МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Университет «Дубна»

Институт системного анализа и управления

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема: Применение методов глубокого обучения для реконструкции событий по данным временных слайсов в эксперименте SPD

Ф.И.О. студента Борисов Максим Сергеевич

Группа 6015 **Направление подготовки** 27.04.03 Системный анализ и управление

Направленность (профиль) образовательной программы Бизнес-аналитика и интеллектуальный анализ данных

Выпускающая кафедра системного анализа и управления

Руководитель работы _____ /проф. Ососков Г.А./

Консультант (ы) _____ /_____/
_____ /_____/
_____ /_____/

Рецензент _____ / к.ф.-м.н. Жемчугов А.С. /

Выпускная квалификационная работа допущена к защите «_____»_____20___г.
(дата)

Заведующий кафедрой _____ / Черемисина Е. Н. /

г. Дубна

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Университет «Дубна»

Институт системного анализа и управления

УТВЕРЖДАЮ
Заведующий кафедрой

_____ /Черемисина Е.Н. /
(подпись) (Фамилия И. О.)

«_____» _____20___ г.

З а д а н и е
на выпускную квалификационную работу – магистерскую диссертацию

Тема: Применение методов глубокого обучения для реконструкции событий по данным временных слайсов в эксперименте SPD

Утверждена приказом № _____ от _____

ФИО студента Борисов Максим Сергеевич

Группа 6015 **Направление подготовки** 27.04.03 Системный анализ и управление

Направленность (профиль) образовательной программы Бизнес-аналитика и интеллектуальный анализ данных

Выпускающая кафедра системного анализа и управления

Дата выдачи задания «___» _____ 20___ г.

Дата завершения
выпускной квалификационной работы «___» _____ 20___ г.

г. Дубна

Исходные данные к работе

1) Справочные материалы по нейросетям и машинному обучению;

2) Справочные материалы и наработки по обработке данных в эксперименте SPD;

3) Наработки в области обработки данных для экспериментов физики высоких энергий;

4) Скрипт для моделирования событий эксперимента SPD;

Результаты работы:

1. Содержание пояснительной записки (перечень рассматриваемых вопросов)

1) Изучение эксперимента SPD. Моделирование данных с использованием предоставленного скрипта;

2) Определение и разработка метрик для оценки качества решаемой задачи;

3) Подготовка среды окружения для апробации алгоритмов;

4) Разработка подхода реконструкции событий на основе вершин;

5) Разработка подхода реконструкции событий на основе кластеризации векторов признаков;

6) Сравнение подходов и выбор оптимального;

7) Подготовка и оптимизация алгоритмов для инференса.

2. Перечень демонстрационных листов (материалов)

Презентация

Консультант(ы) _____ /_____ /

Руководитель работы _____ / проф. Ососков Г.А./

Задание принял к исполнению _____

(дата)

_____

(подпись студента)

Я, Борисов Максим Сергеевич, ознакомлен(а) с требованием об обязательности проверки выпускной квалификационной работы на объем заимствования. Все прямые заимствования из печатных и электронных источников, а также из защищенных ранее выпускных квалификационных работ, научных докладов об основных результатах подготовленной научно-квалификационной работы (диссертации), кандидатских и докторских диссертаций, должны иметь в работе соответствующие ссылки.

Я ознакомлен(а) с Порядком проверки на объем заимствования и размещения в электронно-библиотечной системе текстов выпускных квалификационных работ и научных докладов обучающихся, согласно которому обнаружение тексте выпускной квалификационной работы заимствований, в том числе содержательных, неправомочных заимствований, является основанием для недопуска к защите выпускной квалификационной работы и отчисления из образовательной организации.

_____ /_____Борисов М.С._____/
подпись *Фамилия И.О.*

Аннотация

М. С. Борисов. Магистерская диссертация. Применение методов глубокого обучения для реконструкции событий по данным временных слайсов в эксперименте SPD / Руководитель: профессор Г. А. Ососков; Объединенный институт ядерных исследований, 2024. — 37 с.

Исследование посвящено разработке методов реконструкции событий в рамках эксперимента *SPD*. Основная задача заключается в разработке онлайн фильтра, способного распутывать события внутри временных слайсов. В ходе работы разработаны методы вычисления метрик оценки качества, а также программная среда для апробации алгоритмов. Реализованы подходы на основе предсказания вершины событий, которые продемонстрировали неудовлетворительные результаты. Разработан более эффективный подход с использованием сиамских нейронных сетей, который также способен обрабатывать нефиксированное количество событий внутри временного среза. Полученные результаты были представлены на конференции *GRID*-2023 и семинаре в СПбГУ. По результатам работы опубликована статья в журнале "*Physics of Particles and Nuclei*". Исследование выполнено при поддержке Российского научного фонда № 22-12-00109 и проведено на платформе *HybriLIT* (ЛИТ, ОИЯИ).

Abstract

M. S. Borisov. Master's Thesis. Application of Deep Learning Methods for Event Reconstruction Using Time Slice Data in the SPD Experiment / Supervisor: Professor G. A. Ososkov; Joint Institute for Nuclear Research, 2024. — 37 pages.

The research focuses on developing event reconstruction methods within the SPD experiment. The primary objective is to develop an online filter capable of disentangling events within time slices. During the work, methods for calculating quality assessment metrics were developed, as well as a software environment for testing algorithms. Approaches based on event vertex prediction were implemented, which demonstrated unsatisfactory results. A more effective approach was developed using Siamese neural networks, capable of processing a variable number of events within a time slice. The results were presented at the GRID-2023 conference and a seminar at St. Petersburg State University. Based on the results of the work, an article was published in the journal "Physics of Particles and Nuclei". The research was supported by the Russian Science Foundation No. 22-12-00109 and conducted on the HybriLIT platform (LIT, JINR).

Содержание

Введение

SPD (Spin Physics Detector) разрабатывается для изучения спиновой структуры протона, дейтрона и других явлений, связанных со спином, с помощью поляризованных пучков протонов и дейтронов при энергии столкновения до 27 ГэВ и светимости до 10^{32} $cm^{-2}s^{-1}$. Данные о событиях из *SPD* будут поступать со скоростью 3 МГц в виде тайм-срезов в 10 мс, в каждом из которых будет происходить в среднем 40 событий, т.е. один тайм-слайс будет содержать в среднем 200 треков и 1100 хитов на одну станцию (причем 82,26% всех хитов являются фейками) [1]. Планируется разработать алгоритм для онлайн фильтра, чтобы обрабатывать не менее 100 тайм-срезов в секунду. Данный алгоритм использует треки-кандидаты от алгоритма трекинга [2]. Это позволит в режиме реального времени отбирать интересующие события, отвечающие целям эксперимента, и сохранять только их, тем самым значительно сократить сетевую нагрузку и требуемое дисковое пространство.

Постановка задачи

Целью данного исследования является анализ и разработка методов реконструкции событий в рамках эксперимента *SPD*, используя передовые алгоритмы машинного глубокого обучения для обработки и анализа высокоскоростных потоков данных. Важность задачи обусловлена ожидаемыми высокими темпами получения данных в эксперименте и необходимостью их оперативной обработки. Основная задача заключается в моделировании данных для восстановления событий и последующему сравнительному анализу разработанных подходов с целью выбора наиболее эффективного. Модели обучаются на подготовленных треках из симуляции, в реальности входными данными будут треки-кандидаты от алгоритма трекинга [2]. Треком называется – траектория частицы, зарегистрированной детектором, представленная в виде последовательности сигналов, хитов. Для достижения поставленных целей предполагается выполнение следующих задач:

1) Изучение эксперимента *SPD*. Моделирование данных с использованием предоставленного скрипта;
2) Определение и разработка метрик для оценки качества модели, решающей поставленную задачу решаемой задачи;
3) Подготовка и разработка программной среды для апробации алгоритмов;
4) Разработка подхода реконструкции событий на основе вершин;
5) Разработка подхода реконструкции событий на основе кластеризации векторов признаков;
6) Сравнение подходов и выбор оптимального;
7) Подготовка и оптимизация скорости алгоритмов.

Исследование нацелено на создание комплексного решения, способного эффективно функционировать в условиях высокой нагрузки и обеспечивать точность реконструкции событий для последующего анализа в эксперименте *SPD*.

Генерация модельных данных

Поскольку реальные данные из эксперимента еще недоступны, так как он пока не запущен, в работе используются модельные данные. Эти данные были получены с помощью алгоритма, разработанного физиками на основе доступной информации о событиях эксперимента *SPD*. Данный алгоритм реализован в виде *Python*-скрипта, который аппроксимирует спиральную траекторию движения частицы. В каждом моделируемом событии количество треков изменяется от 1 до 10, а поперечный импульс частицы подчиняется равномерному распределению в пределах от 100 до 1000 МэВ/с. Координаты вершин выбираются случайно (по равномерному распределению) из заданной области вдоль оси *Z* (для первой версии скрипта), что соответствует зоне потенциальных столкновений частиц.

Траектории частиц в модели представлены в виде набора точек спиральной траектории, при этом точки распределены в соответствии со среднеквадратическим отклонением, известным из конструкции трекинговой системы (*Silicon Tracking System* — *STS*) [3]. В исследовании использовалась конфигурация детектора, состоящая из 35 станций (см. рис. 1).

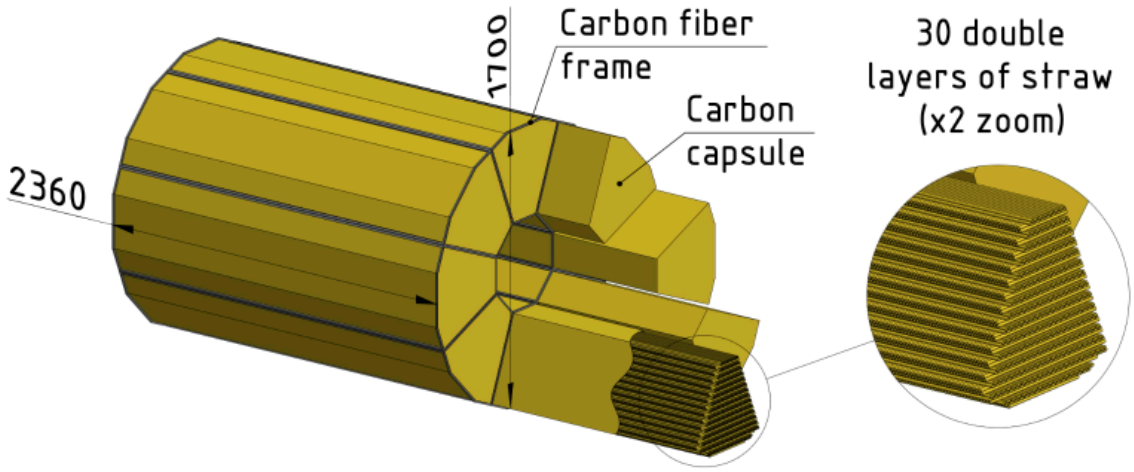

Рис. 1. Схема трубчатой части детектора *SPD*

Неэффективность детектора учитывалась путем введения вероятности исключения хита из набора данных, при этом эффективность детектора оценивалась в 99% и 98%. Итоговый набор данных включает в себя: номер события evt, координаты хитов *(x, y, z)*, номер станции, идентификатор трека *trk* в событии, импульс *p(x, y, z)* и координаты вершины *vtx(x, y, z)* (см. рис. 2).

	evt	x	y	z	station	trk	px	py	pz	vtxx	vtxy	vtxz
0	0	-268.018768	33.173191	565.522303	1	0	-507.704732	94.421070	851.451364	-1.253358	-15.544558	120.099724
1	0	-276.910426	34.872703	580.684414	2	0	-507.312956	96.503876	851.451364	-1.253358	-15.544558	120.099724
2	0	-286.027191	36.580065	595.575991	3	0	-506.912583	98.585324	851.451364	-1.253358	-15.544558	120.099724
3	0	-294.664452	38.385470	610.836264	4	0	-506.503616	100.665389	851.451364	-1.253358	-15.544558	120.099724
4	0	-303.774233	40.166026	625.997983	5	0	-506.086055	102.744042	851.451364	-1.253358	-15.544558	120.099724
...	...	...	...	...	...	...	...	...	...	...	...	...

Рис. 2. Пример данных, полученных в результате работы скрипта генерации

В эксперименте данные записываются в виде временных срезов (*timeslice*), которые состоят из набора событий. Такой подход позволят значительно сократить нагрузку.

Пример модельного временного среза в эксперименте *SPD* при 40 событиях во временном срезе (см. рис. 3 а). Первый вариант (а) отражает полный пример генерации данных, с шумом, треками и вершинами. Шум – это ложное срабатывание детектора на шумные или не относящиеся к эксперименту частицы. Вершина – это точка взаимодействия частиц. Треки показаны цветными линиями, их первичные вершины – точками соответствующего цвета (см. рис. 3 *c*). Фейковые хиты показаны серыми точками. На рисунке 3 *b*, продемонстрированы чистые данные, без фейков, именно такой набор данных будет использован в работе, поскольку использование данного метода планируется после очистки данных от фейков и восстановления треков.

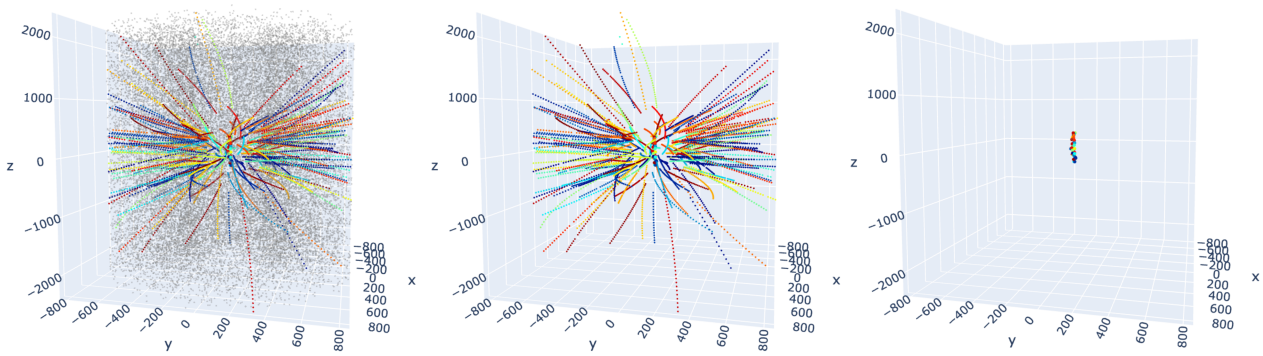

Рис. 3. Визуализация временного среда для 40 событий во временном срезе

В ходе работы над проектом появилась модифицированная, более физическая, версия генератора. Которая включает в себя:

- Добавление *x* и *y* координаты к вершине;
- Нефиксированное количество событий внутри временного среза, которое формируется исходя из Пуассоновского распределения.

Распределение Пуассона — это дискретное вероятностное распределение, которое описывает число событий, происходящих за фиксированный интервал времени или пространства, при условии, что эти событии происходят с постоянной средней

скоростью и независимо друг от друга. Такое распределение характеризуется единственным параметром *lambda* – среднее количество событий, также это значение является дисперсией. Ниже приведен пример генерации 1000 срезов со случайным значением событий (см. рис. 4).

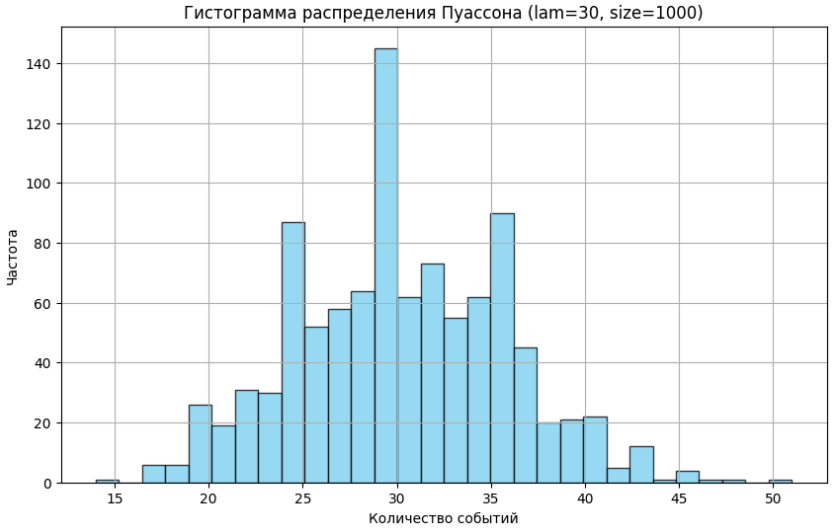

Рис. 4. Визуализация Пуассоновского распределения для 30 событий во временном срезе

Метрики оценки качества

В ходе данной работы рассматриваются две задачи: регрессия и кластеризация, каждая из которых требует применения специфических метрик для оценки результатов.

Для задачи регрессии важно точно измерять степень отклонения предсказанных значений от фактических. В этом контексте используются различные метрики, в работе использовались: средняя абсолютная ошибка (*MAE*) и средняя абсолютная процентная ошибка (*MAPE*).

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\frac{|y_i - \widehat{y_i}|}{|y_i|} \quad (1)$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}| \quad (2)$$

Где:

- y_i – истинное значение;
- $\widehat{y_i}$ – спрогнозированное значение.

Для оценки результатов кластеризации данных широко применяются две основные группы метрик, каждая из которых играет важную роль в интерпретации эффективности используемых алгоритмов [4]:

1. Внутренние – метрики, которые оценивают качество кластеризации, основываясь на структуре самих данных, не используя информацию о лейблах (если такая имеется). Они измеряют, насколько плотно сгруппированы объекты внутри кластеров и как далеко кластеры разнесены друг от друга. Пример: *Silhouette* и *Davies Bouldin index (DBI)*.

$$\text{Silhouette score} = \frac{1}{n}\sum_{i=1}^{n}\frac{b_i - a_i}{\max(a_i, b_i)} \quad (3)$$

$$\text{DBI} = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right) \quad (4)$$

Где:

- n - количество элементов в данных;
- k - количество кластеров;
- ai - среднее расстояние от точки i до других точек в том же кластере;
- bi - минимальное среднее расстояние от точки i до точек в любом другом кластере;

- σi - среднее расстояние от всех точек в кластере i до его центроида;
- $d(ci, cj)$ - расстояние между центроидами кластеров i и j.

2. Внешние – метрики, которые сравнивают результат кластеризации с заранее известной информацией, такой как истинные метки классов (лейблы). Пример: *Precision*, *Recall*, *F1-score*, *Accuracy*.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

Где:

- TP (*True Positive*) - количество истинно положительных предсказаний;
- TN (*True Negative*) - количество истинно отрицательных предсказаний;
- FP (*False Positive*) - количество ложно положительных предсказаний;
- FN (*False Negative*) - количество ложно отрицательных предсказаний.

Расчёт внешних метрик для задачи кластеризации может быть сложной задачей из-за отсутствия прямого и однозначного соответствия между кластерами и истинными классами. Для этих целей были разработаны специальные методы расчёта таких метрик, которые позволяют адекватно оценить качество кластеризации в условиях неопределённости. Эти методы будут рассмотрены подробнее в последующих разделах.

Опираясь исключительно на внутренние метрики, можно получить представление о структурной целостности и внутренней согласованности данных в рамках кластеризации, однако это не всегда корректно отражает реальную эффективность в достижении поставленных задач. Внутренние метрики могут показывать высокие значения, в то время как модель может быть неспособна корректно решать определенную задачу. Поэтому, важно использовать внешние метрики в комбинации с внутренними, что способствует более точной настройке алгоритмов под конкретные применения.

Подходы к решению задачи

В данном эксперименте планируется фильтрация, состоящая из нескольких этапов (см. рис. 5):

a. На этом этапе продемонстрированы сырые данные, приходящие с детектора и не имеющие лейблов принадлежности трекам.
b. На данном этапе произведена очистка данных от фейков, восстановление треков при помощи *TrackNET* [2], а также восстановление пропусков на станциях.
c. На данном этапе производится реконструкция событий по данным временных срезов. Для каждого трека определяется вершина, из которой он был порожден.

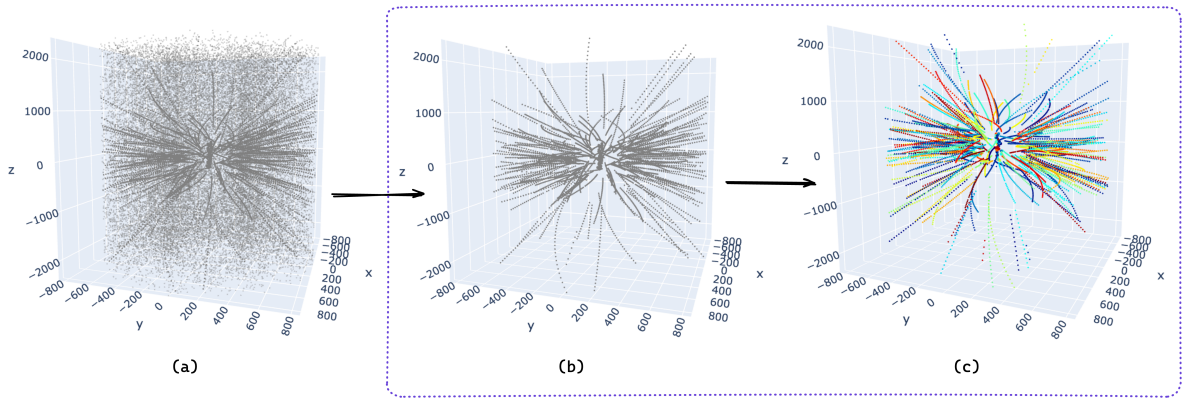

Рис. 5. Этапы фильтрации данных для эксперимента *SPD*

В рамках данной работы будет рассмотрен только этап «Реконструкции событий», для которого будут использоваться подготовленные треки из симуляции, в реальности входными данными будут треки-кандидаты от алгоритма трекинга.

В ходе решения задачи онлайн фильтрации данных, возникающей в эксперименте *SPD*, было разработано несколько методик. В основу первого подхода положено использование вершинных характеристик треков, что позволяет анализировать и фильтровать данные, опираясь на геометрическое положение происхождения частиц. Второй подход основан на применении векторов признаков (*embeddings*), которые формируют такое латентное представление данных, которое обеспечивает их большую разделимость. Вектора признаков формируются с помощью алгоритмов глубокого обучения.

Оба подхода представляют собой уникальные методы обработки, каждый из которых имеет свои преимущества и области применения. Далее, в отдельных блоках, будут подробно рассмотрены технические детали, алгоритмические основы, а также результаты тестирования каждого из методов. Это даст возможность оценить их

эффективность и применимость для решения поставленной задачи фильтрации данных в условиях реального эксперимента.

Вершинный подход

Основная идея метода заключается в восстановлении вершины трека на основе его характеристик (информация с хитов), исходя из предположения, что треки, порождаемые одним и тем же событием, сходятся в общей вершине. Для достижения этой цели использовались методы регрессии и интерполяции.

Интерполяция вершины

В ходе работы был реализован подход для восстановления вершины с использованием нескольких видов интерполяции (линейная/сплайновая). Идея данного подхода заключается в том, что треки из одного события будут иметь близлежащие прогнозы вершин, которые, в дальнейшем, можно будет обособить, используя кластеризацию. Сложность данного подхода выражалась в определении точки, в которую будет производиться интерполировать. Данный подход показал крайне неудовлетворительные результаты на ранних стадиях, поэтому был разработан следующий подход.

Прогнозирование вершины регрессией

Прогнозирование вершины с использованием методов регрессии. В ходе работы протестировано множество реализаций регрессии, но лучшие результаты были достигнуты с применением градиентного бустинга над решающими деревьями (*GBDT*) [5], который показал наилучшие результаты. Это подход к ансамблированию моделей, который последовательно исправляет ошибки предыдущих моделей путем введения новых моделей, оптимизирующих целевую функцию, тем самым улучшая общую «точность» ансамбля. В качестве модели *GBDT* использовалась реализация *CatBoost* [6], которая имеет ряд особенностей: симметричные деревья, таргет энкодинг, гистограмирование.

Для обучения модели бустинга были предобработаны данные, в качестве независимых переменных использовались координаты *(x, y, z)* трека на каждой станции, а в качестве таргета (целевой переменной) использовалась координата *z* (т. к. x и y зафиксированы в модельных данных). Модель была обучена на 100 тыс. треках. Также использовалась байесовская оптимизация с использованием фреймворка *Optuna* [7] для подбора оптимальных параметров модели.

На графике (см. рис. 6) продемострирован результат работы модели: сравнение распределения данных из моделирования и предсказанных значений (координаты

вершины оси *z*). Модель успешно улавливает общую форму распределения исходных данных, что указывает на её способность к воспроизведению статистических характеристик тестового набора данных. Хотя есть схожесть в распределениях, различия в плотности распределения могут указывать на то, что в определенных областях модель менее точна.

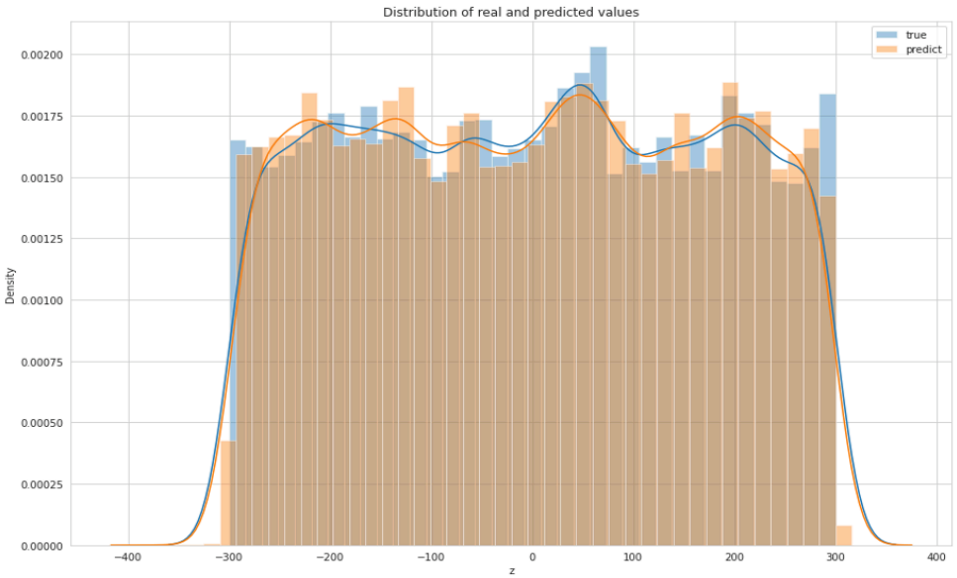

Рис. 6. Распределение исходных и спрогнозированных вершин на тестовых данных

Качество регрессии измерялось на основе метрики *MAPE* (*Mean Absolute Percentage Error*), ее показатель составил порядка 15%, это означает, что предсказанная вершина в среднем находится в 15% от фактического знаначения. Это довольно большой показатель ошибки при работе с 40 событиями внутри временной среза.

Интерпретация прогнозов

Для оценки вклада признаков в модель использовался подход *SHAP* (*SHapley Additive exPlanations*) [8], который представляет собой метод, основанный на концепциях теории игр, он позволяет интерпретировать прогнозы любой модели машинного обучения. Он использует значения Шепли — классическую концепцию теории игр — для определения степени влияния каждого признака на предсказание модели. Для вычисления значений Шепли анализируются все возможные комбинации признаков, исследуется изменение прогноза модели при добавлении или исключении конкретного признака (см. рис. 7).

Графики *SHAP* визуализируют эти влияния следующим образом:

- Ось x отображает величину вклада признака в прогнозируемое значение (влияние на целевую переменную).
- Ось y организует признаки в порядке убывания их значимости.

- Цвет указывает на значение признака для конкретного прогноза (обычно синий обозначает низкие значения, а красный — высокие).
- Толщина линий или размер точек на графике отражает плотность наблюдений, показывая, как часто встречаются различные значения признаков.

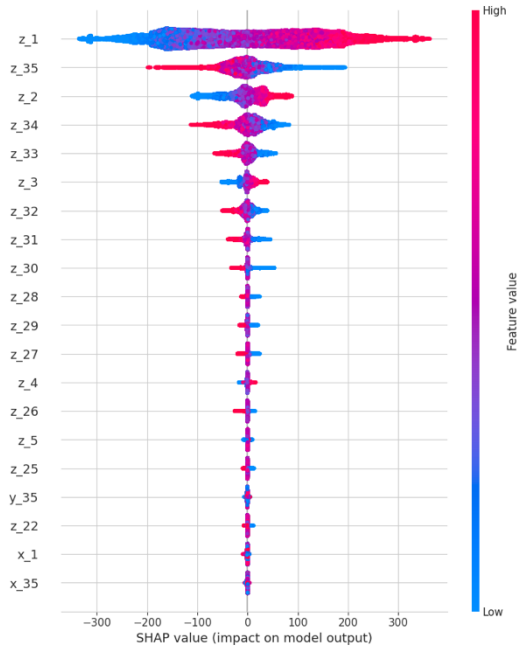

Рис 7. Интерпретация работы *GBDT*-модели используя показатель *SHAP*

Исходя из анализа графика, можно сделать вывод, что наиболее значимыми признаками являются координаты *z* в треках (по станциям).

Разделение спрогнозированных событий (кластеризация)

После предсказания вершин необходимо кластеризовать спрогнозированные вершины треков по событиям. В качестве алгоритма кластеризации использовался *K-means* с фиксированным количеством кластеров (кол-во кластеров=кол-во событий внутри среза) на временной срез.

Алгоритм *K-means* — это метод кластеризации, который группирует данные, минимизируя метрику расстояний от каждой точки до центра ближайшего ей кластера [9]. Этот метод имеет ряд недостатков:

- Работает только с фиксированным количеством кластеров;
- Чувствителен к начальному выбору центров кластера, поэтому, для «стабильных» результатов требуется запускать несколько последовательных запусков алгоритма, с разными начальными значениями центройда. В данном эксперименте использовалось 10 запусков;
- Плохо работает с несферическими кластерами;
- Чувствителен к выбросам.

В случае вершинного подхода такие недостатки не играют никакой роли, поскольку кластеризация осуществляется по одномерным векторам со спрогнозированной *z*-координатой.

Алгоритм сопоставления кластеров и лейблов

Для возможности расчета внешних метрик был разработан специализированный алгоритм сопоставления (матчинга), основанный на пересечении множеств (см. рис. 8). Данный алгоритм позволяет сопоставлять исходный лейбл события и предсказанный кластер, на основе входящих в него треков. Алгоритм реализован в форме метода, принимающего следующие входные параметры:

- *cluster_assignments*: прогнозные метки;
- *labels*: исходные метки событий.

```python
@staticmethod
def link_clusters(cluster_assignments: np.ndarray, labels: np.ndarray):

    labels_dict = defaultdict(set)
    cluster_dict = defaultdict(set)

    # add indices to sets for each label
    for i, val in enumerate(labels.tolist()):
        labels_dict[val].add(i)

    for i, val in enumerate(cluster_assignments.tolist()):
        cluster_dict[val].add(i)

    # calculate intersections and link clusters and events
    cluster_evt_dict = {}
    for cluster, cluster_set in cluster_dict.items():
        max_intersect = 0
        max_evt = None
        for evt, evt_set in labels_dict.items():
            intersect = len(cluster_set & evt_set)
            if intersect > max_intersect:
                max_intersect = intersect
                max_evt = evt
        cluster_evt_dict[cluster] = max_evt

    vfunc = np.vectorize(cluster_evt_dict.get)
    return vfunc(cluster_assignments)
```

Рис. 8 Метод сопоставления лейблов и прогнозов кластеризации

Работа метода организована следующим образом:

1. Формирование набора идентификаторов: для каждого кластера составляется список идентификаторов треков (*track_id*), которые входят в его состав;
2. Вычисление пересечений: осуществляется подсчёт попарных пересечений между множествами идентификаторов кластеров и событий;
3. Сортировка результатов: Полученные пересечения сортируются в порядке убывания для определения наибольшего совпадения;
4. Определение наилучших пар: выбираются пары кластер-событие, имеющие максимальное пересечение;

5. Устранение конфликтов: в случае, когда одно событие может быть ассоциировано с несколькими кластерами, выбор осуществляется в пользу кластера с наибольшим пересечением;
6. Присвоение меток: каждому кластеру присваивается метка на основе наиболее подходящего сопоставления с событием.

Анализ результатов

F1-score составил 0.146, что является неудовлетворительным результатом. Основной проблемой были близость вершин и перекрытие. Учитывая эти результаты, данный метод не оказался эффективным для идентификации 40 событий в срезе, что привело к поиску новых подходов для решения задачи.

В ходе разработки данного подхода было выявлено, что основной проблемой подхода предсказания вершин и дальнейшей кластеризации заключается в том, что они близки и пересекаются. Проблема хорошо проиллюстрирована на рисунке 9.

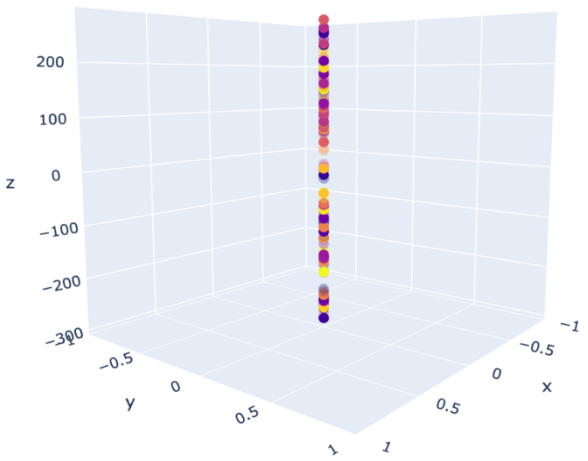

Рис. 9 Визуализация вершин для одного временного среза с 40 событиями

В реальном эксперименте *x* и *y* не будут фиксированными, как при приближенном моделировании. Это усложнит задачу, поскольку придется прогнозировать сразу 3 координаты для вершины, что в свою очередь скажется на скорости работы алгоритма. Но при этом вершины станут более разделимы, что может улучшить качество.

В рамках данного эксперимента, исходя из полученных метрик, данный подход неприменим для распутывания 40 событий во временном срезе.

Таблица. 1 Метрики для «Вершинного» подхода

	5	**10**	**40**
samples	2019	1010	253
Precision	0,864	0,355	0,291
Recall	0,886	0,491	0,224
F1-score	0,857	0,401	0,146
Accuracy	0,921	0,629	0,211

На основе таблицы метрик (строки – метрики, столбцы – кол-во событий внутри временного среза) (таблица 1), можно сделать вывод, что данный подход не применим для распутывания 40 событий во временном срезе. Это обосновывается спецификой решения, модель больше всего ошибается в близко расположенных событиях, поскольку распределение вершин – равномерное, с увеличением вершин, возрастает их плотность, что негативно влияет на разделяющую способность подхода. Поэтому был разработан другой подход, основанный на кластеризации векторов признаков.

Генерация векторов признаков

Переходя от методов восстановления вершин, таких как интерполяция и ансамблевым моделям регрессии, следующий этап разработки в данном исследовании ориентирован на более серьезную обработку событий перед применением кластеризации. Основной задачей данного подхода является обеспечение максимальной разделимости векторов признаков. Для обеспечения максимальной разделимости векторов признаков, в данном исследовании применяется метод обучения сравнению, или *metric learning* [10]. Этот подход позволяет создать пространство признаков, в котором данные, относящиеся к одному классу, группируются ближе друг к другу, в то время как данные разных классов располагаются на значительном удалении. Применение metric learning оправдано его способностью улучшить качество кластеризации, путем оптимизации расстояний между данными напрямую в процессе обучения.

Подход с использованием нейросетей предлагает значительно больше возможностей для настройки и оптимизации по сравнению с традиционными методами, такими как интерполяция и регрессия. Вариативность нейросетевых архитектур позволяет не только выбирать различные структуры и глубины сетей, но и предоставляет полную свободу в определении целевых функций, что является критически важным для точного моделирования сложных зависимостей в данных.

Кроме того, нейросети способны обрабатывать данные с минимальной потерей информации, что особенно важно при работе с большими и многомерными наборами данных, где каждый атрибут может нести важную информацию о поведении системы. В данном случае они обеспечивают «бесшовный» переход от задачи генерации векторов признаков, к разделению их по событиям, с использованием кластеризации. В подходе с прогнозирования вершины происходила «утечка» полезной информации на моменте перехода от задачи регрессии к кластеризации, поскольку рассматривались только спрогнозированные точки (вершины), уже без учета информации о треках.

Данный подход позволяет не только обрабатывать большие объемы данных, но и находить неявные взаимосвязи и закономерности, которые могут ускользнуть от более простых методов анализа. Таким образом, применение нейросетей значительно расширяет горизонты аналитических возможностей, улучшая качество и эффективность научных исследований.

В качестве конкретного решения для реализации подхода с предсказанием признаков была выбрана архитектура сиамской нейронной сети [11]. Сиамская нейросеть, состоит из трех нейросетей с одинаковыми весами, которые сходятся к главному блоку для сравнения полученных признаков (*embeddings*), функционирует как генератор векторов признаков. Архитектура и принцип работы представлены (см. рис. 10). Эти эмбеддеры преобразуют входные треки в вектора признаков (эмбеддинги) во время обучения. Затем выходные данные сети объединяются в модуль для сравнения эмбеддингов (голову). Этот модуль помогает оценить вероятность, что входные треки принадлежат одному событию, используя евклидово расстояние, косинусное расстояние или расстояние *SNR* [12].

На рисунке 10 приведена архитектура сиамской нейронной сети.

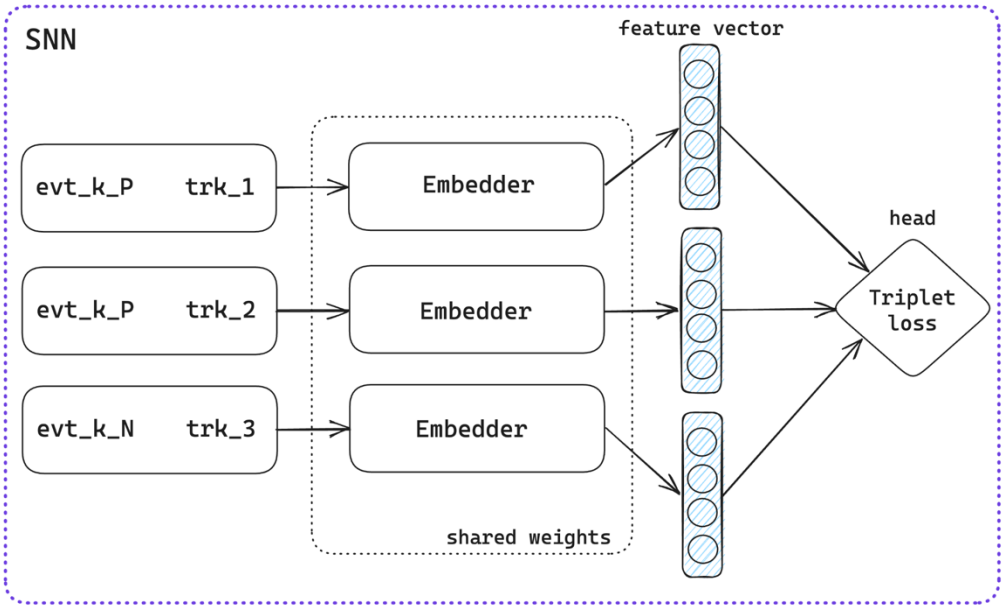

Рис. 10 Архитектура сиамской нейронной сети

Основная идея обучения этой сети заключается в использовании метода триплетной ошибки (*triplet loss)* (см. рис. 11), который направлен на минимизацию расстояния между похожими (*positive* и *anchor*) треками и максимизацию расстояния между различными (*negative*) треками. Это подразумевает переход от общей кластеризации всех треков к попарному сравнению, особенно выбирая треки, которые близки друг к другу в рамках одних и тех же событий. Таким образом, сиамская сеть обучается распознавать треки одного события как положительные примеры и треки

разных событий как отрицательные. Сеть переводит треки в латентное представление, в котором треки из разных событий становятся максимальны разделимыми. В процессе обучения сравниваются не просто пары, а тройки треков: главный трек (*anchor*), трек из того же события (*positive*) и трек из другого события (*negative*). Такое обучение позволяет нейронной сети эффективно извлекать вектора признаков для треков, улучшая разделимость в латентном пространстве данных из разных событий.

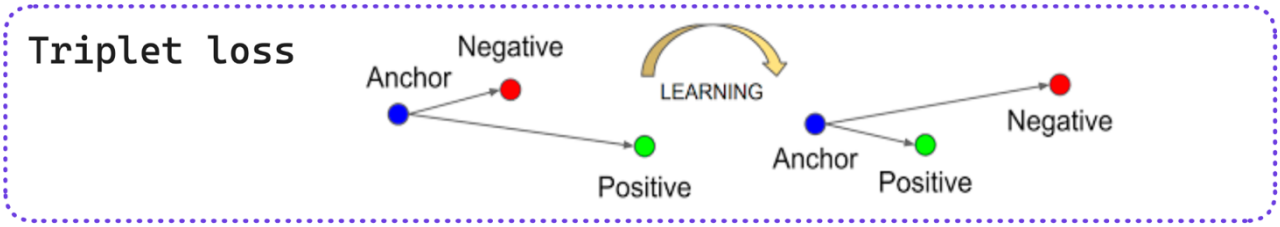

Рис. 11 Визуализация идеи *triplet loss*

Margin, *hard negative* и *semi-hard negative samples* являются регулируемыми параметрами в процессе обучения с использованием *triplet loss*, которые напрямую влияют на формирование обучающей выборки и оптимизацию модели. *Margin* устанавливает пространство разрыва между классами, позволяя модели не только различать классы, но и делать это с определенной степенью уверенности, что способствует более быстрой и качественной сходимости в процессе обучения.

Выборка *hard negative samples* формируется таким образом, чтобы включать те примеры, которые с наибольшей вероятностью будут ошибочно классифицированы моделью. Обучение на таких сложных примерах позволяет углубить сходимость и улучшить способность модели к различению классов, делая её предсказания более точными.

Semi-hard negative samples также выбираются в процессе подготовки обучающего набора данных и используются для обеспечения плавной оптимизации, избегая резких перепадов в градиентах, что способствует устойчивости и эффективности процесса обучения [10]. Их включение в обучающую выборку помогает избежать проблем с переобучением и локальными минимумами, обеспечивая более стабильное и эффективное обучение.

В совокупности эти параметры позволяют формировать обучающую выборку таким образом, чтобы максимизировать производительность модели, делая её более адаптивной и способной к обобщению на новые данные.

Модель эмбеддера

В архитектуре эмбеддера сиамской нейросети используются блоки, состоящие из последовательности нормализации и полносвязных сетей (*Feed Forward Network*, *FFN*),

что является ключевым гиперпараметром [13]. Эти блоки позволяют эффективно стабилизировать процесс обучения и улучшить сходимость, предоставляя гибкость в настройке нейросетевой архитектуры. Архитектура вдохновлена *MLP Mixer* [13] и слоями *FFN* трансформеров, при этом каждый блок нормализации расположен перед *FFN*, что гарантирует последовательное преобразование входных данных.

Входной линейный слой проектирует данные в пространство признаков, после чего применяется активация *GELU* (см. рис. 12), вносящая нелинейность. *GELU* была использована в качестве функции активации, поскольку она объединяет регуляризацию и нелинейность, аналогично dropout и *ReLU*, мультиплицируя входы на значение от 0 до 1, зависимое от значения входа. Это дает стохастический, но в то же время зависимый от входа эффект, что делает *GELU* более предпочтительной в сравнении с *ReLU* и *ELU*, как показали исследования, демонстрирующие улучшение точности на различных наборах данных [14].

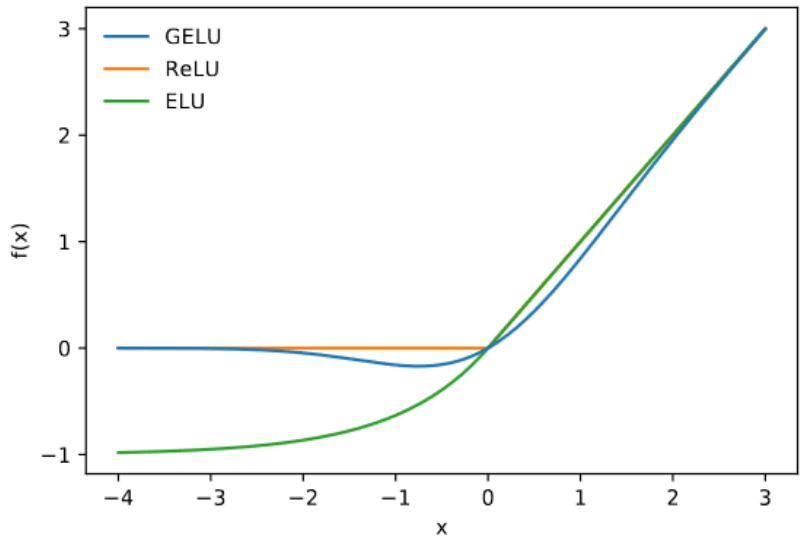

Рис 12. Графики функций активации: *GELU, ReLU, ELU*

Слои исключения (Dropout) используются для предотвращения переобучения, «выключая» случайным образом нейроны, тем самым усиливая регуляризацию сети. Остаточные связи (*residual connections*) интегрированы между блоками нормализации и *FFN*, что позволяет поддерживать градиенты в сети и способствует более глубокому обучению без деградации производительности [15] (см рис. 13).

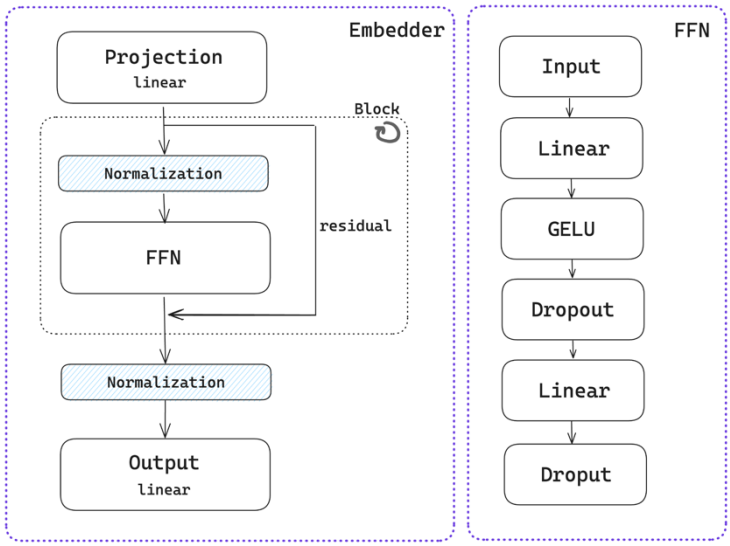

Рис 13. Графики архитектуры *Embedder*-блока

Эти компоненты работают вместе для формирования векторов признаков, или эмбеддингов, которые затем используются для сравнения и классификации входных треков. Финальный линейный слой в блоке эмбеддера генерирует выходные данные, которые подаются на головной модуль для сравнения эмбеддингов, способствуя определению степени схожести входных объектов с определенными классами.

Средства реализации

В ходе работы была разработана программная среда для апробации алгоритмов на основе библиотеки *ARIADNA*, которая специализируется на глубоком обучении и обработке данных с детекторов частиц. Основная архитектура проекта и его модули были реализованы с использованием следующих фреймворков и библиотек:

- *PyTorch* - основной инструмент для построения и тренировки нейронных сетей [16].
- *PyTorch Lightning* - используется для упрощения тренировки моделей, позволяя автоматизировать рутинные задачи тренировки [17].
- *PyTorch Metric Learning* - позволяет легко использовать и настраивать метрические функции потерь и методы выбора триплетов для обучения моделей [18].

Основные модули и классы проекта

Ниже указаны основные компоненты проекта, также приведена диаграмма классов с полями и методами (см. рис. 14)

1. *TrackEmbedder* (src/model.py)
    - Класс, который определяет архитектуру модели.
2. *SPDTimesliceTracksDataset* (src/dataset.py)

- Предназначен для генерации искусственных данных треков в детекторе, которые используются для обучения и валидации моделей. Это обеспечивает возможность тренировки модели в контролируемых условиях с точно настраиваемыми параметрами.

3. *TripletTracksEmbedder* (src/training.py)
   - Реализует обучение модели с использованием триплетного подхода. Применяет различные стратегии для выбора триплетов и метрические функции, что позволяет оптимизировать обучение и повысить качество векторных представлений.и.
4. *ModelEvaluator* (src/evaluation.py)
   - Предназначен для оценки качества и эффективности обученной модели на тестовых данных. Включает функционал для расчета различных метрик качества и визуализации векторных представлений, что позволяет анализировать и улучшать модель
5. *Clustering* (src/clustering.py)
   - Выполняет кластеризацию векторных представлений и матчинг спрогнозированных кластеров с истинными событиями.

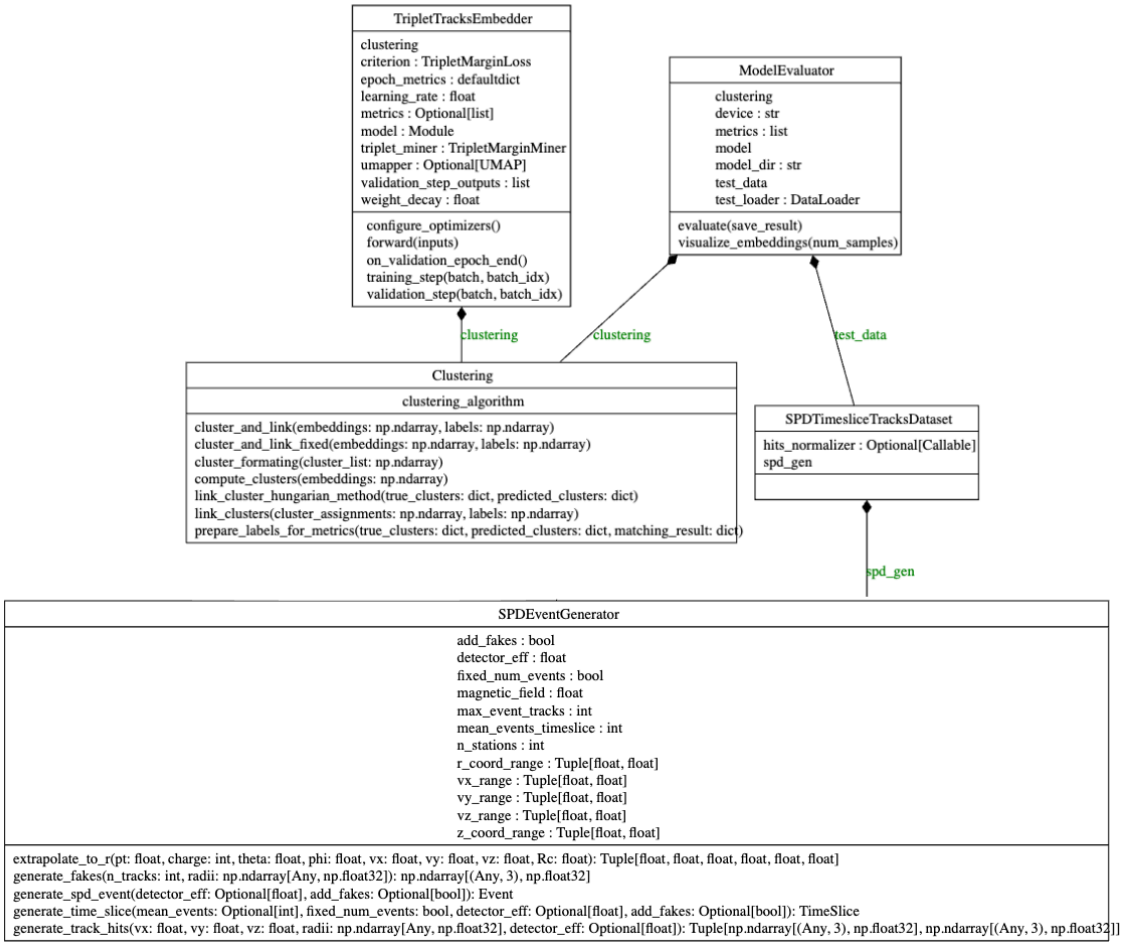

Рис. 14 Архитектура основных компонентов проекта

Конфигурация экспериментов

Конфигурационные файлы играют важную роль в обучении моделей, позволяя удобно настраивать и фиксировать параметры экспериментов. Использование таких файлов обеспечивает повторяемость экспериментов, упрощает тестирование различных гиперпараметров и ускоряет процесс разработки, избавляя от необходимости вносить изменения непосредственно в код.

В данном проекте для управления конфигурациями используется библиотека *gin-config* [19], которая позволяет определять параметры и зависимости в отдельных конфигурационных файлах. Это позволяет изменять параметры модели, процедуры обучения и даже структуру данных без изменения основного кода программы.

В работе используется следующий конфигурационный файл, в котором задается информация об архитектуре модели, параметры обучения, методы кластеризации, а также метрики и другие параметры (см. рис. 15).

```
experiment.model = @TrackEmbedder()
experiment.logging_dir = "experiment_logs"
experiment.random_seed = 42
experiment.num_epochs = 100
experiment.train_samples= 10
experiment.test_samples = 10
experiment.detector_efficiency = 1.0
experiment.learning_rate = 0.0001
experiment.weight_decay = 1e-2
experiment.distance = %DistanceType.euclidean_distance
experiment.type_of_triplets = %TripletType.semihard
experiment.triplet_margin = 0.1
experiment.hits_normalizer = @ConstraintsNormalizer()
experiment.num_workers = 4
experiment.pin_memory = True

experiment.metrics = [
    @SilhouetteScoreMetric(),
    @DaviesBouldinScoreMetric(),
    @CalinskiHarabaszScoreMetric(),
    @AccuracyScoreMetric(),
    @F1ScoreMetric(),
    @PrecisionScoreMetric(),
    @RecallScoreMetric(),
]
#experiment.resume_from_checkpoint =

### model ###
TrackEmbedder.n_blocks = 2
TrackEmbedder.output_dim = 32

### clustering model ###
clustering_algorithm.class_ = 'AgglomerativeClustering'
clustering_algorithm.module = 'sklearn.cluster'
clustering_algorithm.distance_threshold = 2.5
clustering_algorithm.n_clusters = None
# hits normalizer
ConstraintsNormalizer.x_coord_range = (-851., 851.)
ConstraintsNormalizer.y_coord_range = (-851., 851.)
ConstraintsNormalizer.z_coord_range = (-2386., 2386.)
```

Рис. 15 Конфигурационный файл проекта

Использование конфигурационных файлов типа *train.cfg* делает процесс настройки экспериментов более удобным и модульным, что способствует более быстрой и эффективной разработке исследовательских и промышленных проектов в области машинного обучения.

Также весь процесс обучения сопровождается логированием результатов с использованием *TensorBoard* [20], логируются все этапы обучения и тестирования модели.

Обучение и тестирование модели

Модель обучалась на 10 тысячах временных срезов (количество событий в каждом срезе фиксировано и составляет 40). Тестовая выборка состоит из 1 тысячи временных срезов. Лучшие параметры для модели *TripletTracksEmbedder*: *n_blocks*=5, *output_dim*=32. Сходимость достигается на 80 эпохах (см. рис. 16). Время обучения на *GPU Tesla V100* составляет 9 часов.

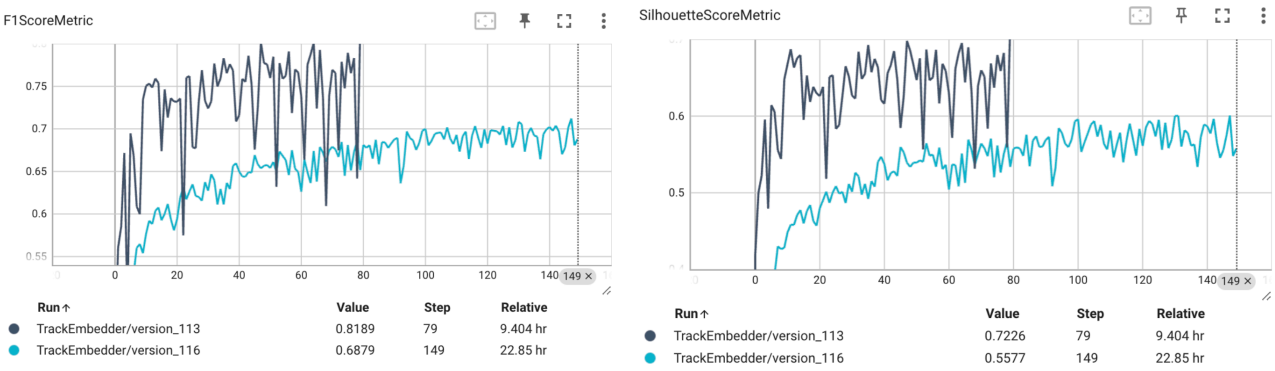

Рис. 16. Визуализация процесса обучения на основе метрик *F1-Score* и *Silhouette score*

Модель обучалась для 2 состояний:

- Серый график – треки не содержат пропусков хитов;
- Синий график – треки содержат пропуски хитов.

Как можно заметить, пропуски негативно влияют на модель, в реальных данных пропуски по станциям присутствуют. На данный момент они обрабатываются константно – заполняются 0.

После получения вектора признаков выполняется кластеризация и рассчитываются внутренние метрики (классификации). В качестве алгоритма кластеризации использовался *K-means* (число кластеров = число событий во временном срезе). Кластеризация применяется к каждому срезу. В качестве мер расстояния между объектами (спрогнозированными векторами) использовались (*cosine* и *euclidean*)

Но в реальном эксперименте мы не знаем точного количества событий во временном срезе. В работе используется фиксированное число кластеров для разработки и тестирования алгоритмов.

Результатом применения обученной модели к трекам из временного среза является визуализация, для которой был применен алгоритм *UMAP* для уменьшения размерности исходных 32-мерных векторов до *2D* [21]. Треки из различных событий внутри одного временного среза выделены разными цветами, что позволяет демонстрировать их разделение и группировку (см. рис. 17).

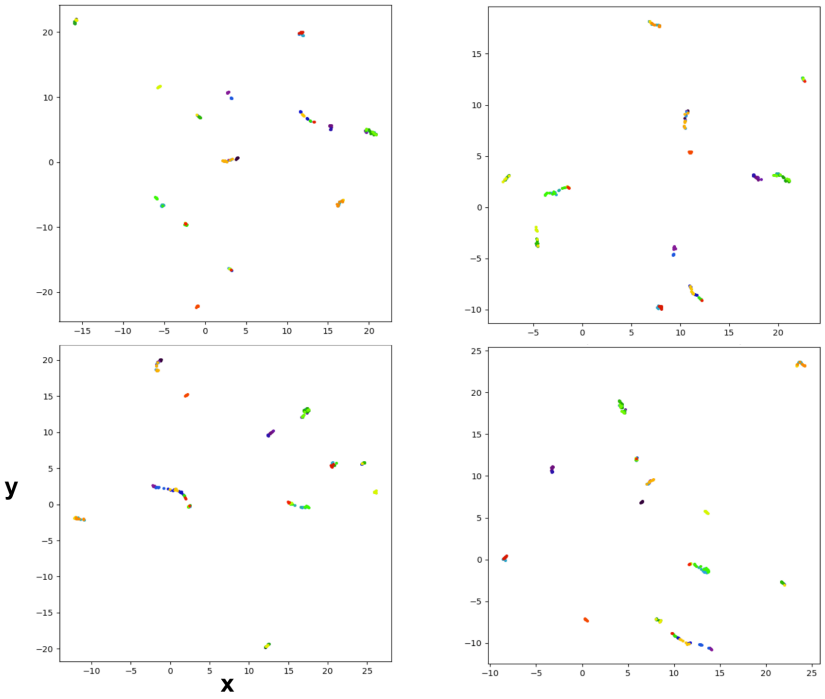

Рис. 17. Визуализация *embedings* полученных на тестовых данных, в ходе прогноза модели

Эта процедура уменьшения размерности и последующей визуализации служит инструментом для оценки качества кластеризации и понимания структуры данных, позволяя лучше интерпретировать, как модель различает разные события на основе извлеченных признаков.

Исходя из представленных метрик (таблица 2), можно сделать вывод о том, что подход, основанный на использовании эмбеддингов, значительно превосходит подход, основанный на вершинах. *Embedding* подход демонстрирует более высокие значения по всем метрикам: *Silhouette score*, который измеряет согласованность внутри кластеров, значительно выше (0,725 против 0,291), что указывает на лучшее разделение кластеров. Индекс *Davies Bouldin*, который оценивает среднее «расстояние» между кластерами, также лучше для эмбеддинг-базированного подхода (0,655 против 0,124), что подтверждает более четкое разграничение кластеров. Это означает, что удалось получить латентное представление треков, в котором они более разделимы, чем просто по координате вершины.

Таблица 2. Сравнение метрик для двух подходов: вершинного и кластеризация *embeddings*

	Подход	
Метрика	***Vertex based***	***Embedding based***
Silhouette	0,291	**0,725**
Davies bouldin	0,124	**0,655**
Precision	0,291	**0,811**
Recall	0,124	**0,843**
F1-score	0,046	**0,818**
Accuracy	0,211	**0,895**

Точность (*Precision*), полнота (*Recall*), *F1-score* и *Accuracy* значительно выше для *embedding* подхода, что демонстрирует его способность более эффективно классифицировать и распознавать различные события.

Таким образом, подход, основанный на *embeddings*, не только обеспечивает более высокую точность и качество кластеризации, но также, с учетом современных вычислительных ресурсов, позволяет достигать этих результатов быстрее, что делает его предпочтительным выбором для решения задач в данном контексте.

Обработка нефиксированного числа событий

Поскольку в условиях реального эксперимента количество событий внутри временного среза не будет зафиксировано, необходимо модифицировать разработанный подход, который будет обрабатывать такие случаи.

Общая архитектура расчета метрик приведена на рисунке 18.

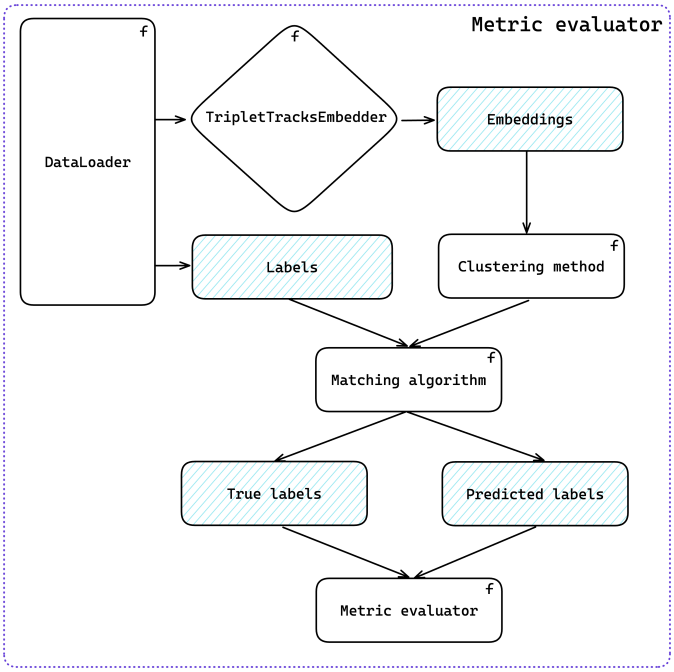

Рис. 18 Схема алгоритма расчета внешних метрик кластеризации

Для расчета метрики, при нефиксированном количестве событий, был применен новый алгоритм для кластеризации и разработан метод сопоставления треков, описанный ниже:

1. Загрузка данных: используется *Dataloader* для загрузки данных.
2. Извлечение эмбеддингов: *TripletTracksEmbedder* принимает на вход исходные треки и вычисляет эмбеддинги, которые будут использоваться для кластеризации.
3. Кластеризация: кластеризация эмбеддингов выполняется с помощью указанного метода кластеризации (*Clustering method*), который может быть *KMeans*, *DBSCAN*, *AgglomerativeClustering* и т. д.

4. Сопоставление меток: сопоставление кластеров с истинными метками выполняется с использованием алгоритма сопоставления. Два метода предлагаются для этого: *link_clusters* и *link_cluster_hungarian_method* (см. рис. 19).
5. Вычисление метрик: для оценки качества кластеризации используются различные метрики, такие как *SilhouetteScore*, *DaviesBouldinScore*, *CalinskiHarabaszScore*, *Accuracy*, *F1Score*, *PrecisionScore* и *RecallScore*.

Методы кластеризации для нефиксированного количества событий

Если при подходе с фиксированным количеством событий внутри временного среза использовался *K-means* в качестве метода кластеризации, то теперь его использовать невозможно, поскольку он требует явного указание количества кластеров. Поэтому в качестве новых методов кластеризации были выбраны 2 метода:

1. Агломеративная кластеризация (*AgglomerativeClustering*). Метод основан на иерархическом подходе. Каждый объект рассматривается как отдельный кластер, затем на каждом шаге объединяет два наиболее близких кластера, пока все объекты не окажутся в одном кластере или не будет достигнуто фиксированное количество кластеров. Поэтому этот метод может обрабатывать произвольное количество кластеров [22]. Основные гиперпараметры:
   - Метод вычисления расстояния между кластерами (например, *single linkage*, *complete linkage*, *average linkage*);
   - Порог расстояния для остановки объединения кластеров или желаемое количество кластеров.
2. *DBSCAN* (*Density-Based Spatial Clustering of Applications with Noise*). Метод, который основан на плотности данных. Он группирует точки, которые расположены близко друг к другу, образуя кластеры, а также помечает точки, которые находятся в минорных областях (то есть, имеющие менее определённого количества соседей в радиусе), как шум. *DBSCAN* не требует предварительного указания количества кластеров и может автоматически обнаруживать кластеры произвольной формы и размера [23]. Основные гиперпараметры:
   - *Eps* (ε): максимальное расстояние между двумя точками для того, чтобы одна считалась соседкой другой.
   - *MinPts*: минимальное количество точек, необходимых для формирования плотного региона (кластера).

Поскольку методы имеют набор чувствительных гиперпараметров, они были подобраны с использованием байесовской оптимизации на валидационной выборке.

Метод сопоставления (матчинга)

Для расчета внешних метрик кластеризации важен метод сопоставления прогнозных кластеров и исходных лейблов. Поскольку прошлый метод не адаптирован под работы с нефиксированным количеством кластеров, был разработан новый, улучшенный метод. Метод *link_cluster_hungarian_method* осуществляет матчинг кластеров с использованием венгерского алгоритма (см. рис. 19) [24].

```python
@staticmethod
def link_cluster_hungarian_method(true_clusters: dict, predicted_clusters: dict):

    intersection_matrix = np.zeros((len(true_clusters), len(predicted_clusters)))

    for i, true_cluster in true_clusters.items():
        for j, predicted_cluster in predicted_clusters.items():
            intersection_matrix[i, j] = len(set(true_cluster) & set(predicted_cluster))

    cost_matrix = np.max(intersection_matrix) - intersection_matrix
    row_ind, col_ind = linear_sum_assignment(cost_matrix)
    matching_result = {true_idx: pred_idx for true_idx, pred_idx in zip(row_ind, col_ind)}

    return matching_result
```

Рис. 19 Метод сопоставления кластеров и лейблов на основе Венгерского алгоритма

Метод принимает два словаря: *true_clusters* – исходные лейблы и *predicted_clusters* – спрогнозированные метки кластеров.

Данный метод состоит из 4 основных этапов:

1. Создается матрица пересечений intersection_matrix, где каждый элемент (*i, j*) представляет количество общих элементов между *i*-м истинным и *j*-м предсказанным кластерами.
2. На основе матрицы пересечений строится стоимостная матрица cost_matrix.
3. С помощью функции *linear_sum_assignment* решается задача о назначениях, минимизирующая общую стоимость, что позволяет найти оптимальное соответствие между истинными и предсказанными кластерами.
4. Возвращается словарь *matching_result*, который содержит пары индексов истинных и предсказанных кластеров, соответствующие оптимальному матчингу.

После чего, на основе сопоставленных кластеров и лейблах рассчитываются метрики.

Обучение модели

Поскольку для экспериментов был использован обновленный, более физический генератор, который включает в себя неопределенное количество событий внутри временного среза, распределенное по Пуассону, и нефиксированные координаты вершин для *x* и *y*, ранее использованная модель была переобучена с новым генератором, с

применением новых методов для расчета метрик (см. рис. 20). В качестве метода кластеризации использовался *AgglomerativeClustering*, поскольку он показал наилучшие результаты, на валидационной выборке, в сравнении с *DBSCAN*.

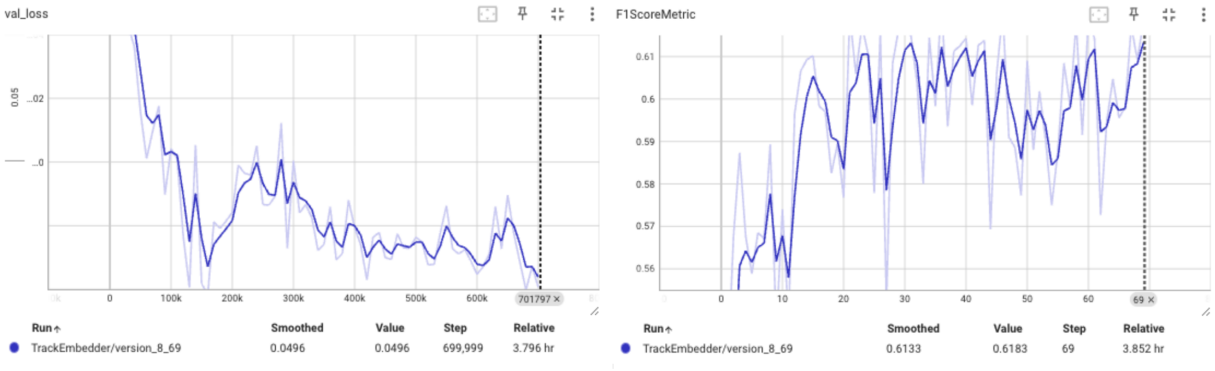

Рис. 20. Визуализация процесса обучения для неопределенного количества событий

Хоть в новом генераторе, в среднем и наблюдается меньшее количество событий, как можно наблюдать на рисунке 20, показатель *F1-score* ~0.61.

Таблица 3. Метрики полученный для нефиксированного подхода

Метрика	***Embedding based (unfixed)***
Silhouette	0,342
Davies bouldin	0,922
Precision	0,574
Recall	0,651
F1-score	0,607
Accuracy	0,587

Метрики при нефиксированном подходе оказались хуже на ~20%, в сравнении с фиксированным (*F1-score* был ~0.81) (таблица 3).

Оценка скорости обработки

Как уже упоминалось, в контексте будущего эксперимента *SPD NICA*, где предполагается обработка огромного объема данных со скоростью 20 Гб/с и частотой столкновений 3 МГц, замеры скорости работы разрабатываемого подхода являются критически важными. Оптимизация времени обработки каждого временного среза, содержащего сотни треков и хитов, напрямую влияет на возможность реализации онлайн-анализа данных и обеспечения своевременной и точной интерпретации событий в условиях высокой интенсивности входящего потока информации [1].

Тестирование скорости обработки данных было проведено с использованием *BATCH_SIZE*, равным одному временному срезу, который содержит случайное количество треков и фиксированное число событий — 40 (*clustering_fixed*). В ходе

испытаний были получены следующие результаты времени обработки одного временного среза на разных устройствах (сводная таблица 4):

1. На *CPU* (*Apple M1 Pro* с 10 ядрами), среднее время работы *embedder* составило 0.0177 секунды со стандартным отклонением ±0.0059, в то время как время кластеризации *K-means* составило 0.1884 секунды (±0.0919).
2. На *GPU* (*Apple M1 Pro* с 10 ядрами), среднее время работы *embedder* увеличилось до 0.0279 секунды (±0.0099), а кластеризация *K-means* заняла 0.2010 секунды (±0.1031). Увеличение скорости обработки на GPU, может быть связано с неверной ее настройкой.
3. Сочетание *GPU Tesla V100* с *CPU* показало наилучшее среднее время для *embedder*, равное 0.0076 секунды (±0.0015), и значительно ускорило процесс кластеризации до 0.1126 секунды (±0.0053).

Таким образом, наиболее эффективной комбинацией для задачи оказалось использование *GPU Tesla V100* вместе с *CPU (Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz*). Однако стоит отметить, что процесс кластеризации с помощью K-means, который выполняется на *CPU*, является "узким местом" в рабочем процессе, для фиксированного числа событий.

Также замеры скорости обработки были произведены для нефиксированного подхода. В качестве алгоритма кластеризации использовался *AgglomerativeClustering* метод. Опираясь на сравнение скорости кластеризации для фиксированного и нефиксированного похода, было достигнуто практически 100 кратное ускорение этапа кластеризации. Такое ускорение достигается из-за того, что *K-means* используется несколько инициализаций, когда агломеративная кластеризация, только 1. Полученные показатели, теоретически, могут обеспечить требуемую скорость обработки.

Таблица 4. Замер времени на каждом этапе, для разных устройств

Устройство	***embedder (s)***	***clustering_fixed (s)***	***clustering (s)***
CPU (Apple m1pro 10 cores)	0.0177 (±0.0059)	0.1884 (±0.0919)	0.0019 (±0.0006)
GPU (Apple m1pro 10 cores)	0.0279 (±0.0099)	0.2010 (±0.1031)	-
GPU Tesla V100 + CPU	0.0076 (±0.0015)	0.1126 (±0.0053)	-

Для ускорения обработки данных предлагаются следующие пути оптимизации:

- Перенос алгоритма кластеризации на *GPU*, чтобы воспользоваться более высокой производительностью и параллельной обработкой данных.

- Использование библиотеки кластеризации, оптимизированной для большей скорости выполнения, возможно, с применением аппаратно-ускоренных вычислений.
- Оптимизация архитектуры модели, включая пересмотр параметров и слоев, для сокращения времени обработки без потери точности.
- Применение *ONNX/TensorRT* [25] для ускорения инференса модели за счет конвертации в оптимизированный формат, который лучше работает на специализированных устройствах.
- Использование *TorchScript* [26] для компиляции и оптимизации *PyTorch* моделей, что позволяет улучшить производительность на различных платформах.

Такие изменения могут существенно повысить скорость обработки, что критически важно для реализации анализа данных в реальном времени в условиях высокой интенсивности потока регистрируемых сигналов от детектора, как это предполагается в эксперименте *SPD NICA*.

Заключение

В ходе работы были получены следующие результаты:

- После изучения описаний детекторов и процедур съема данных планируемого эксперимента *SPD* мегапроекта *NICA* были разработаны программы моделирования данных *SPD*;
- Определены и разработаны программы для вычисления метрик оценки качества решаемой задачи;
- Разработана программная среда для апробации алгоритмов на основе библиотеки *ARIADNA*;
- Реализован первый подход для предсказания вершины события и разработаны оценки качества работы фильтра;
- Реализован конвейер для распутывания событий внутри временного среза для этого подхода. Однако этот подход оказался неприменим при большом количестве событий во временном срезе;
- Реализован альтернативный подход к распутыванию, основанный на кластеризации векторов признаков, полученных с использованием сиамской архитектуры нейронной сети с применением триплетной функции потерь, который показал лучшее качество для зафиксированного количества событий внутри временного среза с *F1-score* ~ 0.82;
- Также разработан подход для обработки нефиксированного количества событий внутри временного среза, который продемонстрировал качество *F1-score* ~ 0.61 (таблица 5). При этом данный подход ускоряет скорость обработки кластеризации в 100 раз по сравнению с фиксированным подходом.

В ходе исследования были достигнуты следующие результаты по метрикам оценки качества распутывания фиксированного и нефиксированного количества событий (таблица 5).

Таблица 5. Финальные метрики для фиксированного и нефиксированного подходов

	Подход	
Метрика	***Embedding based (fixed)***	***Embedding based (unfixed)***
Silhouette	**0,725**	0,342
Davies bouldin	**0,655**	0,922
Precision	**0,811**	0,574
Recall	**0,843**	0,651
F1-score	**0,818**	0,607
Accuracy	**0,895**	0,587

Следует иметь в виду, что на настоящем этапе разработки экспериментальной установки *SPD* проведенное исследование носит предварительный характер. В частности, результаты ВКР базировались в основном на упрощенной имитационной модели данных *SPD* и в дальнейшем будут неизбежно подлежать доработке как с учетом замеченных слабых мест алгоритмов, предложенных в ВКР, так и новой информации об устройстве сооружаемых трековых детекторов *SPD* и темпов поступления данных, которая будет поступать в ходе разработки экспериментальной установки.

Полученные положительные результаты были доложены на конференции *GRID*-2023 в ОИЯИ и на семинаре «Применение методов машинного обучения на комплексе *NICA*» в СПбГУ в августе 2023. По результатам работы опубликована статья в журнале "*Physics of Particles and Nucl*ei" [27].

Расчеты проводились на гетерогенной вычислительной платформе *HybriLIT* (ЛИТ, ОИЯИ). Исследование выполнено при финансовой поддержке Российского научного фонда № 22–12-00109.

Список литературы

1. The SPD Collaboration. Conceptual design of the Spin Physics Detector. — 2022. — Feb. — [Электронный ресурс] — Электрон. текст. — 2022. — Режим доступа: https://arxiv.org/abs/2102.00442, свободный.

2. D. Rusov et al. Particle track reconstruction at scale: online tracking with PyTorch [Электронный ресурс] — Электрон. текст. — 2023. — Режим доступа: http://omega.sp.susu.ru/pavt2023/short/057.pdf, свободный.

3. Silicon Tracking System of the BM@N Experiment: Technical Design Report. — Dubna: JINR, 2020. — 101 p. — ISBN 978-5-9530-0541-8.

4. Julio-Omar. Evaluation Metrics for Unsupervised Learning Algorithms. — 2019. — May. — [Электронный ресурс] — Электрон. текст. — 2019. — Режим доступа: https://arxiv.org/abs/1905.05667, свободный.

5. Robust-GBDT: A Novel Gradient Boosting Model for Noise-Robust Classification — 2023. — October. — [Электронный ресурс] — Электрон. текст. — 2023. — Режим доступа: https://arxiv.org/abs/2310.05067, свободный (дата обращения: 14.05.2024)

6. CatBoost [Электронный ресурс] — Электрон. текст. — 2023. — Режим доступа: https://arxiv.org/abs/1810.11363, свободный.

7. Optuna: A Next-generation Hyperparameter Optimization Framework — 2019. — [Электронный ресурс] — Электрон. текст. — 2019. — Режим доступа: https://arxiv.org/abs/1907.10902, свободный.

8. A Unified Approach to Interpreting Model Predictions — 2017. — [Электронный ресурс] — Электрон. текст. — 2017. — Режим доступа: https://arxiv.org/abs/1705.07874, свободный.

9. Алгоритм K-means [Электронный ресурс] — Электрон. текст. — 2020. — Режим доступа: https://theory.stanford.edu/~sergei/papers/kMeansPP-soda.pdf, свободный.

10. Metric Learning [Электронный ресурс] — Электрон. текст. — 2020. — Режим доступа: http://contrib.scikit-learn.org/metric-learn/, свободный.

11. Архитектура сиамской нейронной сети [Электронный ресурс] — Электрон. текст. — 2020. — Режим доступа: https://nac.spl.harvard.edu/publications/siamese-neural-networks-continuous-disease-severity-evaluation-and-change-detection, свободный.

12. SNR for metric learning [Электронный ресурс] — Электрон. текст. — 2019. — Режим доступа: https://ecs.syr.edu/~jsh/SNR-distance-metric.pdf, свободный.

13. MLP-Mixer: An all-MLP Architecture for Vision [Электронный ресурс] — Электрон. текст. — 2023. — Режим доступа: https://arxiv.org/abs/2105.01601, свободный.

14. GELU (Gaussian Error Linear Units) [Электронный ресурс] — Электрон. текст. — 2016. — Режим доступа: https://arxiv.org/abs/1606.08415, свободный.

15. Residual Connections [Электронный ресурс] — Электрон. текст. — 2015. — Режим доступа: https://arxiv.org/abs/1512.03385, свободный.

16. PyTorch [Электронный ресурс] — Электрон. текст. — 2023. — Режим доступа: https://pytorch.org/, свободный.

17. PyTorch Lightning [Электронный ресурс] — Электрон. текст. — 2023. — Режим доступа: https://www.pytorchlightning.ai/, свободный.

18. PyTorch Metric Learning [Электронный ресурс] — Электрон. текст. — 2023. — Режим доступа: https://kevinmusgrave.github.io/pytorch-metric-learning/, свободный

19. Gin-config [Электронный ресурс] — Электрон. текст. — 2023. — Режим доступа: https://google.github.io/gin-config/, свободный.

20. TensorBoard [Электронный ресурс] — Электрон. текст. — 2023. — Режим доступа: https://www.tensorflow.org/tensorboard, свободный (дата обращения: 14.05.2024)

21. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction [Электронный ресурс] — Электрон. текст. — 2018. — Режим доступа: https://arxiv.org/abs/1802.03426, свободный.

22. Agglomerative Clustering [Электронный ресурс] — Электрон. текст. — 2011. — Режим доступа: https://arxiv.org/abs/1109.2378, свободный.

23. DBSCAN++ [Электронный ресурс] — Электрон. текст. — 2018. — Режим доступа: https://arxiv.org/abs/1810.13105, свободный.

24. Kuhn, H.W. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly. 2(1-2), 83–97 (1955).

25. ONNXRUNTIME [Электронный ресурс] — Электрон. текст. — 2023. — Режим доступа: https://onnxruntime.ai/, свободный.

26. *TorchScript* [Электронный ресурс] — Электрон. текст. — 2023. — Режим доступа: https://pytorch.org/docs/stable/jit.html, свободный.

27. Borisov M., Goncharov P., Ososkov G., Rusov D.: Unraveling Time Slices of Events in the SPD Experiment. Physics of Particles and Nuclei, 2024, Vol. 55, No. 3, pp. 453–455. Pleiades Publishing, Ltd., 2024.