Imperial College
London

Physics Informed Machine Learning

Attila Haas

Department of Mechanical Engineering

December 2022

Abstract

This literature review introduces the concept of Physics Informed Machine Learning (PIML) aimed at an audience new to the subject, but already familiar with classical machine learning (ML) and engineering. Beginning from the limitations of both physics based and purely data driven methods, the motivation for blending the two is presented. This usually involves adapting existing ML solutions to conform to laws of physics. Several methods for such augmentations are then explored: using simulations as training data, customising the cost function and designing network architecture. Next, several compelling examples of applications are presented, which each highlight special features of PIML approaches. These include but are not limited to label-free training, surrogate based modelling and ill-posed problem solving. To showcase these, research from several scientific fields are considered with a focus on fibre reinforced composites. Finally, the current hurdles of PIML are highlighted and future promising areas of research identified.

Contents

1 Introduction

As engineering development keeps reaching new heights, current modelling techniques are pushed to the limit. Cai, Mao, Wang, Yin & Karniadakis (2021) show how the field of fluid dynamics is a prime example. Despite extensive progress in numerically solving the Navier-Stokes equations with finite element (Hughes 1987), spectral (Karniadakis et al. 2005) and meshless methods (Katz 2009) their applicability is still restricted. These methods are computationally too expensive for some industrial use-cases and they cannot handle unknown boundary conditions or material characteristics. Additionally, they highlight how complexity of these software make updating them unrealistic, as for example the OpenFoam software (Jasak et al. 2007) package has over 100 000 lines of code.

The need therefore arises to develop new, more efficient approaches which can be easily customised to any problem. Exploiting Machine Learning (ML) methods is a promising option given its proven effectiveness in a vast variety of different fields including image recognition (Krizhevsky et al. 2017), medical diagnoses (Kononenko 2001) and natural language processing (Mikolov et al. 2013). This literature review begins with describing the motivation for using machine learning, then showing why the classical approaches must be altered to carry physical insight. Next, the mathematical methods for altering existing ML frameworks are described. To explore these further, multiple promising examples of research using Physics Informed Machine Learning (PIML) are highlighted. Finally the reliability of PIML methods is discussed.

2 Motivation for Physics Informed Machine Learning

2.1 Overview of problem-solving approaches

To establish any new methodology for modelling systems, it is first best to take a high-level overview of how existing ones work. All problems can be grouped into two families: direct or inverse problems. Direct problems involve using already established physical laws to predict the behaviour of designs, whereas inverse problems use observations to discover relationships which can then be used directly. In other words, to solve any problem one must first solve an inverse problem first to find a correlation which can then be used to solve specific direct cases.

Most current engineering modelling methods are direct, as the underlying physical laws have already been discovered by physicists. Depending on the complexity and simplifications, either the analytical laws can be solved for a pure, exact solution or numerical methods, like Rouge Kutta (Butcher 1996) or FEA (Zienkiewicz et al. 2005) can provide suitable approximations. If convergence is ensured and the numerical method is set up correctly, the results are guaranteed to conform to physical laws. The deficiencies of these methods lie in their high computational costs, their need for all boundary conditions and material characteristics and their inability to describe any problem where the physical laws are unknown.

Either of these shortcomings can cause classical numerical methods to become insufficient for practical use. As an example, Fabregat et al. (2021) modelled the turbulent flow from a human sneezing with CFD to better understand the social distancing rules needed during the Covid-19 pandemic. On their computer setup, they reported that it took 540 hours or 23 days of computation time for simulating just 1.68s. This illustrates how industrial applications can be limited by computing power. Other then speed, as highlighted by Cai, Wang, Wang, Perdikaris & Karniadakis (2021) solving free-boundary and Stefan problems also pose a significant challenge. As they come up in many areas of engineering (Friedman 2000), some methods have been developed to numerically resolve them (Fix 1982), (Chen et al. 1997), however they are not perfect. Unfortunately, these are highly specific to certain problems and are not general, nor can they incorporate any observations about a system.

Other than making assumptions and approximations, the shortcomings of direct modelling methods can be tackled by changing the problem formulation. Instead of using physical laws to describe the behaviour of the system, one can gathering test results and find problem specific-relationships. In simple cases, a few empirical relationships might suffice, however a more systematic approach is necessary to handle all problems.

2.2 Finding inverse relationships

Given the extensive successes of machine learning cited in Section 1, it is the method of choice solve the inverse problem associated with measured experimental results. Specifically Artificial Neural Networks (ANNs or NNs) are chosen, as they have been mathematically proven to be able to approximate any function, given enough training data (Hornik et al. 1989). Additionally, ANNs boast multiple other advantages. Firstly, ANNs are versatile as they can handle multiple types of inputs, not only raw measurements but images or maybe even descriptions in text. Secondly, implementing ANNs is relatively user friendly using open-source Python libraries like Pytorch (Paszke et al. 2019) and TensorFlow (Abadi et al. 2016). Finally, the extensive development of NNs available from other fields, can often be re-purposed with transfer learning.

Staying with the example of fluid mechanics, such purely data driven models have shown notable achievements in modelling turbulent flows using a database of RANS solutions and flow data (Kutz 2017). This methodology however also has it's drawbacks: Firstly, creating and maintaining databases of sufficient volume and quality is expensive. Secondly, noisy data poses the danger of overfitting. Lastly, Meng et al. (2022) voice concerns about how the cost functions of such ANNs are highly non-convex, thus making their training particularly challenging. They emphasise how converging on a local minima might correspond to solutions which cannot be generalised and could contradict some physical laws or common sense. A closed system gaining or losing energy is the most obvious example.

2.3 The middle ground

In this paper so far the use of direct approaches was discussed and their limitations highlighted. These methods prove to be too rigid, not applicable to ill-posed problems and highly computationally expensive. Next inverse, purely data driven methods were considered, for which it's own set of shortcomings were listed. They are difficult to optimise, and thus cannot be trusted for general problems as the solutions might contradict the laws of physics. The natural question one might ask next is if there is maybe a mix of the two which offers both the data-driven flexibility and the physics knowledge. The answer is Physics Informed Machine Learning (PIML), which is illustrated in Figure 1.

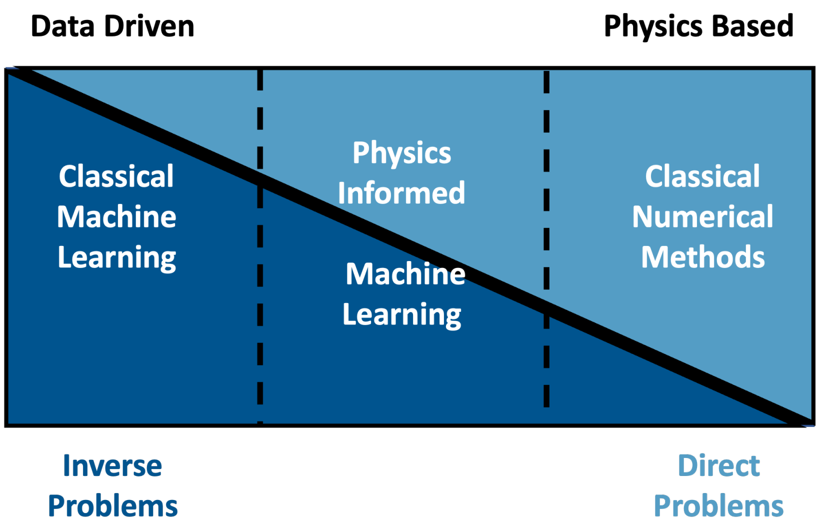

Data Driven
Physics Based
Classical Machine Learning
Physics Informed Machine Learning
Classical Numerical Methods
Inverse Problems
Direct Problems

Figure 1: Schematic of problem solving methods (Based on Karniadakis et al. (2021))

This middle ground between inverse and direct problems will aim to incorporate both information form data collected from tests and prior knowledge about the world. Most solutions will take an existing ML method as a foundation, which will then be augmented to conform to known physical laws.

3 Techniques to build PIML models

3.1 Informed machine learning in general

Before turning attention to PIML, the general case of indroducing any prior knowledge into ML methods is to be studied. This topic is covered in detail in an excellent overview from von Rueden et al. (2021), on which this description is based and interested readers are recommended to start further research there. They define Informed Machine Learning as an ML algorithm that learns form multiple, independent pipelines consisting of both training data and prior knowledge. (Illustrated on Figure 2) For the purpose of this literature review, the definition in some cases will be expanded to include ML methods working together with physics based methods only, without any other training data. Despite technically only having one pipeline of input, these can also play an enormous role in augmenting existing methods.

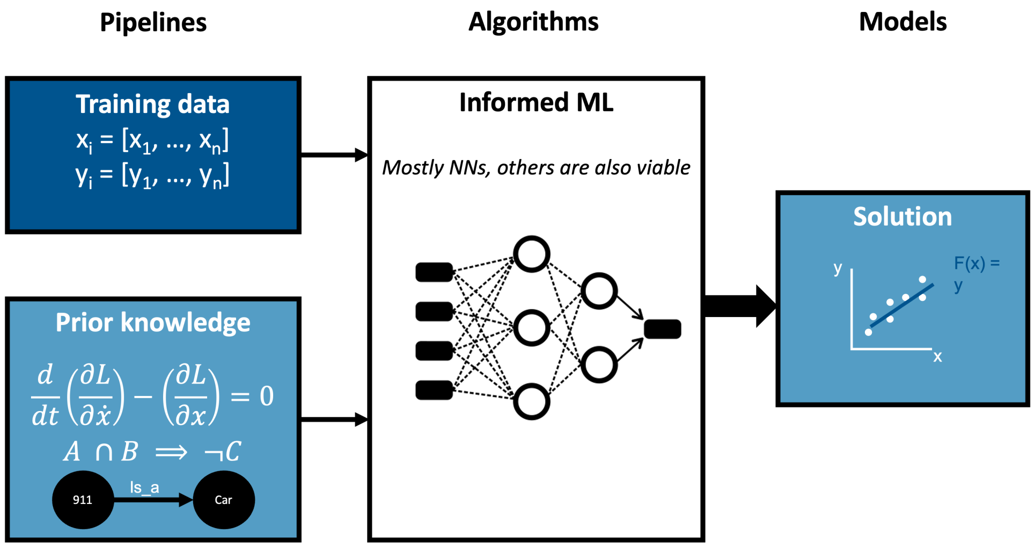

Pipelines
Algorithms
Models
Training data
$x_i = [x_1, ..., x_n]$
$y_i = [y_1, ..., y_n]$
Prior knowledge
$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \left(\frac{\partial L}{\partial x}\right) = 0$
$A \cap B \implies \neg C$
911
Is_a
Car
Informed ML
Mostly NNs, others are also viable
Solution
y
x
F(x) = y

Figure 2: Schematic for informed ML (Based on von Rueden et al. (2021))

When creating such an algorithm, they propose a three step process. First, identify the classical ML method to augment and the source of knowledge to incorporate. Next, represent the knowledge in a format the computer can understand. This could mean logic rules, knowledge graphs or equations. Finally, this representation must be integrated into the machine learning pipeline. Although multiple types of ML methods can be informed with prior knowledge, the most popular is ANNs to which this section will restrict its scope. Other, more obscure methods will be briefly discussed in section 5.

3.2 Secondary pipelines

Still following von Rueden et al. (2021), for ANNs one can add a secondary pipeline of knowledge in one of three means. It is generally good practice to use multiple of these methods simultaneously as they work together in great synergy. These, most popular methods for informing a NN with physics knowledge is summarised in Figure 3.

Firstly, the most straightforward method is to convey information in the training data itself. This quintessentially does not differ from a traditional ML approach, however with physics for example, multiple pipelines of data can be considered. One could use a hybrid of both experimental and simulation results as training data, thus relying on both previously discovered physical rules and measurements. This not only allows for significant cost reductions in data acquisition, but also to reduce noise.

Secondly, the ANNs architecture can be altered to help it better optimise for certain solutions. Here a clear distinction must be made between the classical optimisation of a neural network and informing it. Only changes that have real world meaning are considered a second pipeline. Mapping certain neurons that are known to represent parts of a logic statement to particular neurons is a prime example. CNNs used to add translation invariance is another.

Thirdly, the learning algorithm can be tweaked. The cost function of a NN can usually be altered in a way for the results to conform to known algebraic or differential equations. As one would expect this is the most widely used method in PIML, as most systems can be described by known relationships.

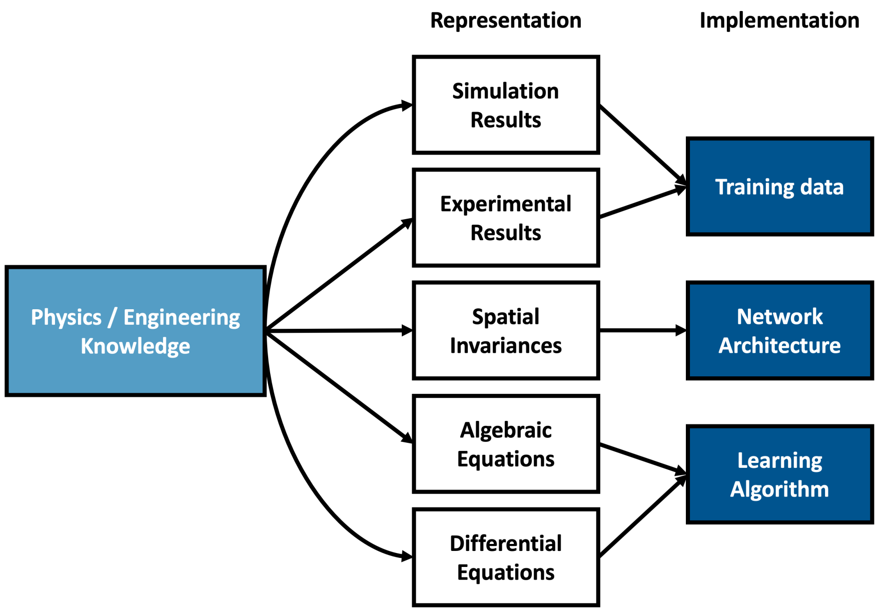


Figure 3: Incorporating different types of knowledge (Based on von Rueden et al. (2021))

3.3 Implementation methods

Next, the three main techniques described in Section 3.2 will be discussed in detail. Each have multiple sub-variants, all with their own strengths and weaknesses. When developing a PIML solution, the challenge lies in combining these to compliment the particular problem, avoid overfitting and overly complex cost functions.

3.3.1 Hybrid training data

Supplementing measurement data with simulation results can be as simple as appending it to physical measurements. In cases where making measurements are difficult or expensive, it is good practice to have a mix of the two sources. Depending on the application, great care must be taken to choose what scenarios to measure and which to simulate considering the biases and limitations of the simulation methods. As such decisions are highly problem specific, it cannot be discussed generally. Although use of such synthetically generated labels by themselves have been shown to be effective, as outlined by Daw et al. (2021), alternative integration methods can benefit more from the underlying physics based model. They propose three different hybrid data pipelines shown in Figure 4 for using a NN to fine tune the physical simulation to experimental results.

These methods don't use the simulation to create extra labels, but instead as an extra input. Say a set of measurements $\epsilon_m = \{(x_1, y_{1,m}), ..., (x_n, y_{n,m})\}$ and a physics simulation $f_P : x \rightarrow y_p$ are available. Using the simulation to predict the values for the same inputs yields $\epsilon_p = \{(x_1, y_{1,p}), ..., (x_n, y_{n,p})\}$. The first possible configuration is to use the original inputs $\boldsymbol{x}$ and the simulated results $\boldsymbol{y}_p$ as inputs to the NN, with the measurements as labels:

$$f_{NN} : [\boldsymbol{x}, \boldsymbol{y_p}] \rightarrow \boldsymbol{y} \text{ where labels are } \boldsymbol{y_m} \tag{a}$$

This will allow the NN to find systematic discrepancies between the measured and simulated data, thus improving original predictions. The framewok is illutstrated on Figure 4/a.

Alternatively, one can use the NN to estimate the error of the simulation result instead of the end result directly. These are termed "Residual Models", as the output of the NN will be the difference between the measured label $\boldsymbol{y_m}$ and the simulation result $\boldsymbol{y_p}$:

$$f_{NN,Res} : \boldsymbol{x} \rightarrow \boldsymbol{y}_{Res} \text{ where labels are } [\boldsymbol{y_m} - \boldsymbol{y_p}] \tag{b}$$

The end result can then be computed by adding the predicted residual from the NN and the simulation result: $\boldsymbol{y} = \boldsymbol{y_p} + \boldsymbol{y_{Res}}$. This model is considerably simpler than the one described in Equation a, as it only finds discrepancies and not a full function from $\boldsymbol{x}$ to $\boldsymbol{y}$. The full architecture of a basic residual model is shown on Figure 4/b.

Finally, the ideas behind Equation a and b can be combined to form a NN which uses both the inputs $\boldsymbol{x}$ and the simulation results $\boldsymbol{y_p}$ to predict the residual $\boldsymbol{y_m} - \boldsymbol{y_p}$:

$$f_{NN,Res} : [\boldsymbol{x}, \boldsymbol{y_p}] \rightarrow \boldsymbol{y}_{Res} \text{ where labels are } [\boldsymbol{y_m} - \boldsymbol{y_p}] \tag{c}$$

This model is called a "Hybrid-Physics-Data-Residual Model" and is illustrated on Figure 4/c. The additional inputs form the physical model aim to ease the the learning at the cost of extra complexity. The final result is calculated same as with the Residual Model, by adding the residual to the simulated result.

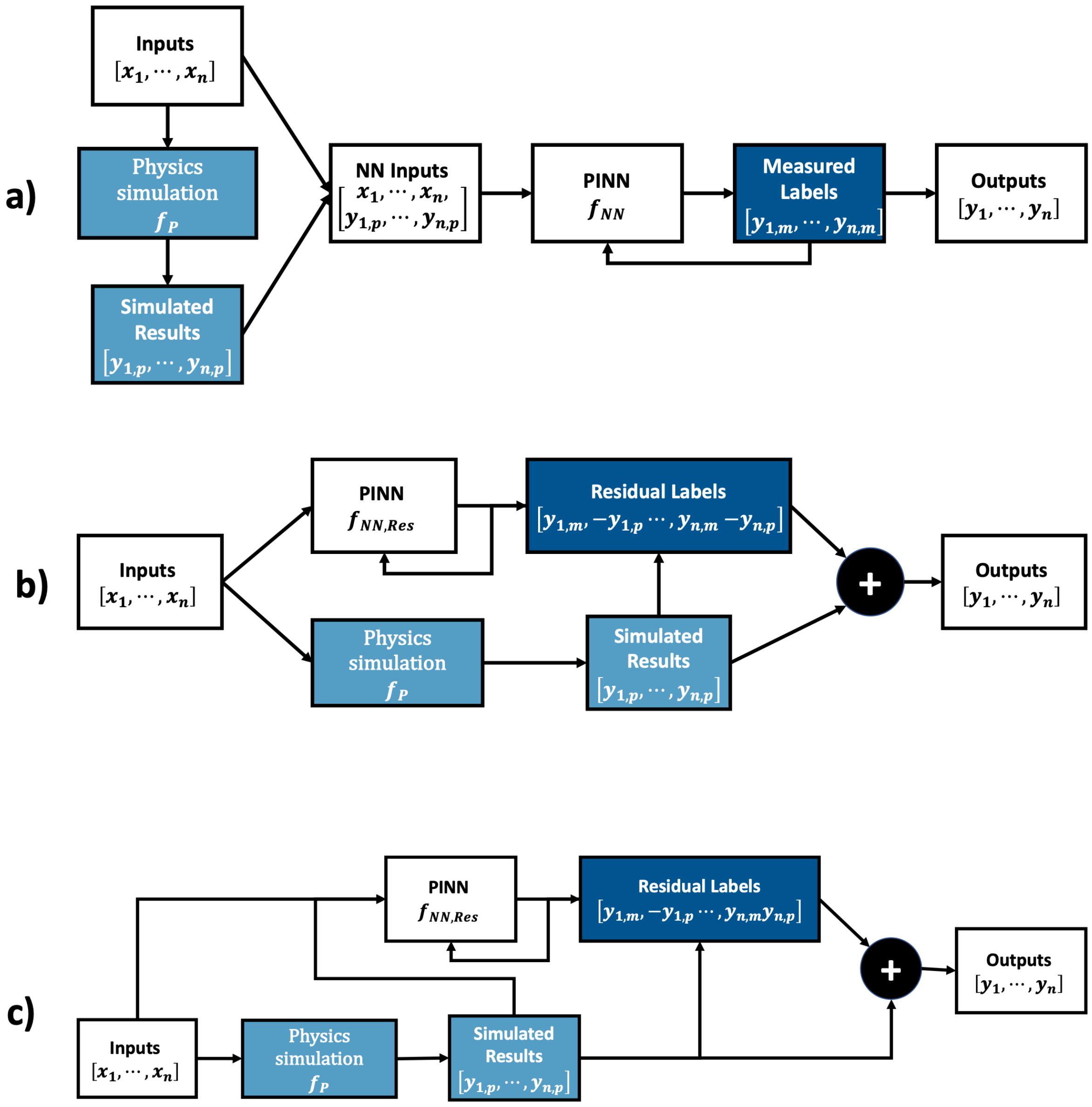

a)
Inputs $[x_1, \cdots, x_n]$
Physics simulation f_P
Simulated Results $[y_{1,p}, \cdots, y_{n,p}]$
NN Inputs $\begin{bmatrix} x_1, \cdots, x_n, \\ y_{1,p}, \cdots, y_{n,p} \end{bmatrix}$
PINN f_{NN}
Measured Labels $[y_{1,m}, \cdots, y_{n,m}]$
Outputs $[y_1, \cdots, y_n]$
b)
Inputs $[x_1, \cdots, x_n]$
PINN $f_{NN,Res}$
Residual Labels $[y_{1,m}, -y_{1,p} \cdots, y_{n,m} - y_{n,p}]$
Physics simulation f_P
Simulated Results $[y_{1,p}, \cdots, y_{n,p}]$
+
Outputs $[y_1, \cdots, y_n]$
c)
PINN $f_{NN,Res}$
Residual Labels $[y_{1,m}, -y_{1,p} \cdots, y_{n,m} y_{n,p}]$
+
Outputs $[y_1, \cdots, y_n]$
Inputs $[x_1, \cdots, x_n]$
Physics simulation f_P
Simulated Results $[y_{1,p}, \cdots, y_{n,p}]$

Figure 4: Hybrid data pipelines(Based on Daw et al. (2021))

At the time of writing this literature review, a rigorous comparison between the integration frameworks described above was not found, however Daw et al. (2021) provide an empirical example.

In their paper they use all three to predict temperatures of lakes and find that Model c outperforms the other two. Model b is less consistent than Model a, however has better performance. Naturally these observations cannot be generalised, however they provide a useful benchmark. This paper on lake temperature prediction is the best example found for use of hybrid training data. As it has been covered here it will not be included in Section 4.

3.3.2 Network architecture

Forms of spacial invariances are inherently implied by laws of physics, however they can be complex to incorporate into a NN. Noether's Theorem states that each spacial invariance of a system implies a corresponding conservation law. (Noether 1918) For example, the conservation of momentum corresponds to transnational invariance. If a PINN is invariant, this implication is more likely to carry over, thus creating more general predictions. (Willard et al. 2020) One option is to translate or rotate the training examples, thus creating additional virtual examples (Niyogi et al. 1998), however this just heuristically tunes the each input rather than capturing true invariance.

Generally, for transnational, rotational and scale invariance can be incorporated into a CNN architecture. As for temporal invariances, Recurrent Neural Networks (RNNs) are model of choice. (Willard et al. 2020) Extensive research on both is available from the field of image recognition (e.g: Wang et al. (2016)), however the solutions are problem specific and must be handled on a case by case basis. These however do not cover all types of invariances. Galilean invariance for example states that laws of motion do not depend on the chosen inertial reference frame. A general method of incorporating this into a PIML method was not mentioned in any research, however it has been achieved in examples. Ling et al. (2016) show how a multiplicative layer with invariant tensor bases can be used to enforce this on the Reynolds Stress anisotropy tensor. Butter et al. (2018) manage to incorporate mass invariance into the model using the Lorentz layer. The technical complexity of these works far outreach the scope of this literature review, therefore they will not discussed further.

Changing the network architecture of a PINN is probably the most difficult to implement from the three options described in this literature review. There is no direct framework, however pre-trained networks can often be used which reduces implementation efforts significantly.

3.3.3 Learning algorithms

During the training of a classical reinforcement learning NN, a cost function is used to quantify the difference between the desired output and result of the NN. In the most classical sense, the desired output is just the label assigned to the training data. Let the training data set be ϵ, which is made up of n pairs of inputs and labels: $\epsilon = \{(x_1, y_1), ..., (x_n, y_n)\}$. Furthermore, let the NN be represented by the function $f : x_i \rightarrow y_i$. The cost function can then be expressed as:

$$f_{cost} = \sum_{x \in \epsilon} L(f(x_i), y_i) \tag{1}$$

Where L somehow describes the difference between the prediction of the NN $f(x_i)$, and the known label y_i. In the most simple case just $L = |y_i - f(x_i)|$. Training minimises this difference by adjusting the weights and biases of the network, thus changing $f(x_i)$. (Lecun 2001) This leads to a set of $\mathcal{F}$ functions to be evaluated, from which the one with the lowest f_{cost} will be selected.

$$f^* = \underset{f \in \mathcal{F}}{\text{arg min}} f_{cost}(f(x_i), y_i)$$

If there are any other criteria which one wishes to impose on the resulting NN, it can be similarly incorporated into the cost function as long as it can be quantified. Overfitting for example is the classical problem with ML when the model is too heavily influenced by training data and loses generality. By adding a term to the cost function that quantifies the complexity of the NN, one can minimise this as well, thus avoiding overfitting. (Srivastava et al. 2014). This is called regularisation. It's details is outside of the scope of this literature review, so it will only be marked by $R : f \to \mathbb{R}$.

Continuing this line of thought, if the outputs of a NN are to follow certain physical rules, the deviation from those laws can also be incorporated into the cost function. Similarly to L, let L_k describe the difference between the NN's prediction $f(x_i)$ and the result expected from previous knowledge. The expected knowledge could be calculated using the inputs or outputs (or both), therefore $L_k = L_k(f(x_i), x_i, y_i)$. Adding both the regularisation and knowledge loss terms to Equation 1, we obtain (Diligenti et al. 2017):

$$f_{cost} = \underbrace{\lambda_L \cdot \sum_{x \in \epsilon} L_l(f(x_i), y_i)}_{Label} + \underbrace{\lambda_R \cdot R(f)}_{Regularization} + \underbrace{\lambda_k \cdot L_k(f(x_i), x_i, y_i)}_{Knowledge} \tag{2}$$

Where λ_L, λ_R and λ_K are just weights of the label, regularisation and knowledge terms respectively. Selecting optimal weights is key to ensure improved trainability of the model. (Wang et al. 2020) Equation 2 is illustrated on Figure 5 and expresses the general idea behind customising the cost function for informed NNs. Naturally one could add multiple L_k functions, or extra terms, however making the cost function too complex is generally not recommended. The exact formulation of a specific L_k function is best illustrated through an example, which will be given in Section 4.1.

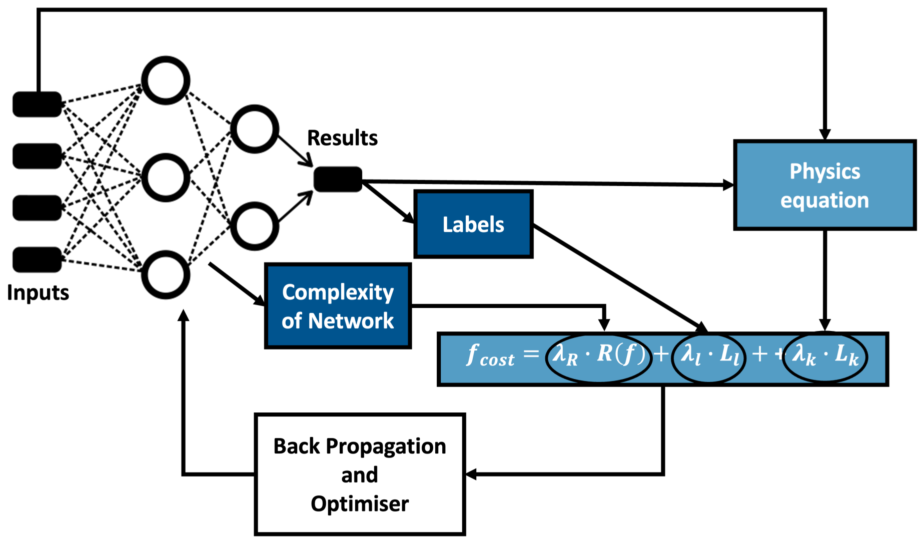

Inputs
Results
Labels
Physics equation
Complexity of Network
$f_{cost} = \lambda_R \cdot R(f) + \lambda_l \cdot L_l + + \lambda_k \cdot L_k$
Back Propagation and Optimiser

Figure 5: General loss function of a informed NN (Based on Karniadakis et al. (2021))

4 Examples and Notable results

The PIML methods described above are not only highly versatile and synergistic with each other, but also facilitate different training methods when compared to classical NNs. In this section, examples of uses and implementations will be detailed to showcase their behaviour on real problems. The examples have been chosen to best illustrate each technique purely and in combinations to highlight the following two points: Firstly, PIML methods could be used in any field of research including fluid dynamics, heat transfer, materials science, manufacturing and so on. Where possible, papers on fibre reinforced composites were preferred, to adhere to the project objectives. Secondly, creative formulations of physics knowledge can significantly influence how PIML methods are trained and how they preform.

4.1 Knowledge loss term example

Karniadakis et al. (2021) introduce the idea of a custom cost function for a PINN, by proposing a solution for the 1D viscous Burger's equation. It will be assumed that a governing equation for the boundary conditions is not known, however the measurements are available. Furthermore it is known if the measurement is form a boundary or other parts of the domain. The known correlation and training data for this problem are therefore:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{3}$$

$$\boldsymbol{u} = \begin{cases} \text{Boundaries} & \boldsymbol{u}_i = \{(x_1, t_1), \ldots, (x_n, t_n)\} \\ \text{Other areas} & \boldsymbol{u}_j = \{(x_1, t_1), \ldots, (x_k, t_k)\} \end{cases} \tag{4}$$

To implement these constraints on a PIML model the following steps are to be taken. Firstly, Equation 3 must to be rearranged to zero and discretised for it to be compatible with the cost function, as it will be minimised and only finite points are available. This will become the L_k term defined in Section 3.3.3, as it draws the resulting solution towards satisfying the governing equation. Secondly, the predicted boundary conditions will be compared to the training data, forming the L_l term from Section 3.3.3. These can be expressed as:

$$\begin{aligned} L_k &= \frac{1}{n}\sum_{j=1}^{n}\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2}\right)\bigg|_{x_j,t_j} \\ L_l &= \frac{1}{k}\sum_{i=1}^{k}(f(x_i,t_i) - u_i)^2 \end{aligned} \tag{5}$$

These two terms from Equation 5 can be added with their respective user tuned weights to form the cost function. Finally, the NN setup and training can commence as with any classical NN. Here no PINN specific guidelines for selecting the number of layers or neurons or optimiser are recommended, therefore it will have to be handled as any other NN.

4.2 Unknown boundary conditions

Cai, Wang, Wang, Perdikaris & Karniadakis (2021) tackle heat transfer problems with unknown boundary conditions and only a few sets of physical measurements. This much better represents real-life problems when compared to classical CFD, where the boundary conditions approximated. They employ PINNs relying on governing equations, thermal and flow velocity measurements alike. A forced convection problem around a cylinder is studied in detail, which is then followed by more complex Stefan problems and applications. Here only the former will be discussed to stay concise.

The governing equations of such flows, assuming incompressibility are the Navier-Stokes equations and the temperature correlations:

$$\begin{gathered} \frac{\partial \theta}{\partial t} + (\boldsymbol{u}\cdot\nabla)\,\theta = \frac{1}{Pe}\nabla^2\theta \\ \frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u} = -\Delta p + \frac{1}{Re}\nabla^2\boldsymbol{u} + Ri\theta \\ \nabla\cdot\theta = 0 \end{gathered} \tag{6}$$

Where θ is the temperature, $\boldsymbol{u} = [u,v]^T$ is the velocity and p is the pressure field and all values are dimensionless. Furthermore, as conventional, Re represents the Reynolds number, Pe the Peclet number and Ri the Richardson number. For the forced convection problem, Ri was taken to be 0, and the scope is restricted to 2D. In addition to the physical laws governing the problem defined in Equation 8, two sets of training data were available. The first from a set of measurements from velocity probes placed behind the cylinder. The second is from k temperature sensors placed on and around the cylinder. With all inputs defined, these have to be formulated into loss terms from Equation 2. Three types of penalisation aspects were defined: One for the governing equation, one for boundary conditions and one for the measurement data. These will be each explored one by one. Conforming the results to the governing equation is achieved by setting up the following residual from Equation 8:

$$e = u\frac{\partial \theta}{\partial x} + v\frac{\partial \theta}{\partial y} - \frac{1}{Pe}\left(\frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \theta}{\partial y^2}\right) \tag{7}$$

Computing this does not require any prior knowledge and can be imposed by itself to a set of random points from the predictions. It must be noted that computing the differential terms is achieved with automatic differentiation (Baydin et al. 2018) which is included in the TensorFlow library (Abadi et al. 2016). Using any other method is highly not recommended as those can become too complex. Next, the boundary conditions must be incorporated into the cost function. Velocity field measurements are compared at a number of given points (N_{ub}) to the predictions of the NN, giving the second loss term. Similarly for temperature boundaries, the environmental temperature (θ_b) is known, and is compared at ($N_{\theta b}$) points to the simulation. Finally, the points known temperature from the measurements ($\theta_{i,data}$) are compared to the predictions as well. These four terms are each expressed in order in Equation 8

$$\begin{aligned}
L_r &= \frac{1}{N_r}\sum_{j=1}^{4}\sum_{i=1}^{N_r}|e_j(x_i, y_i)|^2 \\
L_{ub} &= \frac{1}{N_{ub}}\sum_{i=1}^{N_{ub}}[\boldsymbol{u}(x_i, y_i) - \boldsymbol{u}_{i,b}]^2 \\
L_{\theta b} &= \frac{1}{N_{\theta b}}\sum_{i=1}^{N_{\theta b}}|\theta(x_i, y_i) - \theta_{i,b}|^2 \\
L_{\theta} &= \frac{1}{k}\sum_{i=1}^{k}|\theta(x_i, y_i) - \theta_{i,data}|^2
\end{aligned} \tag{8}$$

After initial training with only 5 temperature sensors, errors were as great as 10%. This hardly satisfies needs for accuracy, therefore a systematic approach for adding extra sensors was developed. The residual defined in Equation 7 can be used to find the maximum difference between desired results and predictions along the boundary of the cylinder. For better estimations at these points, extra sensors were placed in these locations, and then the model re-trained with this extra information. After only seven iterations, errors compared to proven CFD simulations were decreased to as low as 1.25%.

The paper goes on to describe similar solutions for constant flux surfaces and two phase Stefan problems, showing how only a few measurements can accurately predict Nusselt number profiles. This bridges the gap between measurements and idealised physics problems for the first time in heat transfer.

4.3 Label-free training

The method of customising the cost function described in Section 3.3.3 can be leveraged to eliminate the need for labels. Equation 2 can be set up to only depend on the inputs x_i and predictions $f(x_i)$. This allows the NN to be trained with with inputs alone, significantly reducing the cost of data acquisition. Stewart & Ermon (2016) demonstrated this by training a PINN to track the trajectory of a tossed object from a video. Their resulting function was to estimate the height of the object on each frame of the recording. For an input of N frames each with a height of h and width of w the PINN was defined as $f : \left(\mathbb{R}^{w \cdot h \cdot 3}\right)^N \to \mathbb{R}^N$. From elementary physics, the object must follow the trajectory described by:

$$y_i = y_0 + v_0 \cdot (i\Delta t) - \frac{g}{2} \cdot (i\Delta t)^2$$

Where y_i is the height of the object in the ith frame, Δt is the time between frames, v_0 is the initial velocity and g is the acceleration due to gravity. The initial height was considered to be $y_0 = 0m$ as that can be easily offset by a constant. As v_0 is unknown however, the equation cannot be used outright. The trajectory nonetheless is known to be parabolic, hence a parabola with a fixed curvature was fitted to the predictions and any deviations were penalised. This was achieved by first defining a function for the fitted curve $\hat{y}$:

$$\hat{\boldsymbol{y}} = \boldsymbol{a} + \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T \cdot (f(x) - \boldsymbol{a})$$

Where:

$$\boldsymbol{A} = \begin{bmatrix} \Delta t & 1 \\ 2 \cdot \Delta t & 1 \\ \vdots & \vdots \\ N \cdot \Delta t & 1 \end{bmatrix} \text{ and } \boldsymbol{a} = \left[\frac{-g}{2} \cdot (\Delta t)^2, \frac{-g}{2} \cdot (2 \cdot \Delta t)^2, \ldots, \frac{-g}{2} \cdot (N \cdot \Delta t)^2\right]$$

The loss function of the PINN was then constructed form only this and a regularisation term (the paper did not specify the regularisation function):

$$f_{loss}(f(x_i), f) = \sum_{i=1}^{N} |\hat{\boldsymbol{y}} - f(x_i)| + R(f)$$

The resulting PINN achieved a 94% correlation to expectations. This is a remarkable result considering how the only inputs were an equation and a video.

4.4 Alternate boundary condition constraint

Raj et al. (2022) propose solutions to general thermo-mechanics problems using PINNs. Their paper focuses on abstract physics problems, and rigorously goes through how to solve each, serving as a strong foundation for solving real-world problems. Although they provide multiple examples, here only the first will be mentioned below to illustrate how to set up a PINN as a pure PDE solver.

Consider a cantilever beam, of unit length and constant cross sectional area loaded at the end in the longitudinal direction. Let the reference frames origin be at the base of beam, x direction to be along the beam, and assume that the elasticiy of the beam is described by $E = \frac{1}{1+x}$. For a total displacement of 1, the task is to find the relative displacements of finite points $\boldsymbol{u}_i$ for a total displacement of $\boldsymbol{u}_n = 1$. The analytical solution is know to be $u(x) = \frac{x^2+2x}{3}$, so here using a PINN solution is only a demonstration.

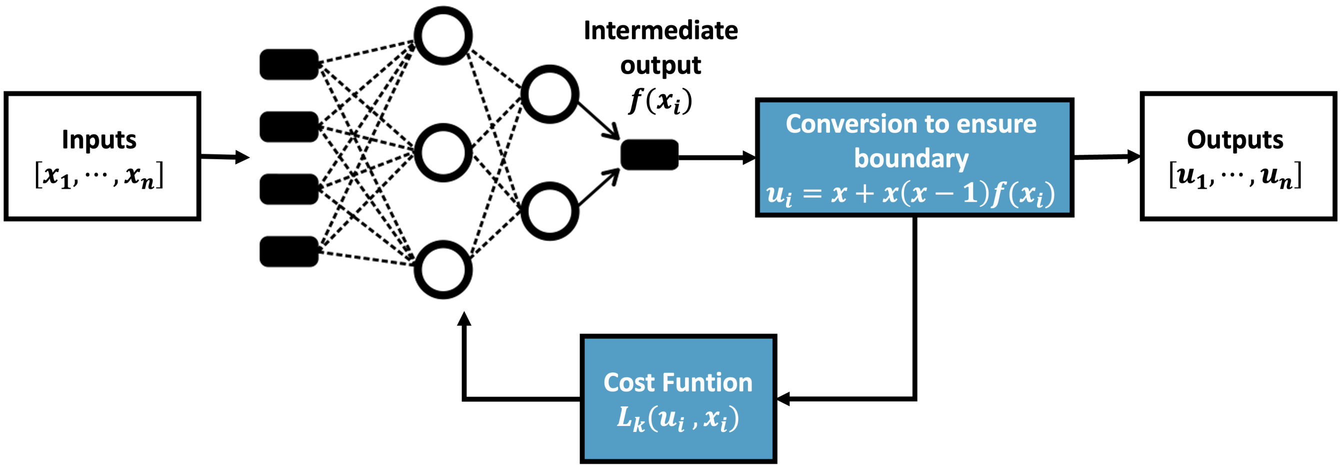
For the governing equation, the difference of the strain energy and work done by the force is taken. This is then discretised into finite points and E is substituted to find the loss function:

$$\frac{d}{dx}\left(E\frac{d}{dx}\right) = 0 \xrightarrow{\text{Discretising}} L_k(u_i, x_i) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2(x_i+1)}\left(\left.\frac{du}{dx}\right|_{x_i}\right) \tag{9}$$

Differentiation again is achieved with automatic differentiation (Baydin et al. 2018) included in the TensorFlow library (Abadi et al. 2016). Furthermore the boundary conditions are $u(0) = 0$ and $u(1) = 1$, which the resulting function f must also satisfy. Raj et al. (2022) take a drastically different approach on how to constrain the end result to the boundary conditions. Instead of adding a loss term to the cost function reflecting any deviation form the boundary, they mathematically constrain them entirely. This is achieved by adding an extra node after the output of the NN which is calculated by:

$$\boldsymbol{u} = x + x(1-x)f(x) \tag{10}$$

This will absolutely ensure $u(0) = 0$ and $u(1) = 1$. Notice how in this configuration, the NN does not predict the actual outputs, but rather some other function which will then be converted by Equation 10. The cost function will be then imposed on this converted value. Furthermore it will only consist of the term defined in Equation 9, and not use any training data similarly to the last section. The whole proscess is illustrated on Figure 6.



Inputs $[x_1, \cdots, x_n]$
Intermediate output $f(x_i)$
Conversion to ensure boundary $u_i = x + x(x-1)f(x_i)$
Outputs $[u_1, \cdots, u_n]$
Cost Funtion $L_k(u_i, x_i)$

Figure 6: Alternative boundary condition constraint on PDE solver (Based on Raj et al. (2022))

As the authors note, one must be careful when formulating Equation 10, as other variations might accidentally impose unintended constraints. For example $u = x^2 + x(1 - x^2)f(x)$ would also work at first glance, however that would also cause $u'(0)$ to be 0.

Their paper goes on to discuss multiple 1D and 2D examples including temperature gradients as well. After evaluating all models PINN models they conclude that 99% accuracy can be achieved for the target variables in each case. The prediction of secondary variables like stresses however are less precise at dipping to about 60-70% in some cases. Furthermore they go on to highlight that the PINNs are overly sensitive both to the activation function used and the convergence of losses, causing training to be troublesome.

4.5 Simulation only as training data

A NN trained on simulation data only is strictly speaking not a PIML method according to the definition in Section 3.1, as it only has one pipeline of training data. This however should not exclude such solutions from discussion, as such NNs can still augment physics based methods enormously. The work by Pfrommer et al. (2018) on the optimisation of composite textile draping serves as an excellent case study on how Physics based simulations and NNs can work in tandem.

Use of Continuous Carbon Reinforced Plastics (CoCRP) has been popularised in the automotive industry due to it's potential for weight reductions. During the manufacturing of these parts, woven fabric is stacked and pressed into a mould by grippers to form the desired 3D shape. As the shear and bending stiffness of the material is relatively low, if shear deformations during forming are not controlled, the resulting part might wrinkle. A pair of studies on a car door reinforcement beam set out to solve this issue. To predict if shearing will exceed a limiting value, Kärger et al. (2018) developed a FE model to simulate the process. In their model, the grippers constraining the part were modelled by springs. By adjusting the positions and stiffnesses of these springs, they managed to decrease the maximum shear for the specific part, thus optimising the process by 12%. Despite this improvement, shear stresses were still found to be too high for the required component. They conclude that the computational complexity of the simulation prevents the feasibility of an automatic optimisation. Pfrommer et al. (2018) proposed using a NN as a surrogate model to overcome this obstacle. Surrogate Based Modelling is a technique to approximate the results of an underlying model with another which is quicker to evaluate. Evaluating a trained NN is relatively computationally inexpensive when compared to physical simulations, making it a prime candidate for such a surrogate model.

Inputs were taken to be the springs stiffnesses for each potential gripper setup: $\boldsymbol{c} = [c_0 \ldots c_n]$. Outputs were set to be the shear angles of each element of the FE model $\boldsymbol{\gamma} = [\gamma_0 \ldots \gamma_n]$. The NN $(f : \boldsymbol{c} \rightarrow \boldsymbol{\gamma})$ will approximate the underlying function (f_p), and the aim is to find the inputs $\boldsymbol{c}$ which best optimise the output $\boldsymbol{\gamma}$. After training the model with simulations using a relatively wide range of inputs, f will approximate the general shape of f_p, however will still be inaccurate for specific examples. Accepting some inaccuracy, the NN is used to initially optimise the selection of the gripper setup. This can be achieved by defining a merit function which quantifies the quality of the output: $\rho : \gamma \rightarrow \mathbb{R}$ and then finding:

$$\boldsymbol{c} = \text{arg max } \rho(f(\boldsymbol{c}))$$

This is then evaluated with the physics based simulation $(f_p(\boldsymbol{c}))$ and added to the training data. The model is re-trained then, re-trained resulting in a better approximation of f_p around the optimum. Next, the new f is optimised again and the process is repeated. The whole iterative process is shown on Figure 7.

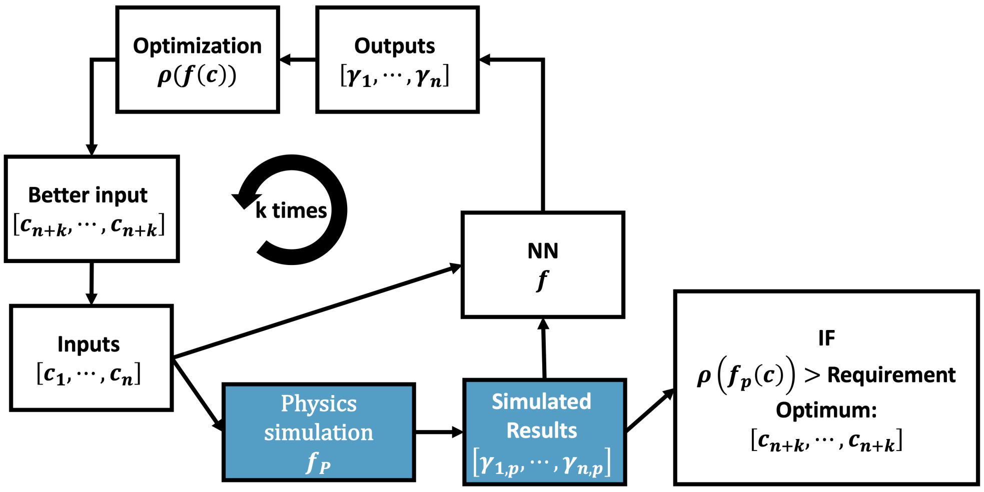

Optimization
$\rho(f(c))$
Outputs
$[\gamma_1, \cdots, \gamma_n]$
Better input
$[c_{n+k}, \cdots, c_{n+k}]$
k times
NN
f
Inputs
$[c_1, \cdots, c_n]$
Physics simulation
f_P
Simulated Results
$[\gamma_{1,p}, \cdots, \gamma_{n,p}]$
IF
$\rho\left(f_p(c)\right) >$ Requirement
Optimum:
$[c_{n+k}, \cdots, c_{n+k}]$

Figure 7: Surrogate modelling (Based on Pfrommer et al. (2018))

After only 19 cycles they managed to decrease the shear angles below the maximum allowed before wrinkling, solving the original problem of the car door reinforcement beam. Contrary to examples so far, this result is not novel work on some simplified problem however an actual use-case from industry, proving the potential of PIML methods.

4.6 Transfer learning

Continuing with the topic of fibre reinforced plastics, once the manufacturing is complete, residual stresses from forming can result in cracks which reduce the strength significantly. Zhou et al. (2021) developed a PINN to predict the uncertainties in strengths of such materials. Inputs similarly to the last section were only from simulations, however the inputs were taken as an image format. Generally such estimations are difficult to accurately carry out as data acquisition is extremely resource intensive. One could use scanning electron microscopes to capture microsturctures of physical materials, however capturing a statistically representative amount of data requires enormous amounts of sample materials, which is costly. Alternatively, numerical generation of Representative Volume Elements (RVE) can simulate the batch of sample materials with set distribution of properties. To find the strength of each sample, an FEA model can be employed, however the computational burden makes it impossible to create a large dataset. Using a pre-trained CNN, Zhou et al. (2021) manage to accurately model uncertainties with only a small set of training examples.

For cases where training data is scarce, ML models that have been previously trained on similar data can be used as a starting point and then re-trained for specific purpose. This is called transfer learning. (Tammina 2019) For this particular case, the RVE model produces images of the fibre structure, which must be mapped to a strength value. The Visual Graphics Group developed an award winning CNN for the recognition of everyday object on images, which is chosen as the base model. (Simonyan & Zisserman 2015) The exact changes to the model are described in the original paper and will not be discussed here, only the principles. With Deep NNs low level layers tend to be similar for related applications, while the top level layers are what solve the specific problem. The authors had no specific methodology to determine which layers to train and which to keep frozen with the pre-trained values, so a trial-and-error based approach was adopted. As a result only the top 2 layers of the model were trained (about only 2.6% of the entire model). Instead of labels of everyday objects this changed the mappings to strength values. The newly trained NN could now serve as a surrogate model for the FEA simulation, allowing for a wide range of inputs to be tested. The resulting distribution of strengths gave the uncertainty associated with strengths of newly manufactured parts.

As a benchmark, in addition to the transfer learning model, a whole new, custom CNN was developed in parallel. Both models successfully achieve desired accuracies without any overfitting, successfully achieving the task. The custom CNN slightly out-preformed the transfer learning model, both in accuracy and training time. This however is not to be discouraging as the development time needed for the transfer learning model is significantly less and it's applicability is more general. Considering that VGG16 was trained on everyday objects which have nothing to do with RVE simulations, the outcome is remarkable. This work serves as an invaluable case study on how transfer learning can be applied to PIML methods.

4.7 Knowledge Graph for reasoning

Moving on from physical laws and other relations that can be quantified in an algebraic equation, one might ask how to incorporate basic common sense into a ML method. In computer science basic human reasoning is usually incorporated into knowledge graphs. Knowledge graphs are made up of so called *Triples*, which consist of two nodes connected by a directed edge. The initial node is termed the *Subject* and the target node is the *Object*, while the directed edge represents the *Relationship* between the two. With this formulation each triple is usually defined to be read out as a sentence: $\boxed{Elephant} \xrightarrow{has} \boxed{Trunk}$. Large collections of these triples can be used to represent complex ideas, and transversing them can lead to discovery of implied knowledge. An illustrative example of how these can be used is how, Marino et al. (2017) use knowledge graphs with similar statements for image classification.

Such logic representations can also be useful in the case of physics problems. Battaglia et al. (2016) model dynamical systems with multiple bodies implementing knowledge graphs to model their interactions. Say a configuration is chosen where a single objects trajectory were described by a single NN $f_o : o_{i,t} \to o_{i,t+1}$. Where $o_{i,t}$ represents the state of the i-th object at time t. If a second object is added, and the same NN applied to describe its motion, the system will break down as interactions between them would not be modelled. To solve this, first the type of interaction has to be identified, then the subsequent relations must be enforced. These can be easily incorporated into a knowledge graph, which for example differentiates colliding objects from springs connected to bodies and points to the correct physical formula for each. Once these are identified a so called relation-centric function f_r takes the two objects and their relations and applies the correct physical rules. This outputs the effects of the interaction e_t, which o_1 imposes on o_2. The mathematical definition of the relation-centric function is therefore

$$f_r : (o_{1_t}, o_{2_t}, r) \to e_{t+1}$$

Where r is the interaction or relation between the two objects which the knowledge graph specified. Next the outputs of f_r and the original inputs from the last state are transferred to f_o, the so called object-centric function, which will then compute the next state:

$$f_o : (o_{2_t}, e_{t+1}) \to o_{2,t+1}$$

Here the framework is only shown for two objects, however the paper has the derivation of the general case of N objects. The entire architecture is illustrated on Figure 8.

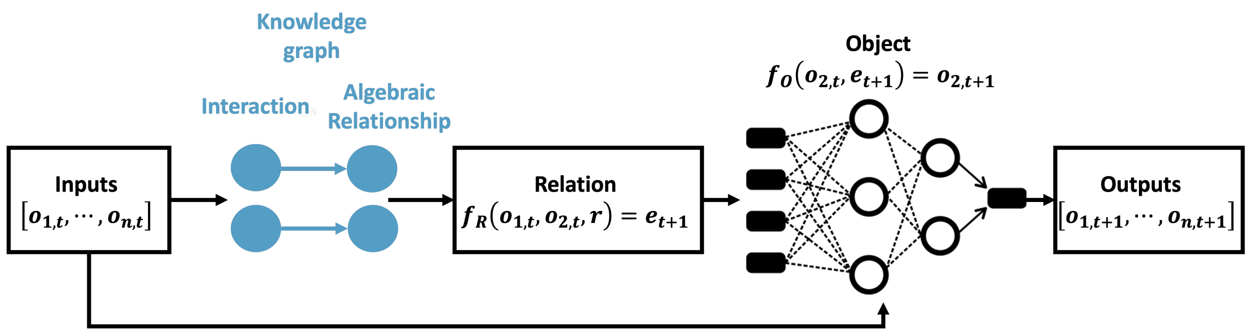

Knowledge graph
Interaction
Algebraic Relationship
Inputs
$[o_{1,t}, \cdots, o_{n,t}]$
Relation
$f_R(o_{1,t}, o_{2,t}, r) = e_{t+1}$
Object
$f_O(o_{2,t}, e_{t+1}) = o_{2,t+1}$
Outputs
$[o_{1,t+1}, \cdots, o_{n,t+1}]$

Figure 8: Knowledge graph implementation (Based on Battaglia et al. (2016))

This format was tested on a set of 2D scenarios with balls colliding, orbiting each other and colliding with strings. Simulation data from a physics engine as served as an input alone. The results when rendered speak for themselves. To the human eye they are identical to the original simulations. The most significant finding however was that this model could predict thousands of steps with great accuracy even if it was trained only on single steps.

Note that multiple other types of implementations using knowledge graphs are viable, so before developing any, the reader is advised to go over examples provided by von Rueden et al. (2021).

4.8 Directory of research papers

Literature surveys by von Rueden et al. (2021) and Willard et al. (2020) both contain a extensive tables of literature grouped by methods, objectives and knowledge sources. Their work both identifies more method types and segregate them further compared to this review. Readers interested in one specific type of implementation are recommended to start further research there. In Table 1 below, a more concise version is presented, with all discussed examples and a few supplementary.

	Training Data	Network Architecture	Learning algorithm
Simulation results	Zhou et al. (2021) Daw et al. (2021) Pfrommer et al. (2018)	Battaglia et al. (2016)	-
Spacial Invariance	(Niyogi et al. 1998)	Ling et al. (2016) Butter et al. (2018)	-
Known Equations	-	Raj et al. (2022)	(Karniadakis et al. 2021) Cai, Wang, Wang, Perdikaris & Karniadakis (2021) Stewart & Ermon (2016) (Amini Niaki et al. 2021) (Zhang & Gu 2021)

Table 1: Classification of research papers

5 Outlook

After presenting examples of uses, implementation types and features of PINN models, the scope is broadened. First the reliability of these methods will be discussed for completely new inputs or problems. Next a few alternatives for NNs will be discussed. As mentioned in Section 3.1, using NNs is not the only option for a PIML method. It is the most versatile, and most commonly used, which is why it was the main focus so far, however one must be aware of alternatives. Two such examples will be brought: using SVM kernel methods and evolutionary symbolic regression.

5.1 Reliability of predictions

When using classical data driven methods, a rule of thumb is to always only interpolate estimates as extrapolations might lead to significant errors. Once some form of physics knowledge is introduced to the network to make it a PINN, one might ask if this should still be followed. Shouldn't the extra secondary pipeline of data carry enough weight to make the model valid outside the original dataset? How does this depend on the problem itself? Kim et al. (2021) set out to answer some of these questions. Their primary focus is the 1D version of the Burgers Equation solution from Section 4.1. If the training were perfect, the function should accurately predict values across the whole input space. They find however, that in this case, the extrapolated values produce significantly larger errors when compared to interpolation. To better understand how this could happen they individually found the values from each of the two loss terms in Equation 5. The findings show that the L_l term responsible for constraining solutions to the boundary conditions converges to 0 in only a few epochs and does not change. The L_k term on the other hand, which is to enforce the governing equation, refuses to converge and it's value is inconsistent along epochs.

With the root cause identified, they go on to develop a novel training algorithm called the "Dynamic Pulling Method" to overcome this. The concrete mathematics of the method is best read from the original paper, however the basic idea is simple. They define a hyper-parameter ϵ above which they do not allow the L_k term. This is achieved by tweaking the gradient based optimisers dynamically through epochs to ensure the decrease of L_k below ϵ. Naturally this might come at an expense of the L_l term, however, this is deemed a fair trade off. The resulting algorithm achieves 72% lower errors when compared to the original. After success with the 1D Burgers equation they employ the same method for the nonlinear Schrödinger's equation and Allen–Cah equation which have mixed results. Although the successfully augmented models in this work are rather simple, it paves the way for optimiser development for PINNs in the future. Upon successful development of such a general method, extrapolations could also potentially be trusted, but for now care still must be taken.

5.2 Failure modes

PIML methods unfortunately also come with their own limitations. Krishnapriyan et al. (2021) set out to characterise the possible failure modes of PINN methods relying on custom cost functions and sadly found significantly troublesome results. Similarly to Kim et al. (2021) from the previous section, they also looked at a relatively simple model with 2 loss terms, with the same format. One constraining the boundary conditions and one the governing equation. The only difference was that in their particular case the 1D diffusion equation was examined instead of the Burger's equation. They too identified that the main errors are generated by the L_k term. This intuitively should be correct, as this term usually involves a complex differential operator, which makes the cost function particularly difficult to optimise.

Investigating further, they find that just by changing the constants of the original PDE, relative errors can jump up to as high as 93%. Additionally, the weight of the term λ_k from Equation 2 also highly influences trainablility of the model. Increasing λ_k causes the optimisation of the NN to run smoother, however at the cost of it conforming to the governing equation less. They conclude that the correct choice of λ_k can significantly improve matters however it cannot completely alleviate them.

Two solutions are proposed. Firstly, the curriculum regularisation training method involves training a NN first on a relatively simpler problem and then upping the difficulty with each re-training. With this technique the PINN was found to be able to approximate the function properly, reducing relative errors by 1-2 orders of magnitude. Secondly, sequence-to-sequence learning can also be leveraged. This involves predicting each time step one-by-one, instead of the whole sequence at once. This means that the PINN first predicts only the first time step from the boundary condition and the governing equation. Next, the prediction for the first time step is taken to be the boundary condition of the second step and so on. The produced results are similarly 1-2 orders of magnitude less compared to the original models.

5.3 Alternate PIML methods

5.3.1 Support Vector Machines

Apart from neural networks, Support Vector Machines (SVM) can also be informed with prior knowledge. Although main focus has shifted to NNs in the past decade, the accuracy of SVM models on small training samples can sometimes be beneficial. Lauer & Bloch (2008) provide an excellent survey of all methods, including mathematical descriptions which can serve as a base for further research. Examples of notable recent uses include how Deist et al. (2019) use simulation based kernel methods and SVM methods to better personalise cancer treatment, and how Jhong et al. (2022) forcasted floods during typhoons in Taiwan.

5.3.2 PIML for physics research

Although this paper has mainly focused on PINNs in engineering applications, in a more broader sense machine learning also shows great potential in classical physics applications. The methods discussed thus-far aimed to model a specific system given some data and laws to conform to. The result of each model was only an approximation to some underlying physical relationship, which is too complex to find analytically. Shifting perspectives, one could also take finding the analytical function as the task. Work by Schmidt & Lipson (2009), show that using evolutionary symbolic regression, one can look for analytical functions for which a set of training data is invariant. In simple terms the algorithm makes a guesses for a governing function of the system and then substitutes the training data into them. It incrementally changes the guess until a function is found for which all inputs give an approximately invariant value. They prove the concept by correctly identifying the equation of motion of a double pendulum, nonlinear energy conservation laws and Newtonian force laws from only experimental data. Their work proves the concept that machine learning methods can be used to discover physical laws from observations alone. For future researchers similar techniques could become useful tools to find new relationships.

6 Discussion

It is evident that despite all technological advancements, today's physics based simulation tools are highly constrained by computational capacity as well as lack of pipelines for experimental data. PIML methods promise efficient computation times mixed with high versatility. This makes PIML a top prospect for solving the hurdles of physics based simulations and purely data driven methods. If it can fully deliver on these promises however remains to be seen.

In cases where the training data drives the "informing" of the network, concrete, pragmatic examples have been shown for it's uses. Daw et al. (2021) manage to improve predictions of lake temperatures with hybrid training data, Pfrommer et al. (2018) successfully optimise composite manufacturing and Zhou et al. (2021) achieve better estimates for composite strength with transfer learning. These works all solve real-world problems and already show how PIML methods can be superior to simulations. All these solutions are somewhere more to the left side of the diagram on Figure 1 as they are more data driven than physics driven.

For the cases where the cost function drives the "informing", most examples cited are only novel work proving the concept of PIML. It must be noted that in this review the easier to understand, illustrative problems have been covered, as the objective was to give an overview feel for all possibilities that PIML enables, without getting lost in details. Most cited papers had more complex problems solved in them, however these were also frequently 1 or 2 dimensional and simplified. The two papers on assessing PINNs cited in Section 5, both show some doubt about PINNs accuracy when extrapolated or when more complex problems are tackled. This is not caused by an intrinsic fault of the methodology itself, but rather the training methods used. In the whole are of machine learning, maybe the greatest challenge is posed by finding the global minima of the cost function. Algorithms based on gradient descent, like the Adam Optimiser (Kingma & Ba 2017) and L- BFGS (Byrd et al. 1995) have had enormous successes however can still struggle with highly non-convex cost functions. (Karniadakis et al. 2021) The PIML methods proposed rely on creating highly complex cost functions, therefore are difficult to properly optimise. Although both Kim et al. (2021) and Krishnapriyan et al. (2021) propose some encouraging solutions, neither seems general enough for a major breakthrough. Additionally Raj et al. (2022) (Section 4.4) voice concerns over how much the result is influenced by the activation function chosen even for complex, but highly simplified from reality problems.

All this is not to say that PIML is just a fancy technique which isn't scalable. It definitely has a bright future, however the mathematical framework has to be better understood, and more empirical studies have to be concluded. At the moment, the PINN methods using custom loss functions is more of a research area, while coupled systems of NN-s working with simulations can be more readily employed in industry.

7 Conclusion

Overall PIML is an extremely promising approach for a new-generation of simulations to augment and/or replace traditional physics based simulations. In theory their ability to incorporate multiple types of input sources and pipelines makes it applicable to more problems and fields. Ill-posed problems can be dealt with, measurements can be incorporated and deep learning libraries in python make them more accessible and relatively easy to develop. In practice however unfortunately the some PIML techniques currently suffer from significant limitations. Despite custom cost functions facilitating huge potential, with today's training methods, correct optimisation cannot be ensured for complex problems. The multiple pieces of research presented on this topic in this review clearly highlight that these can be overcome, however further research is required. On the other hand, more data driven PIML methods like surrogate modelling, and the use of hybrid training data proves to be effective tool for lowering computational and data accusation costs.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. & Zheng, X. (2016), {TensorFlow}: A System for {Large-Scale} Machine Learning, pp. 265–283.

Amini Niaki, S., Haghighat, E., Campbell, T., Poursartip, A. & Vaziri, R. (2021), 'Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture', *Computer Methods in Applied Mechanics and Engineering* **384**, 113959.
URL: *https://www.sciencedirect.com/science/article/pii/S0045782521002966*

Battaglia, P., Pascanu, R., Lai, M., Jimenez Rezende, D. & kavukcuoglu, k. (2016), Interaction Networks for Learning about Objects, Relations and Physics, *in* 'Advances in Neural Information Processing Systems', Vol. 29, Curran Associates, Inc.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A. & Siskind, J. M. (2018), 'Automatic differentiation in machine learning: a survey', *Journal of Marchine Learning Research* **18**, 1–43. Publisher: Microtome Publishing.

Butcher, J. C. (1996), 'A history of Runge-Kutta methods', *Applied Numerical Mathematics* **20**(3), 247–260.

Butter, A., Kasieczka, G., Plehn, T. & Russell, M. (2018), 'Deep-learned top tagging with a Lorentz layer', *SciPost Physics* **5**(3), 028.

Byrd, R. H., Lu, P., Nocedal, J. & Zhu, C. (1995), 'A Limited Memory Algorithm for Bound Constrained Optimization', *SIAM Journal on Scientific Computing* **16**(5), 1190–1208. Publisher: Society for Industrial and Applied Mathematics.

Cai, S., Mao, Z., Wang, Z., Yin, M. & Karniadakis, G. E. (2021), 'Physics-informed neural networks (PINNs) for fluid mechanics: a review', *Acta Mechanica Sinica* **37**(12), 1727–1738.

Cai, S., Wang, Z., Wang, S., Perdikaris, P. & Karniadakis, G. E. (2021), 'Physics-Informed Neural Networks for Heat Transfer Problems', *Journal of Heat Transfer* **143**(6).

Chen, S., Merriman, B., Osher, S. & Smereka, P. (1997), 'A Simple Level Set Method for Solving Stefan Problems', *Journal of Computational Physics* **135**(1), 8–29.

Daw, A., Karpatne, A., Watkins, W., Read, J. & Kumar, V. (2021), 'Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling'.

Deist, T. M., Patti, A., Wang, Z., Krane, D., Sorenson, T. & Craft, D. (2019), 'Simulation-assisted machine learning', *Bioinformatics* **35**(20), 4072–4080.

Diligenti, M., Roychowdhury, S. & Gori, M. (2017), Integrating Prior Knowledge into Deep Learning, *in* '2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)', pp. 920–923.

Fabregat, A., Gisbert, F., Vernet, A., Dutta, S., Mittal, K. & Pallarès, J. (2021), 'Direct numerical simulation of the turbulent flow generated during a violent expiratory event', *Physics of Fluids* **33**(3).

Fix, G. J. (1982), 'Phase field methods for free boundary problems'. Publisher: Carnegie Mellon University.

Friedman, A. (2000), 'Free boundary problems in science and technology', *Notices of the AMS* **47**(8), 854–861.

Hornik, K., Stinchcombe, M. & White, H. (1989), 'Multilayer feedforward networks are universal approximators', *Neural Networks* **2**(5), 359–366.

Hughes, T. J. R. (1987), 'Recent progress in the development and understanding of SUPG methods with special reference to the compressible Euler and Navier-Stokes equations', *International Journal for Numerical Methods in Fluids* **7**(11), 1261–1275.

Jasak, H., Jemcov, A., Tukovic, Z. & others (2007), OpenFOAM: A C++ library for complex physics simulations, *in* 'International workshop on coupled methods in numerical dynamics', Vol. 1000, IUC Dubrovnik Croatia, pp. 1–20.

Jhong, B.-C., Lin, C.-Y., Jhong, Y.-D., Chang, H.-K., Chu, J.-L. & Fang, H.-T. (2022), 'Assessing the effective spatial characteristics of input features through physics-informed machine learning models in inundation forecasting during typhoons', *Hydrological Sciences Journal* **67**(10), 1527–1545. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/02626667.2022.2092406.

Karniadakis, G. E., Karniadakis, G. & Sherwin, S. (2005), *Spectral/hp Element Methods for Computational Fluid Dynamics: Second Edition*, OUP Oxford.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S. & Yang, L. (2021), 'Physics-informed machine learning', *Nature Reviews Physics* **3**(6), 422–440.

Katz, A. J. (2009), *Meshless methods for computational fluid dynamics*, Stanford University.

Kim, J., Lee, K., Lee, D., Jhin, S. Y. & Park, N. (2021), 'DPM: A Novel Training Method for Physics-Informed Neural Networks in Extrapolation', *Proceedings of the AAAI Conference on Artificial Intelligence* **35**(9), 8146–8154.

Kingma, D. P. & Ba, J. (2017), 'Adam: A Method for Stochastic Optimization'.

Kononenko, I. (2001), 'Machine learning for medical diagnosis: history, state of the art and perspective', *Artificial Intelligence in Medicine* **23**(1), 89–109.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R. & Mahoney, M. W. (2021), Characterizing possible failure modes in physics-informed neural networks, *in* 'Advances in Neural Information Processing Systems', Vol. 34, Curran Associates, Inc., pp. 26548–26560.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2017), 'ImageNet classification with deep convolutional neural networks', *Communications of the ACM* **60**(6), 84–90.

Kutz, J. N. (2017), 'Deep learning in fluid dynamics', *Journal of Fluid Mechanics* **814**, 1–4.

Kärger, L., Galkin, S., Zimmerling, C., Dörr, D., Linden, J., Oeckerath, A. & Wolf, K. (2018), 'Forming optimisation embedded in a CAE chain to assess and enhance the structural performance of composite components', *Composite Structures* **192**, 143–152.

Lauer, F. & Bloch, G. (2008), 'Incorporating prior knowledge in support vector machines for classification: A review', *Neurocomputing* **71**(7), 1578–1594.

Lecun, Y. (2001), 'A Theoretical Framework for Back-Propagation'.

Ling, J., Kurzawski, A. & Templeton, J. (2016), 'Reynolds averaged turbulence modelling using deep neural networks with embedded invariance', *Journal of Fluid Mechanics* **807**, 155–166.

Marino, K., Salakhutdinov, R. & Gupta, A. (2017), 'The More You Know: Using Knowledge Graphs for Image Classification'.

Meng, C., Seo, S., Cao, D., Griesemer, S. & Liu, Y. (2022), 'When Physics Meets Machine Learning: A Survey of Physics-Informed Machine Learning'.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013), 'Efficient Estimation of Word Representations in Vector Space'.

Niyogi, P., Girosi, F. & Poggio, T. (1998), 'Incorporating prior information in machine learning by creating virtual examples', *Proceedings of the IEEE* **86**(11), 2196–2209.

Noether, E. (1918), 'Invarianten beliebiger Differentialausdrücke', *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse* **1918**, 37–44.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. (2019), PyTorch: An Imperative Style, High-Performance Deep Learning Library, *in* 'Advances in Neural Information Processing Systems', Vol. 32, Curran Associates, Inc.

Pfrommer, J., Zimmerling, C., Liu, J., Kärger, L., Henning, F. & Beyerer, J. (2018), 'Optimisation of manufacturing process parameters using deep neural networks as surrogate models', *Procedia CIRP* **72**, 426–431.

Raj, M., Kumbhar, P. & Annabattula, R. K. (2022), 'Physics-informed neural networks for solving thermo-mechanics problems of functionally graded material'.

Schmidt, M. & Lipson, H. (2009), 'Distilling Free-Form Natural Laws from Experimental Data', *Science* **324**(5923), 81–85.

Simonyan, K. & Zisserman, A. (2015), 'Very Deep Convolutional Networks for Large-Scale Image Recognition'.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), 'Dropout: a simple way to prevent neural networks from overfitting', *The journal of machine learning research* **15**(1), 1929–1958.

Stewart, R. & Ermon, S. (2016), 'Label-Free Supervision of Neural Networks with Physics and

Domain Knowledge'. arXiv:1609.05566 [cs].

Tammina, S. (2019), 'Transfer learning using vgg-16 with deep convolutional neural network for classifying images', *International Journal of Scientific and Research Publications (IJSRP)* **9**(10), 143–150.

von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., Walczak, M., Garcke, J., Bauckhage, C. & Schuecker, J. (2021), 'Informed Machine Learning – A Taxonomy and Survey of Integrating Knowledge into Learning Systems', *IEEE Transactions on Knowledge and Data Engineering* pp. 1–1.

Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C. & Xu, W. (2016), CNN-RNN: A Unified Framework for Multi-Label Image Classification, pp. 2285–2294.

Wang, S., Yu, X. & Perdikaris, P. (2020), 'When and why PINNs fail to train: A neural tangent kernel perspective'.

Willard, J., Jia, X., Xu, S., Steinbach, M. & Kumar, V. (2020), 'Integrating physics-based modeling with machine learning: A survey', *arXiv preprint arXiv:2003.04919* **1**(1), 1–34.

Zhang, Z. & Gu, G. X. (2021), 'Physics-informed deep learning for digital materials', *Theoretical and Applied Mechanics Letters* **11**(1), 100220.
URL: *https://www.sciencedirect.com/science/article/pii/S2095034921000258*

Zhou, K., Sun, H., Enos, R., Zhang, D. & Tang, J. (2021), 'Harnessing deep learning for physics-informed prediction of composite strength with microstructural uncertainties', *Computational Materials Science* **197**.

Zienkiewicz, O. C., Taylor, R. L. & Zhu, J. Z. (2005), *The Finite Element Method: Its Basis and Fundamentals*, Elsevier.