Science and Technology Facilities Council

Technical Report
DL-TR-2021-001

How surrogate models can enable integration of experiments, big data, modelling and simulation

J Taylor, J Castagna, L Mason, V Alexandrov

February 2021

DOI: 10.5286/dltr.2021001

ISSN 1362-0207

How surrogate models can enable integration of experiments, big data, modelling and simulation

Jack Taylor, Jony Castagna, Luke Mason and Vassil Alexandrov

UKRI-STFC Hartree Centre, Daresbury Laboratory, UK
jack.a.taylor@stfc.ac.uk, jony.castagna@stfc.ac.uk,
WWW home page: https://www.hartree.stfc.ac.uk/Pages/home.aspx

Abstract. AI surrogate models for science, based on Deep Neural Networks (DNN), are becoming increasingly popular. This is especially the case for generative models such as Autoencoders and Generative Adversarial Networks (GANs), which are showing promising results and exciting opportunities in several fields of science.
We present here an overview of the current state of the art for surrogate modelling, starting with a broad view across several fields, ranging from particle methods for Molecular Dynamics to continuum solvers for Computational Fluid Dynamics. We then focus our attention to applications in Nuclear Fusion through plasma-physics simulation, presenting the latest results and achievements
The paper not only outlines current limitations but also proposes the next steps in the research, in order to connect the high accuracy prediction power of the DNN together with scientific understanding and will detail the current state of their scalability on large computing resources. The main intention is to build a road map that bridges the traditional science and the novel *big data* in order to have solid and reliable surrogate models.

Keywords: Deep Learning, Surrogate Model, Nuclear Fusion and High-performance Computing

1 Introduction

The recent fusion of big data methods and traditional scientific modelling has led to a new approach for investigating scientific problems. These methods are applied to find the solution of physical problems that are usually described by partial differential equations, based only on the available experimental and numerical data, rather than solving the equations directly. These surrogate models can find solutions of similar accuracy with no or only partial knowledge of the physics of the problem and are often several orders of magnitudes faster.

This advent is mainly driven by the rise of Deep Learning. The fundamentals of these approaches were developed over 50 years ago but has seen a kind of "big bang" in the 2010s due to the introduction of the Deep Neural Network (DNN), the growing amount of data generated through social media and the introduction of GPUs as programmable hardware for DNN training. Despite initial applications being developed for the likes of image processing and natural language processing tasks their application to scientific problems is relatively straightforward, as the same concepts apply, feed a large amount of data to the DNN, then train and tune the model until it produces satisfactory estimates. In principle no knowledge of the physics is required a priori, which makes the approach attractive for traditional computer scientists, but also the many data scientists working on all social fields.

However, there is no free lunch: a high accuracy in prediction is often associated with a poor understanding of the problem, which gives rise to fundamental questions about the robustness and reliability of the solution. In this paper we will give a resume of the current state of art surrogate models, how they can integrate numerical methods and big data sources and the advantages and drawbacks of the approaches. We will focus particularly on generative DNN models that have seen success in this field, applied to different scientific problems like turbulent flows and instabilities occurring in the controlled nuclear fusion reactors. The intention is not to give an exhaustive list of surrogate models, but rather to create a starting point towards a better understanding on their reliability and what problems need to be tackled in the future to create robust and reliable surrogate models.

The paper is divided as follows: the next section II introduces the fundamentals of DNN and details about their architectures. Section III introduces a classification based on reduced parameters vs full PDEs and physics-constrained vs unconstrained models. Several examples, ranging from *Molecular Dynamic* (MD) to *Computational Fluid Dynamics* (CFD), are given. We then present the latest surrogate alternative in confined nuclear fusion in section IV and we propose a way forward to gradually build higher resolution models. In section V we mention about the role of High Performance Computing (HPC) and its challenges and finally we draw conclusions in section VI.

2 Fundamentals and common DNN used for surrogate models

Despite their simple mathematical structure, NN are still considered a kind of *black box* from their behaviour point of view. It is not yet clear what features are extracted in the different layers and how to build a new network. The probably most important theorem has been given in 1989 by Hornick et al. [1] proving that any continuous function can be learned by even a single hidden layer provided it to have a wide enough (number of neurons) size. In 2017 Rolnick and Tegmark [2] found that by increasing depth and decreasing width, you can perform the same functions with exponentially fewer neurons. However, in 2018 Johnson [3] proved that at a certain point, no amount of depth can compensate for a lack of width. See Foundations built for a general theory of neural networks for a nice introduction on the subject.

Modern NN approaches stem from the introduction of the *perceptron* [4], a linear classifier model inspired by the biological neuron, which accepts an input vector of sensory data x, a weight coefficient vector w and an additional bias term b to offset the linear model. The perceptron produces a prediction by applying a Heaviside activation function to the linear combination of these terms $y = \phi(x^T w)$ the weights are then iteratively adjusted to minimise the loss function (error) of the model, with adjustments scaled with a step-size parameter $\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \eta[t(n) - y(n)]\boldsymbol{x}(n)$ until convergence is satisfied.

The perceptron is proven to converge for linearly separable solutions and spurred the development of the multi-layered perceptron (MLP) design consisting of multiple fully-connected layers of perceptrons with an individual weight w_{ji} for each connection and replaced the activation function with a continuous sigmoidal function, allowing the network to find solutions to non-linear problems and allowed the training of the individual weights due to the differentiable activation function. This allows the network to be trained with the backpropagation algorithm [5] which applies the concepts of gradient descent, minimising the loss function by iteratively adjusting the weights in the direction of the steepest descent of the loss function, this is done by computing gradients of the loss function $w.r.t$ each weight in model $\eta \frac{\partial L}{\partial w_{1_1}}$ by backwards propagating the error through the network, then applying the chain rule to calculate the gradients for all the weights in the network.

DNNs are a further advancement of the MLP, making use of many non-linear hidden layers, allowing the network to build more abstract representations of the input patterns, resulting in the model being able to make good predictions for highly complex and multi-domain problems. The fundamental theory and application of backpropagation remains the same but there have been many new neural network designs that have been introduced and developed with the popularity of field.

Convolutional neural networks (CNNs) [6] are one such architecture, primarily applied in the image classification domain. CNNs process pixel data, taking an input matrix with a height, width and depth representation. These networks have minimal need for feature engineering as they can extract features from the input data. The CNN does this by applying matrix convolution operations on the input image through a kernel matrix with its own trainable weights which can learn to extract the relevant features from the image for inference.

Recurrent Neural Networks (RNN) are another network, where the nodes form a directed graph in a sequence, allowing the network to interpret temporal behaviour, by feeding the outputs of the individual units back into the input stage of the next time step, making new predictions based on both current weights and the temporal sequence of states. This has been used extensively in both time-series forecasting and simulation environments. LSTM models [7] advanced on this method further, introducing *gating mechanisms* for better gradient control to preserve long-range dependencies of the sequence, avoiding the notorious *vanishing gradient* [8] issues of the RNN.

Autoencoders (AE) are unsupervised networks in which the target values are equivalent to the input $t^i = x^i$. The AE attempts to learn an approximation of the identity function from some constrained, usually low-dimensional form, where the loss function is the reconstruction loss of the identity function. The network has two components, the *encoder* and *decoder* (see Fig. 2). The encoder part of the network compresses the input patterns into a latent-space representation in the hidden units and the decoder will reconstruct the original inputs from this latent space representation. There has also been considerable research into using AEs as generative models through *variational autoencoders* [9] by sampling from the latent space of the encoder, with the decoder purposed to sample new data points with the latent distributions as a prior.

Generative Adversarial Networks (GANs) are another branch of generative DNN with the ability to generate new samples of data that are of the same type of the training set through *adversarial training*. In this model two neural networks are being trained: the generative model G that captures the data distribution in attempt to create new realistic data similar to the original data distribution and a discriminative model D which estimates the conditional probability that a sample is from the original dataset. This algorithm takes a batch of noise samples $z(1), ..., z(m)$ from a prior distribution $p_g(z)$ and a batch of samples of the original data $x(1), ..., x(m)$ from the distribution $p_d(x)$. D model then makes a prediction for each generated z and real training example x and updates it's weights through backpropagation to improve the success at recognising the fake samples, while the generator is instead trained to penalise the model if its samples are classified as fake, resulting in a generative model that can generate realistic samples of data. See Fig. 3 for an integrated RNN-GAN architecture.

An important theoretical aspect to consider in creating surrogate models from NNs is the well explained difference between accuracy and understanding by Sanjuán [10]. Real systems are often chaotic with several degrees of non linearity. While in some cases the NN could be trained on a dataset and allowed to extrapolate outside those values, in other cases the strong divergence could easily bring to completely wrong answer and strongly affect its robustness. Moreover, even if an high accuracy is achieved, this does not mean that we understood the phenomena. This is due to the still black box understanding of current NN which limits our interpretation of the several layers.

Moreover, dealing with data and model uncertainties and having comprehensive Uncertainty Quantification approaches for those is very important. Data-related uncertainties are usually common for many practical problems, and in particular many problems discussed are multi-scale and are modelled by PDE's or systems of PDE's, and SPDEs. Often Multi-Level Monte Carlo methods are employed in this case since MLMC ([11], [12], [13]) is a highly efficient variance reduction method. Examples of applying Machine Learning in combination with MLMC in case of environmental modelling can be found in Kani et al. [14]. while example of using MLMC for UQ is used in many areas, for example, in structural engineering [15], in oil fields modelling by SPDE's using MLMC [16].

The basic idea is to accelerate the standard (single level) Monte Carlo algorithm by carefully defining coarser levels, representing the expected value of the quantity of interest at the fine level in identical form as telescopic sum, equal to the expected value at the coarsest level, plus corrections between levels. Thus the main idea being employed is to perform many calculations at coarser levels (which are cheap), and fewer calculations at the finest level(s) (which are expensive). In respect to integrating this approach with surrogate models, the coarser levels can be defined in different ways, e.g, as approximate and easier to solve models, while the surrogate model can be used replacing the finest level(s). In more detail, MLMC will define the levels, will generate realizations for the parameters, and will call the surrogate model for particular applications to solve for each realization of the coefficients at corresponding level(s).

3 Constrained vs unconstrained

We present here a classification of the surrogate models according to 3 criteria (see Fig. 1) the type of equations solved, i.e. full Partial Differential Equations (PDEs) vs Reduced Parameter; 2) the use of physics constraints to enforce the PDEs or conservation laws like mass, energy and momentum vs fully unconstrained NN; 3) the type of constraint which can be soft or hard, according if the equations works as a regularization of the loss function or are actually enforced by the NN.

The constraint usually reduces the flexibility during the design of the NN and then its abstraction power as mentioned in section II. On the other side, fully unconstrained NN can take advantage of the most complex state of art architectures and achieve higher results in terms of accuracy. However, they cannot enforce conservation laws and the understanding of their prediction capability is undermined. Reduced parameters are usually unconstrained, however we will see an examples where a soft constraint has been applied [17].

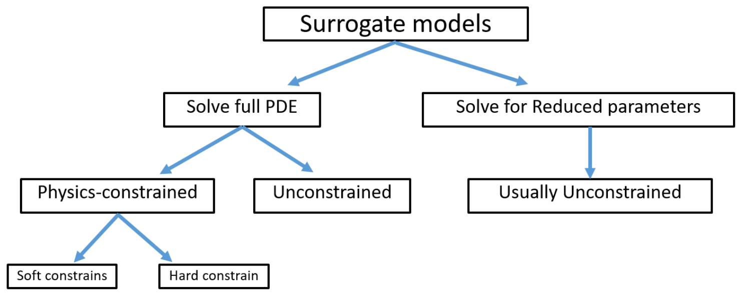

Fig. 1. Type of surrogate models.

3.1 Physics-constrained

A good starting point for Physics-constrained NN has been given by Raissi et al. [18] with the introduction of Physics-Informed Neural Networks (PINNs). The idea is as follows. Given the equation:

$$u_t + \mathcal{N}[u] = 0 \tag{1}$$

we define a function $f(t, x)$ as left-hand side of eqn 1 and approximate $u(t, x)$ via a DNN. The implementation is trivial and a high accuracy is achieved in solving notorious PDEs: *Burgers' equation*, *Shrödingher's equation* and *Allen-Cahn equation*. To note that the DNN used in the paper is rather small and a merely done of a series of fully connected layers. A main drawback of these approaches is in the large amount of sampling points needed for more complex 3D-systems like the Navier-Stokes equations describing fluid dynamics. Moreover, the conservation law of physics happens in a so called *weak* form, in the sense that the solution is driven by the loss function towards values satisfying Eqn. 1, but it does not guarantee that conservation laws are completely satisfied.

A *stronger* form, reported as *hard* constraint, has been provided by Mohan et el. [19] via Embedded-informed NN (see Fig. 3). Here the equations are actually enforced in the solution process via the NN layers itself. The kernels of the convolutional layers downstream the decoder section calculate the derivative components of the PDE. During the training, this constraint, forces the encoder to learn the main features of only those solutions which satisfy the imposed equations. Moreover, boundary conditions can be applied in the same way allowing the model to correctly reproduce also those values far from bulk of the physical system. The NN, named Physics Embedded Convolutional Autoencoder (PhyCAE) has been trained with Direct Numerical Simualtion data of a 3D homogeneous isotropic turbulent flow. Results show very good agreement for the large scales, while at small values the loss of accuracy of the autoencoder creates an acceptable departure. More important, results shows a better satisfaction of the conservation of mass compared to the same autoencoder without the physics-constraints.

3.2 Fully unconstrained

When no physics is enforced into the neural network we have a fully unconstrained model. The main advantage is of course that no knowledge at priori is required. For experimental data this approach would allow to replicate the real system without any tuning ad hoc of parameters of a mathematical representation. A good example are the Reynolds Average Navier-Stokes (RANS) equations where an unique model for all possible flow regimes has never been found. However, on the drawback side, fully unconstrained surrogate models can seriously undermine the confidence in the solution provided, especially in systems where conservation laws apply. Ideally, we would like a mathematical correlation between the feature extracted from the NN and the solution itself such to verify

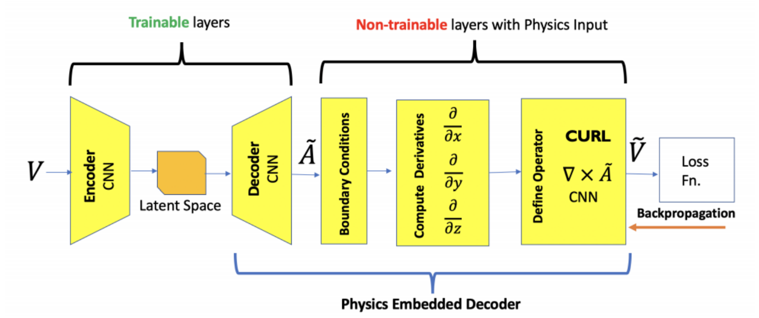

Fig. 2. Mohan et al. [19]: Physics Embedded Convolutional Autoencoder.

that *hidden* constraints are actually enforced from the architecture itself. For example, if the training of a CNN via incompressible velocity flow fields would end up in a series of kernels which enforce the velocity free-divergence, then the mass conservation would be guaranteed. This is of course a naive wish, but preliminary results look promising. Let's see some.

A good example of fully unconstrained model has been given by Breen et al. [20]. In their work their are able to solve the famous three body problem using a simple fully connected NN trained with a time series of data obtained from a very accurate numerical solver (Brutus). The basic idea is very simple: we feed in the NN the position of a particle positions at time t and we give as correct output the values of the second planet at the same time t. The third planet position is obtained via conservation of the center of mass. The importance of this work is in to the strong non linearity of the problem, solved in the difficult scenario of equal masses. Predictions from the NN are in good agreements with the numerical solutions. However, sometimes the error on the energy can have some spikes when the particles are very close to each other. An improved solution is obtained adding a soft physics-constraint on the particle position via a projection layer. The intention of this layer is to reduce the energy imbalance minimizing following problem:

$$Er(x,\nu)^2 + \gamma_1 D_x(x)^2 + \gamma_2 D_\nu(\nu)^2 \tag{2}$$

where $Er(x,\nu)$ is the energy error, while the other terms have the intention to penalize deviations from the initial values of position x and velocity ν. However, it does not enforce the conservation of energy, nor momentum, as a hard constraint would do.

The natural extension of the three many body problem is to larger systems like those used in MD. An excellent work has been presented by Kadupitiya et al. [21] where they used a RNN to simulate an MD system with 16 particles and different force potentials. The algorithm is very robust and allows to use a 4000x larger time step than that used in the Verlet integrator to the generate the training dataset. In the paper author's view, the RNN seems to really have learned the Newton equations of motion as proven by the excellent energy conservation, without this being enforced in any layer of the NN.

Another good example of fully unconstrained NN has been recently given by Kim and Lee [17]. They used a RNN-GAN architecture to create instantaneous of turbulent flows at different Reynolds number after training it with a time series produced via Direct Numerical Simulation (see Fig. 3). Results are very good in agreement in both instantaneous and mean flow field values. However, better results are achieved introducing a soft constraint on the statistical variation enforced into the loss function of the generator (L_g) and RNN (L_{RNN}):

$$L_G = ... + MSE_G + MSE_{RNN-G} \tag{3}$$

$$L_{RNN} = ... + MSE_{RNN-G} + MSE_G \tag{4}$$

where MSE_G and MSE_{RNN-G} are Mean Square Error for the GAN Generator and the RNN, respectively.

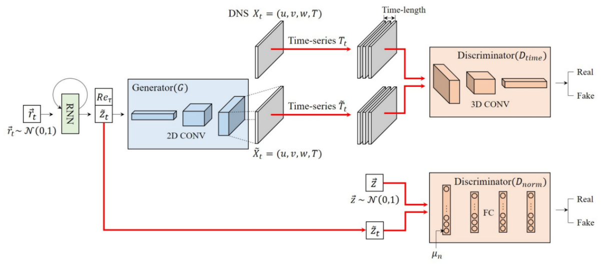
DNS $X_t = (u, v, w, T)$
Time-length
Time-series T_t
Discriminator(D_{time})
Real
Fake
3D CONV
Generator(G)
Re_τ
$\tilde{z}_t$
$\vec{r}_t$
RNN
$\vec{r}_t \sim \mathcal{N}(0,1)$
2D CONV
Time-series $\tilde{T}_t$
$\tilde{X}_t = (u, v, w, T)$
$\vec{z}$
$\vec{z} \sim \mathcal{N}(0,1)$
Discriminator(D_{norm})
FC
$\tilde{z}_t$
μ_n
Real
Fake

Fig. 3. Kim and Lee [22]: RNN-GAN used to generate inflow turbulent flow fields at different Reynolds numbers.

3.3 Examples of surrogate models on Reduced Parameters

A strong mathematical formulation of a reduced parameters example has been given by Lye et al. [23] where the NN learns the mapping from a series of input parameters to observable output using few training samples. For example, given a set of point values from a flow field around and airfoil, they are able to calculate the lift on the airfoil itself. Combined with a Monte Carlo (MC) and Quasi Monte Carlo methods (QMC), this approach allows to efficiently compute uncertainty propagation which compensates for the lack of physical constraints. The speedup achieved is between one and two order of magnitude when compared to the baseline MC and QMC methods.

Another interesting example of constrained surrogate model on reduced parameters has been given by Kim et al. [17] with their *DeepFluid* network. They start from a set of reduced parameters to train a generative CNN using a set of fluid velocity fields from numerical simulations. To conserve the overall mass, they enforce the divergency-free velocity field at all times in the loss function. This is referenced as *stream function* and it is defined as:

$$L_G(\mathbf{c}) = ||\mathbf{u_c} - \nabla \times G(\mathbf{c})|| \tag{5}$$

where $\mathbf{u_c}$ is a sample from the training data and $G(\mathbf{c})$ is the network output ($\mathbf{c}$ indicates a set of parameters, like position and width of the inlet). This constraint is also a *weak form* as mentioned above and does not rigorously enforce boundary conditions. However, results are pretty good in capturing complex turbulent effects, like vorticity in smoke plumes and dam break simulations. Moreover, beside achieving the 700x speedup compared to the CPU solver used to generate the training data, DeepFluid is able to extrapolate, with plausible results, up to a 10% outside the training parameters range.

4 Surrogate models in nuclear fusion

Nuclear fusion is an extremely complex process due to the multi-physics and multi-scale phenomena occurring inside a nuclear reactor like the Tokamak under construction in the ITER project. The design of such a reactor based only first principles is not currently (nor in the near future) feasible.

One of main complexity comes from the turbulence in the plasma which affects the core transport fluxes reducing its temperature and then the capability of ignite the fusion process. If a continuous approach is taken, the plasma behaviour is well described by the magneto-hydrodynamic, which are a combination of the Navier-Stokes equations for fluid dynamic and Maxwell's equations for electromagnetic fields. These are difficult to solve due to their strong coupling and non linearity. On the other side, a particle description is still unfeasible even on next generation of Exascale supercomputers due to the high number of particles involved in a Tokamak ($\sim 10^{22}$) despite the low density of the plasma inside the reactor [24].

4.1 Latest developments

Surrogate model can help at different levels and in different approaches. For example, in the recent work of Ma et al. [25] the Landau closure 1D equation is solved using three different type of Neural Networks: 1) a MLP with only two hidden layers; 2) a CNN with four convolution-pooling layers; 3) a discrete Fourier transform neural network (DFT) which does not contain hidden layers. Results are in good agreement and extension to 2D and 3D seems feasible for the CNN type as the computational cost scales linearly with the number of neurons (while is quadratic with MLP and DFT).

In the work of Meneghini [26], two main solvers of the so called Whole Device Modelling (WDM) are replaced by MLP. Despite its simplicity, the NN is able to simplify the prediction of the turbulent transport fluxes in the core of the plasma satisfying that delicate balance between accuracy and speed required in the WDM concept itself. This is a good example of reduced parameters without enforced physics-constraints.

On a completely different approach, a series of machine learning models have been developed since 1990 on the prediction of disruptive instabilities in controlled fusion plasma ([27],[28]). A recent work using deep learning has been presented by Kates-Harbeck et al. [29]. This is a case of reduced parameters modelling where the signal from different sensors have been used to predict the time evolution of plasma instabilities and then be able to take counter acting measures for an active control mode. The NN used consists essentially of two parts: *a)* a 1D feature extractor and *b)* a LSTM for concatenating the time series data. There are no constraints applied to the output results. The software can be applied to the International Thermonuclear Experimental Reactor (ITER), allowing to detect disruptions within $30ms$ time frame and potentially save the rector from expensive disruptions.

A good example of a reduced parameters model for describing the temporal plasma evolution and neutral characteristics at the edge of a Tokamak has been provided by Gopakumar and Samaddar [30]. They developed a multi CNN with two inputs and two outputs to combine the effects from the plasma behaviour and the neutral particles (see Fig. 4). The focus on the temporal evolution of 7 parameters (5 for the plasma and 2 for the neutral) spread around the poloidal region of the Tokamak. Despite no constraints on the output results, they achieve good results in terms of accuracy (0.01% difference from the numerical solver SOLPS used to generate the training data) and generalization.

Another case of multi input reduced parameters model, but with a soft physics-constrained, to predict the core Tokamak transport heat and particle fluxes named QLKNN (the name comes from the quasilinear gyrokinetic transport model QuaLiKiz NN). It consists of 3 separate feed forward NN, each for a well chosen training dataset, and then combined together (see Fig. 5). The constraints are applied by adding them into the cost function: the sharp instability thresholds, zero flux in the region where no instabilities are predicted, and identical transport flux thresholds for all transport channels.

4.2 A full model for the core plasma turbulence

Despite its importance, we could not find in the literature a surrogate model for the full PDEs governing the plasma core turbulence. This could be due to two main difficulties: 1) the problem itself in formalize a NN with soft or, even more difficult, hard constraints due the complex mathematical structure of the magneto-hydrodynamic equations; 2) the generation of a high fidelity training dataset even on a small sub-geometry of the full Tokamak. On the other side, particle in cell methods can be used as a starting point to build a complete surrogate model, being a fundamental integrator of the Newton's equation under

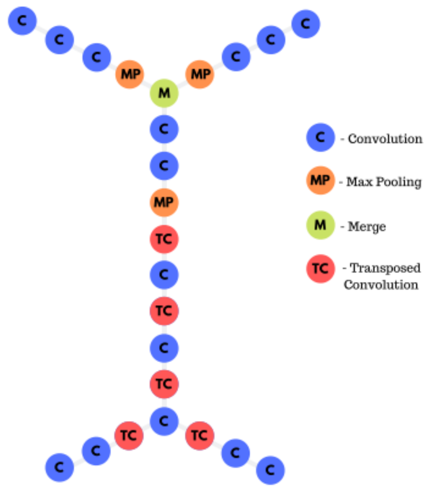

Fig. 4. Gopakumar and Samaddar [30]: NN combining two different physics phenomena.

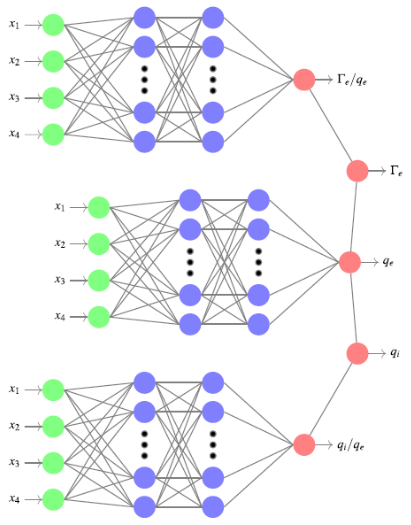

Fig. 5. van de Plassche et al. [31]: physics-constrained multi input NN for particle heat and particle flux estimation.

a Coulomb potential field. The scaling over large systems would recover the above magneto-hydrodynamic equations. A possible approach could then be as follows: we train a NN with particle positions at different time steps t undergoing a simple linear potential and no electromagnetic fields. We iterate on the model until satisfactory results are achieved. If not, we add a physics-constraint and reiterate on the NN model. In a similar manner, the electrostatic field first and then a full electromagnetic potential can be added. We then move to a coarser model where group of particles are replaced by coarser beads, following the same procedure applied in the Dissipative Particle Dynamics (DPD), and then train those beads to replicate the overall behaviour of the smaller beads. We keep scaling until the larger structures of the turbulence are represented with a reasonable number of particles keeping the inter medium/smaller representation for the small scales lengths. The intention is to achieve the best NN for accuracy using the least number of physical constraints. The procedure, which can be generalized also to different physics, is resumed in the Fig. 6 with red comments referred to the plasma physics problem.

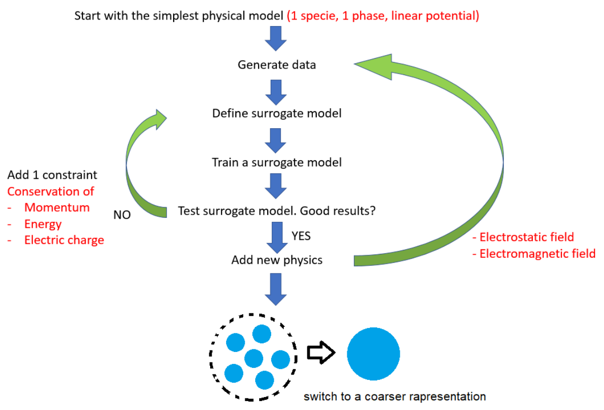

Fig. 6. Building a surrogate model: in red the specifications for a plasma model.

5 Surrogate models and HPC

The forward and backwards pass of the DNN breaks down into matrix multiply-and-accumulate (MAC) operations. MAC are easily parallelised on accelerators [32] (primarily GPUs) and recent hardware innovations, spurred by deep learning, have accelerated the operations further. These devices have become increasingly complex but the size of the DNN, thus the abstraction capabilities, are restricted by the amount of available memory on the device. Scientific data usually comes from extremely large-scale simulations and experiments with a huge number of parameters; therefore, the model needs to be powerful enough to capture the relevant information of the domain. The need for larger models also comes with the requirement of the distributed and parallel training of the networks on modern supercomputers.

DNNs typically involve strict data dependencies between each layer of the network, restricting the inter-layer parallelisation strategies. Therefore one approach for parallelisation is *model parallelism*: which involves dividing the actual network across the nodes of a cluster, with the same batch of data but will perform independent weight updates and computation, depending on the activation functions of the neurons they are distributed. This form of parallelism conserves memory as the full network and parameters are not stored on the GPU, allowing the training of large-scale networks, but conversely require an *all-to-all communication* step at each layer for synchronisation, which can lead to load-balancing issues if the work is not equal. *Data parallelism* approaches instead partition the inputs of the network over the cluster, reducing the communication needs by only synchronising the weights and parameters at the end of a training iteration, but this approach requires a copy of the model for each node, hindering the available memory for each. Often a combination of these methods which are usually designed for individual cases. *Pipelining* is another strategy used for combining these methods which introduces softer dependencies between the layers, allowing for the overlapping of computation between the layers to compute partial results, removing the issue of idle processing time. Google Brain developed GPipe [33] which applied this in a generalised form and achieved state-of-the-art performance on both the ImageNet and CIFAR-10 benchmarks, by estimating the computational cost for each unit given their activation functions and training data, allowing the development of a framework to intelligently divide the work-load. These techniques will be very important to consider when developing large-scale models.

Model design is often a trial-and-error approach; iteratively refining the model and hyperparameters to gain greater performance yield or simply using a predefined standard. This approach is inefficient and can hinder performance, especially in the case of large-scale models which may take a long time to complete training. HPC methods can assist in this process and the MENNDL [34] project is one attempt to automate hyperparameter optimisation, deploying a genetic algorithm with optimises a population of networks across a supercomputer, drastically reducing the engineering of hand-tailoring networks. Similar approaches to network structure have been applied through Neural Architecture Search [35], which attempts to find a DNN structure which is suited for a particular problem, this has given promising results, finding well-performing network structures in surrogate models across several domains of science.

6 Conclusion and Future work

We presented a series of state of art surrogate models for different scientific fields ranging from molecular dynamics to nuclear fusion. We classified them according the capability of reproducing the solution of a full set of PDEs or solve a reduced parameter problem. We also distinguish between physics-constrained models, which can be applied to improve the solution accuracy and its reliability. Such constraints can be classified in soft and hard and, usually, reduced parameter models do not have it.

In the author's knowledge, in nuclear fusion reseach we do not have yet full PDEs surrogate models. A strategy for developing a such surrogate models has been propsed. The same stragegy can be generalized to different problems. As future work we will apply this procedure to the DL_MESO code, a software package developed at Daresbury Laboratory [36] based on Dissipative Particle Dynamics methodology very similar to particle in cell methods when dealing with electrostatic forces between particles.

7 Acknowledgements

This project has been founded by the UKRI Science and Technology Facilities Council - Hartree Centre department (UK).

References

1. K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, 1989.
2. D. Rolnick and M. Tegmark, "The power of deeper networks for expressing natural functions," 2017.
3. J. Johnson, "Deep, skinny neural networks are not universal approximators," 2018.
4. F. Rosenblatt, "The perceptron—a perceiving and recognizing automaton," *Psychological Review*, 1957.
5. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *No. ICS-8506. California Univ San Diego La Jolla Inst for Cognitive Science*, 1985.
6. Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, 1995.
7. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.
8. S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 1998.
9. A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv*, 2015.
10. M. A. F. Sanjuan, "Artificial intelligence, chaos, prediction and understanding in science," 2020.
11. H. S., "Multilevel monte carlo methods," *Lecture Notes in Computer Science*, 2001.
12. G. M. B., "Multilevel monte carlo methods," *Operations Research*, 2008.
13. ——, "Multilevel monte carlo methods," *Acta Numerica*, 2015.
14. J. K. N. Kani and A. Elsheikh, "A machine learning based hbrid milti-fidelity multi-level monte carlo method for uncertainty quantification," *Front. Environ.*, 2019.

15. P. Blondeel, P. Robbe, C. van hoorickx, G. Lombaert, and S. Vandewalle, "Multi-level monte carlo for uncertainty quantification in structural engineering," *arXiv*, 2018.
16. J. M. O. Iliev and N. Shegunov, "Renormalization based MLMC method for scalar elliptic spde," *Large-Scale Scientific Computing - 11th International Conference*, 2017.
17. B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler, "Deep fluids: A generative network for parameterized fluid simulations," *Computer Graphics Forum*, 2019.
18. M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.
19. A. Mohan, N. Lubbers, D. Livescu, and M. Chertkov, "Embedding hard physical constraints in neural network coarse-graining of 3d turbulence." *arXiv: Computational Physics*, 2020.
20. P. G. Breen, C. N. Foley, T. Boekholt, and S. P. Zwart, "Newton versus the machine: solving the chaotic three-body problem using deep neural networks," *Monthly Notices of the Royal Astronomical Society*, 2020.
21. J. Kadupitiya, G. C. Fox, and V. Jadhao, "Deep learning based integrators for solving newton's equations with large timesteps," 2020.
22. J. Kim and C. Lee, "Deep unsupervised learning of turbulence for inflow generation at various reynolds numbers," *Journal of Computational Physics*, 2020.
23. K. O. Lye, S. Mishra, and D. Ray, "Deep learning observables in computational fluid dynamics," *Journal of Computational Physics*, 2020.
24. J. P. Freidberg, *Plasma Physics and Fusion Energy.* Cambridge University Press, 2007.
25. C. Ma, B. Zhu, X.-Q. Xu, and W. Wang, "Machine learning surrogate models for landau fluid closure," *Physics of Plasmas*, 2020.
26. O. Meneghini, S. P. Smith, P. B. Snyder, G. M. Staebler, J. Candy, E. Belli, L. L. Lao, M. Kostuk, T. C. Luce, T. Luda, J. M. Park, and F. Poli, "Self-consistent core-pedestal transport simulations with neural network accelerated models," *Nuclear Fusion*, 2017.
27. J. G. L. L. J. A. Wroblewski, D., "Tokamak disruption alarm based on a neural network model of the high-beta limit," *Nuclear Fusion*, 1997.
28. P. G. T. C. B. R. J. H. T. C. Windsor, C. G., "A cross-tokamak neural network disruption predictor for the jet and asdex upgrade tokamaks," *Nuclear Fusion*, 2005.
29. J. Kates-Harbeck, A. Svyatkovskiy, and W. Tang, "Predicting disruptive instabilities in controlled fusion plasmas through deep learning," *Nature*, 2019.
30. V. Gopakumar and D. Samaddar, "Image mapping the temporal evolution of edge characteristics in tokamaks using neural networks," *Machine Learning: Science and Technology*, 2020.
31. K. L. van de Plassche, J. Citrin, C. Bourdelle, Y. Camenen, F. J. Casson, V. I. Dagnelie, F. Felici, A. Ho, and S. Van Mulders, "Fast modeling of turbulent transport in fusion plasmas using neural networks," *Physics of Plasmas*, 2020.
32. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, 2012.
33. Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, and Y. Wu, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," *n Advances in Neural Information Processing Systems*, 2019.

34. S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," *In Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, 2015.
35. M. Kasim, D. Watson-Parris, L. Deaconu, S. Oliver, P. Hatfield, D. Froula, G. Gregori, M. Jarvis, S. Khatiwala, J. Korenaga, and J. Topp-Mugglestone, "Up to two billion times acceleration of scientific simulations with deep neural architecture search," *arXiv preprint arXiv:2001.08055*, 2020.
36. M. A. Seaton, R. L. Anderson, S. Metz, and W. Smith, "Dl_meso: highly scalable mesoscale simulations," *Molecular Simulation*, 2013.