physical sciences forum

MDPI

Proceeding Paper

Equivariant Neural Networks and Differential Invariants Theory for Solving Partial Differential Equations †

Pierre-Yves Lagrave [1,*,‡] and Eliot Tron [2,‡,§]

[1] Thales Research and Technology, 91767 Palaiseau, France
[2] Ecole Normale Supérieure, 69342 Lyon, France
* Correspondence: pierre-yves.lagrave@thalesgroup.com
† Presented at the 41st International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering, Paris, France, 18–22 July 2022.
‡ These authors contributed equally to this work.
§ The author contributed to this work during an internship at Thales Research and Technology in 2021.

Abstract: This paper discusses the use of Equivariant Neural Networks (ENN) for solving Partial Differential Equations by exploiting their underlying symmetry groups. We first show that Group-Convolutionnal Neural Networks can be used to generalize Physics-Informed Neural Networks and then consider the use of ENN to approximate differential invariants of a given symmetry group, hence allowing to build symmetry-preserving Finite Difference methods without the need to formally derivate corresponding numerical invariantizations. The benefit of our approach is illustrated on the 2D heat equation through the instantiation of an SE(2) symmetry-preserving discretization.

Keywords: geometric deep learning; equivariant neural networks; partial differential equations; differential invariants; physics informed machine learning

Citation: Lagrave, P.-Y.; Tron, E. Equivariant Neural Networks and Differential Invariants Theory for Solving Partial Differential Equations. *Phys. Sci. Forum* **2022**, *5*, 13. https://doi.org/10.3390/psf2022005013

Academic Editors: Frédéric Barbaresco, Ali Mohammad-Djafari, Frank Nielsen and Martino Trassinelli

Published: 7 November 2022

1. Introduction

Numerically solving Partial Differential Equations (PDEs) is of paramount importance for a wide range of applications such as physics, crowd theory, epidemiology and quantitative finance. Conventional methods such as Finite Element or Finite Difference methods have the main advantage of being easy to implement but are highly time consuming. With the rise of Deep Learning in the past decade, new approximate methods based on Physics-Informed Neural Networks (PINN) have been developed [1–3] and allow to significantly improve simulation capacities [4,5].

Nonetheless, usual PDEs typically exhibit symmetries [6,7], and it is therefore natural to expect numerical solving schemes to comply with those. For Hamiltonian systems, symplectic integrators [8–11] have been introduced and have also recently been combined with machine learning techniques for the sake of efficiency [12]. For more general PDEs, symmetry-preserving Finite Difference schemes have been proposed [13,14], with the underlying theory being consolidated in [15]. Practical applications showing improvements with respect to the conventional approach have been presented in [16,17]. However, the formal derivation of the required numerical invariantization of the differential operators becomes more and more challenging as the number of variables increases, hence limiting the applicability of these methods and motivating the need for alternative approaches.

There are mainly two ways to imprint Deep Learning algorithms with symmetries. The first one, recently explored in [18] for PDEs solving, generalizes the data augmentation techniques widely used for image processing tasks and aims at learning symmetries directly from the data. The second one aims at directly encoding the symmetries within the learning algorithms by leveraging the emerging field of Geometric Deep Learning [19,20]. In this context, Equivariant Neural Networks (ENN), initially introduced in [21], have been shown very efficient, leveraging generalized convolution operators such as Steerable

Convolution or $G-$convolution [22–25], and therefore providing equivariance to a wide range of symmetry groups. These equivariance mechanisms are very appealing, as proving theoretical guarantees with respect to the algorithms response to inputs variations and have been shown more efficient than data augmentation techniques from both theoretical [26] and empirical standpoints [27] in several contexts. Yet, these architectures cannot be applied directly to PDEs solving as one would for a conventional PINN and, at the time of writing, only [28] proposes to use steerable convolution to solve PDEs, but it is limited to special cases of symmetries.

Contributions

In this paper, we present two innovative ways of using ENN to solve PDEs while exploiting the associated symmetries. By anchoring in [29], we first show that Group-Convolutionnal Neural Networks can be used to generalize the PINN architecture to encode generic symmetries. By leveraging differential invariant theory [6], we then propose using ENN to approximate differential invariants of a given symmetry group, hence allowing to build symmetry-preserving Finite Difference methods without the need to formally derivate corresponding numerical invariantizations. A key advantage of this approach is that it allows solving any other PDE with the same symmetry group without any retraining. Finally, we illustrate the interest of our approach on the 2D Heat Equation and show in particular that a set of fundamental differential invariants of the roto-translation group SE(2) can be efficiently approximated by ENN for arbitrary functions by training on simple bivariate polynomials evaluations, allowing to easily build SE(2) symmetry-preserving discretization schemes.

2. PDEs and Symmetries

2.1. Systems of PDEs

We are interested in the following in solving systems of PDEs involving one time variable t, p independent space variables $(x_1, \dots, x_p) = x \in \mathcal{X}$ and q dependent variables $(u_1, \dots, u_q) = u \in \mathcal{U}$, for which a solution is of the form $u = f(t, x)$, with $u^j = f^j(t, x)$ for $j = 1, \dots, q$ in terms of components. In the following, we denote by $\mathcal{X} = \mathbb{R}^p$, with coordinates $(x_1, \dots, x_p)$, the space of the independent variables, and by $\mathcal{U} = \mathbb{R}^q$, with coordinates u, that of the dependent variable.

We call $n-$order jet space $J^{(n)}$ the Cartesian product between the space of the independent variables $\mathcal{X}$ and enough copies of the space of the dependent variables $\mathcal{U}$ to include coordinates for each partial derivative of order less or equal than n

$$J^{(n)} = \mathcal{X} \times \underbrace{\mathcal{U} \times \ldots \times \mathcal{U}}_{\binom{p+n}{n}} \tag{1}$$

In the above definition, the binomial coefficient $\binom{p+n}{n}$ corresponds to the number of partial derivatives (assumed to be smooth enough) with order less than or equal to n. A function $f : \mathcal{X} \to \mathcal{U}$ represented as $u = f(x)$ can naturally be prolonged to a function $u^{(n)} = f^{(n)}(x)$ from $\mathcal{X}$ to $J^{(n)}$ by evaluating f and the corresponding partial derivatives, so that $u^{(n)} = \{\partial^\alpha_x u, |\alpha| \leq n\}$ with $\partial^\alpha_x u$ is the spatial cross-derivative corresponding to the multi-index $\alpha = (i_1, \dots, i_p) \in \mathbb{N}^p$. According to this formalism, a PDEs system can be then written as

$$\Delta\left(t, x, u^{(n)}\right) = 0 \tag{2}$$

where Δ is an operator from $\mathbb{R}^+ \times J^{(n)}$ to $\mathbb{R}^q$.

2.2. Symmetry Group and Differential Invariants

We consider a Lie group G of dimension m acting as $g.(x, u)$ on a sub-manifold $\mathcal{M} \subseteq \mathcal{X} \times \mathcal{U}$, with its Lie algebra $\mathfrak{g}$ generated by the vector fields $\zeta_1, \dots, \zeta_m$. We can define the transform of a function $u = f(t, x)$ under the action of G by identifying f with its graph

Γ_f and by defining $g.f = f_g$, where the function f_g is the function associated with the transformed graph $g.\Gamma_f = \Gamma_{f_g}$.

A symmetry group G of a PDEs system is a group G such that if f is a solution, then its transform f_g by the group action is also a solution. We then denote by $\mathrm{pr}^{(n)} G$ the prolongation of the group action of G to $J^{(n)}$ for which a prolonged transform $g^{(n)}$, for $g \in G$, sends the graph $\Gamma_{f^{(n)}}$ onto $\Gamma_{(g.f)^{(n)}}$ and by $\mathrm{pr}^{(n)} \zeta_1, \ldots, \mathrm{pr}^{(n)} \zeta_m$ the corresponding prolonged vector fields. The algebraic invariants I_G of the prolonged group action $\mathrm{pr}^{(n)} G$ are called the differential invariants of order n of the group G and can be obtained by leveraging the infinitesimal invariance criteria $\mathrm{pr}^{(n)} \zeta_i I_G = 0$. A complete set of independent differential invariants of order n in the sense of Theorem 2.17 of [6] is generically denoted by $\partial\phi^G_{u,n} = \left\{\partial\phi^{G,1}_{u,n}, \ldots, \partial\phi^{G,k}_{u,n}\right\}$ in the sequel and is related to the symmetry group of PDEs systems as illustrated in Section 4.2.

3. Equivariant Neural Networks

To incorporate the symmetry information of PDEs into the neural network solver, it is of capital importance to introduce equivariance into neural networks. Multiple approaches have been studied these past years. Those approaches can be separated into two categories: *G-CNN* and *Steerable CNN*. The first method is the one we chose to work with and to generalize. The second one can be explored in [23,28,30–32].

3.1. G-CNN

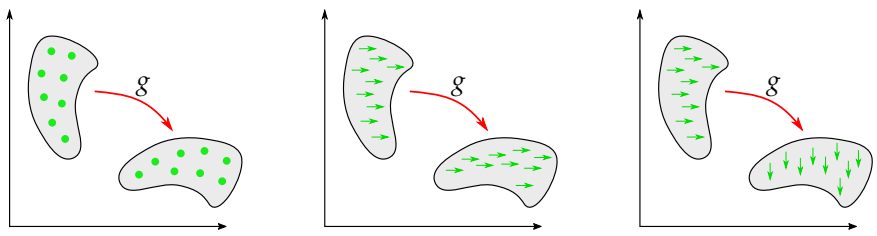
The idea behind a G-CNN is to perform the convolution over the group G one wants equivariance with. These kind of convolution layers were first introduced by Cohen and Welling in [33] for discrete groups. There has been some important work on generalizing this approach to other groups [29,34–36]. Let us first start with some reminders about the group-based convolution operator and its properties.

Definition 1 (Group Convolution)**.** *Let G be a compact Group and V_1, V_2 two vector spaces. Let $K : G \to Ł(V_1, V_2)$ be a kernel, $f : G \to V_1$ be a feature function and μ the Haar measure on G. We define the* group convolution *for any $s \in G$ by*

$$(K * f)(s) = \int_G K\left(r^{-1}s\right) f(r) d\mu(r).$$

Proposition 1. *If the actions of G on V_1^G and V_2^G has regular representations, then the group convolution defined in Figure 1 is G-equivariant.*

As illustrated in Figure 1b, regular representations only allow to describe limited group actions. Indeed, this group convolution does not revoke the constraint on the kernel (see [36]) if one wants equivariance to all kinds of actions and not only ones with regular representations.



(**a**) (**b**) (**c**)

Figure 1. Action of G with various representations. (**a**) Regular representation $\rho(g) = \mathrm{id}$ on scalars. (**b**) Regular representation $\rho(g) = \mathrm{id}$ on vectors. (**c**) Non-regular representation on vectors.

3.2. A New Convolution

Definition 2 (Representative Group Convolution)**.** *Let G be a compact Group and V_1, V_2 two vector spaces. Let $K : G \to Ł(V_1, V_2)$ be a kernel, $f : G \to V_1$ be a feature function and μ the Haar measure on G. If $\rho_1 : G \to Ł(V_1)$ and $\rho_2 : G \to Ł(V_2)$ are the linear representations of the action of G on V_1 and V_2, respectively, we define the* representative group convolution *for any $s \in G$ by*

$$(K \circledast f)(s) = \int_G \rho_2(r) K\left(r^{-1}s\right) \rho_1\left(r^{-1}\right) f(r) d\mu(r) \tag{3}$$

Remark 1. *In what follows, we keep the same definitions for G, K, V, ρ, μ and f.*

Theorem 1. *With the same hypothesis stated in Definition 2, let V denote either V_1 or V_2 and ρ either ρ_1 or ρ_2. If G acts on V^G by*

$$\rho(g) f\left(g^{-1}r\right) \qquad \forall g, r \in G \text{ and } f : G \to V,$$

then the representative group convolution is G-equivariant.

This new convolution layer is thus very powerful to create an estimator to an equivariant function, because it is itself equivariant by construction. However, we cannot compose it with other non-equivariant operations without breaking the equivariance for the whole network. Therefore, alone, a convolution layer is not that powerful, but a chain of multiple convolution layers is much more interesting.

Lemma 1. *Any composition of G-equivariant functions is still G-equivariant.*

Multiple representative-group convolution layers can be composed to obtain a G-equivariant network. Note that the action of G on the output of the ith layer must match the action of G on the input of the $(i+1)$th layer.

In Table 1, there is a chain of representations $\rho_0 \to \ldots \to \rho_L$, but what really matters is only the first one (ρ_0) and the last one (ρ_L). Indeed, by Lemma 1, the whole network is equivariant to G's action with ρ_0 on the input and ρ_L on the output. Thus, we have full choice over the other representations ρ_ℓ for $1 \leq \ell \leq L-1$.

Table 1. Example of an Equivariant Neural Network.

input layer:	$\mathcal{N}^0 = f \in V_0^G$,	
convolution layers:	$\mathcal{N}^\ell = K^\ell \circledast \mathcal{N}^{\ell-1} \in V_\ell^G$,	with $\rho_{\ell-1}$, ρ_ℓ, for $1 \leq \ell \leq L$.

Remark 2. *One can still use non-equivariant functions between two hidden layers of the network, as long as these functions are point-wise and that the representations chosen for theses hidden layers are regulars. This covers the main usual architectures for convolutional neural networks.*

Lifting the Coordinate Space

The Representative Group Convolution cannot be used right away since the convolution is constructed to be performed on G and not on the input data space (denoted $\mathcal{X}$ in the sequel). The problem is obviated by lifting the coordinates from $\mathcal{X}$ to G. One can find more details of this method in [29].

Definition 3 (Lifting)**.** *Let $Q = \mathcal{X}/G$ be the set of orbits of G. If u is a mapping from $\mathcal{X}$ to V, we set $u^\uparrow : G \times Q \to V$ as the lifted version of u, defined by:*

$$\begin{array}{rccc} u^\uparrow & : & G \times Q & \longrightarrow & V \\ & & \left(r, \overline{o_q}\right) & \longmapsto & u(r \cdot o_q) \end{array}$$

An element $x \in \mathcal{X}$ is then lifted to a tuple $\left(r_x, \overline{o_q}\right)$.

Definition 4 (Lifted Action). *If G acts on $\mathcal{X} \times V$, then it has an extended action on the lifted space $(G \times \mathcal{X}/G) \times V$. If $\left((r,q), u^{\uparrow}(r,q)\right)$ is a lifted element, and $g \in G$ is acting by:*

$$\begin{aligned} g \cdot \left((r,q), u^{\uparrow}(r,q)\right) &= g \cdot \left(r \cdot o_q, u\left(r \cdot o_q\right)\right) \\ &= \left(r \cdot o_q, \rho(g) u\left(g^{-1} r \cdot o_q\right)\right) \\ &= \left((r,q), \rho(g) u^{\uparrow}\left(g^{-1} r, q\right)\right). \end{aligned}$$

4. Solving of PDEs with ENN

We discuss in this section two ways of using ENN for solving PDEs, starting with the use of G-CNN to generalize the PINN concept and then by building symmetry-preserving Finite Difference schemes by using the ENN as differential invariant approximators.

4.1. Equivariant PINN

The idea behind PINNs is somewhat straightforward. Let us consider the PDE defined in Section 2.1 but with added boundary conditions on a set B, which gives:

$$(E): \quad \begin{cases} \Delta\left(t, x, u^{(n)}\right) = 0 & \forall t, x \in \mathbb{R}^+ \times \mathcal{X} \\ u(x) = u_b(x) & \forall x \in B \end{cases}$$

Now, we directly estimate a solution u of (E) at $t_0 + dt$ with $\mathcal{N}_\theta$ an Equivariant Neural Network (ENN) parameterized by θ, taking as input the initial profile of the solution, being u at time t_0. This ENN is equivariant to the symmetry group of (E).

In order to train the ENN $\mathcal{N}_\theta$ to approximate the solution of the PDE, we introduce the following optimization problem (P):

$$(P): \theta^* = \arg\min_\theta \{\mathcal{L}(\theta, \mathcal{T})\}.$$

Additionally, the following loss function $\mathcal{L}$:

$$\mathcal{L}(\theta, \mathcal{T}) = w_f \mathcal{L}_f\left(\theta, \mathcal{T}_f\right) + w_b \mathcal{L}_b(\theta, \mathcal{T}_b) \tag{4}$$

with

$$\mathcal{L}_f\left(\theta, \mathcal{T}_f\right) = \frac{1}{\left|\mathcal{T}_f\right|} \sum_{x \in \mathcal{T}_f} \left\| \Delta\left(t, x, \hat{u}^{(n)}\right) \right\|_2^2 \tag{5}$$

$$\mathcal{L}_b(\theta, \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{x \in \mathcal{T}_b} \|\mathcal{B}(\hat{u}_\theta, x)\|_2^2 \tag{6}$$

with boundary conditions $\mathcal{B}(u_b, x) = 0$ on B. Additionally, $\mathcal{T}_f$ and $\mathcal{T}_b$ are two training sets of randomly distributed points, with $\mathcal{T}_b \subset B$ and $\mathcal{T}_f \subset \mathcal{X}$.

Remark 3. *A similar approach has been used by Wang et al. in [28] but with steerable neural networks. They used a* U-Net *and a* ResNet *architecture, common for these types of tasks [37], which they made equivariant in order to predict the PDEs' results. To tackle the constraints on the kernels, they manually design some transformations to make their neural networks equivariant to 4 different actions. Our result is meant to be more general than this case-by-case design. We bypass the kernel's constraints and have a fully equivariant network to any given group.*

4.2. Symmetry-Preserving Finite Difference

We restrict ourselves in the following to PDEs systems of order n with a linear dependency with respect to the time differentials. According to the introduced formalism, we only consider systems of the form

$$\sum_{i=1}^{k_t} a_i \partial_t^i u = \Delta\left(t, x, u^{(n)}\right) \tag{7}$$

where $k_t \in \mathbb{N}$ and Δ is an operator from $\mathbb{R}^+ \times J^{(n)}$ to $\mathbb{R}^q$. The above form covers most of the PDEs encountered in physics, ranging from the heat and wave equations to the Navier Stokes, Shrödinger and Maxwell equations. Assuming that the above PDEs system is regular enough, it admits G as symmetry if and only if the operator Δ can be expressed as a function of a complete set of differential invariants, i.e., if and only if (2) can can be re-written as

$$\sum_{i=1}^{k_t} a_i \partial_t^i u = F\left(\partial\phi_{u,n}^{G,1}, \ldots, \partial\phi_{u,n}^{G,k}\right) \tag{8}$$

with $F : \mathbb{R}^{qk} \to \mathbb{R}^q$.

We propose in the following to approximate the differential invariants by neural networks equivariant to the corresponding group action. More precisely, let us consider a discretisation $x^{(1)}, \ldots, x^{(n_x)}$ (resp. $t^{(1)}, \ldots, t^{(n_t)}$) of the input space $\mathcal{X}$ (resp. time interval $\mathbb{R}^+$) and denote $f^{(i,j)} = f(t^{(i)}, x^{(j)})$ for any $f : \mathbb{R}^+ \times \mathbb{R}^p \to \mathbb{R}^q$. For a given differential invariant $\partial\phi_f^G$, we are then interested in approximating the vector $\left(\partial\phi_f^{G,(i,j)}\right)_{j=1}^{n_x}$ by the output of an equivariant neural network $\mathcal{N}_G$, taking as input the tensor $\left(f^{(i)(j)}\right)_{j=1}^{n_x}$, so that we have for the jth component

$$\mathcal{N}_G\left(\left(f^{(i)(\ell)}\right)_{\ell=1}^{n_x}\right)^j \approx \partial\phi_f^{G,(i,j)} = \partial\phi_f^G(t^{(i)}, x^{(j)}) \tag{9}$$

In the following, we propose to train $\mathcal{N}_G$ on multivariate polynomial functions in the space variables x of degree d, but other choices could be envisioned depending on the considered problem.

The use of equivariant neural networks is motivated here by the fact that the operator $f \to \partial\phi_f^G$ that we are approximating is equivariant. Indeed, for a function $f : \mathbb{R}^+ \times \mathbb{R}^p \to \mathbb{R}^q$, $\partial\phi_f^G$ is a function from $\mathbb{R}^+ \times \mathbb{R}^p$ to $\mathbb{R}^q$, and we can therefore consider its transform $g.\partial\phi_f^G$ according to the action of G by considering its transformed graph, as defined in Section 2.2. As the differential invariant is an algebraic invariant of the prolonged group action $\mathrm{pr}^{(n)} G$, it is possible to write

$$g.\partial\phi_f^G = \partial\phi_{g.f}^G \tag{10}$$

meaning that differential invariant operators are equivariant with respect to the associated group action, as illustrated on Figure 2 for the case of $\mathrm{SE}(2)$.

We now come back to the PDE systems (2) and detail the numerical scheme that we propose here for its integration. The idea is to first train an ENN $\mathcal{N}_G^i$ for approximating each of the k differential invariants $\partial\phi_{.,n}^{G,j}$ which are involved and then to integrate by using an explicit scheme in which the differential invariants are replaced by their approximation with the ENN, leading to

$$\sum_{\ell=1}^{k_t} a_\ell \partial_t^\ell u(t^{(i)}, x^{(j)}) \approx F\left(\mathcal{N}_G^1\left(\left(u^{(i,\ell)}\right)_{\ell=1}^{n_x}\right)^j, \ldots, \mathcal{N}_G^k\left(\left(u^{(i,\ell)}\right)_{\ell=1}^{n_x}\right)^j\right) \tag{11}$$

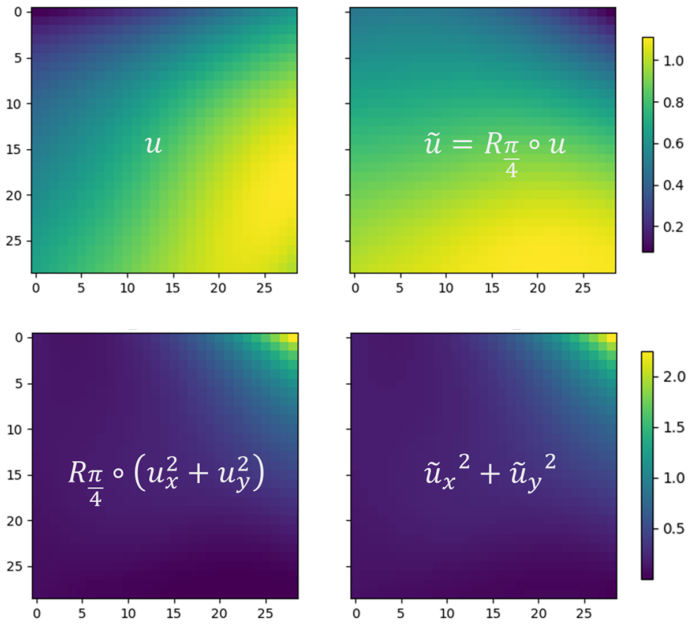


Figure 2. From left to right and top to bottom: the initial function u, its rotated version $\tilde{u}$, the rotated version of the SE(2) differential invariant $u_x^2 + u_y^2$ (see Section 4.3) and the differential invariant of the rotated function. As expected, computing the differential invariant from u and applying the rotation (bottom left) gives the same results as computing the differential invariant from the rotated function (bottom right).

4.3. Numerical Experiments

4.3.1. Approximating SE(2) Differential Invariants

Here, we considered the case of $SE(2)$ for which a generating set of second-order differential invariants is given by

$$\partial\phi_{u,2}^{SE(2)} = \left\{ u, u_x^2 + u_y^2, u_{xx} + u_{yy}, u_x^2 u_{xx} + 2u_x u_y u_{xy} + u_y^2 u_{yy}, u_{xx}^2 + 2u_{xy}^2 + u_{yy}^2 \right\} \quad (12)$$

We trained in the following 2 Neural Networks, namely one conventional Convolutional Neural Network $\mathcal{N}^{\mathbb{R}^2}$ with $\mathbb{R}^2-$equivariant layers and one $SE(2)-$ENN $\mathcal{N}^{\mathrm{SE}(2)}$, both of them built to have roughly the same number of parameters ($\approx 2.2 \times 10^6$). The training set consists in 29×29 evaluations of 2D-polynomials in $\mathbb{R}[X, Y]$ with degrees up to 10 and generated from random coefficients drawn uniformly in $[-1, 1]$. Polynomials evaluations were performed on the discrete grid $(i/29, j/29)_{i,j=-14,\ldots,14}$. An example of prediction with the trained $\mathcal{N}^{\mathrm{SE}(2)}$, together with the corresponding theoretical value, is given in Figure 3.

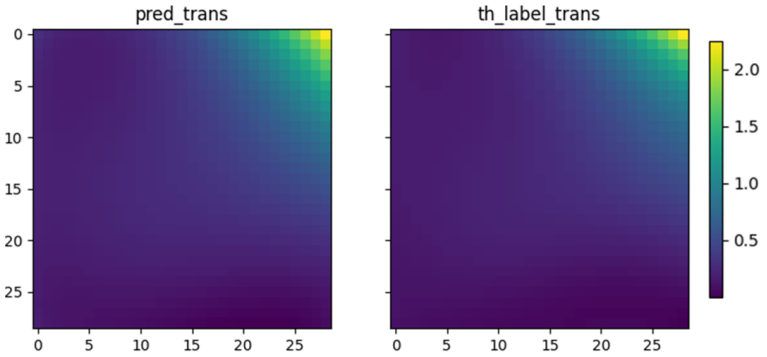


Figure 3. The SE(2) differential invariant $u_x^2 + u_y^2$ computed for the function u depicted in Figure 2 with an SE(2)-CNN (**left**) and its theoretical value (**right**).

4.3.2. Solving the 2D Heat Equation

Here, we consider the 2D heat equation $u_t = u_{xx} + u_{yy}$ defined on a square domain [−a,a]×[−b,b], with the boundary condition $u(t, \pm a, b) = 0$ and $u(t, \pm a, -b) = 100$ and the initial condition $u_{t=0} = f$, for $f : \mathbb{R}^2 \to \mathbb{R}$ an arbitrary function. Below, we give the results that we obtained by using an FD scheme relying on the approximation of the 2D-Laplacian, as described in Section 4.2 , i.e., by computing the solution according to the following update rule:

$$u^{n+1} = u^n + \delta_t \times \mathcal{N}^{\Delta}_{\mathrm{SE}(2)}(u^n) \tag{13}$$

where $u^n = u(n \times \delta_t, .)$. We ran 10^5 steps of the simulation for $\delta_t \approx 10^{-7}$ using the two trained architectures considered in Section 4.3.1, namely $\mathcal{N}^{\mathbb{R}^2}$ and $\mathcal{N}^{\mathrm{SE}(2)}$, and compared the obtained heat profiles with that of the ground truth. The boundary condition was taken into account by overriding the predicted outputs by the conventional second-order derivative approximation for the corner cases. The obtained results are depicted in Figure 4, where we can, in particular, observe the high benefit of preserving the SE(2) symmetry during the numerical integration.

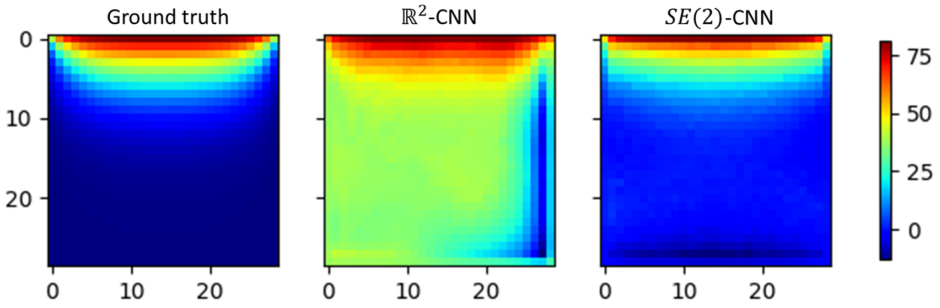


Figure 4. Comparison of the theoretical heat profile of the 2D heat equation with a top 100° boundary condition (see Section 4.3.2) with those obtained through simulation with two symmetry-preserving FD schemes (see Section 4.2) by leveraging $\mathbb{R}^2$ (middle) and SE(2) (right) equivariant neural networks.

5. Conclusions and Further Work

We presented two innovative ways of using ENN to solve PDEs while exploiting the associated symmetries. We first showed that G-CNN can be used to generalize the PINN architecture to encode generic symmetries and then proposed using ENN to approximate differential invariants of a given symmetry group, hence allowing to build symmetry-preserving Finite Difference methods. Our approach is illustrated on the 2D Heat Equation for which we, in particular, showed that a set fundamental differential invariants of SE(2) can be efficiently approximated by ENN for arbitrary functions by training on simple bivariate polynomials evaluations, allowing to easily build SE(2) symmetry-preserving discretization schemes.

Additional work will include performing proper benchmarking between the two approaches and more conventional numerical schemes for PDE integration. More complex PDEs with richer symmetry groups such as Maxwell equations could be considered in this context.

Author Contributions: Both authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: Eliot Tron contributed to this work during an internship at Thales Research and Technology in 2021.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
2. Lu, L.; Meng, X.; Mao, Z.; Karniadakis, G.E. DeepXDE: A deep learning library for solving differential equations. *arXiv* **2020**, arXiv:cs.LG/1907.04502.
3. Sirignano, J.; Spiliopoulos, K. DGM: A Deep Learning Algorithm for Solving Partial Differential Equations. *J. Comput. Phys.* **2018**, *375*, 1339–1364. [CrossRef]
4. Raissi, M.; Yazdani, A.; Karniadakis, G.E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* **2020**, *367*, 1026–1030. [CrossRef] [PubMed]
5. Wang, R.; Kashinath, K.; Mustafa, M.; Albert, A.; Yu, R. Towards Physics-Informed Deep Learning for Turbulent Flow Prediction. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '20, Virtual Event, 6–10 July 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 1457–1466. [CrossRef]
6. Olver, P. Applications of Lie Groups to Differential Equations. In *The Handbook of Brain Theory and Neural Networks*; Springer: New York, NY, USA, 1993.
7. Fushchich, W.; Nikitin, A. *Symmetries of Maxwell's Equations*; Mathematics and Its Applications; Springer: Dordrecht, The Netherlands, 2013.
8. Morrison, P. Structure and structure-preserving algorithms for plasma physics. *Phys. Plasmas* **2016**, *24*, 055502. [CrossRef]
9. Kraus, M. Metriplectic Integrators for Dissipative Fluids. In *Geometric Science of Information*; Nielsen, F., Barbaresco, F., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 292–301.
10. Coquinot, B.; Morrison, P.J. A general metriplectic framework with application to dissipative extended magnetohydrodynamics. *J. Plasma Phys.* **2020**, *86*, 835860302. [CrossRef]
11. Luesink, E.; Ephrati, S.; Cifani, P.; Geurts, B. Casimir preserving stochastic Lie-Poisson integrators. *arXiv* **2021**, arXiv:2111.13143.
12. Zhu, A.; Jin, P.; Tang, Y. Deep Hamiltonian networks based on symplectic integrators. *arXiv* **2020**, arXiv:2004.13830.
13. Dorodnitsyn, V. Finite Difference Models Entirely Inheriting Symmetry of Original Differential Equations. *Int. J. Mod. Phys. C* **1994**, *5*, 723–734. [CrossRef]
14. Shokin, I.; Shokin, J.; Shokin, Y.; Šokin, Û.; Roesner, K. *The Method of Differential Approximation*; Computational Physics Series; Springer: Berlin/Heidelberg, Germany, 1983.
15. Olver, P.J. Geometric Foundations of Numerical Algorithms and Symmetry. *Appl. Algebra Eng. Commun. Comput.* **2001**, *11*, 417–436. [CrossRef]
16. Marx, C.; Aziz, H. Lie Symmetry Preservation by Finite Difference Schemes for the Burgers Equation. *Symmetry* **2010**, *2*, 868. [CrossRef]
17. Razafindralandy, D.; Hamdouni, A. Subgrid models preserving the symmetry group of the Navier–Stokes equations. *C. R. Méc.* **2005**, *333*, 481–486. [CrossRef]
18. Brandstetter, J.; Welling, M.; Worrall, D.E. Lie Point Symmetry Data Augmentation for Neural PDE Solvers. *arXiv* **2022**, arXiv:2202.07643.
19. Bronstein, M.M.; Bruna, J.; Cohen, T.; Veličković, P. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *arXiv* **2021**, arXiv:2104.13478.
20. Gerken, J.E.; Aronsson, J.; Carlsson, O.; Linander, H.; Ohlsson, F.; Petersson, C.; Persson, D. Geometric Deep Learning and Equivariant Neural Networks. *arXiv* **2021**, arXiv:2105.13926.
21. Cohen, T.; Welling, M. Group Equivariant Convolutional Networks. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; Balcan, M.F., Weinberger, K.Q., Eds.; PMLR: New York, NY, USA, 2016; Volume 48, pp. 2990–2999.
22. Cohen, T.S.; Geiger, M.; Weiler, M. A General Theory of Equivariant CNNs on Homogeneous Spaces. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32, pp. 9145–9156.
23. Weiler, M.; Cesa, G. General E(2)-Equivariant Steerable CNNs. *arXiv* **2019**, arXiv:1911.08251.
24. Worrall, D.; Welling, M. Deep Scale-spaces: Equivariance Over Scale. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32.
25. Kondor, R.; Trivedi, S. On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Dy, J., Krause, A., Eds.; PMLR: Stockholm, Sweden, 2018; Volume 80, pp. 2747–2755.
26. Elesedy, B.; Zaidi, S. Provably Strict Generalisation Benefit for Equivariant Models. *arXiv* **2021**, arXiv:2102.10333.
27. Gerken, J.E.; Carlsson, O.; Linander, H.; Ohlsson, F.; Petersson, C.; Persson, D. Equivariance versus Augmentation for Spherical Images. *arXiv* **2022**, arXiv:2202.03990.

28. Wang, R.; Walters, R.; Yu, R. Incorporating Symmetry into Deep Dynamics Models for Improved Generalization. *arXiv* **2020**, arXiv:2002.03061.
29. Finzi, M.; Stanton, S.; Izmailov, P.; Wilson, A.G. Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data. *arXiv* **2020**, arXiv:2002.12880.
30. Cohen, T.S.; Welling, M. Steerable CNNs. *arXiv* **2016**, arXiv:1612.08498.
31. Lang, L.; Weiler, M. A Wigner-Eckart Theorem for Group Equivariant Convolution Kernels. *arXiv* **2020**, arXiv:2010.10952.
32. Cohen, T.S.; Weiler, M.; Kicanaoglu, B.; Welling, M. Gauge Equivariant Convolutional Networks and the Icosahedral CNN. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019.
33. Cohen, T.S.; Welling, M. Group Equivariant Convolutional Networks. *arXiv* **2016**, arXiv:1602.07576.
34. Cohen, T.S.; Geiger, M.; Weiler, M. Intertwiners between Induced Representations (with Applications to the Theory of Equivariant Neural Networks). *arXiv* **2018**, arXiv:1803.10743.
35. Kondor, R.; Trivedi, S. On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups. *arXiv* **2018**, arXiv:1802.03690.
36. Cohen, T.; Geiger, M.; Weiler, M. A General Theory of Equivariant CNNs on Homogeneous Spaces. *arXiv* **2018**, arXiv:1811.02017.
37. Wang, R.; Kashinath, K.; Mustafa, M.; Albert, A.; Yu, R. Towards Physics-informed Deep Learning for Turbulent Flow Prediction. *arXiv* **2019**, arXiv:1911.08655.