ROBUST NEURAL NETWORK-BASED DISCOVERY OF DYNAMICAL SYSTEMS

Santiago Andrés Serrano-Vacca

santiagoserrano334@gmail.com

Tecnológico de Monterrey*

ABSTRACT

Using neural networks to represent the dynamics of a system has multiple advantages over sparse regression techniques such as SINDy, as the applicability of the former is not constrained to systems with dynamics that can only be accurately expressed using analytical expressions. On top of that, progress has been made in the use of physics-informed architectures for neural networks, making them more precise when certain aspects of the laws governing the system are known. However, existing approaches of Physics-Informed Neural Networks (PINNs) are not protected enough against noise in their training data, which makes them less reliable in real-life scenarios where information obtained from sensors or real observations is not noise-free. In this work, I propose two approaches for PINNs tailored to address Gaussian and adversarial noise, respectively, in the training data, by combining previous work on PINNs with existing techniques that aim to improve the generalization and robustness of models. These new approaches, named GPPhysMLP and GRPhysMLP respectively, demonstrate superior performance compared to counterparts that do not take noise into account significantly enough.

Keywords Physics-constrained learning · Neural networks · Robust · Gaussian noise · Adversarial noise · Deep learning

1 Introduction

Discovering the dynamics of an unknown system is crucial for several reasons, since it allows for prediction of its future states and enables the design of controllers to manipulate the system's behavior, among other advantages. This is specially true in engineering and robotics, where being able to predict and control the behavior of machinery and vehicles can lead to improved efficiency, safety, and reliability. Furthermore, it can open up new possibilities for innovation and technological advancement, pushing the boundaries of what is currently achievable.

In the specialized literature, different techniques exist to model unknown dynamical systems. Algorithms such as the Sparse Identification of Nonlinear Dynamics (SINDy) [3] are among the most widely known, with subsequent approaches such as GPSINDy [8] and SINDy-PI [10] aiming at discovering governing equations from noisy data. These computational methods are based on sparse regression techniques. However, their applicability is constrained to systems with dynamics that can be accurately expressed through analytical expressions. On top of that, an optimal set of candidate functions must be selected beforehand. Complex systems (systems with many components that interact in non-linear, often unpredictable ways), for instance, may exhibit properties that are not easily reducible to simple

*This work was independently conducted by the sole author of this paper, primarily at Purdue University within the Machine Intelligence & Networked Data Science Group, under an undergraduate research fellows agreement with Tecnológico de Monterrey. Guidance was provided by Purdue Professor Dr. Abolfazl Hashemi. Code from this work is freely available at: `https://github.com/santiago-a-serrano/robust_physics_constrained_nn`.

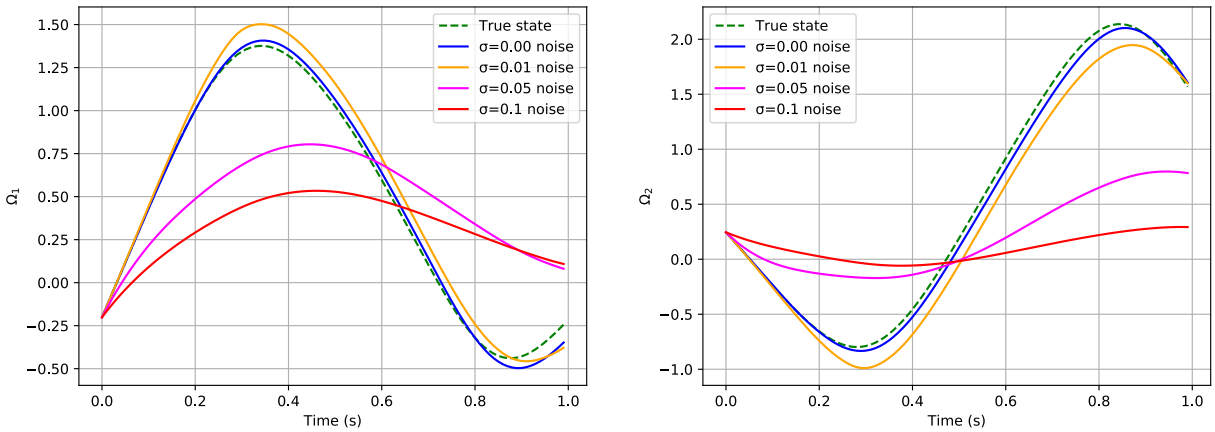

Figure 1: Angular velocity predictions for a double pendulum system by a multilayer perceptron (MLP) under different Gaussian noise levels, compared with true values.

equations due to the intricate interplay of their components [6], making the previously mentioned algorithms unsuitable for discovering them.

Conversely, neural networks are not constrained by the need for pre-defined analytical expressions and can learn complex representations directly from data, making them well-suited for modeling systems with intricate or unknown dynamics. Furthermore, Djeumou *et al.* show that effective inclusion of physics-based knowledge into deep neural network models can greatly improve data efficiency and generalization [5], making deep learning also a very powerful tool when some (but not all) knowledge about the system and/or its constraints is available.

Nonetheless, in a real-life scenario, the data available (and that will be fed to the model) will very likely be perturbed by a significant amount of noise, as it is frequently sourced from sensors and observations, meaning that it can be subject to various environmental and operational disturbances. Additionally, many dynamical systems expose chaotic behavior, which means that a slight disturbance in its initial conditions can result in the prediction of a vastly different future trajectory, as it can be seen in Figure 1.

Noise is not explicitly mentioned or discussed in Djeumou *et al.*'s publication, and although the code that accompanies it enables the incorporation and testing of noise in many examples (excluding the double pendulum, which will be the main chosen benchmark for this work), it does not propose or implement any specific technique to mitigate its effects or make the model more robust.

As neural networks are a very promising and rapidly developing field, and methods such as the one proposed by Djeumou *et al.* have the potential of outperforming techniques such as SINDy in many cases, striving to find optimal robustness against noise approaches is of crucial significance.

2 Foundations

2.1 Benchmarking Neural Network Architecture

As this work is largely focused on improving Djeumou *et al.*'s method to include robustness, the same architectures proposed on their work will be used (with the addition of various techniques for denoising and robustness that will be mentioned in Section 3).

Most of the benchmarks that will be performed in the following experiments will be based on double pendulum dynamics, and, unless specified otherwise for particular cases, will conform to one or two multilayer perceptrons (MLP) with ReLu activation functions and two hidden layers with 256 nodes each [5]. The loss function uses a rollout horizon of 5, meaning that it takes into account the current state and 5 predicted future states of the system to calculate itself. The training is optimized using Adam, with a learning rate of 0.005, a minibatch size of 64, and early stopping patience of 1000. More information can be found in Djeumou *et al.*'s paper [5].

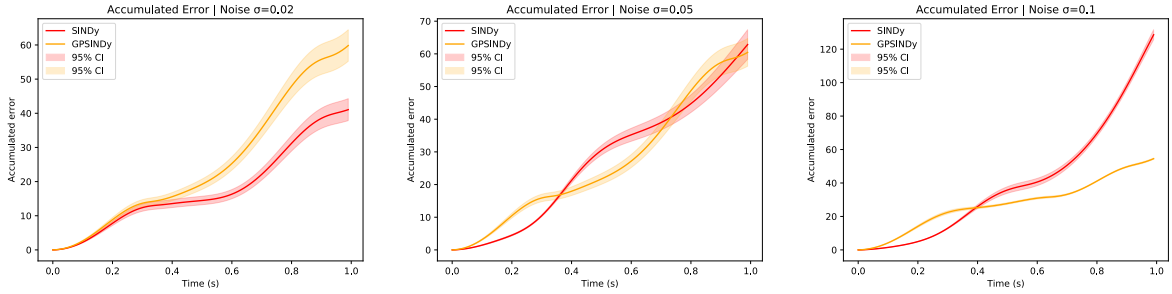

Figure 2: SINDy (red) vs. GPSINDy (orange) performance for training data with varying levels of Gaussian noise, calculated for 10 trajectories (each) and a 95% confidence interval.

Although double pendulum dynamics can accurately be described with differential equations, it was decided to perform the benchmarks on this system as it is a prototypical example of a chaotic system where small differences in initial conditions can lead to vastly different outcomes, making it an ideal candidate for testing the robustness and predictive power of various modeling approaches.

2.2 Calculation of Accumulated Error

The accumulated error, which will be the metric used for benchmarking the different approaches exposed throughout this work, will be calculated as $\sum_{i=0}^{n} \|X_i - Y_i\|_2$, where n is the amount of timesteps elapsed since the beginning of the trajectory, X_i is the state of the predicted trajectory at timestep i, and Y_i is the state of the real trajectory at the same timestep.

This calculation will be the same regardless of the technique (whether a neural network or the result of solving for equations) used for generating the predicted trajectory (or sequence of states) of the dynamical system, since it depends on the predicted trajectory itself and not in any loss function or feature specific to a type of technique.

3 Techniques for Denoising and Robustness

3.1 Gaussian Process Regression (GPR)

As the distribution of noise in numerous engineering systems is well captured by the Gaussian distribution [13], it is an usual choice, when the exact distribution of noise is unknown, to assume it is Gaussian (or at least close to Gaussian). Inspired by this and the problem of discovering dynamical system models from noisy data mentioned earlier, Hsin *et al.* proposed combining Gaussian process regression with SINDy [8]. In their publication, improved performance over SINDy was demonstrated for some dynamics. For the case of the double pendulum example mentioned previously on this work, GPSINDy was implemented and improved performance was also achieved for $\sigma \geq 0.1$, although lower levels of noise didn't produce significantly better results (in some cases, results were even worse) as it can be seen in Figure 2.

Motivated by GPSINDy, this study opted to pre-process noisy training data for the double pendulum system with GPR before feeding it to a neural network. The results are shown in Figure 3.

3.1.1 Definition of Gaussian Processes:

As explained by Wang *et al.*, a GP model[2] describes a probability distribution over possible functions that fit a set of points [18]. In this paper, we experiment with using the mean function of this model, which is the maximum likelihood estimate, as a "denoised version" of the noisy trajectory, and feeding this new version as the data for the neural network's training.

In order to find the mean function of the GP model, we compute the posterior mean:

$$X_{\text{denoised}} = K(t,t)[K(t,t) + \sigma_n^2 I]^{-1} X \tag{1}$$

Where:

[2]Not to be confused with the actual neural network that will be trained for describing the dynamics of the system, which can also be called a "model".

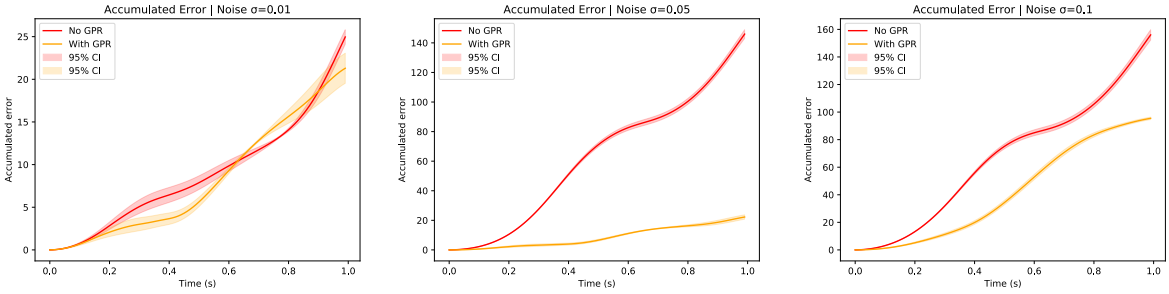

Figure 3: Simple MLP performance for training data with varying levels of Gaussian noise, with (orange) and without (red) GPR denoising, calculated for 10 trajectories (each) and a 95% confidence interval.

- X represents the trajectory of the dynamical system in question that we want to denoise. Every row X_i represents the state at time t_i.
- t is the vector of timesteps, for example: $[0, 1, 2, 3, ..., n]$ (this example is the specific case for my code implementation).
- $K(t, t)$ is the covariance matrix between all pairs of timesteps.
  - Every item in the covariance matrix is calculated as follows:

$$k(t_i, t_j) = \sigma_f^2 \exp\left(-\frac{1}{2\sigma_l^2}\|t_i - t_j\|^2\right) \tag{2}$$

    Where σ_f and σ_l are Gaussian process hyperparameters.
- σ_n is the last of the three Gaussian process hyperparameters, and it is proportional to the amount of noise that we expect to find. Experimentally, we found this parameter to almost always be very close to two times the standard deviation of the noise present in the trajectory.

The Rprop algorithm was used to find the optimal Gaussian process hyperparameters, as suggested by Blum and Riedmiller [2], by finding the minimum negative Log Marginal Likelihood (LML) with respect to them:

$$-\log p(X) = \frac{1}{2}X^\top (K + \sigma_n^2)^{-1} X + \frac{1}{2}\log|K + \sigma_n^2| + \frac{n}{2}\log(2\pi) \tag{3}$$

The Adam algorithm was also attempted, but resulted in worse convergence. The values of σ_f and σ_l, for most of the experiments performed throughout this research for varying levels of Gaussian noise, fell under the following ranges: $1.12 < \sigma_f < 2.91$ and $35.47 < \sigma_l < 40.36$. For different trajectories perturbed with approximately the same amount of noise, variation between the obtained hyperparameters was very negligible. Because of this, for time optimization purposes (since optimizing the negative LML is considerably time-consuming), the hyperparameters used for the training of each model were obtained from only the first trajectory in the training dataset.

3.2 Gradient Regularization

Gradient regularization is a technique used in machine learning to prevent overfitting by adding a penalty term to the loss function. The optimization problem can be expressed as the following:

$$\min_\theta \mathcal{L}(\theta) + \lambda R(\theta) \tag{4}$$

where λ is a regularization coefficient that controls the balance between the original loss and the regularization term, θ are the parameters of the model, and $R(\theta)$ is the regularization term, typically a function of the gradient's magnitude.

In this work, we aim to prevent the neural network from overfitting to noise. Lyu *et al.* have introduced a specialized family of gradient regularization techniques designed to mitigate the impact of adversarial examples [11], that can be approximated as:

$$\min_\theta \mathcal{L}(x) + \sigma \left\|\nabla_x \mathcal{L}\right\|_{p^*} \tag{5}$$

where x is the input data, σ is the p-norm of the perturbation we would like our model to be robust against, $\nabla_x \mathcal{L}$ is the gradient of the loss function at point x, and p^* is the dual of p, i.e. $\frac{1}{p^*} + \frac{1}{p} = 1$.

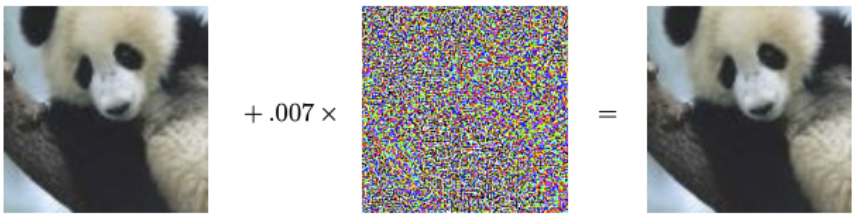

Figure 4: Adding adversarial noise to the image of a panda in order for a classifier model to label it as a gibbon [7].

As this approximation is a result of a first-order Taylor expansion, it can be made more precise by adding the second-order term. Specifically, in the case where $p = 2$ we obtain:

$$\min_{\theta} \mathcal{L}(x) + \sigma \left\|\nabla_x \mathcal{L}\right\|_2 + \frac{\sigma^2}{2} \mathrm{Tr}\left(H_{\mathcal{L}}(x)\right) \tag{6}$$

3.2.1 Adversarial Noise Generation

As Szegedy *et al.* have exposed, neural networks can often be fooled by a specific type of noise called the "adversarial noise" [17]. This noise comes from perturbing datapoints in a specific way, with the concrete intention of deceiving the model as much as possible. Arguably, the most famous example of an adversarial attack is the one from Goodfellow *et al.*'s *Explaining and Harnessing Adversarial Examples* paper, which shows how noise that is imperceptible for humans can easily mislead a neural network into wrongly classifying data (see Figure 4) [7].

In scenarios where security is paramount and maintaining a minimum standard for worst-case performance is crucial, optimizing against adversarial noise becomes highly pertinent. Take, for instance, the deployment of a neural network for trajectory prediction within a military context; the greater its susceptibility to adversarial attacks, the higher the potential for enemy exploitation and resultant harm.

There are multiple approaches to attempt generating the most deceiving adversarial noise, such as the Carlini-Wagner attack [4] and the Fast Gradient Sign Method (FGSM) [7]. However, these methods (and most papers on adversarial noise) primarily target classification tasks. In contrast, for the purpose of predicting the next states of a dynamical system, the model's output is not limited to a finite set of possible labels. Instead, it encompasses an infinite set, $\mathbb{R}^n$, of possible data points.

Because of this, the following optimization problem was employed in order to generate adversarial noise ϵ:

$$\max_{\|\epsilon\| \leq \sigma} \mathcal{L}(x + \epsilon; \theta) \tag{7}$$

The Adam optimization algorithm with a learning rate of $\alpha = 0.001$ was used, along with the following constraints for the norm of the adversarial noise $\sigma = 0.005, 0.01, 0.02, 0.03, 0.05, 0.1, 0.2$. In order to enforce the hard norm constraint, projection was performed at every step where $\|\epsilon\|$ became greater than σ, by scaling down ϵ's components proportionally to make its norm exactly σ. Two approaches were implemented and tested:

- A1 Having x as a batch (size 64) of individual and independent datapoints of the double pendulum trajectory, resulting in an adversarial noise of shape $(4,)$. This can be interpreted as finding the worst perturbation for a point in a double pendulum trajectory.
- A2 Having x as a complete trajectory (comprised of 300 datapoints), resulting in an adversarial noise of shape $(300, 4)$. This can be interpreted as finding the worst perturbation for the whole double pendulum trajectory.

Results of training the double pendulum multilayer perceptron with and without gradient regularization, for varying levels of σ, and with approaches A1 and A2 can be seen in Figure 5. Observations reveal that A2 leads to a more powerful attack on the neural network, which is intuitive since the adversarial noise optimization process has the complete trajectory to learn from, opposed to individual datapoints. However, it is very interesting finding out that, for A2, there is no significant difference between using and not using gradient regularization. For the case of A1, one can also see that, as σ gets bigger, gradient regularization stops outperforming the original loss function for the model training. This behavior can also be seen on bigger values of σ, although they are not included in the figure.

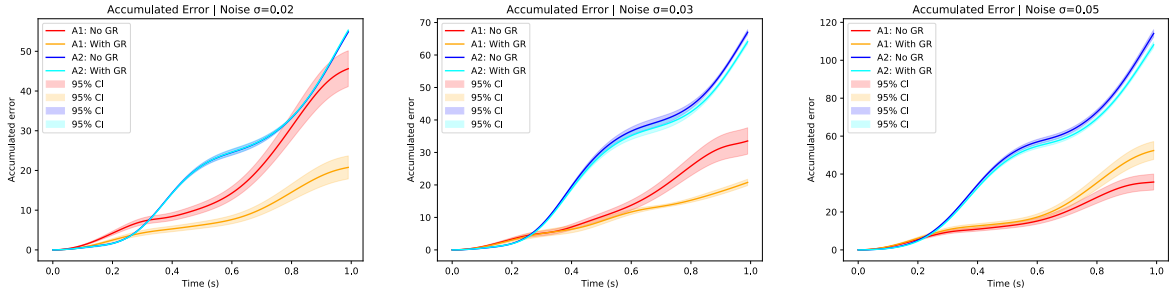

Figure 5: Simple MLP performance for training data with varying levels of adversarial noise, following approaches A1 (red and orange) and A2 (blue and cyan), with (orange and cyan) and without (red and blue) gradient regularization. Calculations were performed (and show the average) for 10 trajectories.

It's worth mentioning that, regardless of the approach, σ represented the maximum norm of the perturbation made to an individual datapoint (in average, for the case of the second approach). More specifically, for the second approach, the maximum norm of the perturbation made to the whole trajectory was calculated as $\sigma * \sqrt{300}$, in order to keep noise proportions equal among both approaches. Also, p was set to 2 in both approaches.

Due to time constraints and the increased computational burden associated with calculating the Hessian matrix of the loss function, we opted to employ Equation (5) over Equation (6). This decision was made despite the potential for Equation (6) to yield substantially improved results. Future research may benefit from exploring this alternative approach to fully assess its impact on performance.

4 Integrated results

Almost as a conclusion of this work, it can be seen on Figure 6 that, for the majority of cases, specially those with a significant amount of Gaussian noise, the variants of the neural networks that implement Gaussian Process Regression outperform their counterparts.

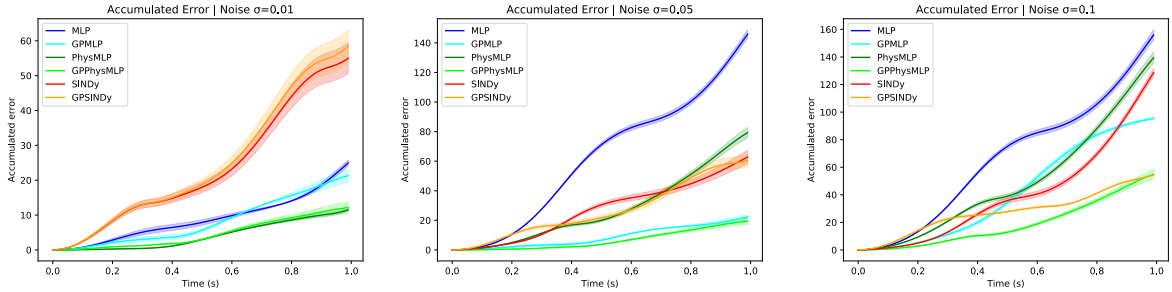

Figure 6: Comparison of performance between the base MLP (dark blue) against itself but with Gaussian Process Regression (cyan), the MLP with physics information (structural knowledge + symmetry constraints) as proposed by Djeumou *et al.* (dark green) against itself but with Gaussian Process Regression (lime), and SINDy (red) against GPSINDy (orange), trained with data perturbed with varying levels of Gaussian noise and a confidence interval of 95%.

It can also be observed that GPPhysMLP, the first of my two proposed approaches (the MLP with physics information as proposed by Djeumou *et al.* with GPR added) is, for the most part, the best-performing and most reliable option of the six (with some slight exceptions), which reinforces my belief about the great importance of PINNs and the huge advantage of being able to enhance them through robust training or denoising strategies. On top of my approach's improved accuracy, its applicability is not constrained to systems with dynamics that can be accurately expressed through analytical expressions or that require an optimal set of candidate functions selected beforehand, making this improvement highly relevant.

For the case of adversarial noise, it can be seen in Figure 7 that results are more varied and not as conclusive. Although GRPhysMLP, the second of my two proposed approaches, is good in general, sometimes it does not outperform

significantly most other candidates. Still, specially for amounts of adversarial noise greater or equal than $\sigma = 0.03$, it appears to be among the best options. The four approaches shown in Figure 7 were tested against approach A1 (mentioned on Section 3.2.1), as approach A2 did not reveal significant difference between using or not using gradient regularization during the training of the model when benchmarked in Figure 5. Nonetheless, further testing would be beneficial.

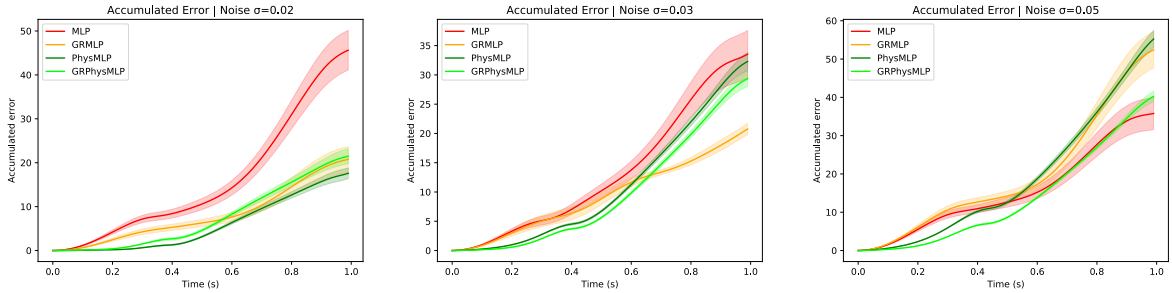

Figure 7: Comparison of performance between the base MLP (red) against itself but with gradient regularization as proposed by Lyu *et al.* (orange), and the MLP with physics information (structural knowledge + symmetry constraints) as proposed by Djeumou *et al.* (dark green) against itself but with gradient regularization as proposed by Lyu *et al.* (lime), trained with data perturbed with varying levels of adversarial noise generated by approach A1, and a confidence interval of 95%.

It is worth mentioning that gradient regularization, specifically using Equation 5, was also tested against Gaussian noise. However, most results showed worse performance than the original models that did not implement the technique. It is important to take into account, however, that Lyu *et al.* claim that, specifically, Equation 6 is the approach that approximately marginalizes over Gaussian noise [11], which emphasizes the need for further testing and implementation using Equation 6.

5 Conclusion

In this paper, it is concluded that GPPhysMLP, one of my two proposed approaches, is highly valuable for inferring the dynamics of a system when dealing with Gaussian (or similar to Gaussian) noise in the training data, as it significantly outperforms other relevant approaches in this scenario. It is also worth mentioning that, as opposed to GRPhysMLP, which is built with Lyu *et al.*'s gradient regularization approach, GPPhysMLP does not require knowledge of the norm of the noise that the training data might contain, making its implementation simpler in real-life scenarios where insight on the specific size of perturbation in the obtained training data might not be easily gathered.

It must also be mentioned that, for similar levels of noise, GPR hyperparameters can be reused with negligible detriment on inference precision from the model, making the overhead computational costs of GPPhysMLP, which are already considerably small to begin with, lowered very significantly.

For GRPhysMLP, I recognize the high potential of replacing Equation (5) with Equation (6), which would probably result on an approach that, aside from being more robust against adversarial noise, could deliver promising results in Gaussian (and even other) noise scenarios. Still, GRPhysMLP performs well (although not with a big significant difference) in the scenario it was designed to work with.

6 Code

As it was mentioned previously in Section 2.1, this work is largely focused on improving Djeumou *et al.*'s method. Code from this work is freely available at: `https://github.com/santiago-a-serrano/robust_physics_constrained_nn`, and is built on top of Djeumou *et al.*'s original code, available at: `https://github.com/wuwushrek/physics_constrained_nn`.

7 Future Research Directions

The discoveries made throughout this work pave the way towards other research directions that pose significant possibilities of yielding insightful findings. Among those directions are:

- Implementing the second-order term of the Taylor expansion that approximates Lyu *et al.*'s technique, as shown in Equation (6). Adding this term to the loss function of the neural network and training with it can potentially lead to better results for minimizing the effect of adversarial noise in trajectories, aside from giving it support against Gaussian noise.
- Evaluating the use of other methods for finding adversarial noise that might produce better results than optimizing Equation (7). However, it is important mentioning that, in my original approach, the loss function reached a point of no apparent significant further maximization, so the advantages of trying other methods might be slim, specially taking into account the very small size of the data being trained (in comparison with what is usual in other machine/deep learning scenarios).
- Regardless of it not being mentioned previously throughout this work, noise was also added to weights and constraints before attempting different experiments (constraints for the case of the physics-informed neural networks proposed by Djeumou *et al.*). However, because of time limitations, not enough experiments could be performed in order to prove something worth reporting, aside from the logical conclusion that adding noise to weights and constraints, in general, worsens model performance. It is relevant noting that running experiments on physics-informed neural networks is very significantly more time-intensive than using more simple MLPs that do not take any side information into account. However, we believe further research and algorithm development in this direction is highly likely to be beneficial, since it might lead to more robustness in general, as explained by Noh *et al.* in *Regularizing Deep Neural Networks by Noise: Its Interpretation and Optimization* [12].
- Testing, finding, and developing algorithms or techniques to mitigate the effects of different types of noise that might also be present in real-life measurements, such as impulse noise, photon noise, or even speckle noise, is also relevant.
- Performing experiments on techniques that are not specifically designed for dealing with the type of noise that the training data contains. For example, performing Gaussian process regression on data with adversarial noise, or using gradient regularization with Gaussian noise. This approach can uncover novel insights into the robustness and adaptability of various algorithms under conditions they were not explicitly optimized for.
- Other robust training techniques, such as the ones outlined in [19, 1, 15, 9, 16] could give interesting insights if they are compared with the work performed in section 3.2 regarding adversarial noise.
- Relevant work, such as the one produced by Raissi *et al.*, uses neural networks to solve problems that necessarily involve solving or discovering partial differential equations [14]. Although one of the most important advantages of Djeumou *et al.*'s approach is, as mentioned in section 1, not being constrained by the need of analytical expressions to describe the dynamics themselves, we believe Raissi *et al.*'s approach is also worth considering, specially taking into account that it already implements robustness against noise and delivers good results when dealing with perturbed data. Because of time limitations, my work was not able to benchmark against Raissi *et al.*'s approach, much less research or implement improvements.
- Applying multiple adversarial perturbations at random to training data, instead of using the same for every datapoint or trajectory (A1 or A2), could lead to more powerful attacks and better benchmarking.
- Benchmarking for dynamical systems other than the double pendulum would be beneficial for a greater confidence in both my approaches' and revevant work performance.

References

[1] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness, 2021.

[2] Manuel Blum and Martin A Riedmiller. Optimization of gaussian process hyperparameters using rprop. In *ESANN*, pages 339–344, 2013.

[3] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, March 2016.

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2017.

[5] Franck Djeumou, Cyrus Neary, Eric Goubault, Sylvie Putot, and Ufuk Topcu. Neural networks with physics-informed architectures and constraints for dynamical systems modeling, 2022.

[6] Nigel Goldenfeld and Leo P. Kadanoff. Simple lessons from complexity. *Science*, 284(5411):87–89, 1999.

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[8] Junette Hsin, Shubhankar Agarwal, Adam Thorpe, and David Fridovich-Keil. Gpsindy: Data-driven discovery of equations of motion, 2023.

[9] Xiaojun Jia, Yong Zhang, Baoyuan Wu, Ke Ma, Jue Wang, and Xiaochun Cao. Las-at: Adversarial training with learnable attack strategy, 2022.

[10] Kadierdan Kaheman, J. Nathan Kutz, and Steven L. Brunton. Sindy-pi: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2242), October 2020.

[11] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. A unified gradient regularization family for adversarial examples, 2015.

[12] Hyeonwoo Noh, Tackgeun You, Jonghwan Mun, and Bohyung Han. Regularizing deep neural networks by noise: Its interpretation and optimization, 2017.

[13] Sangwoo Park, Erchin Serpedin, and Khalid Qaraqe. Gaussian assumption: The least favorable but the most useful [lecture notes]. *IEEE Signal Processing Magazine*, 30(3):183–186, 2013.

[14] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[15] Alexander Robey, Luiz F. O. Chamon, George J. Pappas, and Hamed Hassani. Probabilistically robust learning: Balancing average- and worst-case performance, 2022.

[16] Alexander Robey, Fabian Latorre, George J. Pappas, Hamed Hassani, and Volkan Cevher. Adversarial training should be cast as a non-zero-sum game, 2023.

[17] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[18] Jie Wang. An intuitive tutorial to gaussian processes regression. *Computing in Science Engineering*, pages 1–8, 2023.

[19] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy, 2019.