BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 69(6), 2021, Article number: e139391
DOI: 10.24425/bpasts.2021.139391

Physics-guided neural networks (PGNNs) to solve differential equations for spatial analysis

Bartłomiej BORZYSZKOWSKI, Karol DAMASZKE, Jakub ROMANKIEWICZ, Marcin ŚWINIARSKI, and Marek MOSZYŃSKI*

Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology, ul. G. Narutowicza 11/12, 80-233 Gdańsk, Poland

Abstract. Numerous examples of physically unjustified neural networks, despite satisfactory performance, generate contradictions with logic and lead to many inaccuracies in the final applications. One of the methods to justify the typical black-box model already at the training stage involves extending its cost function by a relationship directly inspired by the physical formula. This publication explains the concept of Physics-guided neural networks (PGNN), makes an overview of already proposed solutions in the field and describes possibilities of implementing physics-based loss functions for spatial analysis. Our approach shows that the model predictions are not only optimal but also scientifically consistent with domain specific equations. Furthermore, we present two applications of PGNNs and illustrate their advantages in theory by solving Poisson's and Burger's partial differential equations. The proposed formulas describe various real-world processes and have numerous applications in the area of applied mathematics. Eventually, the usage of scientific knowledge contained in the tailored cost functions shows that our methods guarantee physics-consistent results as well as better generalizability of the model compared to classical, artificial neural networks.

Key words: physics-guided neural networks; spatial analysis; differential equations; machine learning.

1. INTRODUCTION

Constant advances in the field of data science lead to massive technological progress in numerous branches of industry and science, beating recent state-of-the-art solutions thanks to improved algorithms, datasets or hardware. Deep learning technologies are certainly one of the favorites to widen scientific discoveries in current applications and therefore enhance their capabilities, which is already taking place [1].

Nevertheless, classical artificial intelligence (AI) techniques such as deep neural networks (DNNs) often fail to achieve the expected generalizability because of different reasons. In some cases the artificial neural network (ANN) model is simply not enough to satisfy the demanding needs of a certain task [2] while in others the training process or its parameters are not optimal to guarantee the best possible results [3]. Popular difficulties are encountered when providing data that is fed to the model during training, validation and test stages. Creation of a complete, consistent and objective dataset is not only difficult but also expensive and time consuming. Problems such as lack of information, bias, interference or annotation errors appear very often and are challenging to overcome [4]. The solution may be data augmentation [5] or different methods of evaluation such as cross-validation [6], which in fact allow to minimize the problems, but do not solve them completely. One of the basic limitations of typical "black-box" models is their clear dependence on data presented to them in the learning process [7]. For example, a typical ANN for a supervised problem can be only as good as the representative quality of the labeled data that it is fed with. This means that even if the model achieves a good performance in terms of accuracy on the test set, it could be still useless in real-world scenarios due to critical contradictions occurring in the given data.

It is very difficult to obtain precise information about the factors on the basis of which the neural network draws conclusions in the decision-making process during its training. Typical "black-box" models adjust the weights between their neurons, thus minimizing the cost function to achieve the best results on presented data. Nevertheless, frequently decisions of AI are based on completely different factors than expected [8]. If the results are correct beyond doubt, it may not matter, but the problem arises when the prediction of a neural network is in conflict with basic laws of physics or logic. Explainable AI (XAI) is a field that explores the decision process of neural network after its training [9]. It allows to apply a comprehensive set of methods such as debugging, visualization, evaluation, testing, explainable decisions, cohort analysis, performance and fairness monitoring. Thanks to these types of techniques, we are able to quite accurately determine on the basis of what factors the network made decisions, and in case of some incompatibilities, modify the dataset and re-train the model [10]. This approach certainly brings many benefits, but XAI cannot be interpreted as any form of preventive interference to a structure of the model that would completely eliminate physical contradictions. It is rather a solution that allows to provide information about the deduction factors, while later combating the effects of incorrect training of the model must be done independently. Despite many advantages, such analysis is often not per-

*e-mail: marmoszy@pg.edu.pl

Manuscript submitted 2021-04-16, revised 2021-08-25, initially accepted for publication 2021-09-28, published in December 2021

formed due to the lack of awareness, its complexity, and time consumption.

The solution that eliminates or significantly reduces the described problems already in the training process is making a cost function of the model dependent on a scientific formula. This approach was utilized in so called physics-guided neural networks (PGNNs) that combine opportunities given by classical ANNs with physics-based knowledge. In this case, we are not only able to ensure that the results presented by the network will comply with a specific dependence (based on the given equation), but also to prevent potential contradictions already at the training stage. Of course, it is also possible to supplement PGNN with XAI analysis, and thus close the system in a loop, which further checks the consistency of results and leads to further minimization of contradictions. In this case, PGNN can be treated as a typical neural network that is a subject to XAI methods. Differences in justification of results in case of PGNN and XAI approach are shown in the Fig. 1.

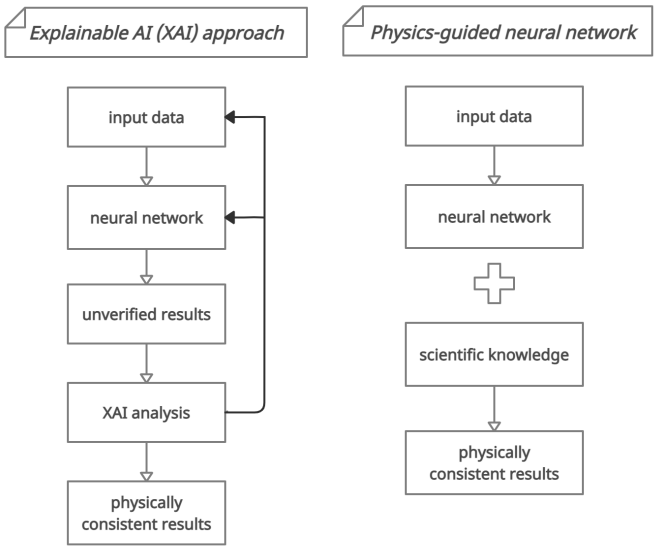

Fig. 1. Two approaches to guarantee the scientific consistency of predictions in DNNs. (Left) Explainable AI – information from the evaluation phase is passed in a closed loop to analyze the results and further improve the training. (Right) PGNN – a combination of a neural network with scientific knowledge that allows to explain the decision process already in the training phase

There are two types of original physics-based models that can be distinguished: (1) scientific equations, formulas and rules that implement relations between variables and thus allow to describe practical scenarios in theory; (2) numerical models of complex physical systems that accurately simulate real-world phenomenon. It is worth to outline that while abilities of physics-based models to correctly interpret relations in scientific terms are greatly beneficial, they are limited in interpretation of data. Therefore, a certain compromise between neural networks that take advantage of data processing as well as physical models that interpret scientific relations is the right direction to allow combining the power of both solutions. The distance between physics-based models and black-box neural networks is schematically depicted in Fig. 2. Both approaches occupy the two extreme ends of knowledge discovery, either relying only on scientific knowledge (physics-based models) or only on the data (neural networks). Their connection is presented as an intersection between both solutions placed in center of the plot.

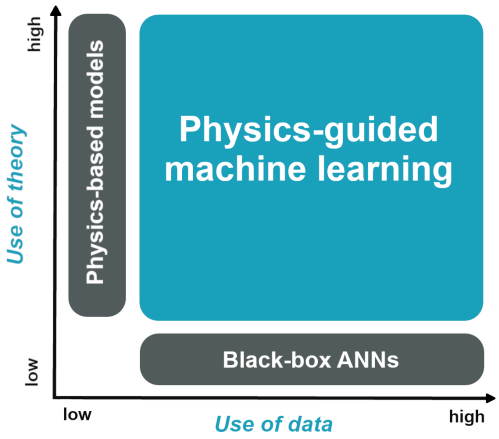

Fig. 2. A schematic representation of PGNNs in the context of other knowledge discovery approaches that either use physics or data. The X-axis measures the use of data while the Y-axis measures the use of scientific knowledge

In this paper, we describe a fundamental concept of PGNN and make an overview of solutions already proposed in the domain. Furthermore, we explain the construction of physics-based cost function and show its numerous benefits that could be utilized in various scenarios. Moreover, there are two primary contributions of this work which introduce applications of PGNNs to make full use of both physics and data in theoretical and real-world examples using spatial analysis. To demonstrate the effectiveness of PGNNs, we consider a Poisson's non-homogeneous second-order linear partial differential equation being an important formula especially in electrostatics or fluid dynamics. In the homogeneous case, it is likewise known as the Laplace's differential equation [11], which is widely used e.g., to describe a behaviour of electric and gravitation potential as well as heat conduction. Second application explains the effectiveness of PGNN in solving a Burger's partial differential equation [12] occurring in various areas of applied mathematics, such as fluid mechanics, nOnlinear acoustics, gas dynamics, and traffic flow.

Our choice of these specific equations was motivated by several factors. First of all, the work aims to illustrate the general mechanisms behind PGNNs, which should be presented on the basis of the most accessible issues possible. Although many partial differential equations cannot be solved analytically due to the computational complexity, their solution can be often approximated by numerical methods. These methods are particularly efficient for low-dimensional problems. Nevertheless, finding an appropriate discretization for a complex space can be sometimes as demanding as solving the actual equation it-

self. Both Poisson's and Burger's equations are characterized by a relatively simple mathematical description and have been widely studied, which makes them accessible even to a reader who does not specialize in applied mathematics. On the other hand, finding their approximate solution can be difficult i.e., when using very few data points, which makes them especially interesting candidates for this study. Moreover, the equations have been successfully used in many areas, therefore their effective solution has real scientific value. Thus, the presented work could ultimately contribute to many applications not only on the theoretical side, but also in practice.

The remainder of this paper is organized as follows. Section 2 presents the overview of research discoveries and applications of PGNNs in numerous domains of science and industry. Section 3 describes the general intuition behind physics-based loss function and explains the potential of PGNNs that can be applied in various novel scenarios. Section 4 introduces a theoretical application of the framework for Poisson's differential equation, including a tailored loss function for the problem, discussion on model architecture and comparison of results between PGNN and ANN. Section 5 presents a corresponding study on Burger's differential equation which might give new insights on connection between data science and physic-based models in practical applications. Finally, Section 6 provides a conclusion and discussion about opportunities for a future work in this domain.

2. BACKGROUND AND RELATED WORK

Intelligent methods of computations are known since the beginning of the second half of the twentieth century when the first research laboratories for this purpose were created. Already in the year 1950, Alan Turing proposed a test of machine's ability to exhibit intelligent behaviour equivalent to, or indistinguishable from, that of a human [13]. In 1955 the term "Artificial Intelligence" was used for the first time with definition proposed by John McCarthy: "the science and engineering of making intelligent machines" [14]. Since this time we observe ongoing innovations of intelligent computing in connection with various technical fields of science which become more and more popular in modern times [15]. Taking into account the period of AI development, it can be concluded that physically-inspired neural networks are a relatively new concept that has been implemented successively over the last few years. One of the fundamental works in this domain was introduced in 2018 by A. Karpatne *et al.* in their paper "Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling" [16]. Since then, the combination of capabilities of neural networks with scientific knowledge constantly finds its utilization in new fields.

When analyzing applications of PGNNs, it is worth emphasizing their basic work requirement resulting from commonly known laws of science. Modification of the model in order to ensure its physical compliance bases on the use of already existing physical relationship, which is integrated into the cost function formula as precisely presented in Section 3.1. Thus, the application of this solution is limited to problems with clearly defined scientific basis, which can be argued by the laws of physics. Consequently, PGNNs find numerous applications in spatial analysis, where they extend possibilities of basic physical models. A comprehensive overview of solutions in the field of integrating physics-based modeling with machine learning was presented in a survey by J. Willard *et al.* [17]. In this chapter we aim to introduce a reader to selected literature items that contributed to finding new applications of PGNNs or suggested improvements to their model architecture.

The above-mentioned example of lake temperature modeling can be considered one of the main inspirations in the field, due to the formalization of PGNN framework and introduction of hybrid models (Section 3.2). Results of this work have been extended by the authors to include modifications in the ANN structure by introducing PGNN concept in recurrent neural networks [18], including particular focus on LSTM layers [19]. Another interesting experiments with architecture of models have been presented by Y. Yang *et al.* [20] and R. Singh *et al.* [21] who show possibilities of physics-informed deep generative models. Furthermore, L. Wang *et al.* [22] use physics-guided autoencoders to estimate states of a power system. These examples show that capabilities of scientifically constrained loss-functions are not limited to any particular network type, but can function in models of varying complexity and layer character.

PGNN applications in spatial problems may occur wherever there is a specific physical dependence. In addition to the previously mentioned examples, it is worth citing the work of N. Muralidhar *et al.* about predicting drag force on particle suspensions in moving fluids [23]. Moreover, PGNNs have been studied in several emerging use cases such as wind-farm power estimation [24], turbulent flow prediction [25], flood forecasting [26] or Fourier Ptychography [27]. Applications of PGNN in the field of medicine and bioinformatics may also seem important e.g., for tomography image processing [28] or cardiac activation mapping [29].

In addition to practical solutions with real-world collected data, related work was carried out on generated examples, using various types of formulas. M. Raissi *et al.* present trainings of PGNNs to solve partial differential equations (PDE) such as Shrödinger and Allen-Cahn Equations [30, 31]. Moreover, Z. Fang *et al.* present numerous approaches to solving PDEs on surface, including 3D generated data [32]. These examples can be interpreted as a broad spectrum of theoretical solutions to which our examples in Sections 4 and 5 refer. In this case, our contribution shows an example of solving a relatively simple Poisson's and Burger's equations, which illustrate the operation of PGNN, and at the same time can be generally applicable in practice or extended to more complex formulas, as presented in the cited works.

In opposition to conventional ANNs, purely bio-inspired computing techniques such as Spiking Neural Networks (SNNs) are being intensively developed [33]. This third generation of artificial intelligence aims to extend machine learning capabilities into areas that correspond to human cognition, such as interpretation and autonomous adaptation. Thanks to their asynchronicity, SNNs are known for their impressive speed and

energy savings when combined with dedicated hardware [34]. They already find promising applications in many areas including robotics [35], physics [36], or language and vision processing [37]. However, due to the non-differential character of spiking activation functions, this technology is still considered to be in its incubation phase. Importantly, a connection of SNNs with physics-based reasoning has not yet been demonstrated, therefore we outline Spiking-PGNNs as a particularly interesting research direction that might improve the performance of neuromorphic technologies.

3. CONCEPT OF PHYSICS-GUIDED NEURAL NETWORKS

In this section, we describe the technical idea behind physics-guided neural networks and explain the usage of scientific knowledge in the cost function. Moreover, we discuss a hybrid concept of PGNN that combines physics-based models with machine learning algorithms.

3.1. Physics-based Loss Function

The basic principle of PGNN's operation is very simple on the conceptual level. The main assumption is to introduce a physical formula to the cost function so as to enable the usage of known theoretical laws when executing the network. The basic cost function formula for an artificial neural network can be represented as follows:

$$\arg\min_{f} \underbrace{Loss\left(\hat{Y}, Y\right)}_{\text{Typical loss function}} . \tag{1}$$

In this case, an expected output (ground truth) is compared to the achieved results of a model, creating a so-called loss function which can be represented with various equations depending on the type of problem. Squared Error Loss, Absolute Error Loss or Cross Entropy Loss are just few examples of the most poplar functions utilized in this case [38]. Regardless of the function applied, the goal of the algorithm is to minimize the cost so as to ensure the greatest generality of the neural network by later updating the model weights, i.e., thanks to stochastic gradient descent (SGD) algorithm [39].

A standard approach for training the PGNN model is very similar and bases on the same rules. Moreover, the architecture of the neural network does not change either. The only difference is an extension of the minimized function with a commonly known physical formula (so-called physical inconsistency) as presented below:

$$\arg\min_{f} \underbrace{Loss\left(\hat{Y}, Y\right) + \lambda R(f)}_{\text{Typical loss function}} + \underbrace{\lambda_{PHY}\, Loss.PHY(\hat{Y})}_{\text{Physicsl inconsistency}} . \tag{2}$$

In this case, λ_{PHY} is a hyper-parameter chosen by the scientist who decides about the relative importance of a physical influence on the model. This parameter is crucial for generated results, because its proper selection allows to increase the reliability, genericity and compliance of results with physical laws. On the other hand, its incorrect setting (too high value) may lead to a drastic deterioration of effectiveness of the neural network, the results of which will be based too much on the physical formula, thus the model will not be able to maximize a potential of the information contained in a training data. Too low value of the λ_{PHY} parameter may cause a very low influence of the formula, and thus the network will take form of a classic ANN model.

A concept of PGNN loss function may seem relatively simple, but its impact on the generated results is often significant. A key aspect in this case is an appropriate selection of the physical formula, which is usually a difficult task that requires knowledge of a specific field of science in which the model is to be used. It is natural that a specific formula must combine parameters that are available in the dataset used for the problem. Unfortunately, this is the main disadvantage of PGNN, which prevents their use in any problem, and imposes on the user a good knowledge of the field of application. For this reason, physically inspired neural networks are treated rather as a narrow spectrum of research applications. Nevertheless, their capabilities can improve the results achieved, which in certain cases can lead to large profits.

3.2. Hybrid Models

Much easier to apply and more often used is the concept of hybrid models. It is considered a separate intersection of the laws of physics with data science capabilities. In this case classic, artificial neural networks are used without making any changes to their structure. The difference is in combining the training data with results of purely physical model. These types of physics-based numerical models represent part of a real physical phenomenon through extensive data collections from a specific area. Thanks to mathematical modeling they are capable of approximately representing a real behavior of given process or object. A simple way for combining physics-based models with data science is to use the simulated outputs of the model as another input to the neural networks as presented below:

$$f_{HPD} : \mathbf{X} = [\mathbf{D},\, Y_{PHY}] \rightarrow Y. \tag{3}$$

In this setup, it is possible to benefit from physically-consistent features as an output of physics-based model as well as take advantage of other input drivers to the neural network, thus reducing possible knowledge gaps.

It is important to emphasize that physics-based models are very often difficult to obtain or may provide incomplete representation of target due to simplified or missing physics relations, thus resulting in model discrepancies. Moreover, they usually require calibrating which is a time-consuming and ineffective process. It is worth to remember about their multiple advantages also as hybrid-physics-data models, however in this work we focus on physics-based loss function and investigate pure PGNN architectures.

4. POISSON'S DIFFERENTIAL EQUATION

The Poisson's partial differential equation is used as an example theoretical case study that directly shows work the of PGNNs basing on primary results of T. Dockhorn [40]. In subsection 4.1 of this chapter we introduce the problem and discuss

its physics-guided relation. The subsection 4.2 shows the architecture of neural network used and explains its training process. In the subsection 4.3 results of experiments and comparisons between ANN and PGNN are depicted.

4.1. Problem description and physics-guided relation

The Poisson's equation in combination with Dirichlet boundary conditions can be written as:

$$\begin{aligned} -\nabla^2 u(x) &= f(x) \quad \text{on area } \Omega, \\ u(x) &= g(x) \quad \text{on area } \partial\Omega. \end{aligned} \tag{4}$$

The function f of spatial variables is treated as known. The Poisson's equation can be also written explicitly for space with a given dimension d. In case of this work we consider a two-dimensional space ($d = 2$), therefore it takes form of a partial differential equation:

$$\frac{\partial^2}{\partial x_1^2} u(x_1,x_2) + \frac{\partial^2}{\partial x_2^2} u(x_1,x_2) = f(x_1,x_2), \tag{5}$$

where for purpose of our work:

$$u(x_1,x_2) = \sin(\pi x_1)\cos(\pi x_2), \tag{6}$$

$$f(x_1,x_2) = 2\pi^2 \sin(\pi x_1)\cos(\pi x_2). \tag{7}$$

The equation is considered on the area $\Omega = [0,1]^2$

For boundary conditions, the loss function can be written as:

$$loss = MSE\,(\hat{u} - g)\,. \tag{8}$$

After applying physics-guided relationship, resulting from the Poisson's equation, modified loss function takes a following representation:

$$loss = MSE\,(\hat{u} - g) + MSE\left(\nabla^2 \hat{u} - f\right), \tag{9}$$

where $MSE\left(\nabla^2 \hat{u} - f\right)$ is considered a physical loss.

Eventually, after substitution of the Mean Squared Error (MSE), the overall loss can be presented as:

$$loss(\hat{u};\theta) = \frac{1}{N_{\text{bou}}} \sum_{i=1}^{N_{\text{bou}}} (\hat{u} - g)^2 + \frac{1}{N_{\text{int}}} \sum_{j=1}^{N_{\text{int}}} \left(\nabla^2 \hat{u} - f\right)^2. \tag{10}$$

To explain this equation better, we could say that the algorithm approximates some certain $u(x)$ using a neural network $\hat{u}(x,\theta)$, where θ is a stacked vector of the algorithm parameters (weights), which we adjust in the training process. Furthermore, N_{bou} and N_{int} mean a number of boundary as well as interior (physics-inspired) data points respectively. More details on the dataset and training process of the model are presented in Section 4.2. Importantly, the functions f and g are a known true, as given in (4) and further explained in (6) and (7). To be precise, we want to minimize the error of our algorithm (10), therefore our ultimate goal is to find such an approximate $u(x)$ that both parts of the equation are minimized.

4.2. Neural Network Architecture and dataset

The goal of this work is not to experiment with the architecture of the neural network by introducing complex layers, such as e.g., Convolutions, Long Short-Term Memory (LSTM), or Gated Recurrent Units (GRU). Various implementations of exotic physics-guided neural networks on the layer structure side were already presented in Section 2, an example of which may be autoencoders or generative models. Our goal is to show the possibilities of PGNNs in a way that is accessible to the reader, therefore we assume to emphasize the impact of the modified cost function on the algorithm's result rather than the novel network architectures.

Secondly, we show that the considered differential equations can be solved using relatively simple neural networks in terms of the class of their layers and the number of learnable parameters, assuming the addition of a physical formula to the cost function. The results obtained in this way appeared to be characterized by satisfactory accuracy, which is why modifying the structure of the algorithm would not bring any greater benefit in terms of performance. In fact, an increased complexity of the algorithm could lead to negative consequences such as making the training process more difficult, e.g., by extending its time and cost. We can say that when choosing the type of an algorithm we refer directly to the Occam's razor principle [41] and do not complicate our approach beyond the absolute necessity.

For the selected Poisson's problem we consider a set of fully-connected neural networks with different number of hidden layers and units. We did not observe any significant increase in performance of the models with more than three dense layers. In fact, accuracy of the architecture with 16 units in each cell appeared to be empirically optimal. Nevertheless, differences between workloads bigger than three hidden layers appeared to be minimal, therefore we decided to limit our study to the model presented in Fig. 3.

The depicted architecture uses three hidden layers and sigmoid activation functions [42] combined with Xavier initializa-

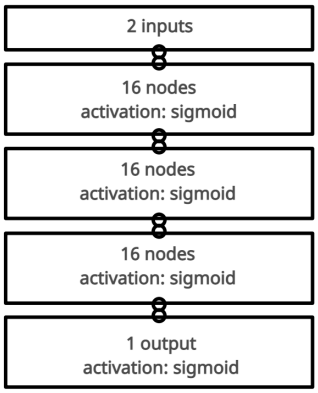

Fig. 3. Architecture of a fully-connected neural network used to solve Poisson's partial differential equation. The model consists of three hidden layers, each of which is activated with the sigmoid function. Quality measure between the approximated and true solutions is calculated on a single output, considering two inputs in the given Poisson's formula. Experimentally, the model appeared to guarantee the best results on the given task

tion [43] of weights. Thanks to the initial weights, our loss gradients are not too large or too small at the beginning, hence the network converges faster. Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [44] is used to optimize the weights, where we know the ground truth and can therefore measure the quality of the approximate solution. We define the maximum number of iterations equal 1000. However, to eliminate any possible overtraining of the neural network, we apply the gradient tolerance equal 10^{-14}. This optimization parameter measures the differences in sizes of gradients on consecutive steps and stops the training once the criteria is violated.

The network is trained on five datasets with 1000, 2000, 4000, 8000, and 16 000 interior and boundary data points with batch size 1000. Because Ω is a rectangle, the grid points are chosen in a way such that two neighboring points have the same value in $d-1$ coordinates and only differ by 0.01 in one coordinate. Training data was generated as a set of uniformly spaced grid points over $\Omega \subset R$ from the square area [0, 1] and is stored in the CSV format with number of records corresponding to the dataset size. Each record contains 4 values placed in separate cells as given in the demonstration samples in Table 1. Subsequently, points are read by a data loader and divided into interior and boundary samples given for each part of the loss function respectively.

Table 1

Example training data for the Poisson's task. Interior x and y coordinates distinguish a point inside the studied area in comparison to the boundary x and y coordinates, which describe the points on the edge of the rectangle. The expected output value is calculated using physical equations (6) and (7) respectively

interior x	interior y	boundary x	boundary y
0.70024601	0.25877339	0.38812045	0.0
0.41249577	0.24691931	0.24181088	1.0
0.57185811	0.72342523	0.0	0.75604303
0.68693357	0.20367522	1.0	0.91711312

The project was carried out entirely in Python, while the pure Tensorflow was used to implement the neural network. This framework selection allows the reader to understand various processes, especially mathematical operations taking place during the training stage. Thanks to a direct declaration of operations in low-level API, a convenient insight and control over the code is possible, particularly when working with operations like weights or gradients. In case of further interests or novel ideas to improve the process, the reader can benefit from our source code (please see Section 6) and introduce the modifications to the presented problem.

4.3. Results

In order to observe the operation of PGNN, we train the neural network using both loss functions (8) (9) and compare the obtained estimates with the scientifically proven solution as presented in Fig. 4. This example shows that a small neural network is able to accurately learn the complex solution for a system of partial differential equations. Moreover, the presented results were obtained when training the model on 1000 data samples to emphasize the advantages of PGNN when having a very small number of data points. From this perspective, it can be stated that both PGNN and ANN estimate the real values over the entire area, which was shown by a heatmap for each point of the space.

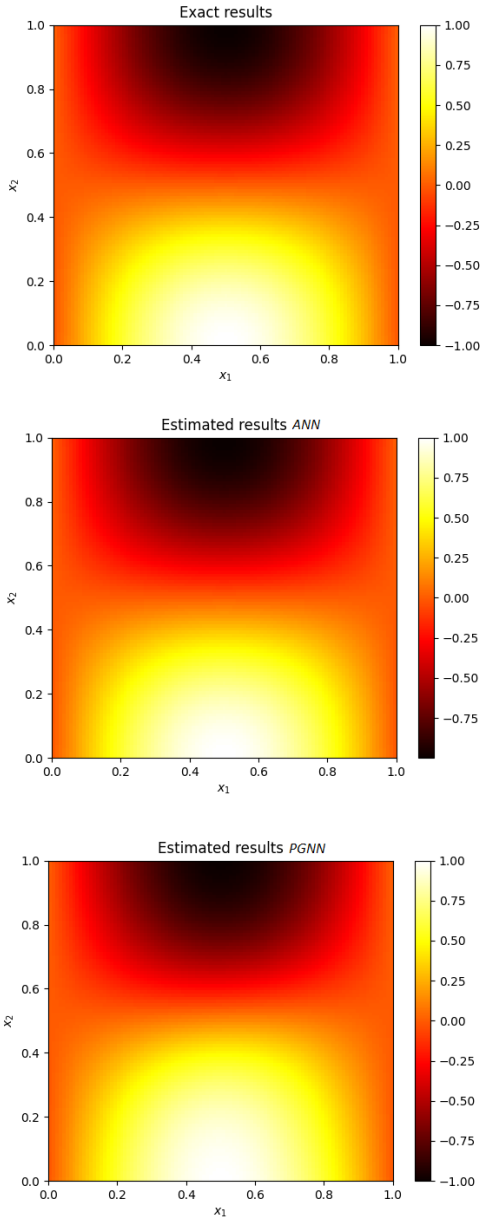

Fig. 4. Results of experiments on Poisson's differential equation for each datapoint on the square area [0, 1]. Ground truth (top), estimates of artificial neural network (middle), and estimates of physics-guided neural network (bottom). Both algorithms achieve the results that are close to exact, however PGNN can be characterized by a significantly smaller error as presented in Fig. 5

More information about the quality of the estimation is shown in Fig. 5, which compares the mean squared error at each point in the region for both ANN and PGNN. Thanks to the

physical consistency, MSE of the model has been reduced by several orders of magnitude, therefore a simple network trained on a small dataset appeared to be surprisingly effective. In case of PGNN, achieved results are very close to exact, with MSE around 10^{-5} while clear ANN reaches an approximate error on the level of 10^{-1}. Thus, although both algorithms approximate a solution of the Poisson's equation successfully, the performance of PGNN is significantly better. In order to obtain similar results without physics-guided relation in the loss function, the network would have to be more complex and trained on much larger dataset, leading to an increase in cost and time of the experiment.

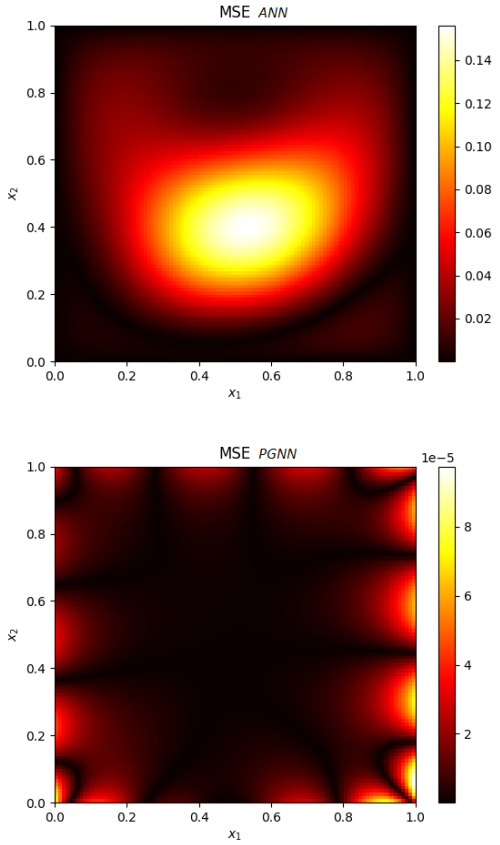

Fig. 5. Comparison of achieved MSE between ANN (top) and PGNN (bottom) showing an increase of performance by several orders of magnitude. Results using physics-inspired loss function are very close to exact, with MSE around 10^{-5} while clear ANN reaches an approximate error on the level of 10^{-1}

5. BURGER'S DIFFERENTIAL EQUATION

As another example, let us consider the Burger's equation basing on a study of M. Raissi *et al.* [30,31] who show the work of PGNN for continuous time models. Burger's formula is considered as a fundamental differential equation and can be derived from the Navier-Stokes equations [45] for the velocity field by dropping the pressure gradient term. Similarly as in case of the Poisson's example, in the following sections of this chapter we present a theoretical background, training process and results of experiments for Burger's equation.

5.1. Problem description and physics-guided relation

In one space dimension, the Burger's equation along with Dirichlet boundary conditions can be written as:

$$\begin{gathered} u_t + uu_x - (0.01/\pi)u_{xx} = 0, \\ u(0,x) = -sin(\pi x), \\ u(t,-1) = u(t,1) = 0, \text{ and } x \in [-1,\ 1],\ t \in [0,\ 1]. \end{gathered} \tag{11}$$

In this case $f(t,x)$ is defined as:

$$f := u_t + uu_x - (0.01/\pi)u_{xx}\,. \tag{12}$$

This representation can be processed by approximating $u(t,x)$ with a deep neural network. Similarly to the Poisson's example, we apply a physics-guided relationship, resulting in a common mean squared error loss:

$$loss = MSE\,(\hat{u} - g) + MSE\left(\nabla^2\hat{u} - f\right), \tag{13}$$

where MSE_f is a physical loss.

Eventually, after substitutions, the overall loss function can be given as:

$$MSE = \frac{1}{N_u}\sum_{i=1}^{N_u}(\hat{u}(t_i,x_i) - u(t_i,x_i))^2 + \frac{1}{N_f}\sum_{j=1}^{N_f} f(t_j,x_j)^2. \tag{14}$$

In this notation, N_u denotes the number of initial and boundary training data on $u(t,x)$ and N_f specifies the number collocations points for $f(t,x)$. Similarly, the MSE_u corresponds to the initial and boundary conditions while MSE_f enforces the structure imposed by equation (11) at a finite set of collocation points and is therefore considered a physical loss. The shared parameters of the neural network for $u(t,x)$ and $f(t,x)$ can be learned by minimizing the loss function.

5.2. Neural Network Architecture and dataset

Similarly as in the case of the Poisson's equation, our goal is to illustrate the operation of PGNN on a simple model of the neural network and limited data in order to emphasize the advantages of physics-inspired loss function. For this purpose, we use a very similar neural architecture as in Section 4.2 with minor modifications that slightly improved the effectiveness for this specific task.

We consider a simple fully-connected model depicted in Fig. 6. The architecture consists of three hidden layers having 32, 16 and 32 neurons respectively, which give a total number of 1201 trainable parameters. Hyperbolic tangent (tanh) [46] is used as an activation function while kernels are initialized using He normal [47]. Similarly to the Poisson's example, loss functions are optimized with Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [44]. Nevertheless, for larger datasets and mini-batches, a stochastic gradient descent (SGD) [39] might be more effective.

As already motivated, we rely on the datasets where the total number of training points is relatively small (a few hundred up to a few thousand points). As a reference, Table 2 presents

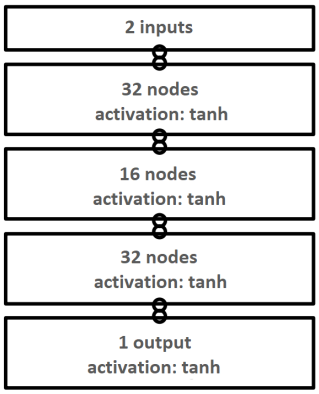

Fig. 6. Architecture of a fully-connected neural network used to solve Burger's partial differential equation. The model consists of three hidden layers, each of which is activated with the hyperbolic tangent. Quality measure between the approximated and true solutions is calculated on a single output, considering two inputs t and x in the given Burger's formula

sample data with dimensions 100×256 for t and x values respectively. Specifically, given a set of $N_u = 100$ randomly distributed initial and boundary data, we learn the latent solution $u(t,x)$ considering 256 points. The presented floating-point values are stored in the matrices exported to the Matlab file format with number of columns and rows corresponding to the dataset size N_u and N_f.

Table 2

Example rounded training data for the Burger's task. Matrix takes 100×256 dimensions for t and x values respectively. Considering uniformly spaced $N_u = 100$ initial and boundary data, we learn the latent solution $u(t,x)$ by training the weights of a neural network given in Fig. 6, thanks to the minimization of mean squared error loss (10). Presented real data is visualized in Fig. 7 and 8 for the three selected snapshots

x \ t	0.00	0.01	0.02	...	0.97	0.98	0.99
–1.00	0	0	0	...	0	0	0
–0.99	0.03	0.02	0.02	...	0.01	0	0
–0.98	0.05	0.04	0.03	...	0.02	0.01	0
–0.97	0.08	0.07	0.06	...	0.03	0.02	0.01
–0.96	0.1	0.09	0.08	...	0.04	0.03	0.02
...	...	...	...	...	...	...	...
0.99	–0.03	–0.02	–0.02	....	–0.01	0	0
1.00	0	0	0	...	0	0	0

5.3. Results

Physical information in loss function is optimized while training the neural network on set of 100 randomly distributed data given in Table 2. In order to observe the operation of PGNN, we optimize the loss function (13) on two experiments: with and without the physical information. A detailed assessment of the predicted solution $u(t,x)$ for PGNN is presented in Fig. 7. In an analogous way, the results of conventional ANN trained without the MSE_f counterpart in equation (13) are depicted in Fig. 8. In particular, we present a comparison between the obtained results and the previously known (exact) solutions at three different timesteps $t = 0.25, 0.50, 0.75$. The results for each of the considered time instances were generated considering 256 data points.

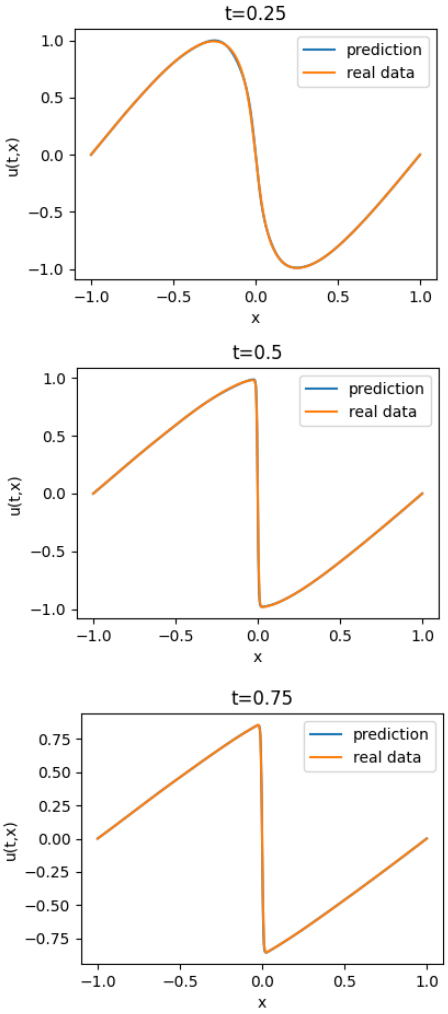

Fig. 7. Solution of the Burgers' nOnlinear partial differential equation using the PGNN. Comparison of the predicted solutions $u(t,x)$ and ground truth corresponding to the three temporal snapshots. The algorithm is able to accurately follow differences in the form of sharpening the sine wave in successive instances of time

Using only limited initial and boundary data, the physics guided neural network can accurately capture the nonlinear behavior of the Burgers' equation. Right after $t = 0.4$ a sharp internal layer starts to develop, which is visible on middle and bottom plots. This solution is usually very demanding to be precisely solved with classical numerical methods because it required a spatio-temporal discretization of equation (11). We are therefore able to determine that although a small dataset used, predictions of a model with physical loss are precisely matching the expectations.

An interesting feature of term MSE_f in equation (13) is its capability to act as a regularization mechanism that penalizes solutions that do not satisfy the Burger's equation. Therefore, we observe that a model without physics-inspired information (Fig. 8) cannot guarantee reliable results. Moreover, the example prove that one of a key advantages of physics-guided neural networks is their effectiveness when trained on small datasets. This property makes usage of PGNNs important for applications where the cost of data acquisition might be expensive or sometimes even impossible.

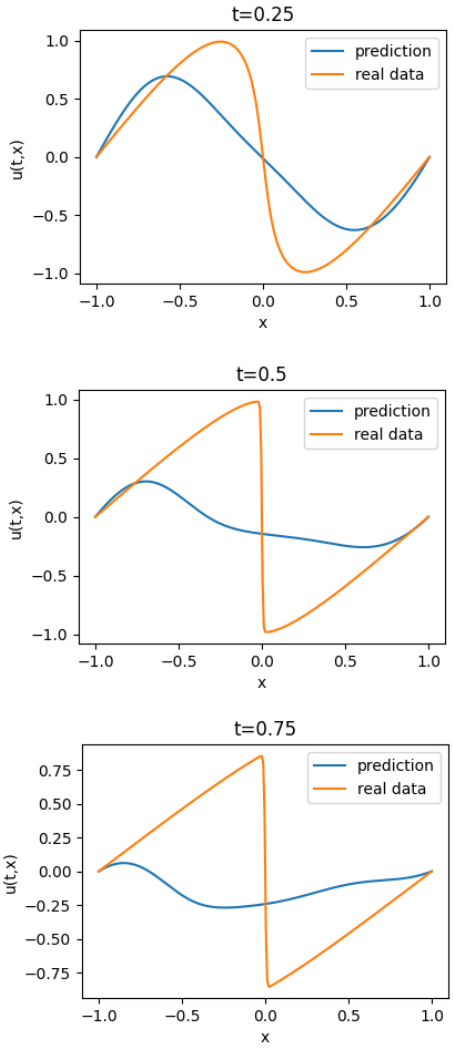

Fig. 8. Results of the ANN analogous to Fig. 7 given for the PGNN. Despite the identical training process, the lack of a physically inspired factor in the cost function makes the model unable to achieve a satisfactory performance for the presented problem. After successive moments of time its results become random

6. CONCLUSION AND FUTURE WORK

In this work, we showed that neural networks can be used to learn complex solutions of partial differential equations for spatial analysis. Moreover, we benefited from scientific relations included in the loss functions and applied Physics-Guided Neural Networks to solve Poisson's and Burger's equations. In both cases, thanks to modifications in cost formula, the accuracy of models increased, although their training process was identical for ANN and PGNN examples. These observations lead to advantages that can be widely used in all applications where it is possible to find a physical relation in the data. Thanks to PGNNs, predictions of models are not only optimal but also scientifically consistent. Presented results show that the combination of data science models with domain specific equations give the best results in terms of accuracy while simultaneously allowing to explain the deduction process and thus eliminate possible contradictions with logic or science which often happen in machine learning.

The primary goal of this publication was to emphasize the concept of PGNN basing on two examples of partial differential equations. Nevertheless, applications of this technology can be broadly extended to a number of theoretical and practical examples such as presented in Section 2. Moreover, it is possible to directly use a presented method of solving Burger's and Poisson's equations by adding custom modifications. Nevertheless, we must remember that presented methods should not be interpreted as replacements of classical numerical methods for solving partial differential equations mainly due to their robustness and computational efficiency. Techniques such as finite elements or spectral methods [48], although their age, are still widely used in many applications. Development of PGNNs is a relatively new field of AI, taking into account a dynamic evolution of this domain. Nevertheless, further papers and constantly new applications of physicsbased loss functions start to gain on popularity. This publication aims to popularize PGNNs and explain how to utilize scientific knowledge contained in the cost functions, basing on theoretical examples. We hope that this contribution will stimulate future work on applications of PGNNs and data-driven scientific computing.

Repository with code for this project is public and fully accessible at: github.com/Borzyszkowski/Experimental-PGNNs.

REFERENCES

[1] R. Vinuesa *et al*., "The role of artificial intelligence in achieving the Sustainable Development Goals," *Nat. Commun*., vol. 11, no. 1, pp. 1–10, 2020.

[2] M. Grochowski, A. Kwasigroch, and A. Mikołajczyk, "Selected technical issues of deep neural networks for image classification purposes," *Bull. Pol. Acad. Sci. Tech. Sci*., vol. 67, no. 2, pp. 363–376, 2019.

[3] T. Poggio and Q. Liao, "Theory II: Deep learning and optimization," *Bull. Pol. Acad. Sci. Tech. Sci*., vol. 66, no. 6, pp. 775–787, 2018.

[4] A. Lüdeling, M. Walter, E. Kroymann, and P. Adolphs, "Multilevel error annotation in learner corpora," *Proc. Corpus Linguistics Conf*., vol. 1, pp. 14–17, 2005.

[5] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," *Proc. Int. Interdiscipl. PhD Workshop (IIPhDW)*, 2018, pp. 117–122.

[6] A.W. Moore and M.S. Lee, "Efficient algorithms for minimizing cross validation error," *Proc. 11th Int'l Conf. Machine Learning*, 1994, pp. 190–198.

[7] J.M. Benitez, J.L. Castro, and I. Requena, "Are artificial neural networks black boxes?," *IEEE Trans. Neural Networks*, vol. 8, pp. 1156–1164, 1997.
[8] T. Hagendorff, "The ethics of AI ethics: An evaluation of guidelines," *Minds Mach*., vol. 30, pp. 99–120, 2020.
[9] T. Miller, P. Howe, and L. Sonenberg, "Explainable AI: Beware of inmates running the asylum," *Proc. IJCAI Workshop Explainable AI (XAI)*, 2017, pp. 36–42.
[10] A. Rai, "Explainable AI: from black box to glass box," *Journal of the Academy of Marketing Science*, vol. 48, pp. 137–141, 2020.
[11] G. Shortley and G. Weller, "Numerical solution of Laplace's equation," *J. Appl. Phys*., vol. 9, no. 1, pp. 334–336, 1938.
[12] R.C. Mittal and P. Singhal, "Numerical solution of Burger's equation," *Commun. Numer. Methods Eng*., vol. 9, no. 5, pp. 397–406, 1993.
[13] R. French, "Subcognition and the limits of the Turing test," *Mind*, vol. 99, no. 393, pp. 53–65, 1990.
[14] J. McCarthy, "What is artificial intelligence?," 1998.
[15] I. Rojek, M. Macko, D. Mikołajewski, M. Saga, and T. Burczyński, "Modern methods in the field of machine modelling and simulation as a research and practical issue related to industry 4.0," *Bull. Pol. Acad. Sci. Tech. Sci*., vol. 69, no. 2, p. e136717, 2021.
[16] A. Karpatne, W. Watkins, J. Read, and V. Kumar, "Physics-guided neural networks (PGNN): An application in lake temperature modeling," 2017, [Online], Available: http://arxiv.org/abs/1710.11431.
[17] J. Willard *et al*., "Integrating Physics-Based Modeling with Machine Learning: A Survey," 2020, [Online], Available: http://arxiv.org/abs/2003.04919.
[18] X. Jia *et al*., "Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles," *Proc. SIAM Int. Conf. Data Mining*, pp. 558–566, 2019.
[19] A. Daw *et al*., "Physics-Guided Architecture (PGA) of neural networks for quantifying uncertainty in lake temperature modeling," *Proc. SIAM Int. Conf. Data Mining*, pp. 532–540, 2020.
[20] Y. Yang and P. Perdikaris, "Physics-informed deep generative models," 2018, [Online], Available: http://arxiv.org/abs/1812.03511.
[21] R. Singh, V. Shah, B. Pokuri, and S. Sarkar, "Physics-aware deep generative models for creating synthetic microstructures," 2018, [Online], Available: http://arxiv.org/abs/1811.09669.
[22] L. Wang, Q. Zhou, and S. Jin, "Physics-guided deep learning for power system state estimation," *J. Mod. Power Syst. Clean Energy*, vol. 8, no. 4, pp. 607–615, 2020.
[23] N. Muralidhar *et al*., "Physics-guided design and learning of neural networks for predicting drag force on particle suspensions in moving fluids," 2019, [Online], Available: http://arxiv.org/abs/1911.04240.
[24] J. Park and J. Park, "Physics-induced graph neural network: An application to wind-farm power estimation," *Energy*, vol. 187, p. 115883, 2019.
[25] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, "Towards physics-informed deep learning for turbulent flow prediction," *Proc. 26th SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2020, pp. 1457–1466.
[26] T. Yang *et al*., "Evaluation and machine learning improvement of global hydrological model-based flood simulations," *Environ. Res. Lett*., vol. 14, no. 11, p. 114027, 2019.
[27] Y. Zhang *et al*., "Pgnn: Physics-guided neural network for fourier ptychographic microscopy," 2019, [Online], Available: http://arxiv.org/abs/1909.08869.
[28] M.G. Poirot *et al*., "Physics-informed deep learning for dual-energy computed tomography image processing," *Sci. Rep*., vol. 9, no. 1, 2019.
[29] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, E. Kuhl, "Physics-informed neural networks for cardiac activation mapping," *Front. Phys*., vol. 8, p. 42, 2020.
[30] M. Raissi, P. Perdikaris, and G.E. Karniadakis, "Physics informed deep learning (part I): Data-driven solutions of nOnlinear partial differential equations," 2017, [Online], Available: http://arxiv.org/abs/1711.10561.
[31] M. Raissi, P. Perdikaris, and G.E. Karniadakis, "Physics informed deep learning (part II): Data-driven solutions of nOnlinear partial differential equations," 2017, [Online], Available: http://arxiv.org/abs/1711.10566.
[32] Z. Fang and J. Zhan, "A physics-informed neural network framework for PDEs on 3D surfaces: Time independent problems," *IEEE Access*, vol. 8, pp. 26328–26335, 2019.
[33] B. Paprocki, A. Pregowska, and J. Szczepanski, "Optimizing information processing in brain-inspired neural networks," *Bull. Pol. Acad. Sci. Tech. Sci*., vol. 68, no. 2, pp. 225–233, 2020.
[34] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Front. Neurosci*., vol. 12, pp. 774, 2018.
[35] Z. Bing *et al*., "A survey of robotics control based on learninginspired spiking neural networks," *Front. Neurorob*., vol. 12, pp. 35, 2018.
[36] B. Borzyszkowski, "Neuromorphic Computing in High Energy Physics," 2020, doi: 10.5281/zenodo.3755310.
[37] J. George, C. Soci, M. Miscuglio, and V. Sorger, "Symmetry perception with spiking neural networks," *Sci. Rep*., vol. 11, no. 1. pp. 1–14, 2021.
[38] K. Janocha and W.M. Czarnecki, "On loss functions for deep neural networks in classification," 2017, [Online], Available: http://arxiv.org/abs/1702.05659.
[39] L. Bottou, "Stochastic gradient descent tricks," in Neural networks: Tricks of the trade, Berlin, Heidelberg: Springer 2012, pp. 421–436.
[40] T. Dockhorn, "A discussion on solving partial differential equations using neural networks," 2019, [Online], Available: https://arxiv.org/abs/1904.07200.
[41] A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth, "Occam's razor," *Inf. Process. Lett*., vol. 24, no. 6, pp. 377–380, 1987.
[42] A. Marreiros, J. Daunizeau, S. Kiebel, and K. Friston, "Population dynamics: variance and the sigmoid activation function," *Neuroimage*, vol. 42, no. 1, pp. 147–157, 2008.
[43] S.K. Kumar, "On weight initialization in deep neural networks." 2017, [Online]. Available: http://arxiv.org/abs/1704.08863.
[44] N. Nawi, M. Ransing, and R. Ransing, "An improved learning algorithm based on the Broyden-fletcher-goldfarb-shanno (BFGS) method for back propagation neural networks," *Proc. 6th Int. Conf. Intell. Syst. Design Appl*., 2006, vol. 1, pp. 152–157.
[45] P. Constantin and C. Foias, "Navier-stokes equations," Chicago: University of Chicago Press, 1988.
[46] G.A. Anastassiou, "Multivariate hyperbolic tangent neural network approximation," *Comput. Math. Appl*., vol. 61, no. 4, pp. 809–821, 2011.
[47] B. Hanin and D. Rolnick, "How to start training: The effect of initialization and architecture," *Proc. Adv. Neural Inf. Process. Syst*., 2018, pp. 571–581.
[48] R. Ghanem and P. Spanos, "Stochastic finite elements: a spectral approach," New York: Springer, 1991.