Применение физически информированного глубокого обучения для решения дифференциальных уравнений

Даниил Лаптев

2 апреля 2024 г.

Содержание

Введение

Physics-Informed Neural Networks (PINNs) [16] обрели популярность в исследовательской среде благодаря, с одной стороны, успехам глубокого обучения для решения самых разных научных и инженерных задач [15, 2], а с другой стороны - огромному количеству вопросов, возникающих в рамках применения этого подхода, и чрезвычайной гибкости его применения. Частный, но достаточно распространённый случай применения этой методологии - решение дифференциальных уравнений и их систем, который мы и рассматриваем в данной работе.

Значительная часть литературы адресует проблему конкретной реализации PINNs, например, для решения тех или иных специфических задач и проблем [1, 8, 7], ускорения их обучения [19] или репрезентации специфических свойств рассматриваемой системы [9, 12, 4]. Кроме того, довольно развита область оценки погрешности и обобщающей способности PINNs [7, 18, 14]. Относительно недавний обзор PINNs и соответствующей литературы можно найти в [3], а также в [5].

Тема гиперпараметров в этой области в большой степени фрагментирована и неоднородна. Этап оптимизации гиперпараметров выступает скорее как инженерная рутина, результаты которой имеют ценность исключительно для какой-либо конкретной ситуации и которая выполняется в полуавтоматическом режиме - больший интерес представляют способы уменьшения числа гиперпараметров, умный дизайн самой PINN и функции потерь. Это не является неожиданностью, поскольку в рамках PINNs существует большое множество различных способов решения той или иной задачи, из-за чего довольно тяжело делать обобщённые выводы.

Тем не менее, в практике обучения PINNs известна серьёзная зависимость качества решения от способа инициализации нейросетевой модели, не в полной мере изучен вопрос выбора оптимальной функции потерь и коэффициентов её термов, не до конца разработан вопрос оптимального выбора точек коллокации для оценки вычисления функции потерь, не вполне ясна роль размера нейросетевой модели и так далее. Литература в этой области существует [13, 17, 6], однако работ, посвящённых общим методологическим выводам и широкому кругу задач, пока не появилось (за исключением некоторых обзорных статей, в которых эти темы практически не обсуждаются). Возникает также ряд теоретических вопросов: например, почему найденные оптимальные гиперпараметры являются оптимальными? Почему для данной задачи необходим такой размер нейросетевой модели и такое число точек коллокации, а для другой задачи - другие? При анализе оптимального learning rate возникает также вопрос о том, каковы особенности ландшафта функции потерь для моделей PINNs.

В нашей работе мы предпринимаем попытку начать движение в сторону ответа на эти вопросы, проведя систематическое экспериментальное исследование гиперпараметров на различных задачах и предложив способ классификации сеттинга PINNs. Целью нашей работы является исследование влияния гиперпараметров на качество решения задачи в сеттинге PINNs. Для достижения цели нами были поставлены следующие задачи:

1. Реализовать PINNs для решения выбранных дифференциальных уравнений.
2. Выявить связь между гиперпараметрами и качеством решения задачи.
3. Зафиксировать основные вопросы и проблемы, которые возникают при использовании PINNs, и выдвинуть гипотезы, которые могут разрешить их.

В данной работе мы концентрируемся на решении дифференциальных уравнений (ДУ) с помощью метода PINNs, ограничившись классическим вариантом непрерывного PINNs, который описан в работе [16]. Кроме того, наша реализация методологии использует так

называемые мягкие ограничения (soft constraints), которые, в отличие от жёстких ограничений (hard constraints), позволяют нейросетевой модели совершать ошибки на границе домена [3]. Тема обнаружения ДУ, а также различных расширений метода остаётся вне поля нашего внимания.

Обзор литературы

1 Методология Physics-Informed Neural Networks

Рассмотрим общую постановку проблемы. Решением ДУ называется функция u, определённая на пространстве Ω, которая удовлетворяет условиям:

$$\begin{aligned} \mathcal{D}u &= f, \\ \mathcal{B}u &= g, \end{aligned} \tag{1}$$

где $\mathcal{D}$ - дифференциальный оператор для всего домена, $\mathcal{B}$ - дифференциальный оператор на границе домена, f и g - какие-либо заданные функции.

Опираясь на свойство универсальной аппроксимации [10], мы можем пожелать приблизить искомую функцию некоторой нейросетевой моделью с параметрами θ. Тогда проблема решения ДУ сведётся к задаче оптимизации (обучения) относительно какой-либо функции потерь.

Пусть $\mathcal{N}(x;\theta)$, где $x \in \Omega$ - нейросетевая модель. Для удобства будем писать $\mathcal{N}_\theta$. То, насколько нейросетевая модель удовлетворяет условиям (1), можно выразить следующим образом:

$$\begin{aligned} L_{\mathcal{D}}(\theta) &= \|\mathcal{D}\mathcal{N}_\theta - f\|, \\ L_{\mathcal{B}}(\theta) &= \|\mathcal{B}\mathcal{N}_\theta - g\|, \end{aligned} \tag{2}$$

Тогда оптимизируемую функцию потерь можно записать как

$$\mathcal{L}(\theta) = w_1 L_{\mathcal{B}}(\theta) + w_2 L_{\mathcal{D}}(\theta), \tag{3}$$

где коэффициенты w_1 и w_2 нужны для того, чтобы балансировать между важностью граничного и внутреннего термов функции потерь. В тех или иных условиях может потребоваться пожертвовать точностью на границе для того, чтобы улучшить точность внутри домена, и наоборот. Для удобства мы введём параметр C и определим $w_1 = C$, $w_2 = (1-C)$, как в работе [13], тем самым уменьшив число гиперпараметров на один.

В конечном итоге для какой-либо фиксированной нейросетевой архитектуры мы ищем такой набор параметров θ^*, что

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta)$$

Рассмотрим (2) и (3). Минимизацию функции потерь $L_{\mathcal{B}}$ можно воспринимать как классическое обучение с учителем, поскольку на границе нам известны значения искомой функции и/или её производных; в свою очередь, $L_{\mathcal{D}}$ является регуляризационным термом, который ограничивает пространство допустимых параметров. L1 и L2 регуляризация направляет оптимизатор в те области, в которых норма параметров будет небольшой; регуляризация с помощью $L_{\mathcal{D}}$, который часто называется PINN-термом функции потерь, направляет оптимизатор в те области, в которых нейросетевая модель будет удовлетворять дифференциальному уравнению внутри домена. Иными словами, мы ищем параметры нейросети в таком множестве, которое можно описать следующим образом:

$$\Theta = \{\theta : \quad \|\mathcal{D}\mathcal{N}_\theta - f\| \approx 0 \quad \text{и} \quad \|\mathcal{B}\mathcal{N}_\theta - g\| \approx 0\}$$

Поскольку на практике мы не можем достичь равенства нулю, мы используем знак $\approx$. В разных контекстах близость к нулю может пониматься по-разному. Если решение поставленной задачи существует и единственно, то минимизация описанных выше функций потерь означает также и приближение искомой функции, причём ровно одним способом (подробнее см. в разделе 4).

Чтобы получить значение функции потерь, обычно используется Монте-Карло оценка L2 нормы, по-другому известная как Mean Squared Error. Например, если на границе нами выбрано (или нам дано) $N_{\mathcal{B}}$ пар вида (x_i, g_i), а внутри домена мы каким-либо образом выбираем $N_{\mathcal{D}}$ точек x_i, то функцию ошибки можно оценить как

$$L_{\mathcal{D}}(\theta) = \frac{1}{N_{\mathcal{D}}} \sum_{i=1}^{N_{\mathcal{D}}} (\mathcal{D}\mathcal{N}_\theta(x_i) - f(x_i))^2,$$
$$L_{\mathcal{B}}(\theta) = \frac{1}{N_{\mathcal{B}}} \sum_{i=1}^{N_{\mathcal{B}}} (\mathcal{B}\mathcal{N}_\theta(x_i) - g(x_i))^2$$

Формально разницы между этими оценками нет, однако нередко на границе заданы сами значения искомой функции (например, если граничные условия поставлены в форме Дирихле) и/или её производных (например для задачи Коши, в которой известно значение функции и её производных в начальный момент времени). Интуитивно мы можем понимать $L_{\mathcal{B}}$ как условие на точное значение искомой функции, а $L_{\mathcal{D}}$ как условие на характер поведения искомой функции на всём домене.

Дифференциальный оператор, как граничный, так и внутренний, легче всего приблизить с помощью алгоритма автоматического дифференцирования. Тогда частные производные любого порядка могут быть вычислены для любой точки, в которой мы оцениваем значение нейросетевой модели. Единственное условие - наличие функции активации подходящей гладкости, чтобы производные высших порядков вообще существовали.

Процесс оптимизации чаще всего реализуется с помощью алгоритмов Adam и L-BFGS [3], нередко в комбинации - сначала происходит тренировка с помощью Adam, затем тренировка продолжается с помощью L-BFGS. В нашем исследовании мы ограничимся алгоритмом Adam.

В качестве нейросетевой архитектуры мы используем модели прямого распространения (FF), которые представляются как композиция аффинных и нелинейных преобразований:

$$\begin{aligned} Linear_i(x) &= W_i x + b_i, \\ FF(z) &= Linear_L(...\phi(Linear_1(z))), \end{aligned} \tag{4}$$

где ϕ - некая функция активации.

Для наших экспериментов мы возьмём такие модели, у которых количество строк в матрицах $W_1, ..., W_{L-1}$ одинаково. Пусть это количество (ширина нейросети) записывается как W. Тогда модель глубины L и ширины W будет обозначаться как $\mathrm{FF}_{L,W}$. Если каждый параметр в нейросети пронумерован и всего их p, то вектор $\theta \in \mathbb{R}^p$ можно взаимно-однозначно соотнести с параметрами нейросетевой модели.

В общем случае нейросетевые модели могут быть произвольными.

2 Решение дифференциальных уравнений

2.1 Гармонический осциллятор с затуханием

На данный момент нас не интересуют её физические свойства, только математическая формулировка. Система описывается следующим образом:

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + 2\zeta\omega_0\frac{\mathrm{d}x}{\mathrm{d}t} + \omega_0^2 x = 0, \tag{5}$$

где функция $x(t) : \mathbb{R} \to \mathbb{R}$ описывает положение осциллирующей массы, а параметры ζ и ω_0 влияют на характер затухания осцилляции.

Пусть дано начальное положение массы $x(0) = x_0$ и начальная скорость $v(0) = \frac{\mathrm{d}x}{\mathrm{d}t}(0) = v_0$. Зафиксируем некоторую нейросетевую архитектуру $\mathcal{N}(z;\theta)$. Тогда мы должны отыскать такие параметры θ, которые минимизируют следующий функционал:

$$L(\theta) = \frac{C}{2}\left\| \begin{bmatrix} \mathcal{N}_\theta(0) - x_0 \\ \frac{\mathrm{d}\mathcal{N}_\theta}{\mathrm{d}t}(0) - v_0 \end{bmatrix} \right\|_2^2 + \frac{1-C}{N_\mathcal{D}} \sum_{i=1}^{N_\mathcal{D}} \left(\Big[\frac{\mathrm{d}^2}{\mathrm{d}t^2} + 2\zeta\omega_0\frac{\mathrm{d}}{\mathrm{d}t} + \omega_0^2 \Big] \mathcal{N}_\theta(t_i) \right)^2$$

2.2 Система Лотки-Вольтерры

Теперь рассмотрим модель взаимодействия двух биологических видов, один из которых выполняет роль хищника, другой - жертвы. Такая система часто моделируется с помощью стандартной системы Лотки-Вольтерры. Пусть число особей в их популяциях описывается, соответственно, $y(t)$ и $x(t)$, которые являются непрерывными функциями. Тогда динамика числа особей может быть выражена с помощью системы ОДУ первого порядка

$$\begin{cases} \frac{\mathrm{d}x}{\mathrm{d}t} = \alpha x - \beta xy \\ \frac{\mathrm{d}y}{\mathrm{d}t} = \delta yx - \gamma y \end{cases}, \tag{6}$$

где параметры $\alpha, \beta, \delta, \gamma$ характеризуют рождаемость и смертность популяций.

Пусть нейросетевая модель $\mathcal{N} : \mathbb{R} \to \mathbb{R}^2$ будет аппроксимировать значение сразу двух искомых функций в момент времени t посредством двумерного вектора на её выходе, и мы будем писать

$$\mathcal{N}(t;\theta) = \begin{bmatrix} \mathcal{X}(t) \\ \mathcal{Y}(t) \end{bmatrix}_\theta \approx \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$$

Пусть даны начальные значения x_0, y_0. Нам необходимо найти такие параметры θ, которые минимизируют следующие функции ошибки:

$$\begin{aligned} \mathcal{L}_\mathcal{I} &= \frac{1}{2}\Big(\mathcal{N}(0;\theta) - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \Big)^2 \\ \mathcal{L}_\mathcal{X} &= \frac{1}{N_\mathcal{D}} \sum_{i=1}^{N_\mathcal{D}} \Big(\frac{\mathrm{d}\mathcal{X}}{\mathrm{d}t}(t_i) - \alpha\mathcal{X}(t_i) + \beta\mathcal{X}(t_i)\mathcal{Y}(t_i) \Big)^2 \\ \mathcal{L}_\mathcal{Y} &= \frac{1}{N_\mathcal{D}} \sum_{i=1}^{N_\mathcal{D}} \Big(\frac{\mathrm{d}\mathcal{Y}}{\mathrm{d}t}(t_i) - \delta\mathcal{X}(t_i)\mathcal{Y}(t_i) + \gamma\mathcal{Y}(t_i) \Big)^2 \end{aligned} \tag{7}$$

В качестве итоговой функции потерь мы возьмём их линейную комбинацию:

$$L_\mathcal{T} = CL_\mathcal{I} + (1-C)(L_\mathcal{X} + L_\mathcal{Y})$$

2.3 Система Лоренца

2.4 Уравнение диффузии

Эта модель используется как самостоятельное описание процесса диффузии или как особый случай некоторых других моделей. Мы рассмотрим одномерный случай и задачу с граничными условиями.

Пусть имеется функция $u(\mathbf{x}, t) : \mathbb{R}^{N+1} \to \mathbb{R}$, которая описывает, например, концентрацию вещества в какой-либо точке пространства и времени (в нашем случае $N = 1$). Пусть число D, называемое коэффициентом диффузии, является постоянным на всей ограниченной области $[A, B] \times [0, T]$. Кроме того, на границах этой области нам известно поведение искомой функции. Тогда краевую задачу для уравнения диффузии мы запишем как

$$\begin{aligned} \frac{\partial u}{\partial t} &= D\frac{\partial^2 u}{\partial x^2}, \\ u(x, 0) &= f(x), \\ u(A, t) &= g_1(t), \\ u(B, t) &= g_2(t), \end{aligned} \tag{8}$$

причём вместо явно заданных функций на границах мы можем иметь лишь некоторый набор измерений. Методы машинного обучения позволят приблизить решение на основе неполных данных, что является ещё одним достоинством PINNs.

Пусть при $t = 0$ и на границах выбирается соответственно N_I, N_A, N_B точек вместе с заданными в них значениями искомой функции. Внутри области мы берём N_D произвольных точек. Тогда функции потерь мы можем записать как

$$\begin{aligned} L_{\mathcal{I}} &= \frac{1}{N_I}\sum_{i=1}^{N_I}(\mathcal{N}(x_i, 0) - f(x_i))^2, \\ L_{\mathcal{B}} &= \frac{1}{N_A}\sum_{i=1}^{N_A}(\mathcal{N}(A, t_i) - g_1(t_i))^2 + \frac{1}{N_B}\sum_{i=1}^{N_B}(\mathcal{N}(B, t_i) - g_2(t_i))^2, \\ L_{\mathcal{D}} &= \frac{1}{N_D}\sum_{i=1}^{N_D}\Big(\frac{\partial \mathcal{N}}{\partial t}(x_i, t_i) - D\frac{\partial^2 \mathcal{N}}{\partial x^2}(x_i, t_i)\Big)^2 \end{aligned} \tag{9}$$

Полная функция потерь в нашей реализации выглядит следующим образом:

$$L = C(L_{\mathcal{I}} + L_{\mathcal{B}}) + (1 - C)L_{\mathcal{D}}$$

2.5 Модель Грея-Скотта

3 Анализ гиперпараметров

Мы занимаемся поиском точки θ^* из пространства параметров $\mathbb{R}^p$, которая удовлетворяет условиям **??**. Поиск производится итеративно с помощью алгоритма градиентного спуска. На i-й итерации для набора данных $\mathbf{X}$ мы вычисляем градиент функции потерь $\nabla\mathcal{L}(\mathbf{X}; \theta_i)$ относительно параметров θ_i с помощью автоматического дифференцирования, и выполняем переход к θ_{i+1} согласно какому-либо правилу (алгоритму оптимизации), тем самым обновляя параметры нашей нейросетевой модели.

Этот процесс можно представить как движение оптимизатора по *ландшафту потерь*, который определяется как поверхность, заданная сильно невыпуклой функцией потерь $\mathcal{L} : \mathbb{R}^p \to \mathbb{R}$. В силу невыпуклости ландшафта, а также неоптимальной желаемой скорости

сходимости оптимизатора, мы можем столкнуться с различными проблемами его движения (например, застревание в локальных минимумах или резкие минимумы, в которые оптимизатор зайти не может), которые обычно преодолеваются либо конструированием нового ландшафта потерь, либо подбором более подходящих гиперпараметров.

Таким образом, в вопросе сходимости процесса обучения является важным рассмотрение геометрии ландшафта потерь и движения по нему оптимизатора. Теоретические исследования показывают, что геометрия ландшафта потерь имеет некоторую связь со способностью нейросетевой модели к обобщению, и, кроме того, выбор нейросетевой архитектуры может значительно влиять на сложность оптимизации из-за изменения ландшафта потерь [11]. Анализ ландшафта потерь как такового мы оставляем на другой раз, однако для лучшего понимания процесса обучения мы должны разделять эти две его стороны.

Говоря о ландшафте потерь, мы также понимаем под этим вычислительный граф, который формирируется при вычислении значения функции потерь. Всё, что влияет на результат этих вычислений, влияет и на геометрию ландшафта потерь.

Введём следующую классификацию параметров:

1. Краевая задача: дифференциальное уравнение, граничные условия, параметры γ.
2. Нейросетевая модель: архитектура и конкретный способ её использования.
3. Задача оптимизации: функция потерь и коэффициенты входящих в неё термов.
4. Обучение: параметры оптимизатора, инициализация модели, точки коллокации.

Эта классификация поможет нам быть более осмотрительными в вопросе анализа гиперпараметров. В общем случае можно сказать, что в ней существуют только 3 и 4 пункты, т.к. нейросеть и постановка задачи встроены в функцию потерь, однако мы рассчитываем использовать нейросетевую модель после обучения, поэтому выделяем её в отдельную сущность, а постановку задачи рассматриваем как наиболее важный пункт, определяющий все остальные. Ландшафт потерь определяется первыми тремя пунктами, движение по нему - четвёртым.

Ради упрощения мы предположим, что на каждой итерации процесса оптимизации используются одни и те же граничные точки и соответствующие им значения искомой функции, т.е., используется фиксированный заранее полный тренировочный датасет. Кроме того, мы рассмотрим лишь решение модели Лотки-Вольтерры (второй подход) и уравнение диффузии.

Для анализа гиперпараметров мы выберем четыре различные постановки задачи, обнаружим для него удовлетворительное решение и будем отталкиваться от полученного набора гиперпараметров и итогового RMSE. Детали описаны в приложении **??**.

4 Теоретические гарантии

4.1 Теоретические гарантии сходимости

4.2 Оценка глобальной погрешности решения

Простейший способ проанализировать, насколько хорошо работает наше нейросетевое решение - сравнить его с общепринятыми методами численного решения, а ещё лучше - рассмотреть задачи с доступным точным решением. Доверие к нему, таким образом, будет опираться на опыт. Однако существует ряд теоретических методов, которые мы рассмотрим далее.

Возьмём нотацию, описанную в [3]. Пусть $u(z) : \Omega \to X$ - искомая функция, $\mathcal{N}(z;\theta)$ - нейросетевая модель с фиксированной архитектурой. Мы хотим получить оценку величины

$$\|\mathcal{N}(z;\theta) - u(z)\|,$$

основываясь на информации об архитектуре модели и тренировочных данных, а также, может быть, на догадках о каких-либо свойствах искомой функции.

Стоит ещё раз заметить, что функция $\mathcal{N}(z;\theta)$ не определяется единственным образом своими параметрами, а имеет также и архитектуру - конкретный способ реализации параметров. Он может заметно влиять на качество решения и его также необходимо брать в расчёт. Для того, чтобы стандартизовать анализ, можно ввести понятие сложности функции, или её экспрессивности, и выработать способ количественного измерения этой характеристики. Возможно, это позволит упростить методологию.

Пусть дан набор точек z_i и соответствующих значений функции $u(z_i)$, где $i \in \{1, ..., N\}$ (например, это какая-либо совокупность измерений искомого решения ДУ, начальных и граничных условий). Тогда *эмпирический риск* для модели с фиксированными параметрами θ определяется как

$$\hat{\mathcal{R}}[\mathcal{N}] = \frac{1}{N}\sum_{i=1}^{N} \|\mathcal{N}(z_i) - u(z_i)\|^2$$

Общий *риск* модели, характеризующий её способность аппроксимировать функцию на всём Ω, будет определяться как

$$\mathcal{R}[\mathcal{N}] = \int_{\Omega} (\mathcal{N}(z) - u(z))^2 dz$$

Качество решения можно оценить, исходя из трёх факторов.

Ошибка оптимизации. Обозначим модель, полученную в результате тренировки, как $\mathcal{N}^*$. Мы введём *ошибку оптимизации* как меру качества тренировочного процесса для данной нейросетевой архитектуры на данных тренировочных точках. Мы ожидаем, что если достигнут минимум ошибки оптимизации, дальнейший способ улучшения качества - поиск новых данных и использование новых архитектур.

Обычно ошибка оптимизации нам неизвестна на практике, т.к. функция потерь является сильно невыпуклой, а стандартные методы оптимизации дают лишь приближённое решение (нередко застревая в локальном минимуме). Математически она будет выражаться как разница между эмпирическим риском полученной модели $\hat{u}^*$ и наименьшим эмпирическим риском из всех возможных:

$$\varepsilon_O = \hat{\mathcal{R}}[\mathcal{N}^*] - \inf_{\theta} \hat{\mathcal{R}}[\mathcal{N}]$$

Равенство $\varepsilon_O = 0$ будет означать, что мы нашли глобальный минимум.

Ошибка аппроксимации. Введём *ошибку аппроксимации*, описывающую, насколько хорошо мы можем аппроксимировать искомую функцию, используя данную архитектуру. Математически она определяется как наименьший возможный общий риск для данной архитектуры:

$$\varepsilon_A = \inf_{\theta} \mathcal{R}[\mathcal{N}]$$

Способность нейросетей аппроксимировать различные классы функций широко изучена в литературе и именно в данном контексте играют роль теоремы универсальной аппроксимации, которые, в сущности, утверждают, что ϵ_A можно сделать сколь угодно

малой при правильном выборе архитектуры модели и количества параметров. Редко, однако, говорится о том, как именно выбирать архитектуру и размер модели - эта область ещё мало изучена, хоть в ней и есть свои результаты.

Ошибка обобщения. Она характеризует способность нейросети делать верные предсказания на тех данных, которые не встречались в тренировочном наборе. Она выразится как максимальное отклонение общего риска от эмпирического риска при данной архитектуре и данных тренировочных точках:

$$\varepsilon_G = \sup_\theta |\mathcal{R}[\mathcal{N}] - \hat{\mathcal{R}}[\mathcal{N}]|$$

Заключение

Список источников

[1] Shengze Cai и др. *Physics-informed neural networks (PINNs) for fluid mechanics: A review*. 2021. arXiv: 2105.09506 [physics.flu-dyn].

[2] Hugh Bishop Christopher M. Bishop. *Deep Learning: Foundations and Concepts*. Springer, 2024.

[3] Salvatore Cuomo и др. «Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next». В: *Journal of Scientific Computing* 92.3 (июль 2022), с. 88. ISSN: 1573-7691. DOI: 10.1007/s10915-022-01939-z. URL: https://doi.org/10.1007/s10915-022-01939-z.

[4] Jan E. Gerken и др. «Geometric deep learning and equivariant neural networks». В: *Artificial Intelligence Review* 56.12 (дек. 2023), с. 14605—14662. ISSN: 1573-7462. DOI: 10.1007/s10462-023-10502-7. URL: https://doi.org/10.1007/s10462-023-10502-7.

[5] Zhongkai Hao и др. «Physics-informed machine learning: A survey on problems, methods and applications». В: *arXiv preprint arXiv:2211.08064* (2022).

[6] Zhongkai Hao и др. «PINNacle: A Comprehensive Benchmark of Physics-Informed Neural Networks for Solving PDEs». В: *arXiv preprint arXiv:2306.08827* (2023).

[7] Ruimeng Hu и др. *Higher-Order error estimates for physics-informed neural networks approximating the primitive equations*. 2023. arXiv: 2209.11929 [math.NA].

[8] Ameya D Jagtap и George Em Karniadakis. «Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations». В: *Communications in Computational Physics* 28.5 (2020), с. 2002—2041.

[9] Ameya D Jagtap, Ehsan Kharazmi и George Em Karniadakis. «Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems». В: *Computer Methods in Applied Mechanics and Engineering* 365 (2020), с. 113028.

[10] Anastasis Kratsios. «The Universal Approximation Property». В: *Annals of Mathematics and Artificial Intelligence* 89.5 (июнь 2021), с. 435—469. ISSN: 1573-7470. DOI: 10.1007/s10472-020-09723-1. URL: https://doi.org/10.1007/s10472-020-09723-1.

[11] Hao Li и др. *Visualizing the Loss Landscape of Neural Nets*. 2018. arXiv: 1712.09913 [cs.LG].

[12] Jie-Ying Li и др. «Utilizing symmetry-enhanced physics-informed neural network to obtain the solution beyond sampling domain for partial differential equations». В: *Nonlinear Dynamics* 111.23 (дек. 2023), с. 21861—21876. ISSN: 1573-269X. DOI: 10.1007/s11071-023-08975-w. URL: https://doi.org/10.1007/s11071-023-08975-w.

[13] Remco van der Meer, Cornelis Oosterlee и Anastasia Borovykh. *Optimally weighted loss functions for solving PDEs with Neural Networks*. 2021. arXiv: 2002.06269 [math.NA].

[14] Siddhartha Mishra и Roberto Molinaro. «Estimates on the generalization error of physics-informed neural networks for approximating PDEs». В: *IMA Journal of Numerical Analysis* 43.1 (янв. 2022), с. 1—43. ISSN: 0272-4979. DOI: 10.1093/imanum/drab093. eprint: https://academic.oup.com/imajna/article-pdf/43/1/1/49059512/drab093.pdf. URL: https://doi.org/10.1093/imanum/drab093.

[15] Maithra Raghu и Eric Schmidt. *A Survey of Deep Learning for Scientific Discovery*. 2020. arXiv: 2003.11755 [cs.LG].

[16] M. Raissi, P. Perdikaris и G.E. Karniadakis. «Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations». В: *Journal of Computational Physics* 378 (2019), с. 686—707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[17] Franz M. Rohrhofer, Stefan Posch и Bernhard C. Geiger. *On the Pareto Front of Physics-Informed Neural Networks*. 2021. arXiv: 2105.00862 [cs.LG].

[18] Tim De Ryck, Ameya D. Jagtap и Siddhartha Mishra. *Error estimates for physics informed neural networks approximating the Navier-Stokes equations*. 2023. arXiv: 2203.09346 [math.NA].

[19] Khemraj Shukla, Ameya D Jagtap и George Em Karniadakis. «Parallel Physics-Informed Neural Networks via Domain Decomposition». В: *Journal of Computational Physics* 447 (2021), с. 110683.