PID-GAN: A GAN Framework based on a Physics-informed Discriminator for Uncertainty Quantification with Physics

Arka Daw*
Virginia Tech
Dept. of Computer Science
darka@vt.edu

M. Maruf*
Virginia Tech
Dept. of Computer Science
marufm@vt.edu

Anuj Karpatne
Virginia Tech
Dept. of Computer Science
karpatne@vt.edu

ABSTRACT

As applications of deep learning (DL) continue to seep into critical scientific use-cases, the importance of performing uncertainty quantification (UQ) with DL has become more pressing than ever before. In scientific applications, it is also important to inform the learning of DL models with knowledge of physics of the problem to produce physically consistent and generalized solutions. This is referred to as the emerging field of physics-informed deep learning (PIDL). We consider the problem of developing PIDL formulations that can also perform UQ. To this end, we propose a novel physics-informed GAN architecture, termed PID-GAN, where the knowledge of physics is used to inform the learning of both the generator and discriminator models, making ample use of unlabeled data instances. We show that our proposed PID-GAN framework does not suffer from imbalance of generator gradients from multiple loss terms as compared to state-of-the-art. We also empirically demonstrate the efficacy of our proposed framework on a variety of case studies involving benchmark physics-based PDEs as well as imperfect physics. All the code and datasets used in this study have been made available on this link [1].

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

KEYWORDS

Uncertainty Quantification; Physics-informed neural networks; Generative Adversarial Networks

ACM Reference Format:
Arka Daw, M. Maruf, and Anuj Karpatne. 2021. PID-GAN: A GAN Framework based on a Physics-informed Discriminator for Uncertainty Quantification with Physics. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467449

*Both authors contributed equally to this research.
[1] https://github.com/arkadaw9/PID-GAN

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467449

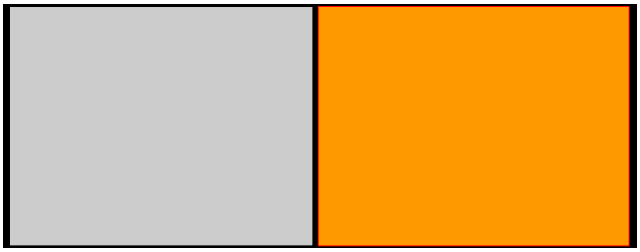


Figure 1: Visualizing differences between our proposed PID-GAN innovations and the state-of-the-art, PIG-GAN [22]

1 INTRODUCTION

As applications of deep learning (DL) continue to seep into critical scientific and engineering use-cases such as climate science, medical imaging, and autonomous vehicles, the importance of performing uncertainty quantification (UQ) with deep learning has become more pressing than ever before. UQ in DL is especially important to ensure trust or confidence in deep learning predictions by end-users of DL frameworks such as scientists and real-world practitioners.

Another aspect of DL formulations that is highly relevant while solving real-world problems in scientific and engineering domains is their ability to incorporate knowledge of the governing physics of the problem in the design and training of DL models. Indeed, there is a rapidly growing body of work in the emerging field of *physics-informed deep learning* [9, 21], where the primary goal is to use physics as another form of supervision for learning *generalizable* DL solutions, even when the number of training labels is small (a problem encountered in many real-world settings). This is commonly achieved by adding *physics-based loss* functions in the training objective of DL models, for capturing the consistency of DL predictions with physics-equations on unlabeled instances (which are plentifully available in many applications).

In this paper, we consider the problem of performing UQ while making use of both physics and data supervision for learning DL models. This is relevant in any scientific application where it is important to generate distributions of output predictions instead of point estimates, while ensuring the learning of physically consistent and generalizable DL solutions. For example, in the domain of climate science, it is important to generate distributions of projected climate variables such as temperature in the future, while ensuring that our predictions comply with the laws of physics such as conservation of mass and energy. A state-of-the-art formulation for this category of problems involves a recent work by Yang et al. [22], where the training of conditional Generative Adversarial Network (cGAN) models were informed using physics-based loss

functions. We refer to this formulation as Physics-Informed Generator (PIG)-GAN, to reflect the fact that the physics-supervision was explicitly used to inform the generator (but not the discriminator).

Since GAN-based frameworks are naturally amenable to generating output distributions by varying the built-in input noise vector, PIG-GAN serves as a pivotal study in incorporating physics knowledge in GAN frameworks for performing UQ. However, PIG-GAN fails to exploit the full potential of the adversarial optimization procedure inherent to GAN-based frameworks for minimizing complex physics-based loss functions (e.g., those encountered in real-world problems involving physics-based PDEs). This is because the discriminator of PIG-GAN is still uninformed by physics and hence, does not leverage the physics-supervision on unlabeled instances. Further, the generator of PIG-GAN suffers from the imbalance of gradient dynamics of different loss terms, as demonstrated theoretically and empirically later in this work.

To address the challenges in state-of-the-art for physics-informed UQ, we propose a novel GAN architecture, termed as **Physics Informed Discriminator (PID)-GAN**, where physics-supervision is directly injected into the adversarial optimization framework to inform the learning of both the generator and discriminator models with physics. PID-GAN allows the use of unlabeled data instances for training both the generator and discriminator models, whereas PIG-GAN uses unlabeled data only for training the generator but not the discriminator. Figure 1 illustrates the major differences between our proposed PID-GAN framework and the state-of-the-art framework, PIG-GAN. These differences impart several advantages to our proposed PID-GAN framework. We theoretically show that PID-GAN does not suffer from the imbalance of Generator gradients, in contrast to PIG-GAN. We also present an extension of PID-GAN that can work in situations even when the physics-supervision is imperfect (either due to incomplete knowledge or observational noise). On a variety of case studies (three problems involving physics-based PDEs and two problems involving imperfect physics), we empirically show the efficacy of our framework in terms of prediction accuracy, physical consistency, and validity of uncertainty estimates, compared to baselines.

The remainder of the paper is organized as follows. Section 2 presents background and related work on physics-informed UQ. Section 3 describes the proposed framework. Section 4 describes experimental results while Section 5 provides concluding remarks.

2 BACKGROUND AND RELATED WORK

2.1 Uncertainty Quantification with DL

Uncertainty quantification (UQ) is an important end-goal in several real-world scientific applications where it is vital to produce distributions of output predictions as opposed to point estimates, allowing for meaningful analyses of the confidence in our predictions. In the context of deep learning, a number of techniques have been developed for UQ, including the use of Bayesian approximations [8, 18, 19] and ensemble-based methods [6, 13, 14, 24]. A simple approach for performing UQ given a trained DL model is to apply Monte Carlo (MC)-Dropout on the DL weights during testing [4]. While MC-Dropout is easy to implement, they are quite sensitive to the choice of dropout rate, which can be difficult to tune [3].

Another line of work for performing UQ in DL is to use generative models like variational autoencoders (VAEs) [2, 11] and cGANs [22, 23]. In a cGAN setting, the generator $G_{\boldsymbol{\theta}}$ learns the mapping from input vector $\mathbf{x}$ and some random noise vector $\mathbf{z}$ to $\mathbf{y}$, $G : (\mathbf{x}, \mathbf{z}) \rightarrow \mathbf{y}$. The generator is trained such that its predictions $\hat{\mathbf{y}} = G(\mathbf{x}, \mathbf{z})$ cannot be distinguished from "real" outputs by an adversarially trained discriminator $D_{\boldsymbol{\phi}}$. The discriminator D, on the other hand, is trained to detect the generator's "fake" predictions. The built-in noise vector z in cGANs can be varied to obtain a distribution on the output predictions $\hat{\mathbf{y}}$. The learning objective of such a cGAN can be written as the following mini-max game:

$$\min_{G_{\boldsymbol{\theta}}} \max_{D_{\boldsymbol{\phi}}} \mathbb{E}_{x,z}\left[\log D(G(x,z),x)\right] + \mathbb{E}_{x,y}\left[\log\left(1 - D(y,x)\right)\right] \quad (1)$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are the parameters of G and D, respectively. In practice, it is common to optimize $\mathbb{E}_{x,z}\left[D(G(x,z),x)\right]$ instead of $\mathbb{E}_{x,z}\left[\log D(G(x,z),x)\right]$ while training the generator [5].

2.2 Physics-Informed Deep Learning (PIDL)

There is a growing volume of work on informing deep learning methods with supervision available in the form of physics knowledge, referred to as the field of physics-informed deep learning (PIDL) [9, 21]. One of the primary objectives of PIDL is to learn deep learning solutions that are consistent with known physics and generalize better on novel testing scenarios, especially when the supervision contained in the labeled data is small. A promising direction of research in PIDL is to explicitly add *physics-based loss functions* in the the deep learning objective, that captures the consistency of neural network predictions on unlabeled instances w.r.t. known physics (e.g., physics-based PDEs [15], monotonic constraints [7, 10], and kinematic equations [17]).

Formally, given a labeled set $\{(\mathbf{x}_{\mathbf{u}_i}, \mathbf{y}_{\mathbf{u}_i})\}_{i=1}^{N_u}$ where $(\mathbf{x}, \mathbf{y})$ denotes an input-output pair, we are interested in learning a neural network model, $\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$, such that along with reducing prediction errors of $\hat{\mathbf{y}}$ on the labeled set, we also want to ensure that $\hat{\mathbf{y}}$ satisfies K physics-based equations, $\{\mathcal{R}^{(k)}(\mathbf{x}, \mathbf{y}) = 0\}_{k=1}^{K}$ on a larger set of unlabeled instances $\{\mathbf{x}_{\mathbf{f}_j}\}_{j=1}^{N_f}$ where $N_f >> N_u$. This can be achieved by adding a physics-based loss in the learning objective that captures the residuals of $\hat{\mathbf{y}}$ w.r.t each physics-equation $\mathcal{R}^{(k)}$ as:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N_u}\sum_{i=1}^{N_u}\left[\mathbf{y}_{\mathbf{u}_i} - \hat{\mathbf{y}}_{\mathbf{u}_i}\right]^2 + \frac{\lambda}{N_f}\sum_{j=1}^{N_f}\sum_{k=1}^{K}\mathcal{R}^{(k)}(\mathbf{x}_{\mathbf{f}_j}, \hat{\mathbf{y}}_{\mathbf{f}_j})^2, \quad (2)$$

where λ is a trade-off hyper-parameter for balancing physics-based loss with prediction errors. Notice that the physics-based loss only depends on model predictions and not ground-truth values, and hence enables the use of unlabeled data to provide an additional form of physics-supervision, along with the supervision contained in the labeled data. A seminal work in optimizing Eq. 2 is the framework of *physics-informed neural networks* (PINN) [15], that was developed for the specific use-case where the physics-equations $\mathcal{R}^{(k)}$ are PDEs. In this paper, we generically refer to all formulations that optimize Eq. 2 as PINN, regardless of the form of $\mathcal{R}^{(k)}$.

While PINN provides a powerful strategy for incorporating physics-supervision in the learning of neural networks, a fundamental challenge in PINN is to balance the gradient flow dynamics of physics loss and prediction errors at different stages (or epochs) of training with a constant λ. To address this challenge, a variant of PINN, referred to as adaptive-PINN (APINN) [20], has recently

been developed to adaptively tune λ at different stages of training for balancing the gradients of different loss terms.

2.3 Physics-Informed UQ

The body of work in PIDL that can perform UQ appears closest to the focus of this paper. Specifically, we are interested in generating uncertainty estimates while ensuring that our predictions lie on a solution manifold that is consistent with known physics. One simple way to achieve this is to implement MC-Dropout for the PINN framework and its variants, so to produce distributions of output predictions. However, as demonstrated in a recent study [3], a major limitation with this approach is that the minor perturbations introduced by MC-Dropout during testing can easily throw off a neural network to become physically inconsistent, even if it was informed using physics during training. We refer to the MC-Dropout versions of PINN and APINN as PINN-Drop and APINN-Drop, respectively, which are used as baselines in our work.

Another line of research in PIDL for UQ involves incorporating physics-based loss functions in the learning objective of cGAN models, which are inherently capable of generating distributions of output predictions. In particular, Yang et al. [22] recently developed a physics-informed GAN formulation with the following objective functions of generator (G) and discriminator (D), respectively:

$$\mathcal{L}_G(\boldsymbol{\theta}) = \frac{1}{N_u}\sum_{i=1}^{N_u} D(\mathbf{x}_{\mathbf{u}_i}, \hat{\mathbf{y}}_{\mathbf{u}_i}) + \frac{\lambda}{N_f}\sum_{j=1}^{N_f}\sum_{k=1}^{K}\left[\mathcal{R}^{(k)}(\mathbf{x}_{\mathbf{f}_j}, \hat{\mathbf{y}}_{\mathbf{f}_j})^2\right], \quad (3)$$

$$\mathcal{L}_D(\boldsymbol{\phi}) = -\frac{1}{N_u}\sum_{i=1}^{N_u}\log\left(D(\mathbf{x}_{\mathbf{u}_i}, \hat{\mathbf{y}}_{\mathbf{u}_i})\right) - \frac{1}{N_u}\sum_{i=1}^{N_u}\log\left(1 - D(\mathbf{x}_{\mathbf{u}_i}, \mathbf{y}_{\mathbf{u}_i})\right), \quad (4)$$

where $\hat{\mathbf{y}}_{\mathbf{u}_i} = G(\mathbf{x}_{\mathbf{u}_i}, \mathbf{z}_{\mathbf{u}_i})$ and $\hat{\mathbf{y}}_{\mathbf{f}_j} = G(\mathbf{x}_{\mathbf{f}_j}, \mathbf{z}_{\mathbf{f}_j})$ are the generator predictions on labeled and unlabeled points, respectively. Notice that in this formulation, the physics-supervision (in terms of $\mathcal{R}^{(k)}$) only appears in the generator objective, while the discriminator only focuses on distinguishing between "real" and "fake" samples on the labeled set (similar to a cGAN discriminator). Hence, we refer to this formulation as Physics-Informed Generator (PIG)-GAN to reflect the fact that only the generator is physics-informed.

While PIG-GAN provides a valid approach for physics-informed UQ, it is easy to observe that it suffers from several deficiencies. First, it makes ineffective use of physics-supervision to only inform the generator, while the discriminator is kept unchanged. As a result, it is unable to use the full power of adversarial optimization procedures inherent to GAN frameworks for jointly minimizing physics-based loss functions along with prediction errors. Instead, it under-utilizes the discriminator to only focus on prediction errors on the labeled set. Second, by design, the discriminator of PIG-GAN is only trained on the small set of labeled instances, thus missing out on the vastly large number of unlabeled instances available in many real-world applications. Third, the use of an explicit trade-off parameter λ to balance physics loss and prediction errors results in a similar problem of gradient flow imbalance as faced by PINN. As we empirically demonstrate later in Section 4.3, this leads to inferior generalization performance compared to our proposed approach.

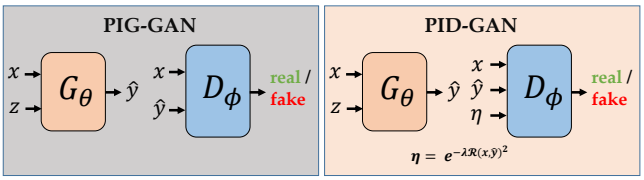


Figure 2: Architecture of PIG-GAN and PID-GAN

3 PROPOSED FRAMEWORK OF PID-GAN

We propose a novel Physics-Informed Discriminator (PID)-GAN formulation that embeds physics-supervision directly into the adversarial learning framework, as shown in Figure 2. Specifically, we utilize the physics residuals to compute a physics consistency score (η) for each prediction, indicating the likelihood of the prediction being physically consistent. These physics consistency scores are fed into the discriminator as additional inputs, such that the discriminator not only distinguishes between real and fake samples by learning from the underlying distribution of labeled points but also using the additional physics-supervision.

Estimating Physics Consistency Scores: Formally, we compute the physics consistency score of a prediction $\hat{\mathbf{y}}$ w.r.t. the k^{th} physical constraint using the following equation:

$$\eta_k = e^{-\lambda \mathcal{R}^{(k)}(\mathbf{x}, \hat{\mathbf{y}})} \quad (5)$$

Larger the value of η_k, it is more likely the prediction $\hat{\mathbf{y}}$ obeys the k^{th} physical constraint, i.e., $\mathcal{R}^{(k)}$ is smaller. The vector notation of would lead to the following: $\boldsymbol{\eta}$, such that $\boldsymbol{\eta} = [\eta_1, \eta_2, \cdots, \eta_K]$.

Training Objective: Our physics-informed discriminator learns the mapping $D : (\mathbf{x}, \mathbf{y}, \boldsymbol{\eta}) \rightarrow \Omega \in [0, 1]$, where Ω represents the probability of a sample being "fake". The objective function of the generator and the discriminator of PID-GAN can then be denoted as:

$$\mathcal{L}_G(\boldsymbol{\theta}) = \frac{1}{N_u}\sum_{i=1}^{N_u} D(\mathbf{x}_{\mathbf{u}_i}, \hat{\mathbf{y}}_{\mathbf{u}_i}, \boldsymbol{\eta}_{\mathbf{u}_i}) + \frac{1}{N_f}\sum_{j=1}^{N_f} D(\mathbf{x}_{\mathbf{f}_j}, \hat{\mathbf{y}}_{\mathbf{f}_j}, \boldsymbol{\eta}_{\mathbf{f}_j}) \quad (6)$$

$$\begin{aligned}\mathcal{L}_D(\boldsymbol{\phi}) = &-\frac{1}{N_u}\sum_{i=1}^{N_u}\log\left(D(\mathbf{x}_{\mathbf{u}_i}, \hat{\mathbf{y}}_{\mathbf{u}_i}, \boldsymbol{\eta}_{\mathbf{u}_i})\right)\\ &-\frac{1}{N_u}\sum_{i=1}^{N_u}\log\left(1 - D(\mathbf{x}_{\mathbf{u}_i}, \mathbf{y}_{\mathbf{u}_i}, \mathbf{1})\right)\\ &-\frac{1}{N_f}\sum_{j=1}^{N_f}\log\left(D(\mathbf{x}_{\mathbf{f}_j}, \hat{\mathbf{y}}_{\mathbf{f}_j}, \boldsymbol{\eta}_{\mathbf{f}_j})\right)\\ &-\frac{1}{N_f}\sum_{j=1}^{N_f}\log\left(1 - D(\mathbf{x}_{\mathbf{f}_j}, \hat{\mathbf{y}}_{\mathbf{f}_j}, \mathbf{1})\right)\end{aligned} \quad (7)$$

where, $\hat{\mathbf{y}}_{\mathbf{u}_i} = G(\mathbf{x}_{\mathbf{u}_i}, \mathbf{z}_{\mathbf{u}_i})$, $\hat{\mathbf{y}}_{\mathbf{f}_j} = G(\mathbf{x}_{\mathbf{f}_j}, \mathbf{z}_{\mathbf{f}_j})$. To interpret generator loss function (Equation 6), let us inspect its two constituent terms. The first and the second term represents the score of the physics-informed discriminator D for the predictions on labeled and unlabeled points, respectively. The generator attempts to minimize both of these scores in order to fool the discriminator into thinking that these generated "fake" predictions $\hat{\mathbf{y}}_{\mathbf{u}}$ and $\hat{\mathbf{y}}_{\mathbf{f}}$ are "real".

On the other hand, the loss function for the discriminator (Equation 7) is comprised of four terms. The first two terms denote the training objective of a binary classifier which distinguishes between

the inputs $(\mathbf{x_{u_i}}, \hat{\mathbf{y}}_{\mathbf{u_i}}, \boldsymbol{\eta}_{\mathbf{u_i}})$ and $(\mathbf{x_{u_i}}, \mathbf{y_{u_i}}, \mathbf{1})$. Note that the consistency score is $\mathbf{1}$ only when $\mathcal{R}^{(k)}(\mathbf{x_{u_i}}, \mathbf{y_{u_i}}) = 0$ for all $k \in \{1, 2, .., K\}$, i.e., the ground truth labels obey the given physics-equations. Similarly, we can interpret the last two terms of equation 7 as the training objective of a binary classifier which tries to distinguish between the inputs $(\mathbf{x_{f_j}}, \hat{\mathbf{y}}_{\mathbf{f_j}}, \boldsymbol{\eta}_{\mathbf{f_j}})$ and $(\mathbf{x_{f_j}}, \hat{\mathbf{y}}_{\mathbf{f_j}}, \mathbf{1})$. Since we don't have labels on the unlabeled set, we use $\hat{\mathbf{y}}_{\mathbf{f_j}}$ as a proxy to the ground truth values. This encourages the generator to increase the physics-consistency score $\boldsymbol{\eta}_{\mathbf{f_j}}$ in order to fool the discriminator on the unlabeled set.

3.1 Analysis of Generator Gradients

Gradient Analysis for PIG-GAN. We assume that the both the inputs $\{\mathbf{x_u}, \mathbf{x_f}\}$ and the outputs $\{\hat{\mathbf{y}}_\mathbf{u}, \hat{\mathbf{y}}_\mathbf{f}\}$ follow same distribution. The backpropagated gradients of the generator can be computed using the chain-rule of differentiation as follows:

$$\begin{aligned}\nabla_\theta \mathcal{L}_G &= \sum_i \nabla_{\hat{\mathbf{y}}_{\mathbf{u_i}}} \mathcal{L}_G . \nabla_\theta \, \hat{\mathbf{y}}_{\mathbf{u_i}} + \sum_j \nabla_{\hat{\mathbf{y}}_{\mathbf{f_j}}} \mathcal{L}_G . \nabla_\theta \, \hat{\mathbf{y}}_{\mathbf{f_j}} \\ &= \sum_i \mathbf{C_{u_i}} . \nabla_\theta \, \hat{\mathbf{y}}_{\mathbf{u_i}} + \sum_j \mathbf{C_{f_j}} . \nabla_\theta \, \hat{\mathbf{y}}_{\mathbf{f_j}}\end{aligned} \tag{8}$$

Let us define $\mathbf{C_{u_i}}$ as the contribution of the i^{th} instance of the labeled set to the backpropagated generator gradients.

$$\mathbf{C_{u_i}} = \nabla_{\hat{\mathbf{y}}_{\mathbf{u_i}}} \mathcal{L}_G = \frac{1}{N_u} \nabla_{\hat{\mathbf{y}}_{\mathbf{u_i}}} D_{\mathbf{u_i}} \tag{9}$$

where, $D_{\mathbf{u_i}} = D(\mathbf{x_{u_i}}, \hat{\mathbf{y}}_{\mathbf{u_i}})$. Similarly, we can define $\mathbf{C_{f_j}}$ as the contribution of the j^{th} instance of the unlabeled set to the backpropagated generator gradients.

$$\begin{aligned}\mathbf{C_{f_j}} = \nabla_{\hat{\mathbf{y}}_{\mathbf{f_j}}} \mathcal{L}_G &= \frac{\lambda}{N_f} \sum_{k=1}^{K} \nabla_{\hat{\mathbf{y}}_{\mathbf{f_j}}} \left(\mathcal{R}_{\mathbf{f_j}}^{(\mathbf{k})}\right)^2 \\ &= \frac{2\lambda}{N_f} \sum_{k=1}^{K} \mathcal{R}_{\mathbf{f_j}}^{(\mathbf{k})} \nabla_{\hat{\mathbf{y}}_{\mathbf{f_j}}} \mathcal{R}_{\mathbf{f_j}}^{(\mathbf{k})}\end{aligned} \tag{10}$$

where $\mathcal{R}_{\mathbf{f_j}}^{(\mathbf{k})} = \mathcal{R}^{(\mathbf{k})}\left(\mathbf{x_{f_j}}, \hat{\mathbf{y}}_{\mathbf{f_j}}\right)$. Since we assume that $\hat{\mathbf{y}}_{\mathbf{u_i}}$ and $\hat{\mathbf{y}}_{\mathbf{f_j}}$ follow similar distributions, $\nabla_\theta \hat{\mathbf{y}}_{\mathbf{u_i}}$ and $\nabla_\theta \hat{\mathbf{y}}_{\mathbf{f_j}}$ would also follow similar distributions. Hence, the overall gradient dynamics would be mostly controlled by the contribution terms $\mathbf{C_{u_i}}$ and $\mathbf{C_{f_j}}$.

Remarks: It is easy to see that $\mathbf{C_{u_i}}$ and $\mathbf{C_{f_j}}$ have widely different functional forms in PIG-GAN. $\mathbf{C_{u_i}}$ depends on $\nabla_{\hat{\mathbf{y}}_{\mathbf{u_i}}} D_{\mathbf{u_i}}$, i.e., it changes as the weights of the discriminator are updated during training. On the other hand, $\mathbf{C_{f_j}}$ depends on $\nabla_{\hat{\mathbf{y}}_{\mathbf{f_j}}} \mathcal{R}_{\mathbf{f_j}}^{(\mathbf{k})}$, which is specific to the set of physics-equations, and does not change as the discriminator is updated. While we can try to balance $\mathbf{C_{u_i}}$ and $\mathbf{C_{f_j}}$ with the help of λ, choosing a constant λ is difficult as $\mathcal{R}_{\mathbf{f_j}}^{(\mathbf{k})}$ changes across epochs and across j.

Gradient Analysis for PID-GAN. The physics consistency score on the labeled and unlabeled sets can be computed as $\boldsymbol{\eta}_{\mathbf{u_i}}^{(\mathbf{k})} = e^{-\lambda \mathcal{R}_{\mathbf{u_i}}^{(\mathbf{k})}}$ and $\boldsymbol{\eta}_{\mathbf{f_j}}^{(\mathbf{k})} = e^{-\lambda \mathcal{R}_{\mathbf{f_j}}^{(\mathbf{k})}}$, respectively, resulting in the following values of $\mathbf{C_{u_i}}$:

$$\begin{aligned}\mathbf{C_{u_i}} &= \frac{1}{N_u} \nabla_{\hat{\mathbf{y}}_{\mathbf{u_i}}} \mathbf{D_{u_i}} + \frac{1}{N_u} \left(\nabla_{\boldsymbol{\eta}_{\mathbf{u_i}}} \mathbf{D_{u_i}}\right) \left(\nabla_{\hat{\mathbf{y}}_{\mathbf{u_i}}} \boldsymbol{\eta}_{\mathbf{u_i}}\right) \\ &= \frac{1}{N_u} \nabla_{\hat{\mathbf{y}}_{\mathbf{u_i}}} \mathbf{D_{u_i}} - \frac{2\lambda}{N_u} \sum_{k=1}^{K} \left(\nabla_{\boldsymbol{\eta}_{\mathbf{u_i}}} \mathbf{D_{u_i}}\right)^{(\mathbf{k})} \boldsymbol{\eta}_{\mathbf{u_i}}{}^{(\mathbf{k})} \mathcal{R}_{\mathbf{u_i}}^{(\mathbf{k})} \nabla_{\hat{\mathbf{y}}_{\mathbf{u_i}}} \mathcal{R}_{\mathbf{u_i}}^{(\mathbf{k})},\end{aligned} \tag{11}$$

where $\left(\nabla_{\boldsymbol{\eta}_{\mathbf{u_i}}} \mathbf{D_{u_i}}\right)^{(\mathbf{k})}$ and $\boldsymbol{\eta}_{\mathbf{u_i}}{}^{(\mathbf{k})}$ denotes the k^{th} term of the gradient $\nabla_{\boldsymbol{\eta}_{\mathbf{u_i}}} \mathbf{D_{u_i}}$ and the vector $\boldsymbol{\eta}_{\mathbf{u_i}}$, respectively. Similarly,

$$\mathbf{C_{f_j}} = \frac{1}{N_u} \nabla_{\hat{\mathbf{y}}_{\mathbf{f_j}}} \mathbf{D_{f_j}} - \frac{2\lambda}{N_u} \sum_{k=1}^{K} \left(\nabla_{\boldsymbol{\eta}_{\mathbf{f_j}}} \mathbf{D_{f_j}}\right)^{(\mathbf{k})} \boldsymbol{\eta}_{\mathbf{f_j}}{}^{(\mathbf{k})} \mathcal{R}_{\mathbf{f_j}}^{(\mathbf{k})} \nabla_{\hat{\mathbf{y}}_{\mathbf{f_j}}} \mathcal{R}_{\mathbf{f_j}}^{(\mathbf{k})} \tag{12}$$

Remarks: Observe that the functional forms of the two contribution terms $\mathbf{C_{u_i}}$ and $\mathbf{C_{f_j}}$ of PID-GANs have similar functional forms. This is not surprising since the generator's objective function is symmetric by formulation, i.e., $\mathbf{x_u}$ and $\mathbf{x_f}$ are interchangeable. If we use the same assumptions on $\{\mathbf{x_u}, \mathbf{x_f}\}$ and the outputs $\{\hat{\mathbf{y}}_\mathbf{u}, \hat{\mathbf{y}}_\mathbf{f}\}$ as made in the case of PIG-GAN, we would see that $\boldsymbol{\eta}_{\mathbf{u_i}}$ and $\boldsymbol{\eta}_{\mathbf{f_j}}$ would have the same distribution. Similarly, it can be shown that $\mathcal{R}_{\mathbf{u_i}}^{(\mathbf{k})}$ and $\mathcal{R}_{\mathbf{f_j}}^{(\mathbf{k})}$ have the same distributions, and $\nabla_{\hat{\mathbf{y}}_{\mathbf{u_i}}} \mathcal{R}_{\mathbf{u_i}}^{(\mathbf{k})}$ and $\nabla_{\hat{\mathbf{y}}_{\mathbf{f_j}}} \mathcal{R}_{\mathbf{f_j}}^{(\mathbf{k})}$ have the same distributions. Hence, each of the individual components of $\mathbf{C_{u_i}}$ and $\mathbf{C_{f_j}}$ would follow the same distribution, and we can thus expect that the magnitudes of the gradients for the labeled and unlabeled components to be similar.

We can also notice the similarities in Equations 10, 11, and 12 to remark that the contribution terms of PID-GAN automatically learn adaptive weights for the physics-based loss that change across training epochs. In particular, all of these contribution formulae have terms involving physics residuals. For the contribution $\mathbf{C_{u_i}}$ in PID-GAN, we can observe that the second term has multiplicative factors, $\left(\nabla_{\boldsymbol{\eta}_{\mathbf{u_i}}} \mathbf{D_{u_i}}\right)^{(\mathbf{k})}$ and $\boldsymbol{\eta}_{\mathbf{u_i}}{}^{(\mathbf{k})}$. While $\boldsymbol{\eta}_{\mathbf{u_i}}{}^{(\mathbf{k})}$ would change as the k^{th} residual varies during training, $\left(\nabla_{\boldsymbol{\eta}_{\mathbf{u_i}}} \mathbf{D_{u_i}}\right)^{(\mathbf{k})}$ depends on the current state of the discriminator and is optimized after every discriminator update. These changing multiplicative factors applied to the physics residual gradients can be viewed as automatically learning adaptive weights for each individual physics residual, as opposed to the use of a constant trade-off parameter λ in PIG-GAN.

3.2 Extension for Imperfect Physics

In some applications, the physics-equations available to us during training may be derived using simplistic assumptions of complex real-world phenomena. For example, while predicting the velocities of two particles just after collision, we usually assume that the energy and the momentum of the system would be conserved. However, such assumptions are only valid in ideal conditions and could easily be misleading when working with noisy real-world labels. To account for situations with imperfect physics, , i.e., when $\mathcal{R}^{(k)}(\mathbf{x}, \mathbf{y}) \neq 0$ for atleast one $k \in \{1, 2, \cdots, K\}$, we provide the following extension of our proposed approach.

The training objective of generator of the PID-GAN under imperfect physics conditions would remain the same (as shown in equation 6. However, the objective for the discriminator would be modified as follows:

$$\mathcal{L}_D(\boldsymbol{\phi}) = -\frac{1}{N_u}\sum_{i=1}^{N_u}\log\left(D(\mathbf{x}_{\mathbf{u}_i}, \hat{\mathbf{y}}_{\mathbf{u}_i}, \boldsymbol{\eta}_{\mathbf{u}_i})\right) - \frac{1}{N_u}\sum_{i=1}^{N_u}\log\left(1 - D(\mathbf{x}_{\mathbf{u}_i}, \mathbf{y}_{\mathbf{u}_i}, \boldsymbol{\eta}'_{\mathbf{u}_i})\right) - \frac{1}{N_f}\sum_{j=1}^{N_f}\log\left(D(\mathbf{x}_{\mathbf{f}_j}, \hat{\mathbf{y}}_{\mathbf{f}_j}, \boldsymbol{\eta}_{\mathbf{f}_j})\right) \tag{13}$$

$$\boldsymbol{\eta}'_{\mathbf{u}_i} = [e^{-\lambda\mathcal{R}^{(1)}(\mathbf{x}_{\mathbf{u}_i}, \mathbf{y}_{\mathbf{u}_i})^2}, e^{-\lambda\mathcal{R}^{(2)}(\mathbf{x}_{\mathbf{u}_i}, \mathbf{y}_{\mathbf{u}_i})^2}, \cdots, e^{-\lambda\mathcal{R}^{(K)}(\mathbf{x}_{\mathbf{u}_i}, \mathbf{y}_{\mathbf{u}_i})^2}] \tag{14}$$

$\boldsymbol{\eta}'_{\mathbf{u}}$ denotes the physics consistency score of the ground truth satisfying the physical constraints on the labeled set. This formulation prevents the generator from predicting samples which blindly satisfies the imperfect physical constraints. Instead, it would learn to mimic the distribution of physics consistency scores of ground truth samples. It must also be noted that the discriminator loss in Equation 13 does not contain the fourth term from the Equation 7, since we cannot compute the physics consistency score of the predictions on the unlabeled set. This introduces an assumption into the model, that the labeled set and the unlabeled set must come from the same distribution. Otherwise, we might end up learning a trivial discriminator D that would look at the input distributions of $\mathbf{x}_\mathbf{u}$ and $\mathbf{x}_\mathbf{f}$ and classify the unlabeled set as fake.

3.3 Mitigating Mode Collapse

GANs are notoriously difficult to train, and it has been observed that sometimes the generator only learns to generate samples from few modes of the data distribution in spite of wide abundance of samples available from the missing modes. This phenomena is commonly known as *mode collapse* [16]. In order to address this problem, Li et. al. [12] provides an information theoretic point of view of using an additional inference model Q_ζ to learn the mapping $Q_\zeta : \{\mathbf{x}, \hat{\mathbf{y}}\} \rightarrow \mathbf{z}$. This inference model delivers stability to the training procedure by mitigating mode collapse while providing a variational approximation to the true intractable posterior over the latent variables $\mathbf{z}$. We utilize this additional network to improve all of our GAN-based formulations.

4 EXPERIMENTAL RESULTS

Case Studies: We evaluate the performance of comparative models on two real-world datasets involving idealized (and imperfect) physics, and three benchmark datasets involving state-of-the-art physics-based PDEs.

Evaluation Metrics: We estimate the deviation of the ground truth values from the mean of our model predictions on every test point using the Root Mean Square Error (**RMSE**). For PDE problems, we use relative L^2-error instead of RMSE, following previous literature [15, 20]. The physical inconsistencies of our predictions are quantified using the absolute **residual** errors w.r.t. the physics-equations. Further, to assess the quality of our generated distributions, we utilize both the **standard deviation** of samples as well as the fraction of ground-truth values that fall within 95% confidence intervals of the sample distributions (referred to as **95% C.I.** fractions). A lower standard deviation with higher 95% C.I. is desired.

Baselines: We compare our proposed PID-GAN with the following baselines: PIG-GAN, cGAN, PINN-Drop, and APINN-Drop.

Table 1: Summary of model performances for collision speed estimation and tossing trajectory prediction in terms of the mean and standard deviation for 10 random runs. For collision speed estimation, the ground truth datapoints have mean residual of 93.96, while for tossing trajectory prediction, the ground truth datapoints have mean residual of 0.59.

Models	PINN-Drop	APINN-Drop	cGAN	PIG-GAN	**PID-GAN**
Collision Speed Estimation					
RMSE	2.34 ± 0.09	2.35 ± 0.06	0.92 ± 0.03	1.65 ± 0.39	**0.73 ± 0.05**
Residual	24.20 ± 2.10	22.80 ± 2.60	96.20 ± 4.50	45.50 ± 23.00	**92.40 ± 0.90**
Std. Dev.	1.60 ± 0.02	1.60 ± 0.02	0.56 ± 0.02	0.03 ± 0.01	0.49 ± 0.04
95% C. I.	79.80 ± 2.80	79.60 ± 0.60	80.60 ± 4.80	1.70 ± 1.10	86.40 ± 1.60
Tossing Trajectory Prediction					
RMSE	0.77 ± 0.01	0.74 ± 0.06	0.81 ± 0.10	0.50 ± 0.05	**0.32 ± 0.04**
Residual	0.48 ± 0.03	**0.62 ± 0.09**	1.68 ± 0.21	0.44 ± 0.08	0.64 ± 0.02
Std. Dev.	1.67 ± 0.02	1.67 ± 0.01	0.45 ± 0.12	**0.09 ± 0.01**	0.09 ± 0.04
95% C. I.	**99.90 ± 0.03**	99.90 ± 0.11	71.39 ± 17.10	29.50 ± 4.34	40.80 ± 11.90

In this section, we briefly describe our results in each case study. Full details of experiments in each case study and additional results focusing on reproducibility are provided in Appendix A.

4.1 Case Study: Collision Speed Estimation

In this real-world problem, we are given the initial speeds $\{v_{a1}, v_{b1}\}$ of two objects $\{a, b\}$ with their respective masses $\{m_a, m_b\}$ and the initial distance d between them as inputs, $X = \{v_{a1}, v_{b1}, m_a, m_b, d\}$. The goal is to estimate their speed after collision $\{v_{af}, v_{bf}\}$. For a perfectly elastic collision, both objects should conserve momentum and energy after the collision, i.e., there is no loss in energy and momentum of the entire system, represented as follows.

$$m_a v_{a_1} + m_b v_{b_1} = m_a v_{a_f} + m_b v_{b_f}$$
$$\frac{1}{2}m_a v_{a_1}^2 + \frac{1}{2}m_b v_{b_1}^2 = \frac{1}{2}m_a v_{a_f}^2 + \frac{1}{2}m_b v_{b_f}^2$$

However, due to the presence of sliding friction during the simulation [1], these ideal physical constraints are violated. Table 1 shows the performance of comparative models on this dataset, where we can see that PID-GAN outperforms all other baselines by a significant margin. At a first glance, it might seem strange that the conservation equations worsen the performance of the models since the cGAN model is the closest competitor to the PID-GAN. However, due to the presence of sliding friction, these equations are violated for ground-truth predictions, and hence models that blindly use these imperfect physics equations (all physics-informed approaches except PID-GAN) show larger test errors. The mean of physics residuals on ground-truth is in fact equal to 93.96. We can see that the residuals of PID-GAN is closest to that of ground-truth. With a lower standard deviation and significantly higher 95% C.I, PID-GAN also produces better uncertainty estimates.

Effect of Varying Training Size: To demonstrate the ability of PID-GAN for showing better generalizability even in the paucity of labeled samples, Figure 3 shows the performance of PID-GAN and PIG-GAN over different training fractions for the collision dataset. We can see that PID-GAN results do not degrade as drastically as PIG-GAN as we reduce the training size. This is because PIG-GAN only uses labeled instances for training the discriminator, which can be susceptible to learning spurious solutions when the labeled

set is small. In contrast, the discriminator of PID-GAN uses both labeled and unlabeled instances and hence leads to the learning of more generalizable solutions.

Analysis of Physics Consistency Scores: Figure 4(a) shows the distribution of the physics consistency score at the last epoch for both PID-GAN and PIG-GAN. Since this problem involves imperfect physics, we can see that physics consistency scores on ground truth labels, $\eta'_{\mathbf{u}}$ (green), is not always exactly equal to 1. While the consistency scores of PIG-GAN predictions on labeled points, $\eta_{\mathbf{u}}$ (red), matches the distribution on ground truth $\eta'_{\mathbf{u}}$, it blindly maximizes the physics consistency scores on the unlabeled (test) points, $\eta_{\mathbf{f}}$ (blue). This is because the discriminator of PIG-GAN is only trained on labeled points and thus is unable to model the physics imperfections on the unlabeled points. On the other hand, the physics consistency scores of PID-GAN predictions accurately match the distribution of consistency scores on ground-truth, on both labeled as well as unlabeled points. This demonstrates the fact that PID-GAN makes effective use of label and physics supervision for training both the generator and discriminator models, thus capturing the distributions of physics consistency scores accurately on both labeled and unlabeled sets.

4.2 Case Study: Tossing Trajectory Prediction

In this problem, we are given the initial three positions of an object as inputs, $X = \{l_1, l_2, l_3\}$, and we want to predict the position of the object for the next 15 time-stamps represented as $\{l_4, l_5, \cdots, l_{15}\}$. In a two-dimensional system, where the position of an object at time i can be represented as $l_i = (l_x, l_y)$, we can adopt the following elementary kinematics equations to model the free-fall physics as additional constraints.

$$l_{x_i} = l_{x_1} + v_x t_i$$

$$l_{y_i} = l_{y_1} + v_y t_i - \frac{1}{2} g t_i^2$$

where, l_{x_i} and l_{y_i} are the object location at time t_i, v_x and v_y are the horizontal and vertical component of the initial velocity, and g is the gravitational acceleration $9.8ms^{-2}$. To introduce imperfect physics scenarios, random accelerations as winds and additional damping factor to imitate air resistance were introduced in the dataset [1]. We can see from Table 1 that PID-GAN outperforms all other baselines for this problem by a significant margin in terms of RMSE. APINN-Drop and PID-GAN perform similarly in terms of residual error (close to the ground truth residual of 0.59); however, APINN-Drop has a much larger RMSE, which makes it worse than PID-GAN. Although the C.I scores of PINN-Drop and APINN-Drop are almost perfect (close to 100), their standard deviation are significantly high, indicating that they are producing under-confident predictions.

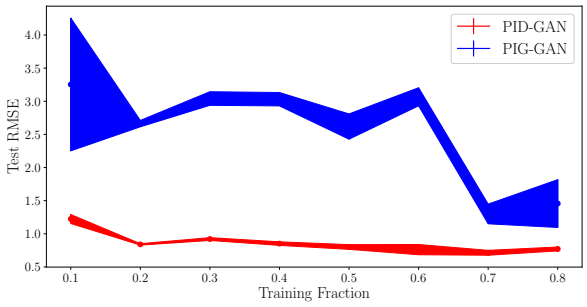
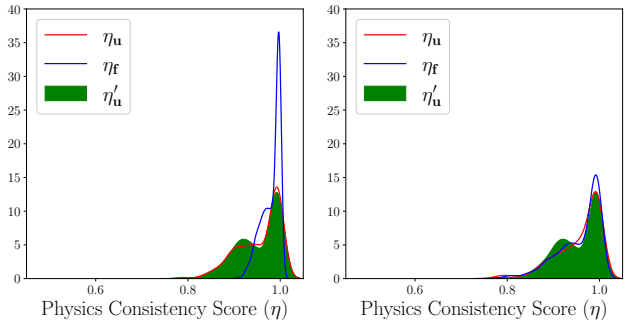


Figure 3: Effect of training size on Collision Dataset



(a) **PIG-GAN** (b) **PID-GAN**

Figure 4: Physics Consistency Score for Collision Dataset

4.3 Case Study: Solving Physics-based PDEs

We test the efficacy of our approach on the problem of solving three physics-based PDEs, which have been used as benchmark in existing literature on PINN [15, 22]. with applications spanning multiple domains. In particular, we study the Burgers', Schrödinger, and Darcy's Equations, briefly described in the following.

Burgers' Equation: We represent the nonlinear time (t) dependent Burgers' equation in one spatial dimension (x) as:

$$u_t + uu_x + vu_{xx} = 0, \quad x \in [-1, 1], \ t \in [0, 1],$$
$$u(0, x) = -\sin(\pi x), \quad u(t, -1) = u(t, 1) = 0.$$

The goal is to predict $u(t, x)$ as output given spatio-temporal coordinates, x and t, as inputs.

Schrödinger Equation: We consider the problem of solving the one-dimensional nonlinear Schrö- dinger equation involving periodic boundary conditions and complex-valued solutions as follows:

$$ih_t + 0.5h_{xx} + |h|^2 h = 0, \quad x \in [-5, 5], \ t \in [0, \pi/2],$$
$$h(x, 0) = 2\,\mathrm{sech}(x), \quad h(-5, t) = h(5, t), \quad h_x(-5, t) = h_x(5, t),$$

where $h(x, t) = u(x, t) + iv(x, t)$ is the complex-valued solution of the equation with $u(x, t)$ as the real part and $v(x, t)$ as the imaginary part. We predict the real and imaginary parts of h as outputs, given x and t as inputs.

Darcy's Equation: We consider the problem of solving Darcy's equation, which is a two-dimensional nonlinear diffusion equation with an unknown state-dependent diffusion coefficient k.

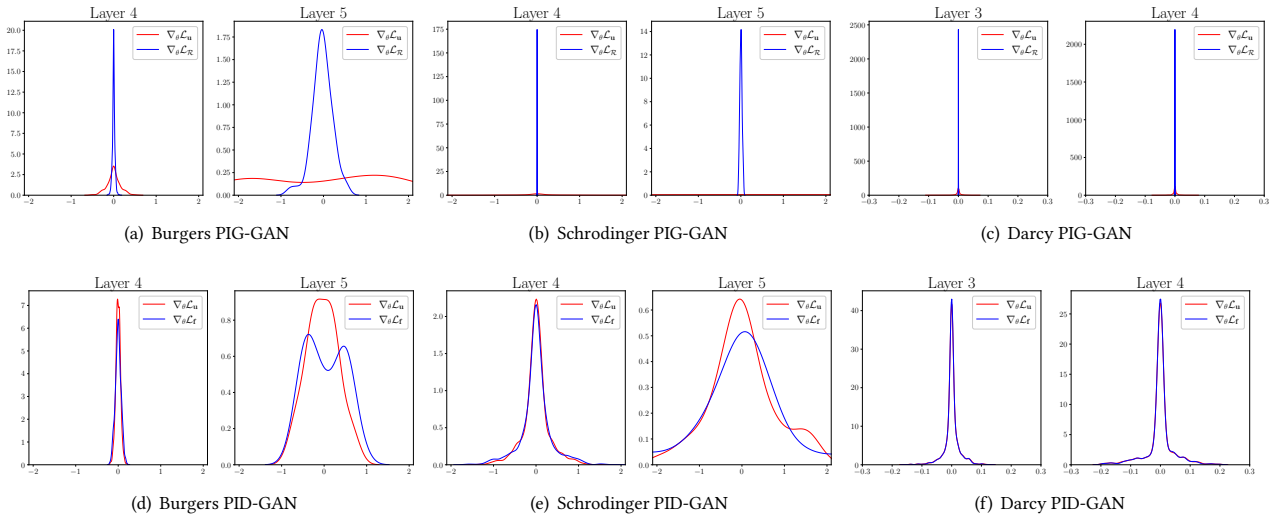
$$\nabla_{\mathbf{x}} \cdot [k(u)\nabla_{\mathbf{x}} u(\mathbf{x})] = 0, \quad \mathbf{x} = (x_1, x_2), \quad u(\mathbf{x}) = u_0, \quad x_1 = L_1$$
$$-k(u)\frac{\partial u(\mathbf{x})}{\partial x_1} = q, \quad x_1 = 0, \quad \frac{\partial u(\mathbf{x})}{\partial x_2} = 0, \quad x_2 = \{0, L_2\}$$

The goal here is predict u given x_1 and x_2 as inputs. Further, the labels for the diffusion coefficient k are not provided during training but is expected to be learned by directly solving the PDE.

Result Comparison on PDEs: For each PDE, we evaluate under two different conditions: deterministic (noise-free) and random boundary conditions (noisy). For noisy conditions, we add 10%

Table 2: Summary of model performances for solving Burgers', Darcy's and Schrödinger equations in terms of the mean and standard deviation for 5 random runs. 95% C.I. corresponds to empirical coverage of 95% predictive intervals.

Condition	Noise-free				Noisy			
Models	PINN-Drop	APINN-Drop	PIG-GAN	**PID-GAN**	PINN-Drop	APINN-Drop	PIG-GAN	**PID-GAN**
Burgers' Equation								
Error-u	0.380 ± 0.027	0.260 ± 0.019	0.215 ± 0.205	**0.100 ± 0.008**	0.370 ± 0.020	0.250 ± 0.040	0.150 ± 0.125	**0.116 ± 0.028**
Residual	0.0040 ± 0.0003	0.0330 ± 0.0040	0.1570 ± 0.2790	**0.0010 ± 0.0004**	0.0040 ± 0.0005	0.0360 ± 0.0100	0.0300 ± 0.0404	**0.0020 ± 0.0003**
Std. Dev.	0.05 ± 0.00	0.06 ± 0.00	0.04 ± 0.01	0.04 ± 0.00	0.05 ± 0.01	0.06 ± 0.00	0.04 ± 0.01	0.04 ± 0.01
95% C. I. (%)	55.89 ± 0.58	68.69 ± 1.10	63.48 ± 35.64	81.16 ± 8.53	56.65 ± 0.88	69.02 ± 4.27	75.54 ± 33.15	80.14 ± 14.63
Schrödinger Equation								
Error-h	0.427 ± 0.001	0.402 ± 0.001	0.112 ± 0.014	**0.045 ± 0.012**	0.428 ± 0.001	0.407 ± 0.002	**0.079 ± 0.008**	0.081 ± 0.014
Residual	0.0100 ± 0.0002	0.0770 ± 0.0002	0.0100 ± 0.0019	**0.0013 ± 0.0002**	0.0096 + ±0.0004	0.0740 ± 0.002	0.0339 ± 0.0058	**0.0007 ± 0.0003**
Std. Dev.	0.04 ± 0.00	0.04 ± 0.00	0.03 ± 0.01	0.01 ± 0.01	0.04 ± 0.00	0.04 ± 0.00	0.02 ± 0.00	0.08 ± 0.02
95% C. I. (%)	69.90 ± 0.36	40.80 ± 1.57	68.72 ± 19.34	47.49 ± 26.75	64.20 ± 1.11	35.90 ± 1.70	49.67 ± 12.96	79.82 ± 11.77
Darcy's Equation								
Error-u	0.010 ± 0.002	**0.009 ± 0.002**	0.013 ± 0.012	**0.009 ± 0.006**	0.012 ± 0.002	0.012 ± 0.001	0.022 ± 0.008	**0.011 ± 0.003**
Error-k	0.450 ± 0.011	0.478 ± 0.013	0.102 ± 0.039	**0.082 ± 0.043**	0.462 ± 0.016	0.540 ± 0.015	0.197 ± 0.045	**0.159 ± 0.020**
Residual	0.0080 ± 0.0005	0.0110 ± 0.0003	0.0005 ± 0.0003	**0.0002 ± 0.0001**	0.0080 ± 0.0004	0.0140 ± 0.0007	0.0030 ± 0.0020	**0.0020 ± 0.0006**
Std. Dev.	0.36 ± 0.01	0.34 ± 0.01	0.11 ± 0.02	0.08 ± 0.01	0.36 ± 0.00	0.35 ± 0.02	0.15 ± 0.05	0.09 ± 0.02
95% C. I. (%)	100.0 ± 0.00	100.0 ± 0.00	84.22 ± 31.56	89.42 ± 13.46	100.0 ± 0.00	100.0 ± 0.00	88.70 ± 13.69	98.97 ± 0.65



(a) Burgers PIG-GAN (b) Schrodinger PIG-GAN (c) Darcy PIG-GAN

(d) Burgers PID-GAN (e) Schrodinger PID-GAN (f) Darcy PID-GAN

Figure 5: Analyzing the Imbalance of Generator Gradients for PIG-GAN and PID-GAN on the three benchmark PDE problems

Gaussian uncorrelated noise to the ground-truth labels which augments the inherent epistemic uncertainty of our predictive model due to the randomness in the boundary conditions.

Table 2 provides a summary of the comparison of different baselines across the three benchmark PDEs for noise-free and noisy conditions. For **Burger's equation**, we see that PID-GAN shows significant improvement in the relative L^2-error of the predictions u in both noisy and noise-free settings. Additionally, the variance of the PID-GAN across 5 random runs is the least, suggesting that it is more robust to random initializations. The PID-GAN also achieves the lowest PDE residual of $1\text{x}10^{-3}$ and $2\text{x}10^{-3}$ in noisy and noise-free setting. This displays the ability of the PID-GAN to generate generalizable and physically consistent solutions. For Burgers' equation, PID-GAN also provides an average 81.16% and 80.14% empirical coverage of the 95% confidence intervals for the two settings, i.e., approximately 80% of the times the ground truth lies between two standard deviations of the predictions. In general, it is preferred to have a higher 95% C.I. with lower standard deviations. On Burger's equation, PID-GAN satisfies both these criteria.

For **Schrödinger Equation**, in terms of relative L^2-error of the predictions h, the PID-GAN performs the best in noise-free setting while PIG-GAN performs slightly better than PID-GAN in the noisy setting (although the difference is not statistically significant and does not carry over for the other PDE experiments). On the other hand, PINN-Drop and APINN-Drop performs significantly worse in both noisy and noise-free conditions. Further, we observe a significantly lower PDE residual for the PID-GAN. The 95% C.I. for the PID-GAN is the best in the noisy setting with lowest std. dev.,

but in the noise-free setting, PINN-Drop shows highest 95% C.I although with a higher std. dev., and a significantly higher L^2-error.

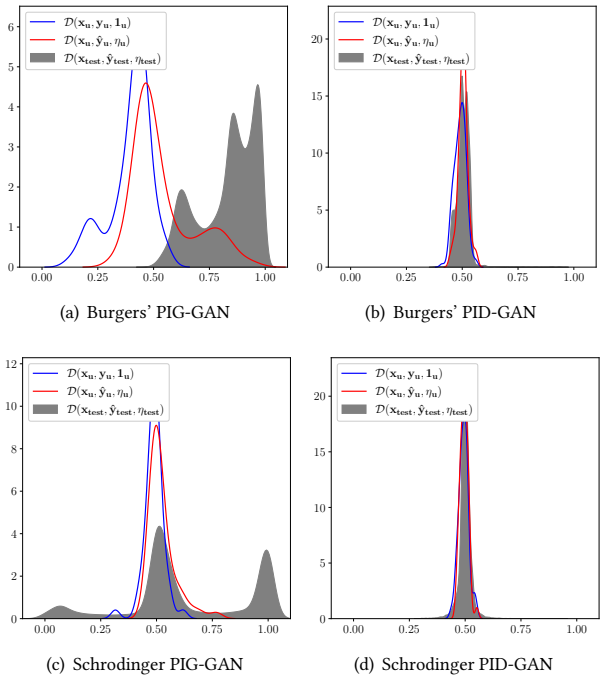
For **Darcy's Equation**, all the baselines show quite similar results w.r.t. the L^2-error on the predictions u, with PID-GAN having a slight edge in both settings. However, when it comes to predicting k (for which no ground truth values were provided during training), PINN-Drop and APINN-Drop performs significantly poor in terms of L^2-errors, with significantly larger standard deviations of samples than other baselines. As a result, even though their 95% C.I. appear perfect, their predictions are not very useful. The GAN based models on the other hand are able to quite precisely estimate the value of k. Also, the PDE residuals are slightly better for the PID-GAN. For the noisy setting, PID-GAN is able to get almost perfect 95% C.I. Additional analyses of the results comparing PID-GAN and APINN-Drop have been provided in Appendix B.

Analyzing Imbalance in Generator Gradients: Figure 5 provides a comparison of the gradients obtained from the last two layers of the generator of PIG-GAN and PID-GAN for the three benchmark PDEs at the last epoch. For the Burgers' equation, we can observe that the gradients of PIG-GAN on the labeled set ($\nabla_\theta \mathcal{L}_\mathbf{u}$) (i.e., the initial and the boundary points) has a much wider distribution than the gradients of the PDE residuals ($\nabla_\theta \mathcal{L}_\mathbf{f}$) computed on the unlabeled set. This imbalance between the gradients indirectly demonstrates the imbalance between the gradient contributions of labeled and unlabeled points in PIG-GAN, supporting the theoretical claims made in Section 3. This problem of PIG-GAN aggravates for the Schrödinger's equation, where we can see that $\nabla_\theta \mathcal{L}_\mathbf{u}$ has an extremely large variance while the variance of $\nabla_\theta \mathcal{L}_\mathbf{f}$ is ultra-low. However, for the Darcy's Equation, the imbalance between the gradients of PIG-GAN is not very visible, indicating that the problem of gradient imbalance in PIG-GAN is use-case dependent. On the other hand, for PID-GAN, the generator gradients for both the labeled and unlabeled set follow the same distribution across all three PDEs, showing no signs of gradient imbalance.

It must be noted that we only visualize the effect of gradient imbalance in the last two layers of the generator. The magnitude of these gradients decreases as we back-propagate deeper into the network. This same phenomena leads to the common problem of vanishing gradients. Thus, the effect of imbalance in the last few layers would be more prominent than its effect in the earlier layers.

Analyzing Discriminator Scores: Figure 6 compares the discriminator outputs of PID-GAN and PIG-GAN on the labeled and unlabeled (test) points after training. For PIG-GAN, we evaluate the fully trained discriminator on three types of inputs: $(\mathbf{x_u}, \mathbf{y_u})$, $(\mathbf{x_u}, \hat{\mathbf{y}}_\mathbf{u})$, and $(\mathbf{x_{test}}, \hat{\mathbf{y}}_\mathbf{test})$. (Note that we evaluate the performance of our model on $\mathbf{x_{test}}$ for PDEs, which is different from the unlabeled set $\mathbf{x_f}$ used during training.) On the other hand, for the PID-GAN, we evaluate the discriminator on three types of inputs: $(\mathbf{x_u}, \mathbf{y_u}, \mathbf{1_u})$, $(\mathbf{x_u}, \hat{\mathbf{y}}_\mathbf{u}, \boldsymbol{\eta}_\mathbf{u})$, and $(\mathbf{x_{test}}, \hat{\mathbf{y}}_\mathbf{test}, \boldsymbol{\eta}_\mathbf{test})$.

On the Burgers' equation for PIG-GAN, we can observe that the discrminator is not able to distinguish between "real" samples $(\mathbf{x_u}, \mathbf{y_u})$ and the generated "fake" samples $(\mathbf{x_u}, \hat{\mathbf{y}}_\mathbf{u})$ since the distribution of these two are similar and centered around 0.5. However, by analyzing its score on $(\mathbf{x_{test}}, \hat{\mathbf{y}}_\mathbf{test})$, we can see that it always tends to predict the samples from the test set as "fake" by scoring them greater than 0.5 on average. This behavior is quite expected since the discriminator of PIG-GAN only learns the decision boundary between the "real" and "fake" samples on the labeled set. However, when we provide the discriminator of PID-GAN with similar inputs, we notice that it has adequately learned the decision boundary between the "real" and generated "fake" samples on both labeled and unlabeled set. Thus, the distribution of discriminator scores for $(\mathbf{x_{test}}, \hat{\mathbf{y}}_\mathbf{test}, \boldsymbol{\eta}_\mathbf{test})$ for PID-GAN is centered at 0.5. Again, for Schrödinger equation, we observe similar behavior for the discriminator scores of PIG-GAN and PID-GAN.

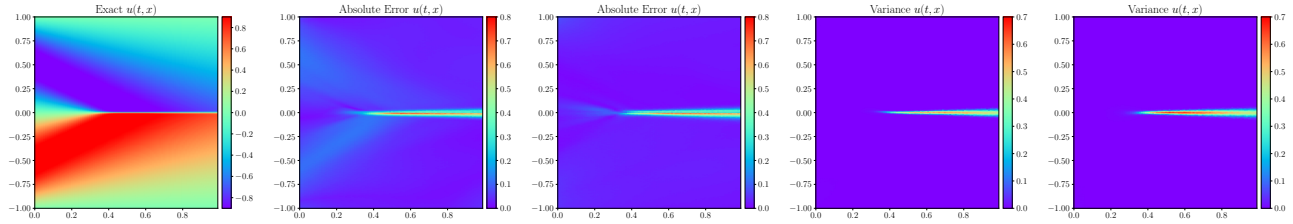


(a) Burgers' PIG-GAN (b) Burgers' PID-GAN

(c) Schrodinger PIG-GAN (d) Schrodinger PID-GAN

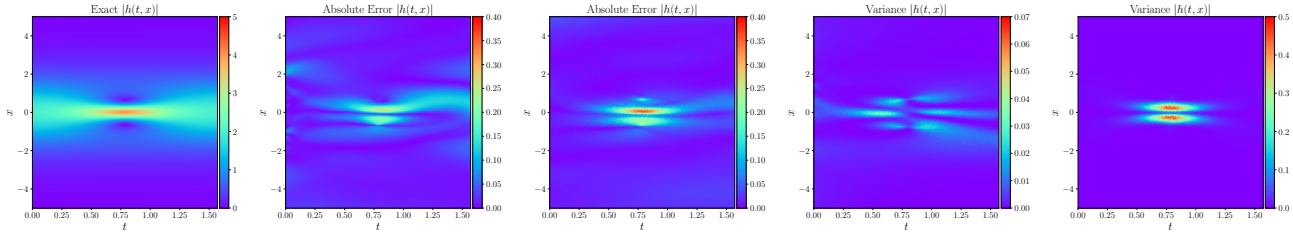
Figure 6: Discriminator Scores of PIG-GAN and PID-GAN

Visualization of PDE Solutions: Figure 7 shows the exact solution of the Burgers' equation along with the absolute errors and variances of PIG-GAN and PID-GAN. Burgers' equation has a non-linear steep response at x=0, where both the predictive models show a strong correlation between regions with higher variances and higher absolute errors, which is a desirable behavior. For example, if the variance of a model is high in a certain region, it suggests that the model is less confident in its predictions and thus can incur larger errors. On the contrary, a model with low variance and a higher value of errors indicates poor and over-confident predictions. Similar to Burgers', Figure 8 shows the exact solutions of the Schrödinger equation along with the absolute errors and variances of PID-GAN and PIG-GAN. Again, we observe two steep responses centered around x = -1 and 1 for t=0.79. Similar to the Burgers', we see PID-GAN has higher variance with a higher absolute error at the steep region. However, the absolute errors and the variances of PIG-GAN are not only concentrated over the steep region but also spread all over the spatio-temporal domain. This means that PIG-GAN is both less confident and more prone to errors in its prediction even in regions with smooth responses. Additional analyses of the results comparing the predicted and exact solutions of Burgers' and Schrödinger equation have been provided in Appendix B.



(a) Exact (b) PIG-GAN Absolute Error (c) PID-GAN Absolute Error (d) PIG-GAN Variance (e) PID-GAN Variance

Figure 7: Visualization of Absolute Error and Variance for Burger's Equation.



(a) Exact (b) PIG-GAN Absolute Error (c) PID-GAN Absolute Error (d) PIG-GAN Variance (e) PID-GAN Variance

Figure 8: Visualization of Absolute Error and Variance for Schrödinger Equation.

5 CONCLUSIONS AND FUTURE WORK

We presented a GAN based framework for UQ with physics-supervision, referred to as PID-GAN. A unique aspect of PID-GAN is its ability to exploit the full potential of adversarial optimization in-built in GAN frameworks for training both the discriminator and generator on labeled and unlabeled points. Future directions of research can explore the use of PID-GANs for inference tasks such as rejection sampling.

6 ACKNOWLEDGEMENT

This work was supported by NSF grant #2026710.

REFERENCES

[1] Yunhao Ba, Guangyuan Zhao, and Achuta Kadambi. 2019. Blending diverse physical priors with neural networks. *arXiv preprint arXiv:1910.00201* (2019).
[2] Vanessa Böhm, François Lanusse, and Uroš Seljak. 2019. Uncertainty Quantification with Generative Models. *arXiv preprint arXiv:1910.10046* (2019).
[3] Arka Daw, R Quinn Thomas, Cayelan C Carey, Jordan S Read, Alison P Appling, and Anuj Karpatne. 2020. Physics-Guided Architecture (PGA) of Neural Networks for Quantifying Uncertainty in Lake Temperature Modeling. In *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 532–540.
[4] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. 1050–1059.
[5] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028* (2017).
[6] Ruihan Hu, Qijun Huang, Sheng Chang, Hao Wang, and Jin He. 2019. The MBPEP: a deep ensemble pruning algorithm providing high quality uncertainty prediction. *Applied Intelligence* 49, 8 (2019), 2942–2955.
[7] Xiaowei Jia, Jared Willard, Anuj Karpatne, Jordan Read, Jacob Zwart, Michael Steinbach, and Vipin Kumar. 2019. Physics Guided RNNs for Modeling Dynamical Systems: A Case Study in Simulating Lake Temperature Profiles. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 558–566.
[8] Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. 2020. Hands-on Bayesian Neural Networks–a Tutorial for Deep Learning Users. *arXiv preprint arXiv:2007.06823* (2020).
[9] Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. 2017. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering* 29, 10 (2017).
[10] Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. 2017. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431* (2017).
[11] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[12] Chunyuan Li, Jianqiao Li, Guoyin Wang, and Lawrence Carin. 2018. Learning to sample with adversarially learned likelihood-ratio. (2018).
[13] Patrick L McDermott and Christopher K Wikle. 2019. Deep echo state networks with uncertainty quantification for spatio-temporal forecasting. *Environmetrics* 30, 3 (2019), e2553.
[14] Tim Pearce, Felix Leibfried, and Alexandra Brintrup. 2020. Uncertainty in neural networks: Approximately bayesian ensembling. In *International conference on artificial intelligence and statistics*. PMLR, 234–244.
[15] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* 378 (2019), 686–707.
[16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498* (2016).
[17] Russell Stewart and Stefano Ermon. 2017. Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*.
[18] Hao Wang and Dit-Yan Yeung. 2016. Towards Bayesian deep learning: A framework and some existing methods. *IEEE Transactions on Knowledge and Data Engineering* 28, 12 (2016), 3395–3408.
[19] Kuan-Chieh Wang, Paul Vicol, James Lucas, Li Gu, Roger Grosse, and Richard Zemel. 2018. Adversarial distillation of bayesian neural network posteriors. In *International Conference on Machine Learning*. PMLR, 5190–5199.
[20] Sifan Wang, Yujun Teng, and Paris Perdikaris. 2020. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv preprint arXiv:2001.04536* (2020).
[21] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. 2020. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919* (2020).
[22] Yibo Yang and Paris Perdikaris. 2019. Adversarial uncertainty quantification in physics-informed neural networks. *J. Comput. Phys.* 394 (2019), 136–152.
[23] Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. 2019. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *J. Comput. Phys.* 397 (2019), 108850.
[24] Ruiyi Zhang, Chunyuan Li, Changyou Chen, and Lawrence Carin. 2018. Learning structural weight uncertainty for sequential decision-making. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1137–1146.

Appendices

A IMPLEMENTATION DETAILS

In this section, we describe the model, data, and training details for reproducing the experiments.

A.1 Procedure for selecting hyparparameters

For the baselines, we choose the exact hyperparameters and model architectures as described in their respective papers.

Architecture selection: In the GAN-based formulation, the discriminator is much easier to train than the generator. To infer the PDE solutions, for every discriminator update, the generator is updated 5 times for all of our GAN-based models. Since we are providing additional information (physics consistency score) to the discriminator, it can easily overfit, which further leads to the vanishing gradient problem. To address this problem, we choose a simpler architecture for the discriminator in the PID-GAN model when compared to its PIG counterpart.

Procedure for selecting λ: We tune the trade-off hyperparameter λ using random search, where we set a heuristic to select the initial $\lambda = 1/(\mathcal{R}_{\theta_0})$ where θ_0 are the generator weights after random initialization. For the baselines shown in PDE problems, we choose the exact trade-off λ as reported in the respective papers, while for other datasets, we perform a random hyperparameter search to select the optimal λ.

We use the Adam optimizer with a learning rate of 10^{-4} for PDE problems and 10^{-3} for other datasets.

A.2 Experimental Setup

In this section, we provide additional information regarding each of the datasets used in the main text. For each of our experiments, we normalized the inputs to follow zero mean and unit standard deviations.

Real-world Dataset: To evaluate our model performance on imperfect physics, we use two real-world datasets: collision prediction dataset, and tossing trajectory prediction dataset. We select 108 labeled points and 436 unlabeled points to train our models on the collision dataset. Meanwhile, we use 217 labeled points with 327 unlabeled points to train on the tossing dataset. All of the comparative baselines are trained for 5,000 epochs and 10,000 epochs on collision and tossing dataset respectively.

Partial Differential Eqations (PDE):

Burgers Equation: Burgers' equation is a fundamental partial differential equation that has multiple applications in areas ranging from applied mathematics like fluid mechanics, nonlinear acoustics, to traffic flow problems. For Burgers' equation, we train our model on 150 labeled points, which are uniformly distributed across the initial condition data $\{(x,t)|t = 0\}$ and the boundary points $\{(x,t)|x \in \{-1,1\}\}$. We further use 10,000 randomly chosen unlabeled points (collocation points for PDEs) to optimize our models. We train the baseline models for 30,000 epochs, which is common practice in the existing literature.

Schrödinger Equation: We use 100 labeled points for Schrodinger equation, where we select 50 uniformly distributed data points across the initial condition $\{(x,t)|t = 0\}$ and 50 uniformly distributed data points across the boundary points $\{(x,t)|x \in \{5,-5\}\}$. Moreover, we select 20,000 unlabeled points (collocation points for PDEs) that are randomly chosen from the input space using the Latin Hypercube Sampling strategy. Similar to the previous works, we train our baseline models for 50,000 epochs.

Darcy's Equation: For Darcy's equation, we use 200 scattered labeled points uniformly distributed over the data space. Additionally, we use 400 boundary points uniformly distributed over the four boundaries. Moreover, we select 10,000 unlabeled points (collocation points for PDEs) that we randomly choose from the input space. All of the baselines are trained over 30,000 epochs.

B ADDITIONAL ANALYSIS OF RESULTS

B.1 Comparing PID-GAN and APINN-Drop on Darcy's Equation

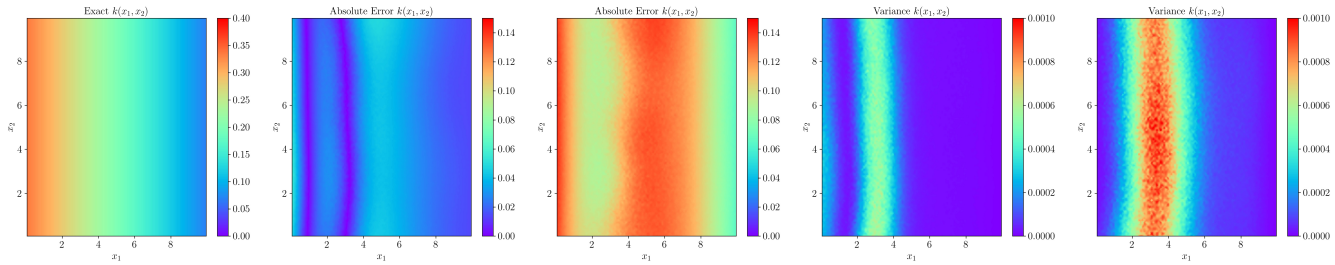
We visualize the performance of the PID-GAN and APINN-Drop for noisy conditions to gain more insights into each of these models. From Table 2, we observed that PID-GAN and APINN-Drop had similar relative L^2 error - u. However, the relative L^2 error - k of APINN-Drop is significantly worse than that of PID-GAN. This is also evident from the plots of Absolute error of k. Also, the variances of the PID-GAN are much lower than those of the APINN-Drop. We observe the trend that PINN variants usually have much larger variances in their predictions, thus achieving higher values of 95% C.I. It can be inferred that APINN-Drop is usually underconfident in its predictions. However, this behavior might not be desirable. Ideally, we would want to have a higher value of 95% C.I. with lower values of standard deviations. PID-GAN on the other hand, generates significantly lower errors in k while having lower variances with a relatively high value of 95% C.I.

B.2 Prediction comparison for Burgers' equation

Figure 10 shows the comparison of the predicted and the exact solutions of Burgers' equation for different baselines at t=0.5 snapshot. It is evident from the figure that at x=0, there is a steep slope, which is hard to predict by conventional neural networks. The original PINN paper, which does point estimates, shows better performance on finding the exact solution of Burgers' equation than our dropout based PINN-Drop method. This justifies our observation on MC-Dropout that its minor perturbation can easily throw-off a model to become physically inconsistent, which motivates us to estimate the uncertainty using GAN-based models.

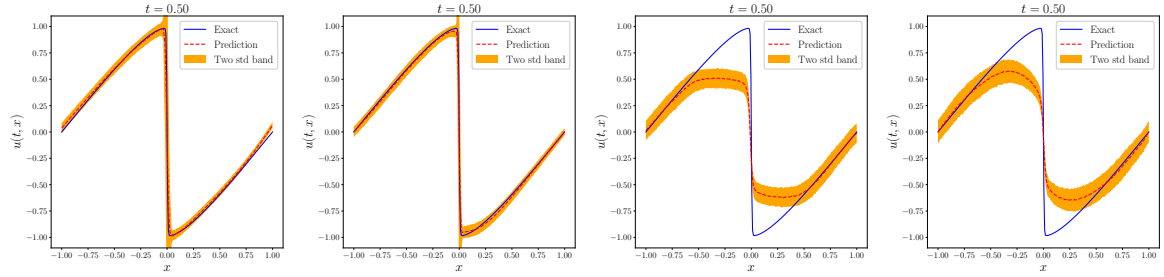
B.3 Prediction comparison for Schrödinger equation

Figure 11 illustrates a similar observation for MC-Dropout based models. PINN-Drop and APINN-Drop perform well on the smooth response region, whereas, for the steep response region, the predictions and the uncertainty of these models are inconsistent. GAN-based model predictions are close to the exact solutions, and from the results, the PID-GAN performs much better than the PIG-GAN in terms of residual error and uncertainty estimate values.



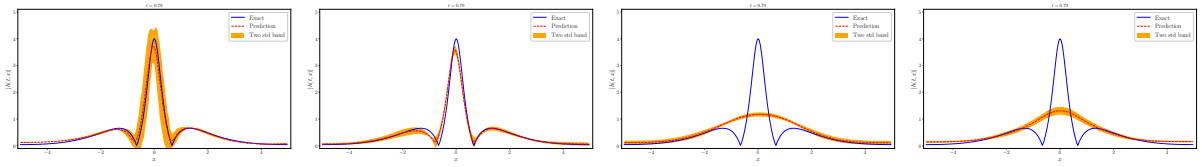
(a) Exact k (b) PID-GAN Absolute Error k (c) APINN-Drop Absolute Error k (d) PID-GAN Variance k (e) APINN-Drop Variance k

Figure 9: Comparison of PID-GAN and Adaptive PINN-Drop in terms of absolute error and variance on Darcy's equation.



(a) PID-GAN (b) PIG-GAN (c) PINN-Drop (d) APINN-Drop

Figure 10: Comparison of the predicted and exact solutions of Burgers' equation corresponding to $t = 0.50$ snapshot.



(a) PID-GAN (b) PIG-GAN (c) PINN-Drop (d) APINN-Drop

Figure 11: Comparison of the predicted and exact solutions of Schrödinger equation corresponding to $t = 0.79$ snapshot.