
Proceedings Paper:
Dennis, A., Stirling, C. and Rigby, S. orcid.org/0000-0001-6844-3797 (2023) Towards the development of Machine Learning tools for blast load prediction. In: Proceedings of the 6th International Conference on Protective Structures (ICPS6). 6th International Conference on Protective Structures (ICPS6), 14-17 May 2023, Auburn, AL, United States. International Association of Protective Structures .

6th International Conference on Protective Structures (ICPS6)
14 – 17 May 2023, Auburn University USA

TOWARDS THE DEVELOPMENT OF MACHINE LEARNING TOOLS FOR BLAST LOAD PREDICTION

Adam. A. Dennis
Department of Civil & Structural Engineering, University of Sheffield,
Mappin Street, Sheffield, S1 3JD, UK. aadennis1@sheffield.ac.uk (corresponding author)

Christopher. G. Stirling
Viper Applied Science (www.viper.as), Glasgow, United Kingdom

Samuel. E. Rigby
Department of Civil & Structural Engineering, University of Sheffield,
Mappin Street, Sheffield, S1 3JD, UK.

ABSTRACT
As explosive events are inherently unpredictable, probabilistic approaches featuring large batches of models with varying input conditions are becoming more prominent in risk assessments and design. Computational fluid dynamics (CFD) allows for the direct solution of blast wave propagation in complex geometries. However, the parameter-rich calculation process can result in prohibitively high computation times, or the need to analyse only a subset of the problem space, making the development of rapid analysis tools essential. This article presents a new analysis approach that leverages the computational benefits from two studies conducted by the author and colleagues to highlight developments made towards this aim. Starting with the Branching Algorithm (BA), informed data mapping reduces the required computation time of a batch of similar explosive scenarios by determining when each model's parameter field would deviate from the others in the CFD process. Thus requiring fewer models to run from birth to termination. The dataset being generated by the BA is then used to incrementally train the Direction-encoded Neural Network (DeNN), a novel approach for peak parameter predictions in complex domains, in series. Once the DeNN reaches a prescribed performance threshold, it replaces the CFD models for the remainder of the required batch. Together, these approaches allow for robust assessments of varied geometries to be generated with a reduction in computation time of 80%, and average percentage errors of 10.46%, when compared to using CFD models exclusively for a batch of 20 models.

Keywords: Artificial Neural Network; Machine Learning; Computation time; Batch.

INTRODUCTION

Rapid analysis tools provide a means of investigating the consequences of variability in explosive events by enabling users to simulate tens, or hundreds of unique domains with limited computational effort when compared to parameter-rich numerical solvers. However, this is often achieved by compromising predictive accuracy and/or tool versatility. For example, the well-established Kingery and Bulmash method provides accurate predictions based on semi-empirical equations, but for a limited number of free air scenarios [1]. An alternative would therefore be required when exploring the variability and risk posed to a complex internal environment or city street, where wave interaction effects will change the topology of the pressure distribution throughout any given domain.

Machine learning (ML) tools, such as Artificial Neural Networks (ANNs), have been tested for use in blast engineering with various studies reporting high prediction-target correlations, low absolute errors, and fast computation times behind blast barriers, along city streets and in internal environments [2]–[5]. Additional studies have also proved that performance can be improved when using transfer learning, or physics guided networks [6], [7]. However, for each application, the developed ANNs receive a series of inputs that are often specific to a select number of scenarios, limiting their applicability for probabilistic assessments. For instance, Dennis et al. [5] provide their ANN with the location of the charge and each prediction point based on a user-defined origin with Cartesian coordinates when predicting peak specific impulse. This results in the tool being useless when alternative domain shapes and sizes require simulation, or when obstacles in the domain need to be repositioned.

The 'Direction-encoded ANN' (DeNN) presented by [8] averts these issues by considering how the blast wave travels to each point of interest, referencing the surroundings instead of the domain itself on a point by point basis. This novel approach enables the developed ANN to be used with various geometries, allowing the user to add, remove or reposition objects. It is therefore well suited to the analysis of probabilistically-derived models that may feature unique geometries or varying charge locations.

Despite this, as with all ML tools, performance is dictated by quality and quantity of data in a training dataset that is used by the ANN to understand the problem being modelled. It is common for numerical models to be used when compiling these datasets [9], [10] due to the expense and cost associated with physical experiments. Thus, reintroducing the need to simulate many domains with potentially prohibitively large computation times. The 'Branching Algorithm' (BA) introduced in [11], provides an approach to reduce this requirement by removing repeat analysis steps from batches of numerical models through informed data mapping so that only one model runs from birth to termination.

The paper summarises the BA and the DeNN before introducing the Direction-encoded Neural Network in series (DeNNIS) training process that utilises both method's advantages when conducting probabilistic assessments. It is shown that by incrementally training the DeNN with outputs from models in the simulation framework produced by the BA, computation time can be saved when analysing a batch of similar domains.

THE DIRECTION-ENCODED NEURAL NETWORK

Introduction to Artificial Neural Networks (ANNs)

Artificial neural networks used for regression analyses typically comprise of three types of layers, each containing a number of neurons. An input layer contains predefined variables related to the problem being modelled. Then, a number of hidden layers, with hidden neurons, process this information before the output layer is reached to provide the network's predictions.

Fully connected networks include connections between every neuron on one layer, with all neurons in the layers one step in front, and one step behind. In the forward pass through the network, each neuron processes its input information (x_i) by summing each input, multiplied by a connection weight (w_i), with a predefined neuron-specific bias (b). The sum is then evaluated with an activation function (K) to form the neuron's output (y) that is passed along the next connection, or output as a model prediction. This process is summarised by Equation 1 for a neuron in the hidden layer.

$$y = K(b + \sum_i w_i x_i) \tag{1}$$

Where i is the number of inputs to the neuron.

The weights and biases are iteratively updated to improve the predictive performance of the ANN through a process of backpropagation using an optimiser algorithm. This is most commonly achieved in a supervised learning process whereby a training dataset is formed with known input-output combinations. A trained ANN is able to generalise problems with multiple variables to provide predictions for inputs that were not included in the training dataset. They require little computational effort and can generate prediction in less than a second, hence making them a useful tool for the rapid analysis of complex problems.

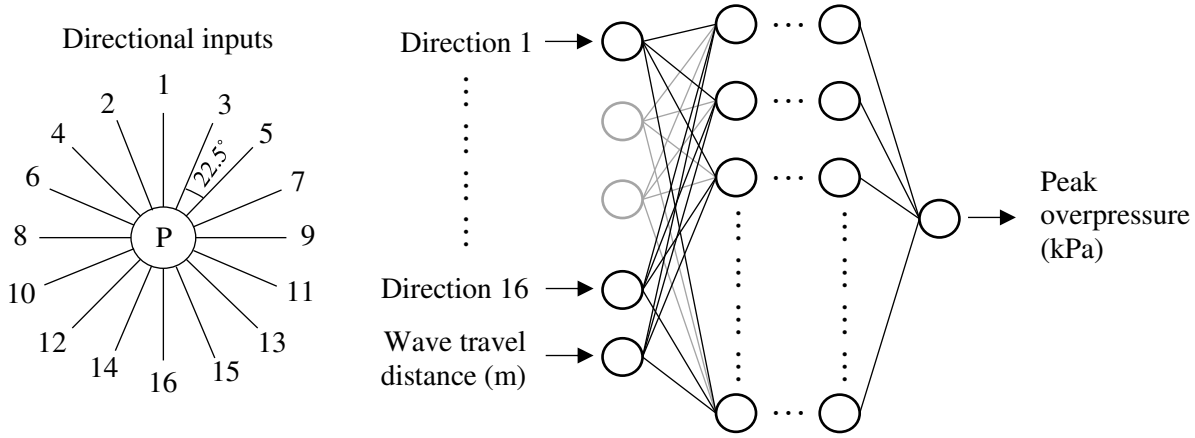
Application of the Direction-encoded Neural Network (DeNN)

The Direction-encoded ANN is a novel approach to machine learning for complex blast scenarios whereby each point of interest is translated to the network via inputs associated with its surroundings [8]. This differs from previous articles studying ANNs where the domain is referenced instead. Therefore the key contribution of this approach was that the DeNN could be used for domains of any shape or size with movable objects, whereas previously a new network would have to be developed and trained for each new scenario.

Figure 1 provides a summary of the DeNN showing that the fully connected network includes 16 directional lasers as inputs to the model along with the blast wave's shortest travel distance to the point of interest (POI). The rosette of lasers is rotated such that direction 1 faces the charge directly, ignoring any obstacles in the domain, and the magnitude for all directions is calculated using a bespoke wave reflection equation. Peak overpressure is calculated as the single output to the model at a fixed height of 1.5 m above a rigid reflecting floor. This corresponds to a 1kg spherical TNT charge, also at height of 1.5 m. An example of how the directional lasers are applied is shown in Figure 2.

This network structure was shown to effectively predict the effects of shielding, clearing and pressure amplification in front of rigid obstacles following a training process that utilised a dataset formed from 25 randomised domains [8]. It was noted by the authors that even with average absolute errors of ~5 kPa for unseen inputs, it is possible that prediction accuracy could be improved if the network was tailored to specific types of structural arrangements. This would require compiling a training dataset of domains that

are similar to the ones that need to be modelled, thus ensuring that the intrinsic processes occurring for those domains are well understood by the machine learning tool.



With,

- Direction 1 being projected towards the charge.
- Obstructions caused by ambient boundaries providing an input of 0.
- Directional inputs = max(Wave travel distance – Obstruction distance , 0)
- Predictions not formed within 1.5 m of the charge centre.

Figure 1. Schematic of the 'Direction-encoded Neural Network', adapted from [8]

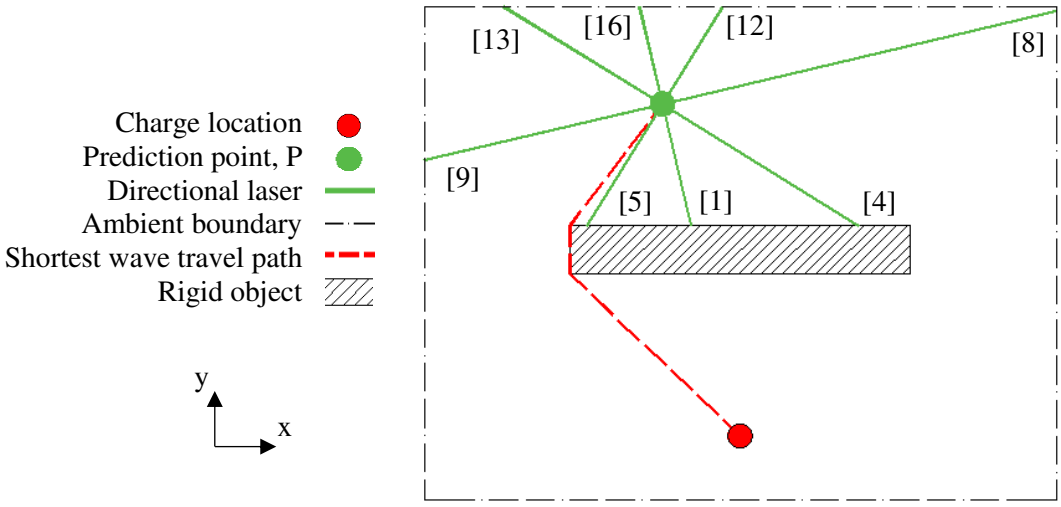


Figure 2. Example input arrangement for the DeNN. Only 8 of the 16 directional lasers shown for brevity. Values in square brackets indicate the laser number.

This paper will explore this hypothesis through incremental training of the DeNN with training data taken from a batch of similar domains with the Branching Algorithm being used to save computation time in developing this dataset. The next section summarises how this algorithm is applied before a combined analysis approach is presented.

THE BRANCHING ALGORITHM (BA)

A summary of the BA is provided in Figure 3. For a batch of models, informed data mapping enables repeat simulation steps to be removed from the calculation process [11]. This is achieved by determining when the parameter field of each model in the batch would diverge from the others. A 'trunk' model is identified as the only model that is required to run from birth to termination. All others benefit from data mapping to reduce computation time.

Seeing as the DeNN operates on a 2D plane, the adaption of this approach for this article differs slightly from the 2D application given previously by the authors. Here, the calculation of the wave travel distance to each POI uses a discretised domain and all points are included in the influence table when determining the locations where parameter fields diverge. Previously, points were only included for each panel face and object vertex. However, implementing a more rigorous process ensures robustness in the calculation of these parameters that are essential for determining the deviation locations.

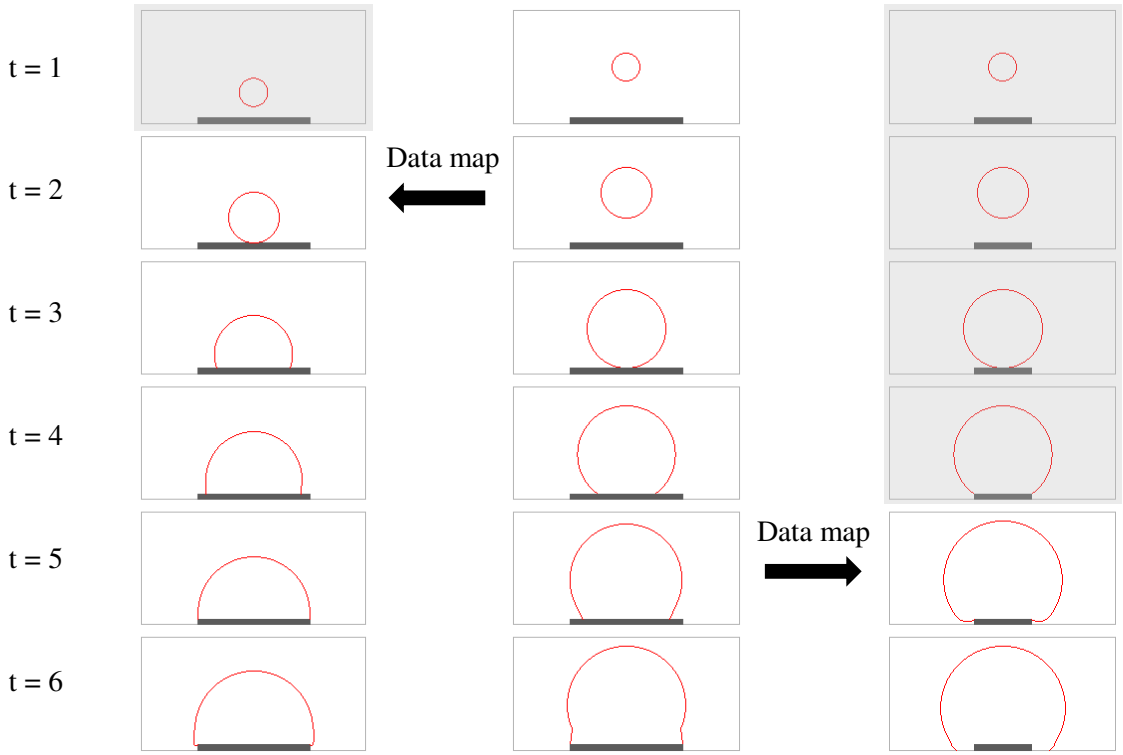


Figure 3. Example mapping stages for a batch of models that has been evaluated by the Branching Algorithm. Grey images indicate saved simulation steps.

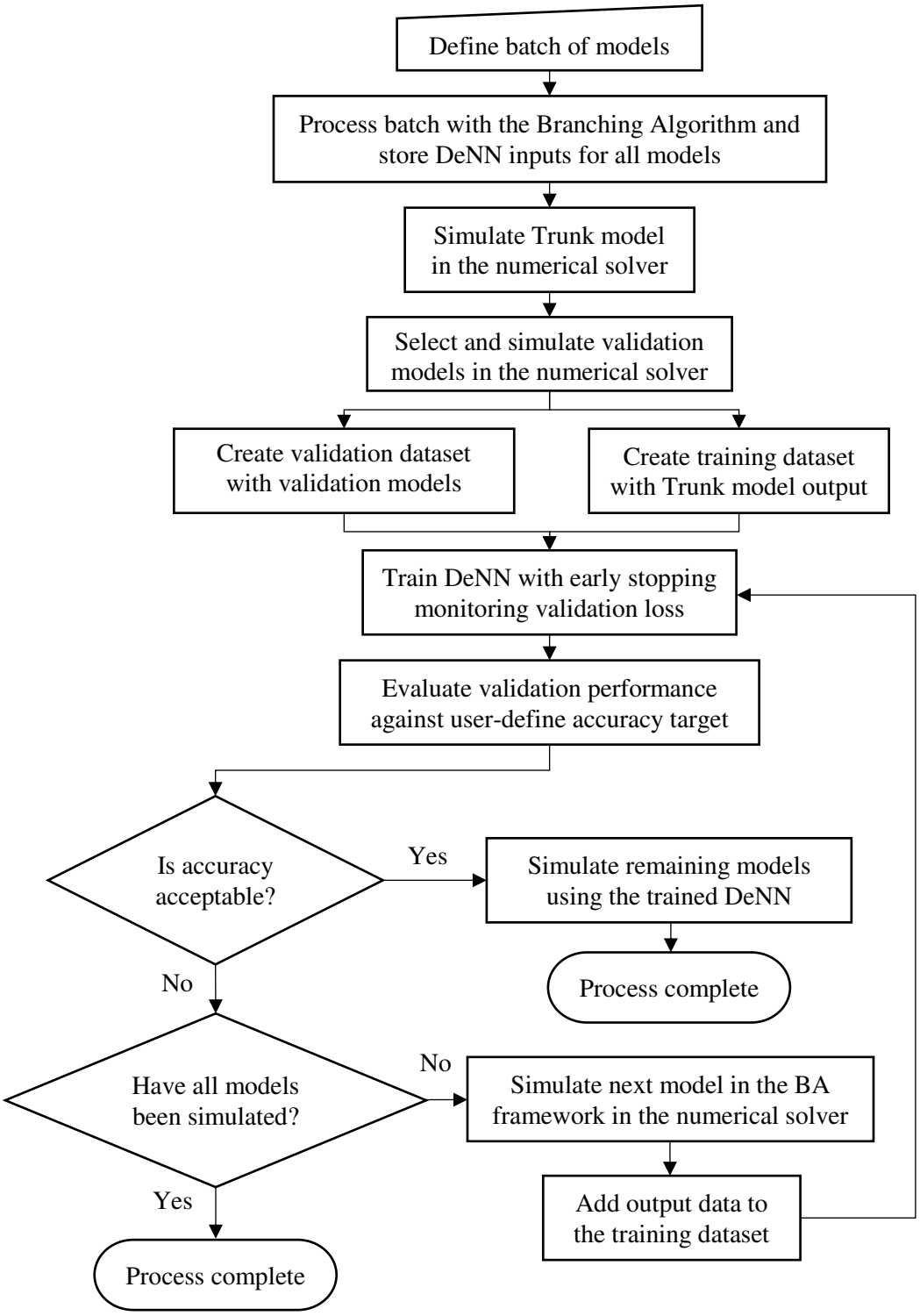


Figure 4. DeNNIS procedure allowing for incremental training.

Incrementally training the DeNN using data from the BA simulation framework combines the computational benefits of both approaches. This Direction-encoded Neural Network in series (DeNNIS) procedure, shown in Figure 4, removes the need to simulate entire domains using computational fluid dynamics (CFD) if the performance of the tool surpasses a user-defined target before additional training data is required. The performance comparison is made by evaluating a validation dataset that is formed using values that are independent of the training process and the target performance could be related to various parameters including the average absolute error, the average percentage error, or the overall correlation coefficient of the validation data.

Once the required performance is achieved by the DeNN, it replaces the chosen numerical solver to evaluate the remaining models in the batch. Hence, reducing the required computation time. Furthermore, both the BA and the DeNN require shortest path analysis following discretisation of each domain in the batch. Combining both approaches with DeNNIS training therefore also reduces the number of calculation steps compared to running each method independently, since the ANNs inputs can be extracted and stored during the BA sorting process.

PROBLEM SCENARIO AND SIMULATION

Figure 5 presents the batch of domains that will be used to highlight the benefit of the method introduced in this paper. Four unique geometries and five independent charge locations are included to form a batch of 20 models. All domains are the same size with equivalent charge positions.

In each geometry, the charge positioned at (11, 1) corresponds to the first model of that arrangement. This progresses up the *y* axis to the fifth domain for each geometry corresponding to a charge at (11, 7). The study aims to replicate the detonation of an explosive along a road next to a building entrance being protected by various structural forms.

Table 1. Viper::Blast input parameters

Charge material:	TNT	Mapping:	1D – 3D
Charge mass:	1 kg	CFL, 1D:	0.5
ρ_0:	1600 kg/m^3	CFL, 3D:	0.4
E_0:	4.52×10^6 J/kg	Cell size, 1D:	0.001 m
Detonation model:	Ideal Gas	Cell size, 3D:	0.02 m
Ambient pressure:	101325 Pa	Termination time:	50 ms
Ambient temperature:	288 K		

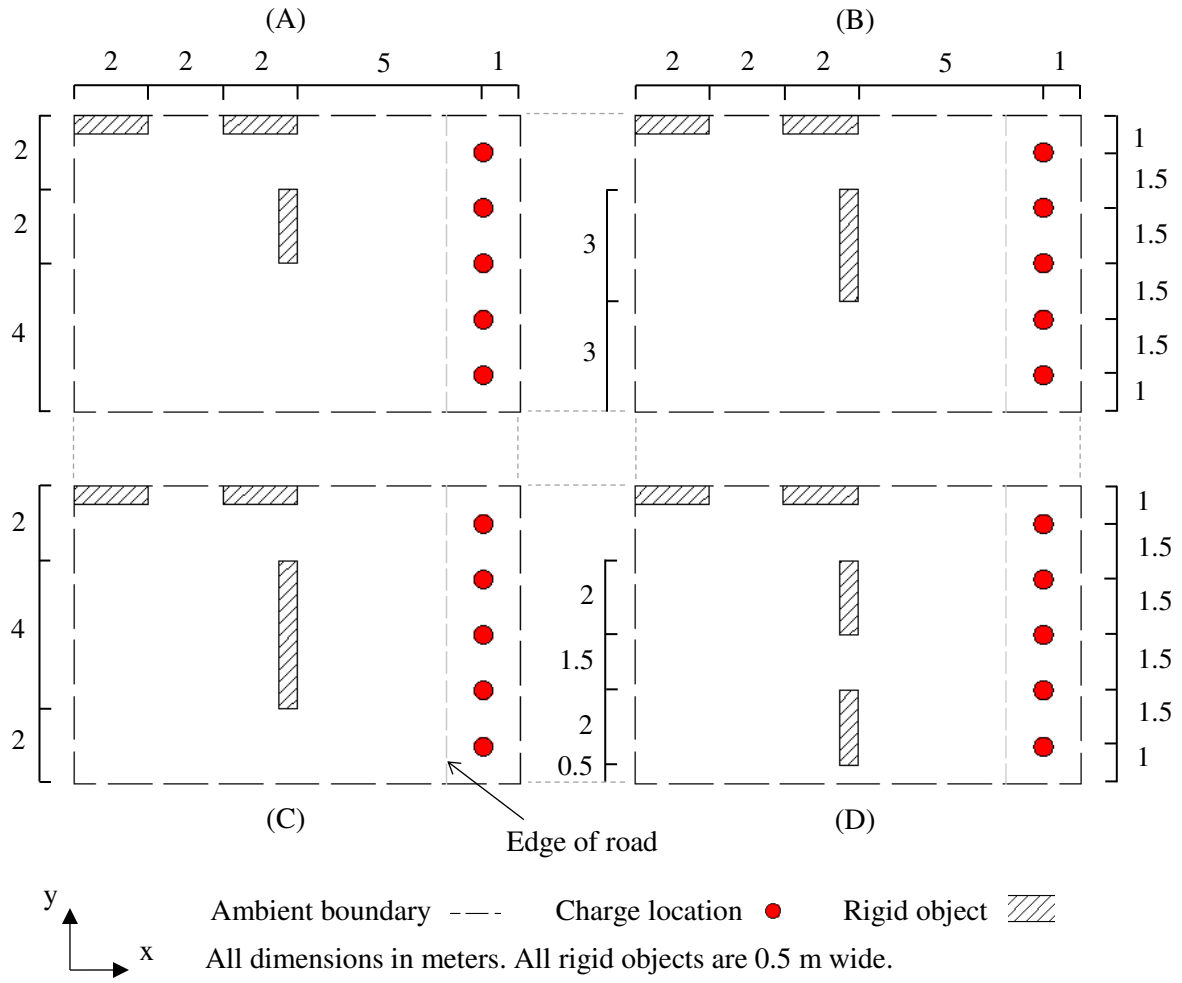


Figure 5. Four geometries and five independent charge locations creating 20 models in the batch. Model numbers are as follows:
domain A (1 – 5), domain B (6 – 10), domain C (11 – 15), domain D (16 – 20).

Each domain is simulated using Viper::Blast, version 1.20.6a, by a computer utilising a NVIDIA T1000 dedicated graphics card, 16 GB of system RAM and an Intel Core i7-10700 processor with the setup parameters shown in Table 1. This allows for performance comparisons between the DeNN and an established numerical solver.

The DeNN comprises of two independent networks. One used to predict POIs that are unobstructed by an obstacle, hence having a direct line of sight to the charge, and another for obstructed points. The hyperparameters and input variables that were determined partially through a tuning process in [8] are adopted for this study and shown in Table 2.

Table 2. Direction-encoded Neural Network variables

Variable	ANN–1 (Unobstructed points)	ANN–2 (Obstructed points)
Neuron structure	17 – 550 – 900 – 550 – 800 – 1	17 – 800 – 650 – 950 – 600 – 1
Learning rate	0.0170	0.0033
Dropout rate	0.0290	0.0139
Activation function	ReLU (linear at output)	
Loss function	Mean squared error (MSE)	
Optimiser	AdaGrad	
Batch size	100	
Regularisation	L2	
Weight initialiser	Glorot Normal	
Bias initialiser	Zeros	

RESULTS

For this analysis the target performance metric was defined as a mean absolute error (MAE) of 3 kPa for both ANN–1 and ANN–2. Incremental training therefore stopped once both networks forming the DeNN achieve a MAE below 3 kPa when evaluating the validation dataset. Here, this dataset is formed using two (i.e. 10% of the batch) randomly selected models.

Figure 6 presents the computation times required for the two analysis options explored in this study. Exclusively using Viper::Blast requires 41825 seconds, whereas the DeNNIS requires only 8217 seconds to evaluate all 20 models. This corresponds to an 80% saving in computation time. Prepossessing of the Viper::Blast models, and DeNN inputs are omitted as they are dependent on the expertise of the user.

Model 13 was defined as the trunk model that formed the initial training dataset when analysing the batch with the BA. Models 5 and 18 were then selected at random to form the validation dataset. During the DeNNIS training process Model 1 was simulated to expand the training dataset so that the desired performance level could be achieved. This rapid training process, requiring only two sets of outputs from CFD models, highlights how the use of similar domains allows the DeNN to learn the wave coalescence effects that are present in the batch with less computational effort when compared to using a dataset formed from randomised models.

Furthermore, the reported saving of 80% relates to this batch of 20 models only. For larger batches, the saving is expected to increase significantly, particularly as the total number of unique domains and model complexity increases. For example, now that the DeNN is trained and validated for the problem scenarios

provided in Figure 5, the analysis could be expanded to explore 5 additional charge locations in each of the four geometries. Use of a numerical solver would require each model to be simulated from birth to termination, taking around 2000 seconds per model. Whereas the DeNN can form predictions in less than a minute for each new arrangement.

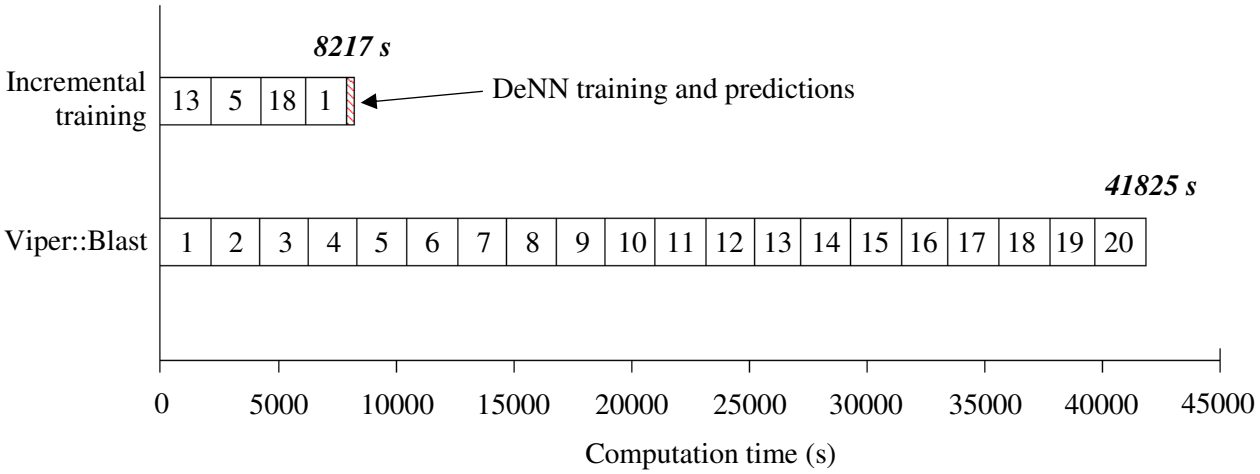


Figure 6. Computation time required by the incremental training process, utilising the DeNNIS, compared to if a numerical model was used to evaluate the entire batch.

Table 3 shows the MAE, Young's correlation coefficient, R_t^2, and the average percentage error for two models and the batch average. These two models relate to the best and worst domain predictions by the DeNN. The batch average metrics are closer to the performance evaluated for model 8 seeing as four models are simulated using CFD in the combined analysis approach, and so the metrics associated to each of these are assumed to have no errors. It should be noted that this assumption, and the performance of the trained DeNN, relies on the accuracy of the chosen numerical solver as training exclusively uses data from simulations using that tool.

Table 3. Example incremental training performance metrics and average from the batch.

Model num.	Sim method	MAE (kPa)	R_t^2	Avr%E
8	DeNN	2.18	0.9978	9.47
20	DeNN	3.49	0.9904	18.05
Batch average	DeNN & Viper::Blast	2.17	0.9963	10.46

Figure 7 provides heat maps of models 8 and 20, showing that the DeNN has qualitatively captured the distribution of peak overpressure in both cases. Model 8's predictions account for shielding and clearing around the central panel, and the pressure build up around the object at (6, 8) is well represented. A similar conclusion can be formed for model 20 in this region, however, channelling is not accurately predicted between the two central panels. Nevertheless this is a known issue of the DeNN [8], and the method is still able to provide good representation of the domain with limited computation effort.

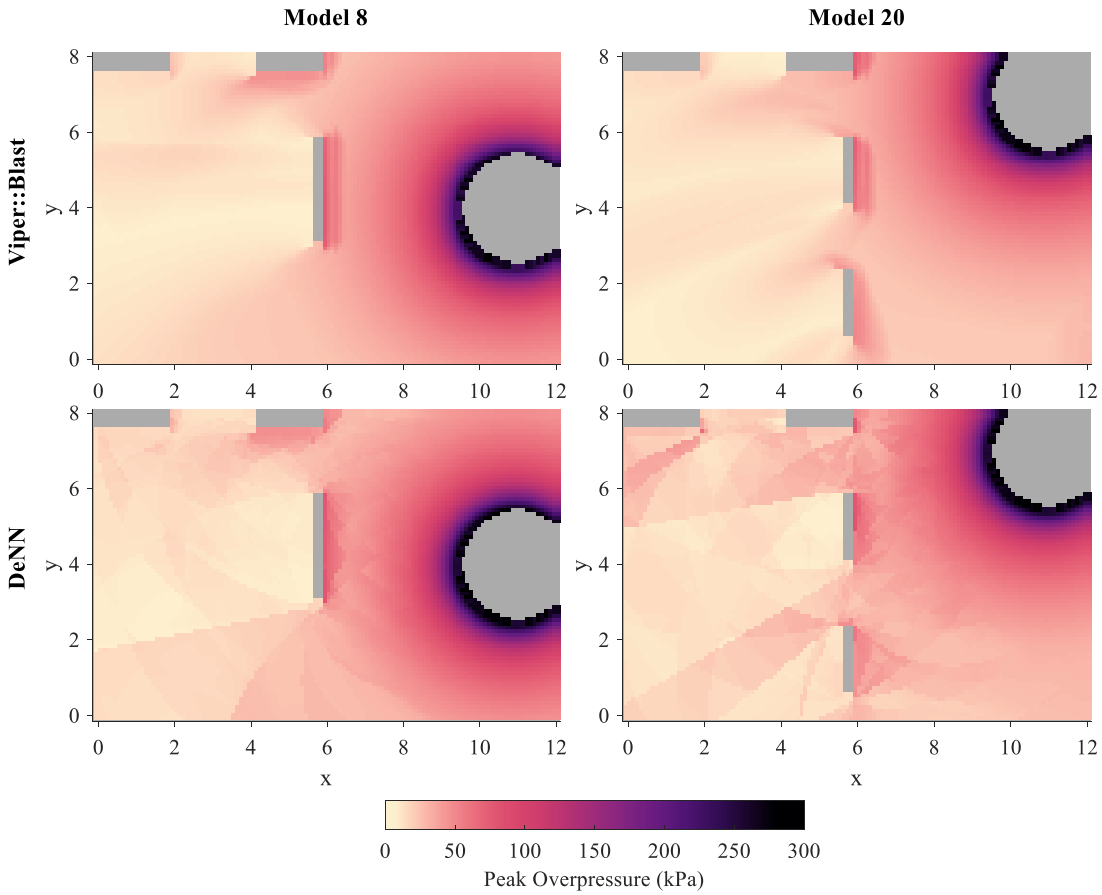


Figure 7. Heat maps showing the peak overpressure distribution for models 8 and 20, simulated using Viper::Blast and the DeNNIS. Grey regions are not predicted, either due to being within rigid obstacles or the 1.5 m exclusion zone around the charge.

CONCLUSIONS

This paper has presented a new methodology that combines the benefits of two previously published approaches to work towards the development of machine learning tools that enable the rapid assessment of various explosive events. It is shown that computational savings of up to 80% are possible, when compared to exclusively using CFD analysis, by incrementally training the Direction-encoded Neural Network in series with the executing the simulation framework produced by the Branching algorithm for a batch of 20 models.

Use of a training dataset related to the scenarios requiring simulation enabled the DeNN to achieve an average absolute error of 2.17 kPa, compared to ~5 kPa when randomised training data was used. Training also required fewer data points seeing as the wave coalescence effects that the DeNN was required to learn were similar among all domains.

This presents a key opportunity for the DeNNIS to be used in probabilistic risk assessments of explosive events, exploring the influence of various uncertainties such as charge location and structural arrangements, with computation times that enable a large number of unique domains to be evaluated.

ACKNOWLEDGMENTS

Adam A Dennis gratefully acknowledges the financial support from the Engineering and Physical Sciences Research Council (EPSRC) Doctoral Training Partnership.

REFERENCES

[1] C. N. Kingery and G. Bulmash, "Airblast parameters from TNT spherical air burst and hemispherical surface blast, Technical Report ARBRL-TR-02555," Aberdeen Proving Ground, Maryland, 1984.

[2] B. Bewick, I. Flood, and Z. Chen, "A neural-network model-based engineering tool for blast wall protection of structures," *Int. J. Prot. Struct.*, vol. 2, no. 2, pp. 159–176, 2011, doi: 10.1260/2041-4196.2.2.159.

[3] A. M. Remennikov and T. A. Rose, "Predicting the effectiveness of blast wall barriers using neural networks," *Int. J. Impact Eng.*, vol. 34, no. 12, pp. 1907–1923, 2007, doi: 10.1016/j.ijimpeng.2006.11.003.

[4] A. M. Remennikov and P. A. Mendis, "Prediction of airblast loads in complex environments using artificial neural networks," *WIT Trans. Built Environ.*, vol. 87, pp. 269–278, 2006, doi: 10.2495/SU060271.

[5] A. A. Dennis, J. J. Pannell, D. J. Smyl, and S. E. Rigby, "Prediction of blast loading in an internal environment using artificial neural networks," *Int. J. Prot. Struct.*, vol. 12, no. 3, pp. 287–314, 2021, doi: 10.1177/2041419620970570.

[6] J. J. Pannell, S. E. Rigby, and G. Panoutsos, "Application of transfer learning for the prediction of blast impulse," *Int. J. Prot. Struct.*, vol. 0, no. 0, pp. 1–21, 2022, doi: 10.1177/20414196221096699.

[7] J. J. Pannell, S. E. Rigby, and G. Panoutsos, "Physics-informed regularisation procedure in neural networks: An application in blast protection engineering," *Int. J. Prot. Struct.*, 2022, doi: 10.1177/20414196211073501.

[8] A. A. Dennis and S. E. Rigby, "The Direction-encoded Neural Network: A machine learning approach to rapidly predict blast loading in obstructed environments," *Submitted for publication, Int. J. Prot. Struct.*, 2023.

[9] L. Chen, H. Hassan, T. N. Tallman, S. S. Huang, and D. Smyl, "Predicting strain and stress fields in self-sensing nanocomposites using deep learned electrical tomography," *Smart Mater. Struct.*, vol. 31, no. 4, 2022, doi: 10.1088/1361-665X/ac585f.

[10] M. Momeni, M. A. Hadianfard, C. Bedon, and A. Baghlani, "Damage evaluation of H-section steel columns under impulsive blast loads via gene expression programming," *Eng. Struct.*, vol. 219, no. June, 2020, doi: 10.1016/j.engstruct.2020.110909.

[11] A. A. Dennis, D. J. Smyl, C. G. Stirling, and S. E. Rigby, "A branching algorithm to reduce computational time of batch models: Application for blast analyses," *Int. J. Prot. Struct.*, pp. 1–33, 2022, doi: 10.1177/20414196221085720.