УДК 004.85:539.121.667

МАШИННОЕ ОБУЧЕНИЕ В ФИЗИЧЕСКИХ ИССЛЕДОВАНИЯХ

Устимчук Р.В., Шубаба М.И. студенты

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Храмович Е.М. – канд. физ.-мат. наук, доцент

Аннотация. В работе рассмотрено применение машинного обучения в физике высоких энергий для преодоления «вычислительного кризиса» на Большом адронном коллайдере. Была описана эволюция методов идентификации струй и использование генеративных моделей (GAN, диффузионные сети) для ускорения симуляций в эксперименте CMS. Особое внимание уделено физически информированным нейросетям (PINN), интегрирующим законы физики в обучение, с примером реализации на PyTorch. Также проанализированы ограничения машинного обучения в физике, включая проблемы обобщения и интерпретируемости.

Ключевые слова. Машинное обучение, физика высоких энергий, эксперимент CMS, идентификация струй, графовые нейронные сети, генеративно-состязательные сети, физически информированные нейронные сети (PINN), автоматическое дифференцирование, триггерная система, Большой адронный коллайдер.

Современная физика столкнулась с огромным потоком данных. Примером может послужить, детектор CMS на Большом адронном коллайдере регистрирует до 40 миллионов столкновений протонов в секунду. Объёмы информации достигают петабайт, и классические методы анализа уже не справляются [1, 2]. Мы решили разобраться, как машинное обучение помогает решать эту проблему. Оно способно быстро фильтровать данные прямо во время эксперимента, а также моделировать сложные физические процессы. В нашей работе мы рассмотрели применение машинного обучения от фильтрации данных до моделирования с помощью физически информированных нейросетей. Особый акцент мы сделали на подготовке к повышению светимости коллайдера (так называемая эра высокой светимости), когда поток данных возрастёт ещё в несколько раз.

Вычислительный кризис и роль машинного обучения в эксперименте CMS. Главная задача физиков высоких энергий заключается в отделении редких сигналов новых частиц от огромного фона. Сложность в том, что мы не измеряем интересующие величины напрямую, а только то, что можем зарегистрировать [2]. Нам приходится решать обратные задачи - связывать измерения с теорией.

В эксперименте CMS для этого используется триггерная система - фильтр реального времени, который за микросекунды решает, сохранять событие или нет. Раньше триггеры работали по простым правилам: она работала на двухуровневой схеме: Первичный уровень (L1) на электронике принимал решение за микросекунды, а последующий - высокоуровневый триггер (HLT) - использовал компьютерные фермы [3]. Однако с ростом интенсивности столкновений старые методы перестали успевать. При мегагерцовой частоте события накладываются друг на друга, и классический фильтр Калмана (метод восстановления треков частиц) не справляется с такими объёмами [4].

Выход нашёлся в глубоких нейросетях, так как они хорошо распознают образы, легко распараллеливаются и масштабируются. Именно это нужно для работы в реальном времени. Сегодня нейросетевые алгоритмы постепенно заменяют классические методы и на этапе триггера, и при восстановлении треков.

Эволюция методов идентификации струй. После столкновения частиц рождаются струи - направленные потоки адронов, которые образуются при превращении кварков и глюонов в частицы. Физикам важно знать, какой именно партон (кварк или глюон) породил струю. Например, нужно отличать струи от b-кварков от струй от лёгких кварков - это необходимо для поиска распада бозона Хиггса на b-кварки [5].

За последние годы методы идентификации струй сильно изменились. Долгое время использовались градиентный бустинг (каждое следующее дерево обучается решать ошибки предыдущих) и случайный лес (использование ансамбля решающих деревьев), которые давали хорошую точность и были понятны человеку. Затем на смену этому пришли свёрточные нейронные сети (CNN). Струю стали представлять как изображение в координатах детектора, где яркость пикселя соответствует энергии частицы. Свёрточные сети научились выделять пространственные закономерности ливня.

Самый современный подход - графовые нейронные сети (GNN). В них частицы внутри струи становятся узлами графа, а связи между ними - рёбрами. Так учитывается сложная структура распада, и не нужно привязываться к фиксированной сетке, как в свёрточных сетях [6]. В работе Саркара показано, что графы и трансформеры достигли рекордной точности идентификации струй в эксперименте CMS [7]. Сравнение методов мы привели в таблице 1.

Отдельно нами была рассмотрена разработка WOMBAT - система триггера на машинном обучении для поиска распадов бозона Хиггса на b-кварки [3]. В этой системе используются две нейросети:

Таблица 1 - Сравнение методов идентификации струй

Метод	Представление данных	Ключевая особенность
Бустируемые деревья	Вектор признаков	Интерпретируемость
Сверточные сети (CNN)	Изображение (η-φ пространство)	Выделение пространственных паттернов
Графовые сети (GNN)	Граф частиц	Учет топологии, отсутствие фикс. сетки
Трансформеры	Последовательность частиц	Механизм внимания, учет контекста

основная и облегчённая версия, которую поместили на FPGA (чип, который можно перестраивать под конкретную задачу). Тесты показали, что новый триггер срабатывает более чем в два раза быстрее старого и может отбирать события с меньшей энергией. Это позволяет не пропускать слабые сигналы, которые раньше терялись.

Ускорение симуляций с помощью генеративных моделей. Симуляции - узкое место любого анализа. Полный просчёт прохождения частиц через детектор в программах типа Geant4 занимает минуты на одно событие. Если нужно набрать статистику для редкого распада, такие расчёты становятся невозможными. Решение нашли в генеративных моделях.

Одними из первых применили генеративно-состязательные сети (GAN). В них одна сеть (генератор) создаёт фальшивый отклик детектора из случайного шума, а другая (дискриминатор) пытается отличить подделку от реальных данных. В результате генератор учится выдавать распределения, неотличимые от настоящих. Симуляция ливней в калориметрах ускоряется на порядки, а ключевые статистические характеристики сохраняются [6]. Например, модель CaloShowerGAN для эксперимента ATLAS легко встраивается в быструю симуляцию и работает лучше предыдущих версий [8].

Более новый подход - диффузионные модели. В работе Петрова показано, что они дают качество на 1,6 % выше, чем GAN [9]. Проблема диффузионных моделей - много шагов генерации, но их научились сокращать, переходя к предсказуемому режиму. Это ускоряет расчёты в десять раз с минимальными потерями качества.

Перевзвешивание событий и доменная адаптация. Когда светимость коллайдера вырастет, хранить отдельные наборы смоделированных событий для каждого варианта расчётов станет невозможно. Мы выяснили, что помогает перевзвешивание с помощью машинного обучения: вместо нового прогона симуляции можно взять готовые события и пересчитать их веса под новые условия [6].

Ещё один метод - доменно-состязательное обучение. Его применили для улучшения поиска мюонов. Классические способы оценки фона опираются на независимость переменных, но если переменные связаны (корреляция), расчёты становятся неточными. Нейросеть, которую обучили отличать нужные мюоны от фоновых, научили также игнорировать информацию о степени изоляции частицы. В итоге связь между решающим признаком и изоляцией уменьшилась, а идентификация мюонов стала точнее [6].

Физически информированное машинное обучение (PINN). Обычные нейросети часто называют “чёрными ящиками”: они могут предсказать что угодно, даже если это противоречит законам физики. Обучаясь на данных, такие сети хорошо работают внутри знакомого диапазона, но за его пределами начинают ошибаться, и объяснить их решения трудно [10].

Физически информированные нейросети (PINN) решают эту проблему иначе: они учатся не только на данных, но и на уравнениях, описывающих процесс [11]. Например, если есть дифференциальное уравнение, сеть принимает на вход координаты и время и пытается восстановить решение. Благодаря автоматическому дифференцированию она может вычислять производные и проверять, выполняется ли уравнение в каждой точке (рисунки 1) [12]. Ошибка считается дважды: если предсказание расходится с известными данными (начальными условиями) и если нарушается само уравнение. Чем меньше вторая ошибка, тем лучше сеть следует физике [13].

В качестве примера рассмотрим затухающие гармонические колебания. Их движение описывается формулой 1:

$$m\frac{d^2u}{dt^2} + \mu\frac{du}{dt} + k\,u = 0, \quad (1)$$

где $\mathrm{m}\frac{\mathrm{d}^2\mathrm{u}}{\mathrm{dt}^2}$ – сила инерции, $\mu\frac{\mathrm{du}}{\mathrm{dt}}$ – сила вязкого трения, $k\,u$ – сила упругости.

```python
import torch
import torch.nn as nn
import numpy as np

# 1. Определение архитектуры нейросети
class FCN(nn.Module):
    def __init__(self, n_input, n_output, n_hidden, n_layers):
        super().__init__()
        layers = [nn.Linear(n_input, n_hidden), nn.Tanh()]
        for _ in range(n_layers - 1):
            layers.extend([nn.Linear(n_hidden, n_hidden), nn.Tanh()])
        layers.append(nn.Linear(n_hidden, n_output))
        self.model = nn.Sequential(*layers)

    def forward(self, x):
        return self.model(x)

# 2. Параметры системы
d, w0 = 2, 20  # коэффициент затухания и частота
mu, k = 2*d, w0**2

# 3. Подготовка данных
t_physics = torch.linspace(0, 1, 500).view(-1, 1).requires_grad_(True)
t_boundary = torch.tensor([[0.]], requires_grad=True)

# 4. Обучение
pinn = FCN(1, 1, 32, 3)
optimizer = torch.optim.Adam(pinn.parameters(), lr=1e-3)
for i in range(10000):
    optimizer.zero_grad()

    # Loss на начальных условиях
    u_b = pinn(t_boundary)
    loss_boundary = torch.mean((u_b - 1)**2)

    # Физический loss (невязка уравнения)
    u = pinn(t_physics)
    u_t = torch.autograd.grad(u, t_physics, torch.ones_like(u), create_graph=True)[0]
    u_tt = torch.autograd.grad(u_t, t_physics, torch.ones_like(u_t), create_graph=True)[0]

    physics_residual = u_tt + mu*u_t + k*u
    loss_physics = torch.mean(physics_residual**2)

    # Общий loss
    loss = loss_boundary + loss_physics
    loss.backward()
    optimizer.step()

    if i % 1000 == 0:
        print(f"Итерация {i}, Loss: {loss.item():.6f}")
```

Рисунок 1 – Реализация PINN на PyTorch

Ключевые моменты: requires_grad_(True) разрешает вычисление производных по времени, torch.autograd.grad вычисляет производные внутри графа, physics_residual - мера соблюдения физики. Обучение идёт без подачи правильных ответов, только на основе уравнения (рисунки 2а, 2б).

Для более сложных задач подход расширяют [14]. Например, для уравнения теплопроводности (формула 2) нужно учитывать производные по всем координатам:

$$\frac{\partial u}{\partial t} = \alpha\left(\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial y \partial z}\right), \tag{2}$$

где $\frac{\partial u}{\partial t}$ – скорость изменения величины u во времени, $\alpha\left(\frac{\partial u}{\partial x}\right)$ – конвективный перенос вдоль оси x с коэффициентом α, $\frac{\partial^2 u}{\partial y \partial z}$ – производная второго порядка, которая может описывать, например, диффузионные процессы в анизотропной среде(как в механике сплошных сред).

```
import torch

def heat_equation_residual(model, x, y, t, alpha):
    """
    Вычисляет невязку уравнения теплопроводности u_t = alpha * (u_xx + u_yy)
    для заданных точек (x, y, t).

    Аргументы:
        model: нейросеть, принимающая тензор [x, y, t] и возвращающая u.
        x, y, t: тензоры формы (N, 1) с координатами точек.
        alpha: коэффициент температуропроводности (скаляр или тензор).

    Возвращает:
        Тензор невязки формы (N, 1) для каждой точки.
    """
    # Объединяем координаты в один входной тензор
    inputs = torch.cat([x, y, t], dim=1)
    # Убедимся, что для входов включено вычисление градиентов
    inputs.requires_grad_(True)

    u = model(inputs)

    # Первые производные
    u_x = torch.autograd.grad(
        u, x,
        grad_outputs=torch.ones_like(u),
        create_graph=True,
        retain_graph=True
    )[0]
    u_y = torch.autograd.grad(
        u, y,
        grad_outputs=torch.ones_like(u),
        create_graph=True,
        retain_graph=True
    )[0]
    u_t = torch.autograd.grad(
        u, t,
        grad_outputs=torch.ones_like(u),
        create_graph=True,
        retain_graph=True
    )[0]
    # Вторые производные
    u_xx = torch.autograd.grad(
        u_x, x,
        grad_outputs=torch.ones_like(u_x),
        create_graph=True,
        retain_graph=True
    )[0]
    u_yy = torch.autograd.grad(
        u_y, y,
        grad_outputs=torch.ones_like(u_y),
        create_graph=True,
        retain_graph=True
    )[0]
    # Невязка дифференциального уравнения
    residual = u_t - alpha * (u_xx + u_yy)
    return residual
def compute_physics_loss(model, x, y, t, alpha):
    """
    Вычисляет MSE-потерь для уравнения теплопроводности.
    Аргументы:
        model: нейросеть.
        x, y, t: тензоры координат.
        alpha: коэффициент температуропроводности.
    Возвращает:
        Скалярный тензор – среднеквадратичная невязка.
    """
    residual = heat_equation_residual(model, x, y, t, alpha)
    loss = torch.mean(residual ** 2)
    return loss
```

Рисунок 2а – Реализация PINN на PyTorch (взвешивание ошибок с коэффициентами)

Разные части ошибки могут конфликтовать, поэтому их взвешивают с коэффициентами (формула 3):

$$\text{Loss} = \lambda_{\text{pde}}\text{L}_{\text{pde}} + \lambda_{\text{bc}}\text{L}_{\text{bc}} + \lambda_{\text{ic}}\text{L}_{\text{ic}} + \lambda_{\text{data}}\text{L}_{\text{data}} \quad , \tag{3}$$

где L_{pde} – штраф за то, что предсказания сети нарушают физический закон, L_{bc} – штраф за нарушение условий на границах области, L_{ic} – штраф за нарушение условий в начальный момент времени, L_{data} – обычная ошибка, если есть экспериментальные данные, λ - весовые коэффициенты, которые могут адаптивно меняться в процессе обучения [11].

```
import torch
import torch.nn as nn
import numpy as np

# ------------------------------------------------------------
# 1. Определим простую полносвязную сеть для PINN
# ------------------------------------------------------------
class PINN(nn.Module):
    def init(self, layers):
        super(PINN, self).init()
        self.activation = nn.Tanh()
        self.linears = nn.ModuleList()
        for i in range(len(layers)-1):
            self.linears.append(nn.Linear(layers[i], layers[i+1]))
            # Инициализация Xavier для лучшей сходимости
            nn.init.xavier_normal_(self.linears[-1].weight)
            nn.init.zeros_(self.linears[-1].bias)

    def forward(self, x):
        for i in range(len(self.linears)-1):
            x = self.activation(self.linears[i](x))
        x = self.linears[-1](x)  # без активации на выходе
        return x

# ------------------------------------------------------------
# 2. Функция потерь с весовыми коэффициентами
# ------------------------------------------------------------
def compute_loss(model, x_pde, t_pde, x_bc, t_bc, u_bc, x_ic, t_ic, u_ic,
                 lambda_pde=1.0, lambda_bc=1.0, lambda_ic=1.0, lambda_data=0.0,
                 x_data=None, t_data=None, u_data=None):
    """
    model: нейросеть, принимающая [x, t] и возвращающая u
    x_pde, t_pde: точки внутри области для PDE (коллокационные точки)
    x_bc, t_bc, u_bc: точки на границе с известными значениями
    x_ic, t_ic, u_ic: точки в начальный момент времени
    x_data, t_data, u_data: экспериментальные данные (если есть)
    lambda_*: весовые коэффициенты для каждого слагаемого
    """
    # ---- PDE loss (невязка уравнения) ----
    # Требуем градиенты для x_pde, t_pde
    x_pde.requires_grad_(True)
    t_pde.requires_grad_(True)
    u = model(torch.cat([x_pde, t_pde], dim=1))

    # Производные по времени и пространству
    u_t = torch.autograd.grad(u, t_pde, grad_outputs=torch.ones_like(u),
                              create_graph=True, retain_graph=True)[0]
    u_x = torch.autograd.grad(u, x_pde, grad_outputs=torch.ones_like(u),
                              create_graph=True, retain_graph=True)[0]
    u_xx = torch.autograd.grad(u_x, x_pde, grad_outputs=torch.ones_like(u_x),
                               create_graph=True, retain_graph=True)[0]

    # Уравнение теплопроводности: u_t = alpha * u_xx
    alpha = 0.01  # коэффициент температуропроводности
    pde_residual = u_t - alpha * u_xx
    loss_pde = torch.mean(pde_residual**2)

    # ---- Boundary loss (граничные условия) ----
    u_bc_pred = model(torch.cat([x_bc, t_bc], dim=1))
    loss_bc = torch.mean((u_bc_pred - u_bc)**2)

    # ---- Initial loss (начальные условия) ----
    u_ic_pred = model(torch.cat([x_ic, t_ic], dim=1))
    loss_ic = torch.mean((u_ic_pred - u_ic)**2)

    # ---- Data loss (подгонка под экспериментальные данные) ----
    loss_data = 0.0
    if x_data is not None:
        u_data_pred = model(torch.cat([x_data, t_data], dim=1))
        loss_data = torch.mean((u_data_pred - u_data)**2)

    # Общий loss
    total_loss = (lambda_pde * loss_pde +
                  lambda_bc  * loss_bc +
                  lambda_ic  * loss_ic +
                  lambda_data * loss_data)

    return total_loss, loss_pde, loss_bc, loss_ic, loss_data

# ------------------------------------------------------------
# 3. Генерация обучающих точек (пример для 1D)
# ------------------------------------------------------------
def generate_training_data(N_pde=1000, N_bc=100, N_ic=100, x_range=[0,1], t_range=[0,1]):
    # Внутренние точки для PDE (Latin Hypercube Sampling – здесь просто случайные)
    x_pde = torch.rand(N_pde, 1) * (x_range[1] - x_range[0]) + x_range[0]
    t_pde = torch.rand(N_pde, 1) * (t_range[1] - t_range[0]) + t_range[0]

    # Граничные условия (например, u=0 на обеих границах)
    x_bc_left = torch.ones(N_bc//2, 1) * x_range[0]
    x_bc_right = torch.ones(N_bc - N_bc//2, 1) * x_range[1]
    x_bc = torch.cat([x_bc_left, x_bc_right], dim=0)
    t_bc = torch.rand(N_bc, 1) * (t_range[1] - t_range[0]) + t_range[0]
    u_bc = torch.zeros(N_bc, 1)  # нулевые граничные условия

    # Начальные условия (t=0) – например, u = sin(pi*x)
    x_ic = torch.rand(N_ic, 1) * (x_range[1] - x_range[0]) + x_range[0]
    t_ic = torch.zeros(N_ic, 1)
    u_ic = torch.sin(np.pi * x_ic)

    return x_pde, t_pde, x_bc, t_bc, u_bc, x_ic, t_ic, u_ic

# ------------------------------------------------------------
# 4. Пример обучения
# ------------------------------------------------------------
if name == "main":
    # Параметры
    layers = [2, 50, 50, 50, 1]  # вход: [x, t], выход: u
    model = PINN(layers)
    optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)

    # Генерация данных
    x_pde, t_pde, x_bc, t_bc, u_bc, x_ic, t_ic, u_ic = generate_training_data()

    # Весовые коэффициенты (можно менять в процессе обучения)
    lambda_pde = 1.0
    lambda_bc  = 10.0  # увеличиваем вес граничных условий для жёсткого соблюдения
    lambda_ic  = 10.0
    lambda_data = 0.0  # данных нет

    # Обучение
    for epoch in range(5000):
        optimizer.zero_grad()
        loss, loss_pde, loss_bc, loss_ic, _ = compute_loss(
            model, x_pde, t_pde, x_bc, t_bc, u_bc, x_ic, t_ic, u_ic,
            lambda_pde, lambda_bc, lambda_ic, lambda_data
        )
        loss.backward()
        optimizer.step()

        if epoch % 500 == 0:
            print(f"Epoch {epoch:5d}, Total Loss: {loss.item():.2e}, "
                  f"PDE: {loss_pde.item():.2e}, BC: {loss_bc.item():.2e}, IC: {loss_ic.item():.2e}")

    print("Обучение завершено.")
```

Рисунок 2б – Реализация PINN на PyTorch (взвешивание ошибок с коэффициентами)

Для очень сложных систем используют ResNet-блоки, чтобы градиенты не затухали, и сеть Siren с синусоидальной активацией - она лучше передаёт высокие частоты [14].

Одно из применений PINN - моделирование магнитного поля в детекторе. Вместо огромных таблиц с замерами обучают сеть со встроенными уравнениями Максвелла. Она становится компактным симулятором поля и автоматически следит за выполнением физических законов (например, div B = 0) [14]. Недавние разработки DAL-PINN, основанные на принципе Даламбера, объединяют уравнения для разных режимов и повышают точность [15].

Ещё одна задача - описание квантовых систем многих частиц. Обычными методами невозможно рассчитать волновую функцию даже для нескольких десятков частиц - информации слишком много. Нейросети предлагают компактное представление - нейронные квантовые состояния [6]. Сеть учится представлять волновую функцию, минимизируя энергию системы. Метод хорошо работает там, где стандартный квантовый Монте-Карло пасует (знаковая проблема). Когда в сеть встроили законы сохранения, расчёты молекул ускорились в десять раз с сохранением химической точности [6].

Другая задача - слабое гравитационное линзирование. Тёмная материя искажает формы далёких галактик, и свёрточные сети помогают выделить эти крошечные искажения из шумов [6]. Недавно китайские физики создали сеть, которая не просто предсказывает, а выдаёт символьные формулы для параметров гало тёмной материи. Она указала конкретный диапазон сечения взаимодействия, который проверят телескопы Euclid и JWST [6].

В эксперименте TAIGA нейросети отсеивают фон адронов (превышение фона в 10 000 раз) и за 21 час наблюдений Крабовидной туманности достигли уровня сигнала выше 5σ [16]. Полносвязные сети определяют направление космических ливней с ошибкой около 0,3 градуса [17].

Тем не менее, нейросети часто остаются “чёрными ящиками” - непонятно, на какие признаки они опираются [10, 18]. Кроме того,, они плохо обобщают результаты за пределы данных, на которых учились, а это критично для поиска новой физики. Также обучение некоторых моделей (особенно GAN и нейроквантовых состояний) бывает нестабильным. И наконец, сами нейросети требуют огромных вычислительных ресурсов, и иногда выигрыш в скорости съедается затратами на обучение. В обзоре по PINN дополнительно выделяют проблемы многоцелевой оптимизации и обработки разрывов [10].

Таким образом, в ходе нашей работы было проанализировано применение машинного обучения в физике высоких энергий. Выяснилось, что современные эксперименты, такие как CMS, уже не могут обходиться без нейросетей - они фильтруют данные в реальном времени и помогают восстанавливать события. Была прослежена эволюция методов идентификации струй от бустируемых деревьев до графовых нейронных сетей и обнаружено, что точность распознавания вышла на новый уровень. Генеративные модели (GAN и диффузионные) позволяют на порядки ускорить симуляцию ливней, сохраняя при этом физику.

В работе разобраны физически информированные нейросети (PINN) на примере кода для колебаний и показали, как их масштабируют для решения уравнений Максвелла и Навье - Стокса. Также было рассмотрено применение машинного обучения в квантовой физике и астрофизике. Нейросети остаются трудно интерпретируемыми, они плохо обобщают за пределы обучения и требуют больших ресурсов. Мы считаем, что гибридные подходы, сочетающие физические законы с гибкостью машинного обучения, - самое перспективное направление для дальнейших исследований.

Список использованных источников:

1. Доленко, С.А. Машинное обучение в физике: вводная лекция [Электронный ресурс] / С.А. Доленко. – М., 2023. – Режим доступа: http://phys.msu.ru/lectures/dolenko_ml_2023.pdf. – Дата доступа: 20.03.2026.
2. CMS and machine learning at the forefront of data management [Электронный ресурс] // CERN. – Режим доступа: https://cms.cern/news/cms-and-machine-learning-forefront-data-management. – Дата доступа: 01.02.2026.
3. CMS releases open data for machine learning [Электронный ресурс] // CERN. – Режим доступа: https://cms.cern/news/cms-releases-open-data-machine-learning. – Дата доступа: 04.02.2026.
4. Bileska, M. Design and FPGA Implementation of WOMBAT: A Deep Neural Network Level-1 Trigger System for Jet Substructure Identification and Boosted H→bb¯ Tagging at the CMS Experiment [Электронный ресурс] / M. Bileska // arXiv preprint. – 2025. – arXiv:2505.05532. – Режим доступа: https://arxiv.org/abs/2505.05532. – Дата доступа: 21.02.2026.
5. Ососков, Г.А. Методы машинного обучения для интеллектуального анализа и обработки экспериментальных данных физики высоких энергий [Электронный ресурс] / Г.А. Ососков // Семинар ЛИТ ОИЯИ. – 2025. – Режим доступа: http://lit.jinr.ru/seminars/ososkov_ml_2025.pdf. – Дата доступа: 19.02.2026.
6. Физически информированное машинное обучение [Электронный ресурс] // Habr / AIRI. – Режим доступа: https://habr.com/ru/companies/airi/articles/752480/. – Дата доступа: 17.02.2026.
7. Sarkar, U. Run 3 performance and advances in heavy-flavor jet tagging in CMS [Электронный ресурс] / U. Sarkar // arXiv preprint. – 2024. – arXiv:2412.05863. – Режим доступа: https://arxiv.org/abs/2412.05863. – Дата доступа: 20.03.2026.
8. Faucci Giannelli, M. CaloShowerGAN, a generative adversarial network model for fast calorimeter shower simulation [Электронный ресурс] / M. Faucci Giannelli, R. Zhang // European Physical Journal Plus. – 2024. – Vol. 139, № 7. – P. 597. – Режим доступа: https://doi.org/10.1140/epjp/s13360-024-05234-6. – Дата доступа: 27.02.2026.
9. Петров, С.А. Диффузионные модели для симуляции физических процессов [Электронный ресурс] / С.А. Петров // Магистерская диссертация, НИУ ВШЭ. – 2025. – Режим доступа: https://www.hse.ru/edu/vkr/503872346. – Дата доступа: 25.02.2026.
10. Guo, J. Advances in physics-informed neural networks for solving complex partial differential equations and their engineering applications: A systematic review [Электронный ресурс] / J. Guo, H. Wang // Engineering Applications of Artificial Intelligence. – 2025. – Vol. 144. – P. 110-135. – Режим доступа: https://doi.org/10.1016/j.engappai.2024.110135. – Дата доступа: 20.02.2026.
11. Raissi, M. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations [Электронный ресурс] / M. Raissi, P. Perdikaris, G.E. Karniadakis // Journal of Computational Physics. – 2019. – Vol. 378. – P. 686-707. – Режим доступа: https://doi.org/10.1016/j.jcp.2018.10.045. – Дата доступа: 18.02.2026.
12. Численное решение системы уравнений Навье-Стокса в случае сжимаемой среды с использованием нейронных сетей [Электронный ресурс] // Национальный агрегатор открытых репозиториев. – Режим доступа: http://www.openrepository.ru/article?id=1009885. – Дата доступа: 15.02.2026.
13. Moseley, B. Physics-informed neural networks (PINNs): an introductory crash-course [Электронный ресурс] / B. Moseley // GitHub repository. – 2022. – Режим доступа: https://github.com/benmoseley/PINN-intro. – Дата доступа: 17.02.2026.
14. Адаптация PINN для сложных задач [Электронный ресурс] / Материалы по масштабированию физически информированных нейронных сетей. – 2026. – Режим доступа: http://www.physics-ml.ru/papers/pinn_scaling_2026.pdf. – Дата доступа: 06.03.2026.
15. Li, X. DAL-PINNs: Physics-informed neural networks based on D'Alembert principle for generalized electromagnetic field model computation [Электронный ресурс] / X. Li, P. Wang, F. Yang // Engineering Analysis with Boundary Elements. – 2024. – Vol. 168. – P. 105-122. – Режим доступа: https://doi.org/10.1016/j.enganabound.2024.105122. – Дата доступа: 07.03.2026.
16. Гресь, Е.О. Gamma/hadron separation in the TAIGA experiment with neural network methods [Электронный ресурс] / Е.О. Гресь [и др.] // Сборник аннотаций конференции DLCP. – МГУ, 2024. – Режим доступа: http://conf.msu.ru/dlcp2024/abstracts/137.pdf. – Дата доступа: 04.03.2026.
17. Дубенская, Ю.Ю. Определение направления ШАЛ по данным TAIGA HiSCORE с помощью полносвязных нейросетей [Электронный ресурс] / Ю.Ю. Дубенская [и др.] // Сборник аннотаций конференции DLCP. – МГУ, 2024. – Режим доступа: http://conf.msu.ru/dlcp2024/abstracts/142.pdf. – Дата доступа: 02.03.2026.
18. Goodfellow, I. Deep Learning [Электронный ресурс] / I. Goodfellow, Y. Bengio, A. Courville. – MIT Press, 2016. – Режим доступа: https://www.deeplearningbook.org/. – Дата доступа: 05.03.2026.

UDC 004.85:539.121.667

MACHINE LEARNING IN PHYSICS RESEARCH

Ustimchuk R.V., Shubaba M.I., students

Belarusian State University of Informatics and Radioelectronics[1]
Minsk, Republic of Belarus

Khramovich E. M. - Candidate of Physical and Mathematical Sciences

Annotation. In this paper, we examined the application of machine learning in high-energy physics to overcome the computational bottleneck at the Large Hadron Collider. We described the evolution of jet identification methods and the use of generative models (GANs, diffusion networks) to accelerate simulations in the CMS experiment. We focused on physically informed neural networks (PINNs), which integrate the laws of physics into training, with an example implementation in PyTorch. We analyzed the limitations of machine learning in physics, including issues of generalization and interpretability.

Keywords. Machine learning, high-energy physics, CMS experiment, jet identification, graph neural networks, generative adversarial networks, physically informed neural networks (PINN), automatic differentiation, trigger system, Large Hadron Collider.