-Blackwell.

Banks, J. (1998). *Handbook of simulation: Modelling, estimation and control*. John Wiley & Sons.

Bender, R., Augustin, T., & Blettner, M. (2005). Generating survival times to simulate Cox proportional hazards models. *Statistics in Medicine*, *24*(11), 1713–1723. https://doi.org/10.1002/sim.2059

Bergmann, S., Stelzer, S., & Strassburger, S. (2014). On the use of artificial neural networks in simulation-based manufacturing control. *Journal of Simulation*, *8*(1), 76–90. https://doi.org/10.1057/jos.2013.6

Bishop, C. M. (1994). *Mixture density networks*. Aston University.

Boos, D. D., & Stefanski, L. A. (2013). *Essential statistical inference: Theory and methods*. Springer.

Bull, G., & Page, C. . (2000). Sublevel caving - today's dependable low-cost "Ore Factory." *Proceedings of the 3rd International Conference and Exhibition on Mass Mining*, 537–556.

Chang, W.-J., & Chang, Y.-H. (2018). Design of a patient-centered appointment scheduling with artificial neural network and discrete event simulation. *Journal of Service Science and Management*, *11*(1), 71–82. https://doi.org/10.4236/jssm.2018.111007

Chollet, F. (2017). *Deep learning with python* (1st ed.). Manning Publications.

Chollet, F. (2017). *Keras*.

Dagkakis, G., & C. Heavey. (2016). A review of open source discrete event simulation software for operations research, *Journal of Simulation*, *10*(3), 193–206.

De la Fuente, R., Smith, R. L., III, & Erazo, I. (2018). Enabling intelligent processes in simulation utilizing the TensorFlow deep learning resources. *Proceedings of the 2018 Winter Simulation Conference*, 1108–1119.

Drakenberg, M. (2007). *An analysis of the boulder-handling system at the Kiruna Mine, LKAB*. Luleå University of Technology.

Göbel, J., Joschko, P., Koors, A., & Page, B. (2013). The discrete event simulation framework DESMO-J: Review, comparison to other frameworks and latest development. *Proceedings - 27th European Conference on Modelling and Simulation, ECMS 2013*, *4*, 100–109. https://doi.org/10.7148/2013-0100

Grieves, M. (2014). *Digital twin: Manufacturing excellence through virtual factory replication. White Paper*.

Haag, S., & Anderl, R. (2018). Digital twin – proof of concept. *Manufacturing Letters*, *15*(Part B), 64–66. https://doi.org/10.1016/j.mfglet.2018.02.006

Harden, J. J., & Kropko, J. (2019). Simulating duration data for the cox model. *Political Science Research and Methods*, *7*(4), 921–928. https://doi.org/10.1017/psrm.2018.19

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. *Neural Networks for Perception*, 1(1), 65–93. https://doi.org/10.1016/B978-0-12-741252-8.50010-8

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*(2), 251–257. https://doi.org/10.1016/0893-6080(91)90009-T

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*. 1–15. http://arxiv.org/abs/1412.6980

Kvapil, R. (2008). *Gravity flow in sublevel and panel caving - a common sense approach*. uleå University of Technology.

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., & Hoffmann, M. (2014). Industry 4.0. *Business & Information Systems Engineering*, *6*(4), 239–242. https://doi.org/10.1007/s12599-014-0334-4

Leemis, L., Shih, L.-H., & Reynertson, K. (1990). Variate generation for accelerated life and proportional hazards models with time dependent covariates. *Statistics & Probability Letters*, *10*(4), 335–339. https://doi.org/10.1016/0167-7152(90)90052-9

Negri, E., Fumagalli, L., & Macchi, M. (2017). A review of the roles of digital twin in CPS-based production systems. *Procedia Manufacturing*, *1*(1), 939–948. https://doi.org/10.1016/j.promfg.2017.07.198

O'Hagan, A., Buck, C. E., Daneshkhah, A., Eiser, J. R., Garthwaite, P. H., Jenkinson, D. J., Oakley, J. E., & Rakow, T. (2006). *Uncertain judgements: Eliciting experts' probabilities*. John Wiley & Sons.

Ross, S. (2012). *Simulation* (5th ed.). Academic Press.

Ruder, S. (2016). *An overview of gradient descent optimization algorithms*. 1–14. http://arxiv.org/abs/1609.04747

Shekhar, G. (2018). *Draw control strategy for sublevel caving mines*. Luleå University of Technology.

Shekhar, G., Gustafson, A., Hersinger, A., Jonsson, K., & Schunnesson, H. (2019). Development of a model for economic control of loading in sublevel caving mines. *Mining Technology: Transactions of the Institute of Mining and Metallurgy*, *128*(2), 118–128. https://doi.org/10.1080/25726668.2019.1586371

Shekhar, G., Gustafson, A., & Schunnesson, H. (2017). *Loading procedure and draw control in LKAB's sublevel caving mines*. Luleå University of Technology.

Skawina, B., Salama, A., Greberg, J., & Schunnesson, H. (2015). Production rate comparison using different load-haul-dump fleet configurations: Case study from Kiirunavaara Mine. *23rd International Symposium on Mine Planning and Equipment Selection*.

Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., & Sui, F. (2018). Digital twin-driven product design, manufacturing and service with big data. *International Journal of Advanced Manufacturing Technology*, *94*(9–12), 3563–3576. https://doi.org/10.1007/s00170-017-0233-1

Vagenas, N. (1996). Simulation modeling of a fleet of remote-controlled/automatic load-haul-dump vehicles in underground mines. *Simulation*, *67*(5), 331–342. https://doi.org/10.1177/003754979606700504

Van Den Oord, A., Kalchbrenner, N., & Kavukcuoglu, K. (2016). *Pixel recurrent neural networks.*

Zen, H., & Senior, A. (2014). Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. *Icassp*, 3872–3876. https://ieeexplore.ieee.org/document/6854321

Appendix A Comparison of regression ANNs and MDNs in predicting stochastic and multi-modal functions

To demonstrate the capabilities of MDNs compared to a conventional regression ANN, consider the function described by Equation. 5:

$$y = \mathcal{N}(1 + x, 0.1x) + Bern(x^2),\ 0 \le x \le 1 \qquad 5$$

where $Bern(p)$ is a Bernoulli distributed random variable with probability p and $\mathcal{N}(\mu, \sigma)$ is a normally distributed random variable with mean μ and standard deviation σ. A training data set, comprising 5000 data points, was generated by randomly sampling x values from the uniform distribution over interval $0 \le x \le 1$ and randomly sampling corresponding y values from Equation 5. Three different dense ANNs were trained on these training data to learn the function described by Equation 5: a conventional regression ANN, an MDN with one normally distributed component and an MDN with two normally distributed components. The three ANNs shared the following identical configuration for the input and hidden layers: an input layer with a single node (representing input x); two hidden layers with 25 nodes each and ReLU activation. They were all trained using the Adam optimiser with a batch size of 25 over 50 epochs. Only the output layers and loss functions were different, reflecting the different approaches:

- Regression ANN: an output layer with a single node (representing output y) and MSE loss function.
- Single component MDN: an output layer with two nodes, representing the mean and standard deviation of a normal distribution describing output y, and loss function computed as the negative natural logarithm of the likelihood. The node representing the standard deviation used a Softplus activation.
- Two component MDN: an output layer with six nodes (representing the weight, mean and standard deviation for each of two normally distributed components of a mixture distribution describing output y) and loss function computed as the negative natural logarithm of the likelihood. The nodes representing the weights used a Softmax activation whilst those representing the standard deviation used a Softplus activation.

A plot of the predicted y values for $0 \le x \le 1$ from the regression ANN is shown in Figure A1, along with the training data points. The ANN predicts y values that approximate the mean average of the training data, but gives no quantification of its distribution, resulting in a poor representation of the underlying function given in Equation 5 from which the training data were generated. For example, at $x \approx 0.8$ it predicts $y \approx 1.4$ which has an extremely low probability of occurrence according to Equation 5.

Figure A2 shows a plot of the mean, along with the interval within one standard deviation, for the predicted normal distribution of y for $0 \le x \le 1$ from the single component MDN, along with the training data points. The predicted mean values for y are similar to the predicted y values from the regression ANN, however by also providing a measure of the deviation it gives a much better representation of the underlying function. Nevertheless, due to being limited to a single normally distributed component, it gives a poor representation of the true distribution of y for certain x values. For example, at $x \approx 0.2$ it significantly underestimates the probability of $y \approx 0.8$.

Figure A3 shows a plot of the weights, means and interval within one standard deviation, for the predicted normally distributed components of y for $0 \le x \le 1$ from the two-component MDN, along with the training data points. It shows that this model accurately predicts the distribution of y over the whole interval $0 \le x \le 1$.

This example shows that MDNs can predict complex distributions for a continuous variable conditioned on one or more covariates by learning from suitable training data, providing more powerful predictive capabilities than conventional regression ANNs when the underlying function to be learned is stochastic and multi-modal.

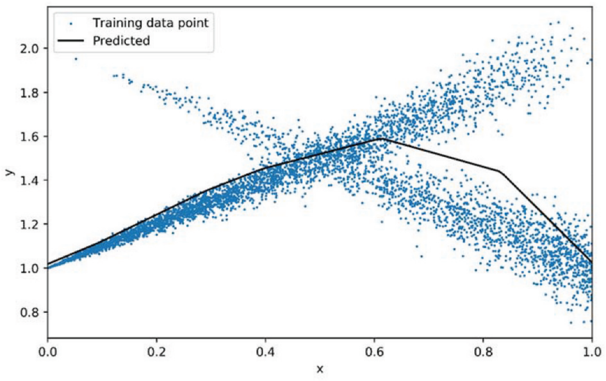

Figure A1. Plot showing predictions from regression ANN trained on 5000 data points generated from model described by Equation 5.

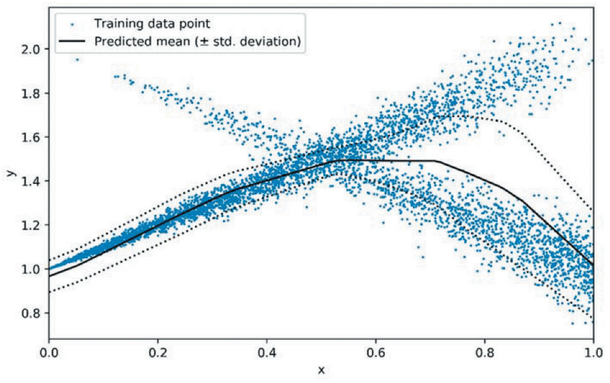

Figure A2. Plot showing predictions from MDN with one normally distributed component trained on 5000 data points generated from model described by Equation 5.

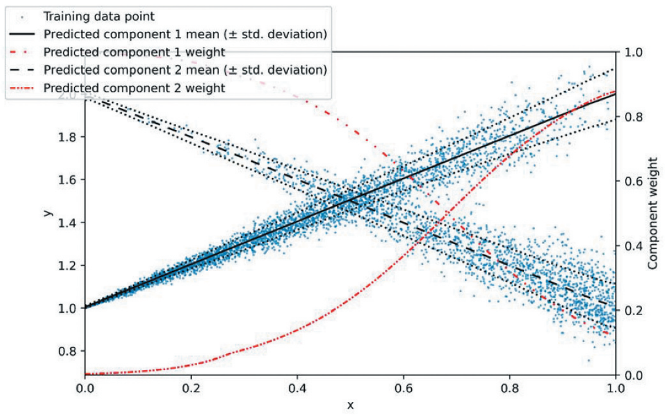

Figure A3. Plot showing predictions from MDN with two normally distributed components trained on 5000 data points generated from model described by Equation 5.