

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

**Московский институт электроники и математики им. А.Н.
Тихонова**

Кацуба Кирилл Олегович

**ФИЗИЧЕСКИ-ИНФОРМИРОВАННЫЕ
ТРАНСФОРМЕРЫ ДЛЯ ОЦЕНКИ ЗАГРЯЗНЕНИЯ
ВОЗДУХА**

Выпускная квалификационная работа
студента образовательной программы бакалавриата
«Прикладная математика»

по направлению 01.03.04 Прикладная математика

Научный руководитель
Доцент, к.ф.-м.н.
Д.А. Деркач

Соруководитель
Преподаватель, к.к.н.
А.С. Рыжиков

Москва 2025 г.

ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ
ИМ. А.Н. ТИХОНОВА

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

студенту группы БПМ213 Кцуба Кириллу Олеговичу

1. Тема работы

Физически-информированные трансформеры для оценки загрязнения воздуха.

2. Цель работы


Определение источников загрязнения воздуха с помощью нейронных сетей, решающих дифференциальные уравнения Навье-Стокса.

3. Формулировка задания

Построение модели, способной решать прямую задачу аппроксимации полей концентраций в определенный момент времени и обратную задачу, то есть определять источники загрязнений воздуха по полю концентраций выбросов в воздух. Необходимо решить задачу для данных по городу Новосибирск.


Проект ВКР представлен студентом до «15» февраля 2025г.

Руководитель ВКР

 Д.А. Деркач


Первый вариант ВКР представлен студентом в срок до 22 апреля 2025г.

Руководитель ВКР


 Д.А. Деркач

Итоговый вариант ВКР представлен студентом в срок до 30 апреля 2025г.

Руководитель ВКР

 Д.А. Деркач

Задание выдано 19 декабря 2024г
студенту

 Д.А. Деркач

Задание принято 19 декабря 2024г
к исполнению
студентом

 К.О. Кацуба

Аннотация

Физически-информированные нейронные сети (PINN) широко используются для численного решения дифференциальных уравнений в частных производных в самых разных областях физики, поскольку эти подходы менее затратны с точки зрения вычислительных ресурсов, памяти и данных. С быстрым развитием искусственного интеллекта в последние годы новые передовые методы также широко используются для построения прокси-моделей для различных физических областей и решения дифференциальных уравнений. Существует ряд алгоритмов, таких как нейронный оператор Фурье (FNO), CoDa-NO (GNO), DeepONet, PINN с динамической стратегией весов и Transolver. Целью этой работы является экспериментальная демонстрация того, как решать уравнение Навье-Стокса, описывающее гидродинамические системы, с помощью физически-информированных трансформеров, таких как Transolver. В этом исследовании делается попытка показать, как Transolver может обрабатывать конкретные примеры уравнений Навье-Стокса на реальных наборах данных. Результатом работы является применение существующей модели глубокого обучения к новому набору данных, которая ранее не тестировалась на этом примере уравнения Навье-Стокса.

Abstract

Physics-Informed neural networks (PINNs) are widely used for numerical solving of partial differential equation in huge variety of physics spheres as these approaches are less expensive in terms of computational costs. With the rapid development of artificial intelligence in recent years new advanced techniques are also popular in area of proxy models of physics domains and partial differential equation (PDE) solvers. There are a number of algorithms such as Fourier neural operator (FNO), CoDa-NO (GNO), DeepONet, PINNs with dynamic weights strategy and Transolver. This thesis aimed to experimentally demonstrate how to solve Navier-Stokes equations applied for fluid dynamics via Physic-Informed Transformers like Transolver. This study endeavors to show how Transolver can handle specific examples of Navier-Stokes equations on real-datasets. The result of the work is an application of an existing deep learning model to the new dataset, which was not tested on this example of Navier-Stokes equation before.

Содержание

Вступление	6
1. Обзор научной области и проблематики	7
2. Описание задачи	9
2.1. Постановка задачи	9
2.2. Данные	9
2.3. Идея Физически-Информированных нейронных сетей.	10
3. Методы решения	11
3.1. Физически-Информированные трансформеры	11
3.1.1. Архитектура	11
3.1.2. Функция потерь	13
3.2. Сверточные нейронные сети	16
3.3. Трансформеры для обработки изображений	17
4. Результаты	18
4.1. Решение прямой задачи	18
4.2. Решение обратной задачи	20
4.2.1. Классификация	20
4.2.2. Регрессия	22
Заключение	28
Список литературы	29
А. Приложения	32

Вступление

Трансформеры [1] в настоящее время широко используются в самых различных областях. Численные методы не являются исключением. Начиная с простой архитектуры физически информированных нейронных сетей (англ. Physics-Informed Neural Networks, PINNs) [2], где процесс обучения модели напрямую использует теоретическое решение дифференциальных уравнений в частных производных (ДУЧП), возникают проблемы с вычислительными затратами при работе с крупными и высоко нелинейными реальными наборами данных [3]. Это создает потребность в более продвинутых физически обоснованных архитектурах, которые стали более актуальными подходами для аппроксимации дифференциальных уравнений. Исследователи активно изучают применение новых методов глубокого обучения в вычислительной физике. Большинство работ [4, 5], посвященных построению аппроксимаций для уравнений Навье-Стокса, сосредоточены на задачах гидродинамики, однако существует недостаток примеров для других физических систем.

Цель данной работы — прогнозирование полей концентрации загрязняющих веществ и идентификация источников их распространения. Для достижения этой цели необходимо, во-первых, выбрать подходящую модель в качестве основного механизма решения обратных задач (построения аппроксимации). Кроме того, критически важным является подбор оптимальных параметров, так как каждый набор данных требует специфических характеристик для суррогатной модели, чтобы обеспечить корректное нахождение решения. В работе рассматривается современный алгоритмы глубокого обучения в физике [6], который может быть использованы для построения аппроксимаций.

В данной статье предлагается исследовать новые данные об уровне загрязнения воздуха в Новосибирске, собранные исследователями компании "CityAir" [7], которые подчиняются уравнению Навье-Стокса. Насколько известно автору, эти реальные данные имеют более сложную структуру по сравнению с примерами из предыдущих исследований. Кроме того, они могут не зависеть напрямую от времени, что усложняет прогнозирование.

Остальная часть статьи структурирована следующим образом. Раздел 1 содержит обзор связанных работ. Раздел 2 описывает данные используемые в задачи. Раздел 3 описывает реализацию функциона-

ла для решения задачи. Раздел 4 представляет результаты. Заключение подводит итоги работы.

1. Обзор научной области и проблематики

Методы машинного и глубокого обучения всё чаще используются исследователями для снижения вычислительных ресурсов и объёма данных, необходимых для построения аппроксимаций [8]. Одной из относительно новых и популярных моделей, разработанных для решения ДУЧП, является физически информированная нейронная сеть (Physics-Informed Neural Network, PINN) [2]. Эта модель призвана учитывать характеристики физической системы путём обучения граничным и начальным условиям, а также законам, которым система предположительно подчиняется. Однако данный подход требует тщательного подбора весов для каждой функции потерь [9, 10] и опирается на точные дифференциальные уравнения, тогда как реальные данные часто содержат значительный шум, что усложняет процесс обучения и снижает точность численной аппроксимации, например, для оценки загрязнения воздуха.

В последние годы нейронные операторы стали перспективной архитектурой для прогнозирования в физических задачах. Фурье-нейронный оператор (Fourier Neural Operator, FNO) [11] аппроксимирует решение уравнений Навье-Стокса, отображая входные данные в спектральную область. Он преобразует входы в коэффициенты Фурье, применяет обучаемые преобразования для улавливания глобальных взаимодействий, а затем использует обратное преобразование Фурье для возврата в физическое пространство. Этот метод эффективно предсказывает эволюцию потоков жидкости или воздуха при различных дискретизациях и разрешениях. Важно, что FNO не решает уравнения напрямую, что позволяет улучшить способность модели работать со сложными системами на реальных данных.

Современные достижения в области нейросетей привели к появлению алгоритмов на основе трансформеров, учитывающих взаимодействие между соседними элементами. Сфера вычислительной физики не стала исключением. Один из популярных подходов — Codomain Attention Neural Operator (CoDA-NO) [12], сочетающий FNO и архитектуру трансформеров для решения многопараметрических ДУЧП. CoDA-NO представляет входные данные как набор функций, где каждая соответствует отдельной физической переменной (например, ско-

рости, давлению). Эти функции токенизируются по их области значений (codomain), что позволяет модели рассматривать каждую переменную как отдельный токен. Каждый токен дополняется позиционным кодированием для учёта пространственной информации, а затем проецируется в пространство большей размерности. Ключевой компонент CoDA-NO — механизм внимания, работающий не в пространственной, а в области значений (codomain). Он вычисляет взаимодействия между физическими переменными (например, связь скорости и давления) через генерацию ключей, запросов и значений для каждого токена, применяя операцию self-attention [1]. Это позволяет модели фиксировать сложные зависимости между переменными, критичные для решения связанных ДУЧП. После обработки вниманием CoDA-NO интегрирует результаты с помощью интегрального оператора, что обеспечивает работу с нерегулярными сетками. Модель, будучи предобученной, демонстрирует высокую точность на симуляционных данных. Однако при прогнозировании долгосрочных зависимостей, таких как оценка загрязнения воздуха, могут возникать сложности.

Следующим шагом в применении трансформеров и нейронных операторов стало создание модели Transolver [6], основанной на механизме физического внимания (physics attention). Её ключевое отличие — способность эффективно обрабатывать реальные зашумлённые и неполные данные. Transolver включает механизм physics self-attention для улавливания сложных зависимостей (например, в уравнениях Навье-Стокса) и обладает высокой адаптивностью к изменениям граничных или начальных условий. Принцип работы physics attention:

1. Сегментация: Пространство разбивается на группы на основе физических состояний (например, области с похожими параметрами).
2. Агрегация: Токены, связанные с каждой группой, агрегируют информацию внутри своего сегмента.
3. Декодирование: Результаты декодируются обратно в узлы сетки.

Transolver демонстрирует превосходные результаты в задачах решения уравнений Навье-Стокса и других физических систем. В данной работе используются достижения численных методов, в частности реализация Transolver, для анализа нового набора данных о загрязнении воздуха в Новосибирске, подчиняющегося уравнениям Навье-Стокса

2. Описание задачи

2.1. Постановка задачи

По данным о концентрации загрязнения воздуха необходимо научиться моделировать поведение этой физической системы в любой точке из области определения (t, x, y) , в различных точках пространства. Научиться определять источник, от которого исходит шлейф загрязнений. В данной задаче рассматриваем дискретное множество источников (имеется 10 различных источников). В результате хотим получить модель, которая по полю концентраций в текущий момент будет способна воспроизвести поле концентраций в произвольный момент времени (цель определить поле в начальный момент времени) и затем находить от какого источника исходит загрязнение.

2.2. Данные

Для решения задачи рассматривается датасет источников загрязнений и выбросов, которые исходят от каждого источника. Данные имеют следующую структуру:

1. **Координата по времени.** Всего для каждого из выбросов рассматриваются 18 временных шагов, каждый такой шаг равен 20 минутам. Начальным временем считается точка равная 20 минутам со старта работы источника.
2. **Вещество.** В данной задаче рассматриваем только пыль.
3. **Источник.** Рассматривается дискретное множество, состоящее из 10 источников.
4. **Координаты сетки.** Данные рассматриваются в двумерном пространстве (x, y) .

Всего имеются 49 наборов данных, в каждом из которых для всех 10 источников находится поле концентраций загрязнений воздуха для 18 временных координат с шагом 20 минут, при этом первый срез по времени соответствует не начальному положению системы, а имеет лаг в 20 минут от начала эмиссии загрязнений. Структура

`xarray` [13], в которой хранятся данные, позволяет в удобном формате содержать информацию для источников загрязнения. Для каждого источника в датасете содержатся координаты в виде широты и долготы.

Таким образом из описанного набора данных будет составлена обучающая, тестовая и валидационная выборка. В качестве необходимых для обучения характеристик будут рассматриваться: координаты поля концентраций в точке (t, x, y) и сам выброс загрязнения в воздух, координаты источника, из которого происходит выброс.

Также стоит отметить, что данные рассматриваемые в задаче не содержат сильного шума или большого внешнего воздействия (например, влияние ветра на распространение шлейфа загрязнений). Все данные собраны за промежуток с 30 августа 2024 года по 17 сентября 2024 года по городу Новосибирск.

2.3. Идея Физически-Информированных нейронных сетей.

Физически-информированные нейронные сети - механизм позволяющий численно решать дифференциальные уравнения и аппроксимировать физические системы, путем обучения законам, описывающим эту систему. Для обучения модели необходимо обучить модель на дифференциальном уравнении, которое задает закон поведения системы. В рассматриваемой задаче необходимо решать уравнение Навье-Стокса (разновидность уравнения Навье-Стокса - уравнение Адвекции-диффузии) [14], которое принимает следующий вид:

$$\frac{\partial c}{\partial t} - u \nabla c - \nabla(k \nabla c) - s = 0, \quad (1)$$

Параметры u, k, s заданы заранее и описывают поведение системы. Здесь u - скорость ветра, k - коэффициент диффузии, s - исходный источник. Таким образом учитывая начальные и граничные условия получаются функции потерь:

$$\mathcal{L}_{pde} = \sum \left(\frac{\partial c_{nn}}{\partial t} - u \nabla c_{nn} - \nabla(k \nabla c_{nn}) - s_{nn} \right) \quad (2)$$

$$\mathcal{L}_{ic} = \sum \left(c_{pinn}(0, x_i, y_i) - c(0, x_i, y_i) \right) \quad (3)$$

$$\mathcal{L}_{data} = \sum \left(c_{pinn}(t_i, x_i, y_i) - c(t_i, x_i, y_i) \right) \quad (4)$$

Таким образом получаем функцию потерь для обучения нейронной сети:

$$\mathcal{L}_{res} = \frac{1}{N_{pde}} \mathcal{L}_{pde} + \frac{1}{N_{ic}} \mathcal{L}_{ic} + \frac{1}{N_{data}} \mathcal{L}_{data} \quad (5)$$

В этом состоит идея физически-информированных нейронных сетей. Передать информацию о дифференциальном уравнении и начальном состоянии системы в процесс обучения, чтобы научиться выделять необходимую информацию для нахождения результата.

3. Методы решения

В этой секции будут описаны методы машинного и глубинного обучения, которые используются для решения задачи. Здесь будет представлено описание подходов для решения прямой задачи - предсказание поля концентраций, и обратной задачи - поиск источников. Результаты работы описанных подходов будут представлены в следующем разделе. Реализация всех алгоритмов с использованием Python фреймворка pytorch [15] будет представлена в Приложении.

3.1. Физически-Информированные трансформеры

3.1.1. Архитектура

В данной работе для решения прямой задачи будет использована модель трансформер Transolver, которая предназначена для работы с различными физическими системами. Архитектура данного алгоритма отличается от классической модели PINN, так как она не учится решать дифференциальные уравнения на прямую, а использует для предсказания слои декодера, энкодера и физического внимания.

В этой работе будет использована архитектура Transolver предназначенная для решения уравнения Навье-Стокса, которое описывает распространение загрязнения в воздухе.

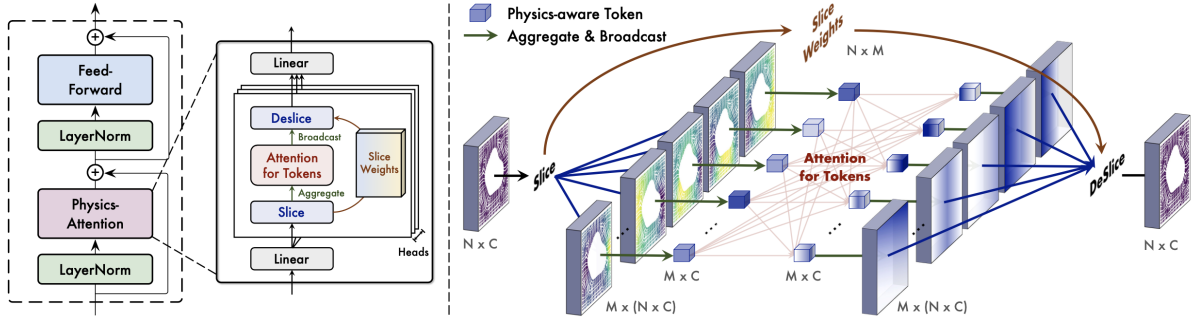


Рисунок 3.1. Архитектура Transsolver.

Архитектура основана на модифицированной трансформерной структуре, где классическое внимание между точками заменяется специализированным механизмом Physics-Attention. Последний обеспечивает агрегацию информации не по геометрической близости, а по сходству физических состояний, что принципиально важно для задач, где физические поля распределены сложным и неоднородным образом.

Входные данные. Передаются координаты сетки $g \in R^{N \times M}$, N - количество точек, M - размерность координатного пространства. Также будет передаваться информация о загрязнении воздуха u - поле концентраций пыли. Эти данные конкатенируются и с помощью линейного слоя получаем следующую проекцию $x_0 = W * [[g, u]] \in R^{N \times C}$.

Общая структура. Далее переходим к стандартной для трансформеров архитектуре. Основное отличие - физическое внимание (Physics-Attention). Оно выполняет внимание не между отдельными точками, а между агрегированными токенами, представляющими физически однородные области. Для каждого эмбединга x_i вычисляется вес принадлежности к M слайсам: $w_i = Softmax(Wx_i)$. Получаем вес принадлежности точки i к слайсу j . Каждый токен z_j для этого слайса равен:

$$z_j = \frac{\sum w_{i,j} \cdot x_i}{\sum w_{i,j}} \quad (6)$$

Таким образом получаются физические состояния для механизма внимания. Следующий шаг — формирование классического внимания между каждым сегментом физики. Цель — вычисление новых конкретных векторов запроса (Q), ключа (K), значения (V) с линейной зависимостью от токенов $Q = W_q Z$, $K = W_k Z$, $V = W_v Z$. Нормализуя,

получается следующая матрица физического внимания

$$A = \text{Softmax}\left(\frac{QK^T}{\sqrt{C}}\right) \quad (7)$$

, где C - нормализующий параметр. Финальным этапом является транслирование обновленных токенов на точки сетки $x'_i = \sum w_{i,j} z'_j$.

Выходной слой. После прохода через несколько слоев физического внимания необходимо получить вектор, размерность которого соответствует поставленной задаче, поэтому здесь просто применяем некоторый линейных оператор: $u = Wx' \in R^{N \times C_g}$

Для применения Transolver также очень важна настройка гиперпараметров модели, об этом более детально будет написано в следующей главе, секция эксперименты.

3.1.2. Функция потерь

Для решения задачи необходимо обучать модель решать две задачи: прямую и обратную, поэтому функция потерь для задачи должна учитывать как разницу между предсказанием полей концентраций загрязнений, так и разницу между оценкой координаты источника и фактическими координатами.

Предлагается использовать составную функцию потерь. Эта функция предназначена для оценки качества предсказаний модели сразу по двум различным целевым значениям: (\hat{y}_1, y_1) и (\hat{y}_2, y_2) , с возможностью взвешенного комбинирования двух соответствующих ошибок. Кроме того, поддерживается выбор между различными метриками: абсолютной, относительной и среднеквадратичной ошибкой.

Обозначения. Пусть:

1. $x_1, x_2 \in \mathbb{R}^{N \times D}$ — предсказания модели по двум задачам;
2. $y_1, y_2 \in \mathbb{R}^{N \times D}$ — соответствующие целевые значения;
3. N — размер батча;
4. D — размерность одного объекта;
5. $p \in \mathbb{R}_{>0}$ — порядок нормы;
6. $d \in \mathbb{N}$ — пространственная размерность (влияет на масштаб при абсолютной ошибке);

7. λ_1, λ_2 — веса для первой и второй компоненты ошибки соответственно.

Общая функция потерь определяется как взвешенная сумма двух компонент:

$$\mathcal{L}_{\text{total}} = \lambda_1 \cdot \mathcal{L}(x_1, y_1) + \lambda_2 \cdot \mathcal{L}(x_2, y_2),$$

где $\mathcal{L}(x, y)$ — выбранная метрика ошибки между предсказанием x и целевым значением y .

Рассмотрим какие функции потерь могут быть использованы при обучении модели для решения прямой задачи:

1. Относительная ошибка (relative error):

$$\mathcal{L}_{\text{rel}}(x, y) = \frac{1}{N} \sum_{i=1}^N \frac{\|x_i - y_i\|_p}{\|y_i\|_p + \varepsilon},$$

где $\varepsilon > 0$ — малое значение для численной устойчивости.

2. Абсолютная ошибка (absolute error), масштабированная по размерности:

$$\mathcal{L}_{\text{abs}}(x, y) = \frac{1}{N} \sum_{i=1}^N h^{\frac{d}{p}} \cdot \|x_i - y_i\|_p, \quad \text{где } h = \frac{1}{D-1}.$$

3. Средняя абсолютная ошибка (MAE):

$$\mathcal{L}_{\text{mae}}(x, y) = \frac{1}{N} \sum_{i=1}^N \frac{1}{D} \sum_{j=1}^D |x_{ij} - y_{ij}|.$$

Для решения обратной задачи могла быть использована другая функция, так например в [14] экспериментальным путем было показано, что стандартные функции потерь для задачи регрессии (MSE, MAE) могут работать хуже для более сложных с физической точки зрения задач, задач по поиску координат источников. Поэтому было предложено адаптировать функцию потерь под конкретную задачу, таким образом для предсказания координат источников загрязнений может быть использована следующая функция потерь.

Для задания пространного распределения, локализованного вокруг некоторой точки (x_0, y_0) . Её целью является формирование маски (или весовой функции), которая принимает наибольшее значение

в окрестности заданного источника и экспоненциально затухает по мере удаления от него.

Пусть x, y — координатные тензоры (или значения), представляющие текущие точки пространства, а (x_0, y_0) — координаты центра источника. Функция определена как:

$$\mathcal{L}(x, y, x_0, y_0) = \exp \left(-\frac{(x - x_0)^2 + (y - y_0)^2}{2 \cdot r^2} \right), \quad \text{где } r = \frac{1}{20}.$$

Это выражение представляет собой двухмерное изотропное (одинаковое по всем направлениям) гауссово распределение без нормирующего коэффициента:

$$f(x, y) = \exp \left(-\frac{\|(x, y) - (x_0, y_0)\|^2}{2r^2} \right),$$

где:

1. $\|(x, y) - (x_0, y_0)\|^2$ — квадрат евклидова расстояния до центра источника;
2. $r = \frac{1}{20}$ — задаёт ширину (радиус влияния) распределения.

Максимальное значение функции — 1 — достигается в точке (x_0, y_0) . При удалении от центра значение экспоненциально уменьшается. Таким образом, функция задаёт локализованную область вокруг источника для выделения зоны на сетке.

В данной работе использование этой функции потерь может быть не самым оптимальным способом обучения модели, поэтому мы остановимся на классическом среднеквадратичном отклонении. При этом важно понимать, что использование этой функции может быть очень полезно для иных задач по локализации объектов.

Редукция. После расчёта индивидуальных ошибок по каждому элементу батча применяется агрегирующая функция. Можно использовать несколько подходов для определения функции потерь.

Полученное для каждого элемента батча значение функции потерь суммируется и усредняется на размер батча. Затем оптимизируется с помощью обратного распространения ошибки. Можно использовать аналогичный подход но без усреднения чтобы добавить мкорости к сходимости.

3.2. Сверточные нейронные сети

Особое внимание в данной работе уделяется локализации источников загрязнений. Предполагается, что множество источников - дискретное множество, поэтому можно предсказывать не только координаты источника, решая задачу регрессии, но и класс источника, которому с наибольшей вероятностью соответствуют поля концентраций загрязнения.

Первой такой моделью будут сверточные нейронные сети [16]. Идея их применения заключается в том, что по сути поля концентраций в двумерном пространстве (координата времени фиксируется) являются изображениями, которые хорошо классифицируются с помощью сверточных сетей.

Сверточный слой. Идея сверточных нейронных сетей заключается в выборе нужных признаков находящихся на изображении и в их последующей обработке. С помощью фильтра свертки это (некоторый тензор W) происходит поэлементное умножение пикселей изображения на тензор свертки:

$$x_{output} = W \odot x_{input} \quad (8)$$

Здесь \odot - поэлементное умножение изображения размера $(channel, h, w)$ на тензор размера $(channel_output, channel_input, h, w)$.

Пулинг слой. После сверточных слоев необходимо агрегировать информацию. Для этого как правило используют два подхода:

- **MaxPooling.** Рассматривается некоторая непрерывная область полученного на выходе свертки изображения размера $(channel, h, w)$ и выбирается максимальное значение из данной области.
- **AvgPooling.** Все происходит аналогично MaxPooling, только вместо максимального значения из области рассматривается среднее значение по данной области.

Таким образом размер изображения уменьшается, но выделяются признаки, которые несут наибольшее количество информации для анализа.

Выход модели. Далее применяется полносвязный слой состоящий из различных функций активации и матрицы операций. И в результате после применения этого слоя получаем вектор состоящий из вероятностей принадлежности исходного поля концентраций к каждому из источников.

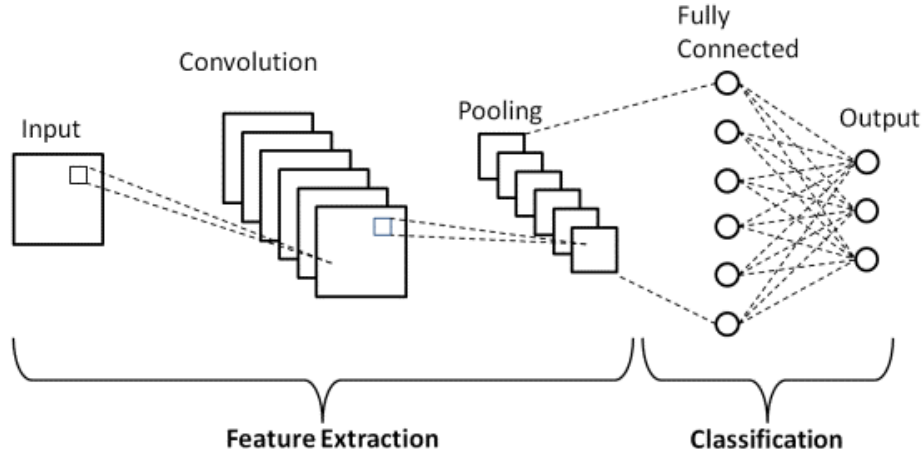


Рисунок 3.2. Архитектура CNN.

3.3. Трансформеры для обработки изображений

В последние годы архитектура трансформеров, изначально разработанная для обработки последовательностей в задачах обработки естественного языка (NLP), была успешно адаптирована для анализа изображений. Одним из ключевых направлений этого подхода стал визуальный трансформер (Vision Transformer, ViT).

В отличие от традиционных сверточных нейронных сетей (CNN), визуальные трансформеры полностью опираются на механизм внимания (self-attention), что позволяет им моделировать долгосрочные зависимости между различными частями изображения без использования свёрток. Основной принцип ViT [18] — представить изображение в виде последовательности патчей (локальных участков), аналогично токенам в тексте.

Пусть входное изображение $x \in \mathbb{R}^{H \times W \times C}$, где H и W — высота и ширина, C — число каналов. Оно разбивается на N непересекающихся патчей размером $P \times P$, каждый из которых векторизуется и проецируется в эмбединг-пространство фиксированной размерности D с помощью линейного слоя:

$$\text{Patch}_i \in \mathbb{R}^{P^2 \cdot C} \xrightarrow{\text{Linear}} \mathbf{z}_i \in \mathbb{R}^D.$$

Таким образом, исходное изображение преобразуется в последовательность $\{\mathbf{z}_1, \dots, \mathbf{z}_N\}$, где $N = \frac{HW}{P^2}$.

Дополнительно добавляется обучаемый *class token* $\mathbf{z}_0 \in \mathbb{R}^D$, аналог CLS-токена в BERT, который впоследствии будет использоваться для предсказания класса.

Так как архитектура трансформера не имеет встроенного механизма учёта порядка, каждому вектору \mathbf{z}_i добавляется позиционный эмбединг p_i , $x_i = \mathbf{z}_i + p_i$.

Полученная последовательность $\{x_0, x_1, \dots, x_N\}$ поступает на вход стандартному трансформеру, состоящему из L блоков encoder, каждый из которых включает:

1. многоголовочное самовнимание (Multi-Head Self-Attention, MHSA);
2. полносвязную нейросеть (MLP);
3. остаточные связи (residual connections) и слой нормализации (LayerNorm).

Для каждой пары токенов i и j вычисляется внимание (по аналогии с Transolver): $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$.

4. Результаты

4.1. Решение прямой задачи

Для начала было рассмотрено решение следующей задачи - по имеющимся данным нужно уметь предсказывать поле концентрации загрязнений в начальный момент времени.

Всего имеется 18 временных меток. Обучение модели будет происходить на 17 временных шагах, а предсказываться будет первый. Таким образом модель будет уметь определять поле концентраций в начальный момент времени.

Для этой задачи была выбрана модель Transolver. Она представляет собой трансформерную архитектуру, работающую с пространственными данными размерности 2. Вход представляет собой двумерную сетку размером 201×251 , которая разбивается на 64 фрагмента (slices). Каждый элемент пространства кодируется в скрытое представление размерности 64. Модель состоит из трёх трансформерных слоёв с многоголовым механизмом внимания, где количество голов равно четырём. Каждый слой включает в себя механизм внимания и многослойный персептрон (MLP) с шириной, равной размерности скрытого слоя (так как коэффициент ширины MLP установлен равным 1).

Область внимания ограничивается локальным окном размером 8×8 . Позиционная информация задаётся в унифицированной форме, то есть одинаковой для всех уровней модели.

Обучение происходило на 35 эпохах на процессоре GPU, количество батчей - 50, размер одного батча - 8. Время обучения заняло 32 минуты 42 секунды.

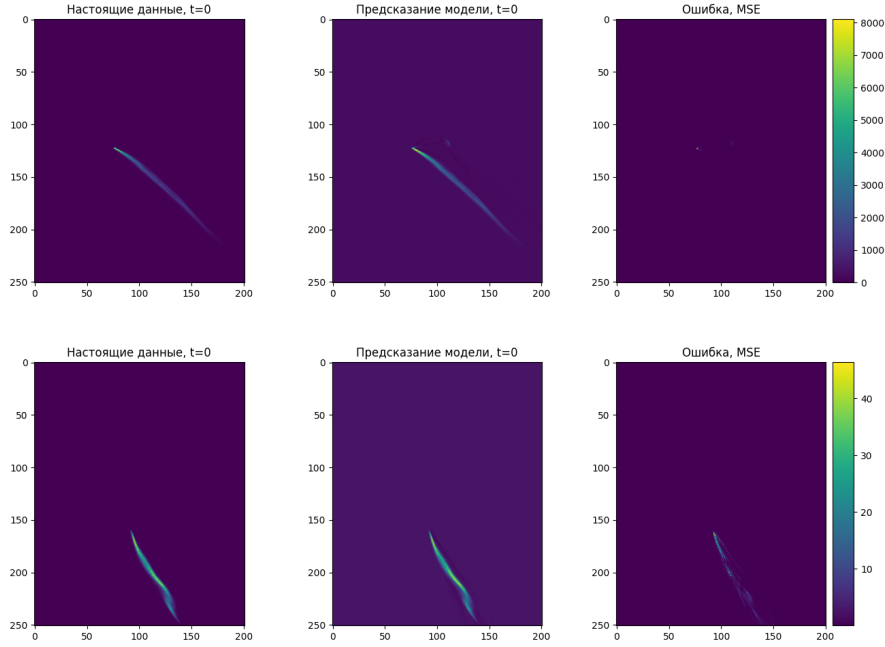


Рисунок 4.1. Результаты на выборке тестовых данных.

Изображения демонстрируют как модель может находить начальное положение системы. Transolver благодаря физическому вниманию может качественно выучивать зависимости между данными и тем самым искать начальное положение.

Шлейф концентрации очень мал и трансолвер хорошо выучивает поведение системы даже для сложных зависимостей, так например в худшем случае MSE мог достигать ошибки порядка 1×10^5 . В нашем случае среднеквадратичная ошибка не превышала четвертого порядка.

Данная модель хорошо выучивает характерное для системы поведение с помощью physics-attention и может как экстраполировать данные на будущее время, как показано в [6], так и искать предыдущий момент времени, зная будущее, как демонстрируется в этой работе.

Метрики качества данной модели на некотором подмножестве валидационной выборки следующие:

№	MAE	RMSE
1	1.05	2.24
2	1.95	4.70
3	1.67	3.45
4	0.80	1.51
5	1.54	3.49

В предсказаниях присутствует ошибка, но тем не менее данный трансформер успешно выполняет задачи по поиску поля концентраций распространения загрязнений в начальный момент времени.

4.2. Решение обратной задачи

Теперь переходим к поиску источников загрязнений. Здесь задача разбивается на два возможных пути решения: классификация и регрессия.

У обоих подходов будет похожая структура. Модель в обоих случаях будет делиться на две части, одна из которых будет одинаковой - предобученная модель Transolver на поиск поля концентраций в начальный момент времени.

Опишем архитектуру более подробно.

Решение прямой задачи. Получаем оценку состояния ситемы в начальный момент времени с помощью обученной в прошлом пункте модели Transolver. Данная часть нейронной сети не будет обучаться отдельно для этой задачи. Здесь будет использована предобученная модель, но она будет работать следующим образом: если обучение модели происходило на временных данных 2 - 18 и предсказывалось время 1, то для этой задачи будут братья также 17 временных шагов, но это будут шаги 1 - 17 и будет предсказываться состояние 0, которое соответствует времени начала распространения загрязнений.

Решение обратной задачи будет разным для этих подходов, далее более подробно будет описана архитектура и результаты для построения решения в обоих случаях.

4.2.1. Классификация

После применения Transolver необходимо правильно обрабатывать поле концентраций, в этом случае будет использоваться модель классификации с использованием сверточных нейронных для решения об-

ратной задачи. Блок для классификации принимает на вход тензор размерности $B \times 1 \times H \times W$, где B — размер батча, H и W — пространственные размеры карты концентраций.

Сначала идёт трёхслойная сверточная часть \square . Первый сверточный слой задаётся ядром размером 3×3 и $P = 1$ пиксель паддинга, преобразуя 1 канал в 32 канала. Затем применяется пакетная нормализация:

$$\tilde{f}_{b,c,i,j}^{(1)} = \gamma_c \frac{f_{b,c,i,j}^{(1)} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} + \beta_c,$$

где μ_c и σ_c^2 — пакетные статистики, γ_c, β_c — обучаемые параметры. После — нелинейность ReLU и пулинг 2×2 с шагом 2 по каждой пространственной оси, уменьшающий размеры вдвое.

Аналогично устроены второй и третий свёрточные блоки, изменяя число каналов: $32 \rightarrow 64$ и $64 \rightarrow 128$. После каждого идёт BatchNorm, ReLU и MaxPool 2×2 . В результате карты имеют размер $128 \times \frac{H}{8} \times \frac{W}{8}$.

Затем применяется адаптивный глобальный пулинг \square по каждому каналу, что даёт вектор $g \in \mathbb{R}^{128}$. Этот вектор проходит через два полносвязных слоя:

$$h^{(1)} = \text{ReLU}(W^{(1)}g + b^{(1)}), \quad \text{logits} = W^{(2)}h^{(1)} + b^{(2)},$$

где $W^{(1)} \in \mathbb{R}^{64 \times 128}$, $W^{(2)} \in \mathbb{R}^{10 \times 64}$. В итоге получаем вектор логитов размерности 10 для классификации источника.

Эта модель обучалась 23 минут 3 секунд на GPU T4×2 на данных описанных во второй главе. Получились следующие результаты

Источник	Precision	Recall	F1-score	Количество данных
Источник 0	1.00	0.89	0.94	9
Источник 1	0.89	0.89	0.89	9
Источник 2	0.90	1.00	0.95	9
Источник 3	0.70	0.78	0.74	9
Источник 4	1.00	1.00	1.00	9
Источник 5	1.00	1.00	1.00	9
Источник 6	0.90	1.00	0.95	9
Источник 7	1.00	0.78	0.88	9
Источник 8	1.00	0.78	0.88	9
Источник 9	0.82	1.00	0.90	9

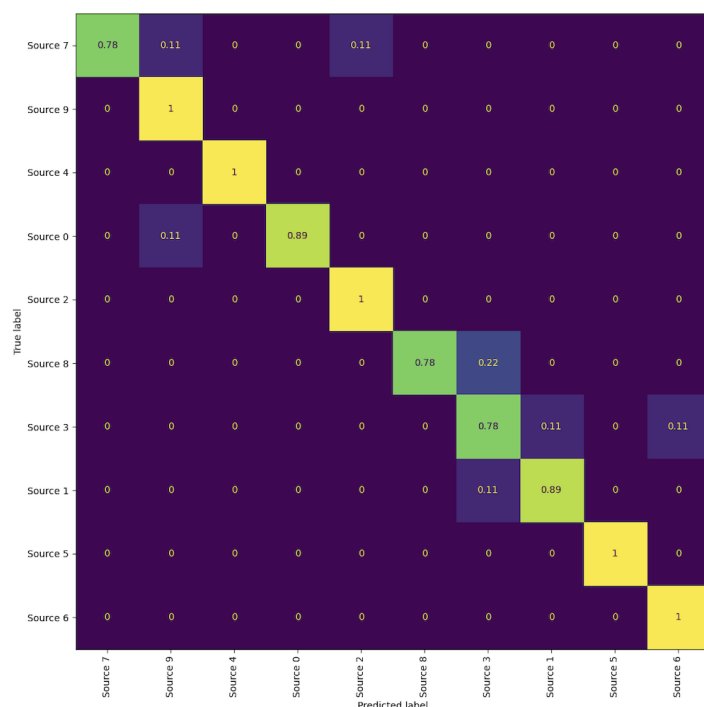


Рисунок 4.2. Точность модели классификации.

Модель определяет классы источников корректно. В основном ошибки возникают из-за близкого расположения источников к друг другу, поэтому не всегда модели удастся определить корректный класс. Тем не менее точность сверточных нейронных сетей очень высокая, так например классификация с помощью метода ближайших соседей для этой же задачи показывала точность равную 30%, что гораздо хуже данного подхода. Определение класса источников хороший подход, если заранее известны все источники и есть гарантия того, что новых источников не появится. Но, если увеличить количество классов источников и требовать как высокую полноту (recall), так и высокую точность (precision), то данный подход может быть не самым оптимальным, так как есть необходимость в точном поиске источнике, а не приблизительном классе, к которому можно его отнести. Поэтому возникает необходимость рассмотреть иной подход - определить не класс, а координаты источника на прямую.

4.2.2. Регрессия

Здесь будет рассмотрено несколько подходов для построение модели регрессии. Первый подход - это использование полносвязного слоя. С помощью него , используя предсказание поля концентраций

в начальный момент времени (выход модели Transolver), необходимо получить координаты. Этот слой представляет собой последовательность из нескольких операций. Входной тензор сначала проходит через операцию адаптивного среднего пула, которая преобразует его из размера (B, C, H, W) в тензор размерности $(B, C, 1, 1)$, где B — размер пакета, C — количество каналов, а H и W — высота и ширина входного изображения. Далее применяется операция уменьшения размерности, которая преобразует тензор размерности $(B, C, 1, 1)$ в вектор размерности (B, C) . Следующий этап — это линейный слой, который преобразует вектор из 64 элементов в вектор размерности 32, где $C = 64$ — это размер скрытого слоя модели. После линейного слоя применяется функция активации ReLU, которая удаляет отрицательные значения и оставляет только положительные. На последнем этапе используется ещё один линейный слой, который преобразует выход из предыдущего слоя размерности 32 в выход размерности 2, что соответствует координатам (x, y) . Итоговый выход этого слоя имеет размерность $(B, 2)$, где 2 — это координаты (x, y) , два неотрицательных числа лежащих в сегменте $[0; 251] \times [0; 201]$. Эта часть нейронной сети будет обучаться именно для этой задачи на поиск координат источников.

Для данной модели обучение будет происходить на координатах источника представленных в виде широты и долготы. После получения предсказания модели данные необходимо будет преобразовать следующим образом:

$$\hat{x}_0 = \frac{|\hat{x} - 54.5|}{0.0067625}, \quad \hat{y}_0 = \frac{|\hat{y} - 82.2975|}{0.0067625} \quad (9)$$

Здесь, \hat{x}, \hat{y} - предсказанные моделью координаты, \hat{x}_0, \hat{y}_0 - координаты сетки, полученные с помощью преобразования широты и долготы.

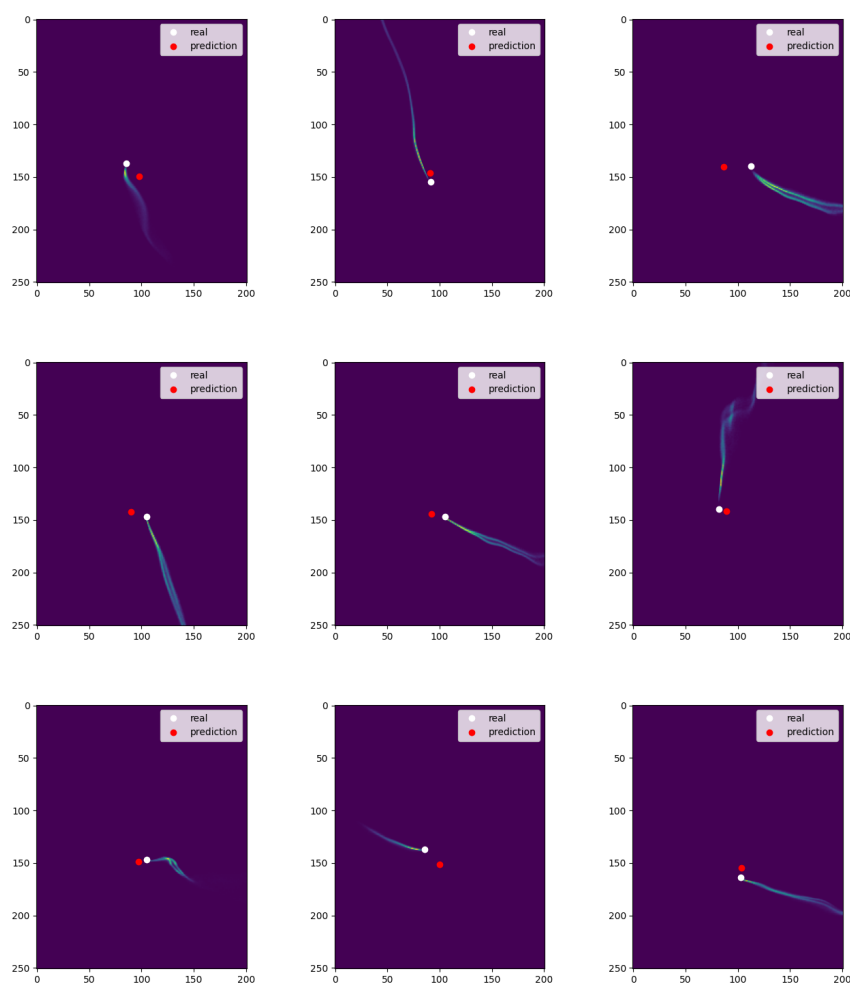


Рисунок 4.3. Оценка координат источника и сам источник находящийся на карте полей концентраций в последний момент времени.

Несмотря на то, что некоторые источники определяются достаточно точно, модель в целом плохо справляется с задачей. Благодаря выходу модели Transolver обычные сверточные слои могут лучше улавливать поведение системы, но тем не менее модели не хватает нелинейности, чтобы лучше с этим справиться.

Модель выучила самые частые паттерны в распространении загрязнений для каждого источника и поэтому может сильно ошибаться на редких для определенного класса данных. Посмотрим на метрики качества, на некотором подмножестве тестовой выборки:

№ батча	MAE	MSE	RMSE
1	67.3340	4255.8212	65.2367
5	43.5031	2275.3025	47.7001
6	43.0556	2220.6473	47.1237
10	45.5665	2267.0678	47.6137
12	84.6338	6620.7268	81.3678

Таблица 4.1

Сравнение метрика модели MAE, MSE и RMSE

Видно, что для данной задачи, такие ошибки являются достаточно большими, и координаты источника не всегда могут быть определены с большой точностью для такой модели. Так как координаты источника лежат в сегменте $[0; 201] \times [0; 251]$. Поэтому расстояние между оценкой координаты и самой точкой по модулю больше чем 40 не является достаточно качественным результатом.

Поэтому возникает необходимость в рассмотрении иного подхода. Сверточные сети благодаря возможности очень качественно агрегировать информацию на изображениях могут достаточно точно выделять нужную точку на карте полей концентраций.

Рассмотрим более продвинутое использование сверточных сетей, а именно в качестве слоя отвечающего за обработку выхода модели Transolver возьмем resnet18 [19]. Эта архитектура благодаря residual блоку ускоряет сходимость модели, препятствует затуханию градиента и быстрее сходится к оптимуму функции потерь. Модель также обладает более сложной архитектурой, что является важным дополнением к прошлой архитектуре, так как это поможет более точно искать необходимые зависимости между полем концентраций и источником эмиссии загрязнений.

Данная модель обучалась на тех же данных, при этом стоит отметить, что сам процесс обучения отличался от предыдущего подхода. Регрессор с обычным сверточным слоем обучался на координатах широты и долготы, так иначе модель начинала расходиться. Данная архитектура с использованием resnet обучалась сразу на координатах источника загрязнений. Обучение происходило на процессоре GPU и на 100 эпохах. Этот процесс занял 24 минуты.

Модель гораздо лучше справилась с поставленной задачей. Она успешно определила источника для всей тестовой выборки, независимо от характерных для каждого источника полей концентраций физически-информированная нейронная сеть смогла извлечь необходимую информацию из данных.

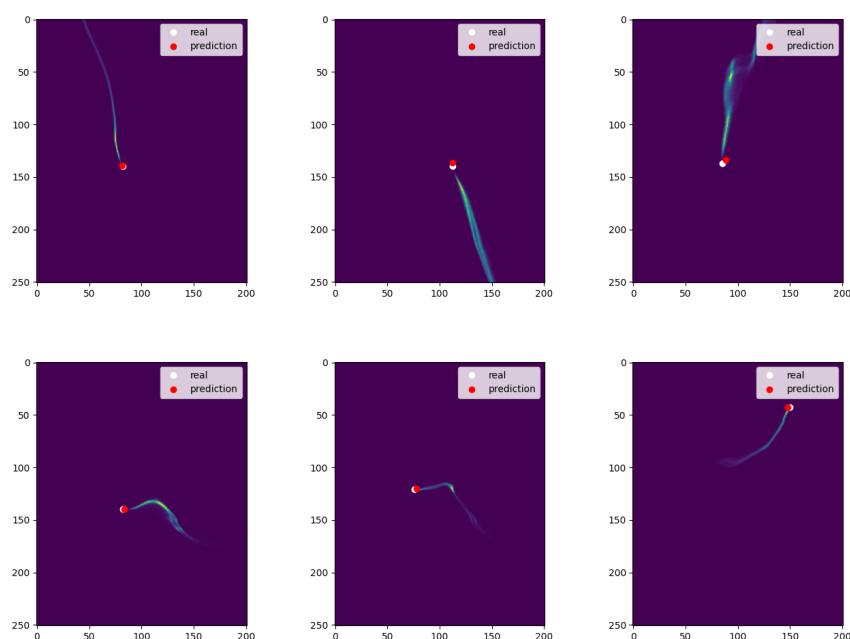


Рисунок 4.4. Оценка координат источника и сам источник находящийся на карте полей концентраций в последний момент времени.

Видим, что в отличии от предыдущего решения

№	MAE	MSE	RMSE
1	3.0982	5.9425	2.4377
2	3.0014	6.4966	2.5488
3	2.3431	3.7208	1.9289
4	2.7110	4.5358	2.1297
5	4.6842	16.2885	4.0359
6	3.4383	10.6662	3.2659
7	2.6241	5.8549	2.4197
8	3.1490	6.6261	2.5741

Таблица 4.2

Метрики MAE, MSE и RMSE для второй модели.

Улучшение модели с помощью resnet гарантировало точное решение. Модель умеет определять источник из каждого класса и не чувствительна к разному распространению загрязнений у одного источника.

Таким образом и модель регрессии, и модель классификации продемонстрировали хорошие результаты по поиску источника загрязнений для разных полей концентраций. При этом важно разделять

то, какой подход нужен для конкретной задачи. При заранее известном классе источников классификация может быть гораздо лучше чем регрессия, если же классы неизвестны заранее модель регрессии может с высокой точностью найти координаты источника.

Заключение

В рамках данной работы были предложены решения задачи по определению координат источников загрязнений воздуха с использованием новейших подходов для аппроксимации и предсказания поведения физических систем. Основной целью было показать, что Transolver может с высокой точностью определять состояние физической системы в начальный момент времени, что позволяет успешно решать обратную задачу по поиску источников. На основе этой модели были рассмотрены как классификаторы, так и регрессоры успешно определяющие координаты источника на синтетических данных по загрязнению воздуха города Новосибирск.

Рассмотренные решения задачи, показывает насколько может быть полезна эта модель основанная на трансформерной архитектуре не только для получения аппроксимации системы в определенный момент времени, но и для поиска других важных характеристик, которые напрямую связаны с различными физическими системы, например поиск источников загрязнений воздуха.

Будущая работа может быть сосредоточена на адаптации Transolver к работе с более сложными данными, где присутствует внешние воздействия, влияющее на распространение пыли в воздухе, например ветер. Также следующим шагом будет переход от двумерной задаче к трехмерной. Здесь потребуются более сложные модели, при этом Transolver может аппроксимировать физические системы и в трехмерном пространстве, что позволит оставить его в качестве основного подхода для решения задачи.

Текущие и дальнейшие результаты данной работы будут интегрированы в работу компании «CityAir» по мониторингу окружающей среды в реальном времени для определения источников загрязнений.

Список литературы

- [1] Vaswani A., Shazeer N., Parmar N., et al. Attention is all you need // Advances in Neural Information Processing Systems. 2017. Pp. 5998–6008. URL: <http://arxiv.org/abs/1706.03762>.
- [2] Raissi M., Perdikaris P., Karniadakis G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations // Journal of Computational Physics. 2019. Vol. 378. Pp. 686–707.
- [3] Савенков Е.Б. Решения уравнений в частных производных на поверхностях: обзор алгоритмов // Препринты ИПМ им. М.В. Келдыша. 2020. №5. С. 1–18. DOI: [10.20948/prepr-2020-5](https://arxiv.org/abs/10.20948/prepr-2020-5). URL: <https://library.keldysh.ru/preprint.asp?id=2020-5>. Полный текст: https://keldysh.ru/papers/2020/2020_prep2020_5.pdf.
- [4] Gao H., Sun L., Wang J.-X. PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving Parameterized Steady-State PDEs on irregular domain // Journal of Computational Physics. 2020. Article ID: 110079. DOI: [10.1016/j.jcp.2020.110079](https://doi.org/10.1016/j.jcp.2020.110079).
- [5] Jin X., Cai S., Li H., Karniadakis G. E. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations // Journal of Computational Physics. 2021. Vol. 426. Article ID: 109951. DOI: [10.1016/j.jcp.2020.109951](https://doi.org/10.1016/j.jcp.2020.109951).
- [6] Wu H., Luo H., Wang H., Wang J., Long M. Transolver: A Fast Transformer Solver for PDEs on General Geometries // Proc. of International Conference on Machine Learning. 2024.
- [7] Сайт CityAir. URL: <https://cityair.ru/ru/> (дата обращения: 13.05.2025).
- [8] Репозиторий работы о сравнении скорости работы PINN и других методов машинного обучения для задачи аппроксимации потока жидкости в скважинах. URL: https://github.com/kirillkatsuba/PINN_project (дата обращения: 13.05.2025).

- [9] Xiang Z., Peng W., Liu X., Yao W. Self-adaptive loss balanced Physics-informed neural networks // *Neurocomputing*. 2022. Vol. 496. Pp. 11–34. DOI: [10.1016/j.neucom.2022.05.015](https://doi.org/10.1016/j.neucom.2022.05.015).
- [10] Deguchi S., Asai M. Dynamic & norm-based weights to normalize imbalance in back-propagated gradients of physics-informed neural networks // *Journal of Physics Communications*. 2023. Vol. 7. DOI: [10.1088/2399-6528/ace416](https://doi.org/10.1088/2399-6528/ace416).
- [11] Li Z., Kovachki N., Azizzadenesheli K., et al. Fourier Neural Operator for Parametric Partial Differential Equations // arXiv preprint arXiv:2010.08895 [cs.LG], 2021. URL: <https://arxiv.org/abs/2010.08895>.
- [12] Rahman M. A., George R. J., Elleithy M., et al. Pretraining Codomain Attention Neural Operators for Solving Multiphysics PDEs // *Advances in Neural Information Processing Systems*. 2024. Vol. 37.
- [13] Hoyer S., Hamman J. xarray: N-D labeled Arrays and Datasets in Python // *Journal of Open Research Software*. 2017. Vol. 5. DOI: [10.5334/jors.148](https://doi.org/10.5334/jors.148).
- [14] Ivan Chuprov, Denis Derkach, Dmitry Efremenko, Aleksei Kychkin. Application of Physics-Informed Neural Networks for Solving the Inverse Advection-Diffusion Problem to Localize Pollution Sources // arXiv preprint arXiv:2503.18849 [cs.NE], 2025. URL: <https://arxiv.org/abs/2503.18849>.
- [15] Paszke A., Gross S., Massa F., et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library // arXiv preprint arXiv:1912.01703 [cs.LG], 2019. URL: <https://arxiv.org/abs/1912.01703>.
- [16] Krizhevsky A., Sutskever I., Hinton G. ImageNet Classification with Deep Convolutional Neural Networks // *Neural Information Processing Systems*. 2012. Vol. 25. DOI: [10.1145/3065386](https://doi.org/10.1145/3065386).
- [17] He K., Zhang X., Ren S., Sun J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification // arXiv preprint arXiv:1502.01852 [cs.CV], 2015. URL: <https://arxiv.org/abs/1502.01852>.

- [18] Dosovitskiy A., Beyer L., Kolesnikov A., Weissenborn D., Zhai X., Unterthiner T., Dehghani M., Minderer M., Heigold G., Gelly S., Uszkoreit J., Houlsby N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale // arXiv preprint arXiv:2010.11929 [cs.CV], 2021. URL: <https://arxiv.org/abs/2010.11929>.
- [19] He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // arXiv preprint arXiv:1512.03385 [cs.CV], 2015. URL: <https://arxiv.org/abs/1512.03385>.

А. Приложения

Listing 1: Функция для считывания данных

```
1 import numpy as np
2 import xarray as xr
3 import pathlib
4 from tqdm import tqdm
5
6 def parse_data(dataset_paths):
7     data = []
8     source_coordinats = []
9     for p in tqdm(dataset_paths):
10         for source in range(10):
11             subset = xr.open_dataset(p)['CONC'][:, 0, source, :, :]
12             source_MIN_LATS = xr.open_dataset(p)['CONC'].attrs['MIN_LATS'][
13                 source]
14             source_MIN_LONGS = xr.open_dataset(p)['CONC'].attrs['MIN_LONGS'
15                 ][source]
16             source_MAX_LATS = xr.open_dataset(p)['CONC'].attrs['MAX_LATS'][
17                 source]
18             source_MAX_LONGS = xr.open_dataset(p)['CONC'].attrs['MAX_LONGS'
19                 ][source]
20             subset = np.array(subset)
21             source_coordinats.append([
22                 source,
23                 source_MIN_LATS,
24                 source_MIN_LONGS,
25                 source_MAX_LATS,
26                 source_MAX_LONGS
27             ])
28             data.append(subset)
29     return np.array(data), np.array(source_coordinats)
```

Listing 2: Архитектура модели Transolver

```
1 import numpy as np
2 import pandas as pd
3 import pathlib
4 from tqdm import tqdm
5 import matplotlib.pyplot as plt
6 import sys
7 import torch
8 from model.Transolver_Structured_Mesh_2D import Model
9
10 tmodel = Model(space_dim=2,
11                 n_layers=3,
12                 n_hidden=64,
13                 n_head=4,
14                 Time_Input=False,
15                 mlp_ratio=1,
16                 fun_dim=17,
17                 out_dim=1,
```



```

18         slice_num=64,
19         ref=8,
20         unified_pos=1,
21         H=201,
22         W=251).cuda()

```

Listing 3: Архитектура модели регрессии для поиска источников

```

1 import torch.nn as nn
2 import torch
3 from TransolverPDE.model.Transolver_Structured_Mesh_2D import Model
4
5 class TransolverForSource(nn.Module):
6     def __init__(self, transolver_path: str):
7         super().__init__()
8         self.backbone = Model(space_dim=2,
9                                n_layers=3,
10                                n_hidden=64,
11                                n_head=4,
12                                Time_Input=False,
13                                mlp_ratio=1,
14                                fun_dim=17,
15                                out_dim=1,
16                                slice_num=64,
17                                ref=8,
18                                unified_pos=1,
19                                H=201,
20                                W=251).cpu()
21         self.backbone.load_state_dict(torch.load(transolver_path,
22                                                  map_location=torch.device('cpu')))
23         self.coord_head = nn.Sequential(
24             nn.AdaptiveAvgPool2d((1,1)),
25             nn.Flatten(),
26             nn.Linear(1, 32),
27             nn.ReLU(),
28             nn.Linear(32, 64),
29             nn.ReLU(),
30             nn.Linear(64, 32),
31             nn.ReLU(),
32             nn.Linear(32, 2),
33         )
34     def forward(self, coords, fx):
35         field_pred = self.backbone(coords, fx)
36         xy = self.coord_head(field_pred.reshape(field_pred.shape[0], 251,
37         201)) # (B,2)
37         return xy

```

Listing 4: Архитектура модели классификации для поиска источников

```

1 import torch.nn as nn
2 import torch

```

```

3 from TransolverPDE.model.Transolver_Structured_Mesh_2D import Model
4
5 class TransolverForSource(nn.Module):
6     def __init__(self, transolver_path: str):
7         super().__init__()
8         self.backbone = Model(space_dim=2,
9                               n_layers=3,
10                              n_hidden=64,
11                              n_head=4,
12                              Time_Input=False,
13                              mlp_ratio=1,
14                              fun_dim=17,
15                              out_dim=1,
16                              slice_num=64,
17                              ref=8,
18                              unified_pos=1,
19                              H=201,
20                              W=251).cpu()
21         self.backbone.load_state_dict(torch.load(transolver_path,
22 map_location=torch.device('cpu'))))
23         self.cnn_head = nn.Sequential(
24             nn.Conv2d(1, 32, kernel_size=3, padding=1),
25             nn.BatchNorm2d(32),
26             nn.ReLU(inplace=True),
27             nn.MaxPool2d(2),
28
29             nn.Conv2d(32, 64, kernel_size=3, padding=1),
30             nn.BatchNorm2d(64),
31             nn.ReLU(inplace=True),
32             nn.MaxPool2d(2),
33
34             nn.Conv2d(64, 128, kernel_size=3, padding=1),
35             nn.BatchNorm2d(128),
36             nn.ReLU(inplace=True),
37             nn.MaxPool2d(2),
38
39             nn.AdaptiveAvgPool2d((1,1)),
40             nn.Flatten(),
41             nn.Linear(128, 64),
42             nn.ReLU(inplace=True),
43             nn.Linear(64, 10)
44         )
45     def forward(self, coords, fx):
46         field_pred = self.backbone(coords, fx) # [B, N, 1]
47         field_pred = field_pred.reshape(field_pred.shape[0], 1, 251, 201)
48
49         logits = self.cnn_head(field_pred)
50         return logits

```

Примеры предсказаний Transolver:

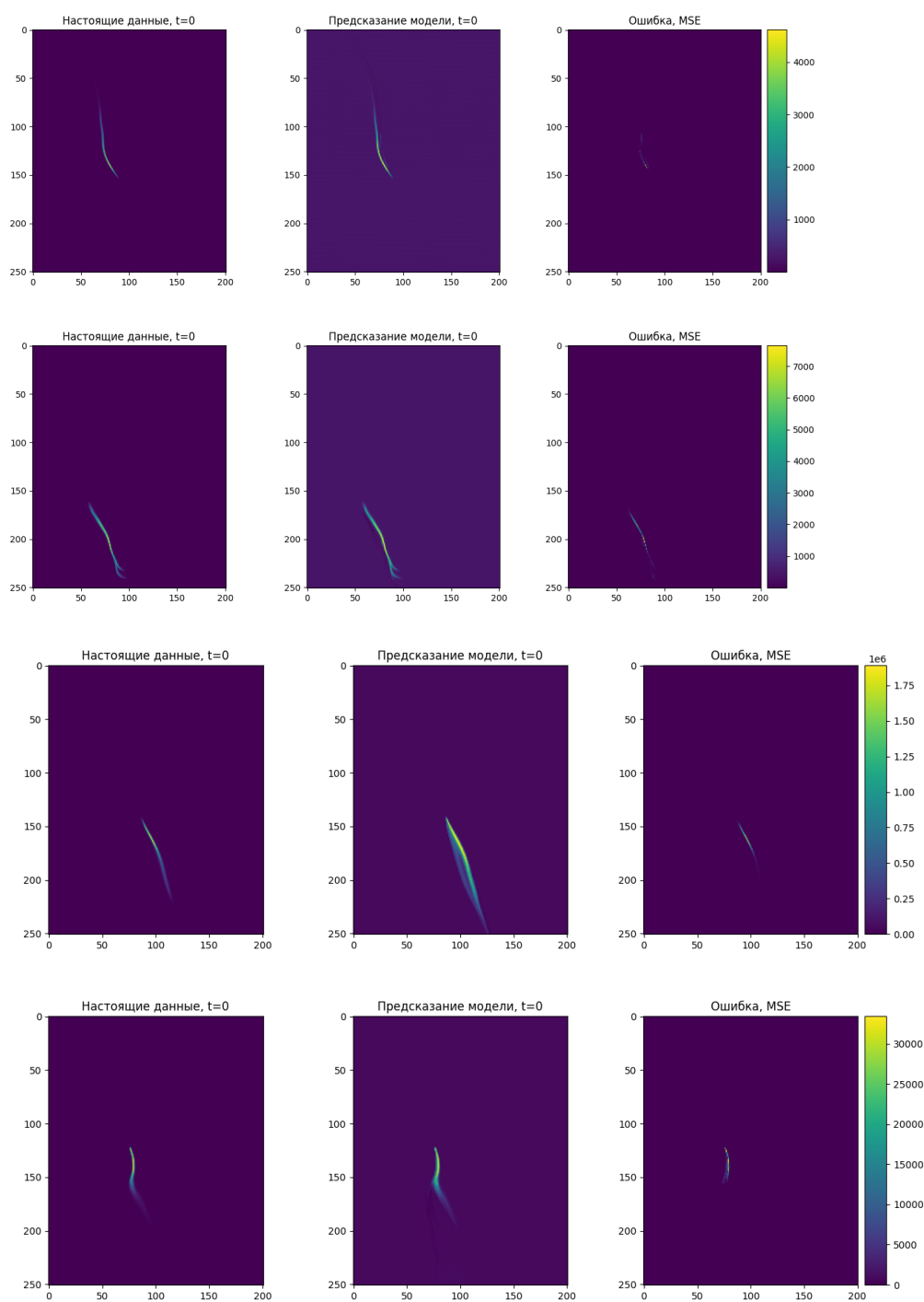


Рисунок А.1. Применение модели Transolver к данным.

Примеры предсказаний регрессионной модели со слоем resnet:

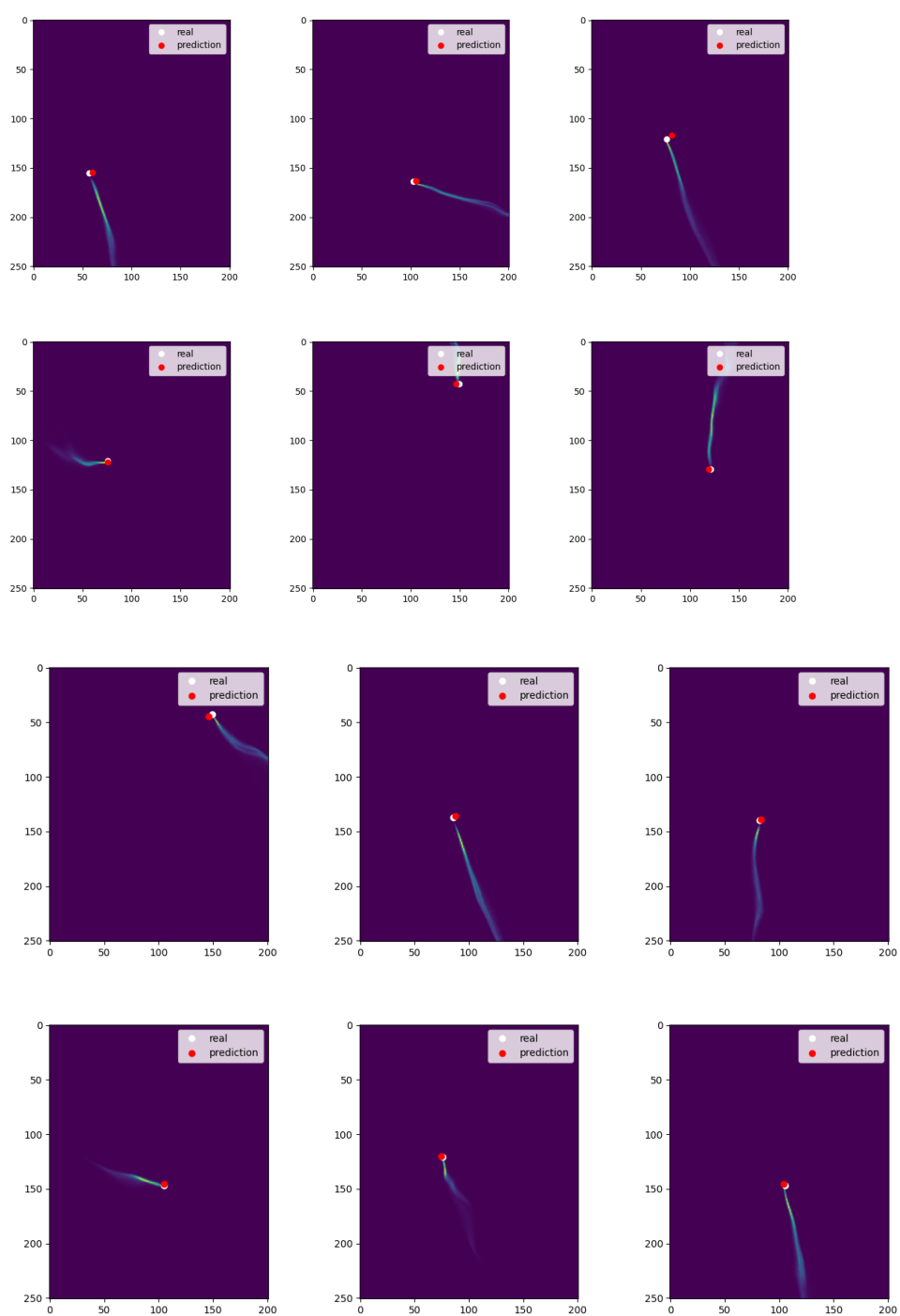


Рисунок А.2. Применение модели Transolver к данным.