Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет «Высшая школа экономики»

Факультет Санкт-Петербургская школа физико-математических и компьютерных наук

Основная образовательная программа

«Анализ больших данных в бизнесе, экономике и обществе»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему

«Построение генеративной нейронной сети для симуляции электромагнитных ливней»

Выполнила студентка группы МБД171, 2 курса,
Демидова Анастасия Анатольевна

Научный руководитель: к.ф.-м.н., доцент
Устюжанин Андрей Евгеньевич

Санкт-Петербург 2019

Аннотация

Моделирование и анализ данных в настоящее время является неотъемлемой частью при проведении экспериментов в физике высоких энергий. Моделирование Монте-Карло используется для интерпретации результатов экспериментов и оценки эффективности, однако является наиболее вычислительно затратной частью при проведении эксперимента. Одной из самых высокозатратных задач является симуляция электромагнитных ливней в ядерных фотоэмульсионных детекторах. Подобные трековые детекторы в физике высоких энергий используются для обнаружения и измерения энергии частиц.

В данной работе предложен альтернативный метод для симуляции электромагнитных ливней, основанный на генеративных нейронных сетях. Электромагнитный ливень можно представить в виде графа, чтобы сохранить историю связи между треками. В связи с этим задача генерации электромагнитных ливней состоит из двух частей: генерации структуры графа и генерации характеристик вершин. Для генерации структуры графа используется подход GrphRNN, который моделирует граф с помощью рекуррентной сети как последовательность вершин и последовательность ребер для каждой новой вершины. Для генерации признаков вершин графа используется генеративно состязательная сеть, где архитектура генератора и дискриминатора состоит из графовых сверточных сетей.

Предложенный подход позволяет генерировать электромагнитные ливни похожие по структуре на реальные ливни, но плохо улавливает ветвление ливня.

Оглавление

1 Введение

OPERA (Oscillation Project with Emulsion-tRacking Apparatus) - международный эксперимент в области физики высоких частиц. Его основной целью было изучение нейтринных осцилляций в канале $\nu_\mu \rightarrow \nu_\tau$. Эти исследования, проведенные различными экспериментами, привели к подтверждению нейтринных осцилляций и Нобелевской премии 2015 года. . В ходе эксперимента OPERA нейтринный луч был отправлен под горными массивами из ЦЕРНа (Женева, Швейцария) в подземную лабораторию Гран-Сассо (Италия), где находится детектор OPERA. Путь в 732 километра от ЦЕРН до лаборатории Гран-Сассо частица пролетает за 3 миллисекунды [1].

OPERA имеет огромный детектор, в котором используется специальный вид электромагнитных сэмплирующих калориметров на основе ядерных фотоэмульсий, которые также имеют название «эмульсионные кирпичи» (ECC bricks). Детектор содержит около 150 тысяч кирпичей 8,5 кг каждый. [2].

Рисунок 1. Эмульсионный кирпич

Один кирпич состоит из 56 слоев свинца 67 пластиковых подложек с нанесенными с двух сторон ядерными фотографическими эмульсиями.

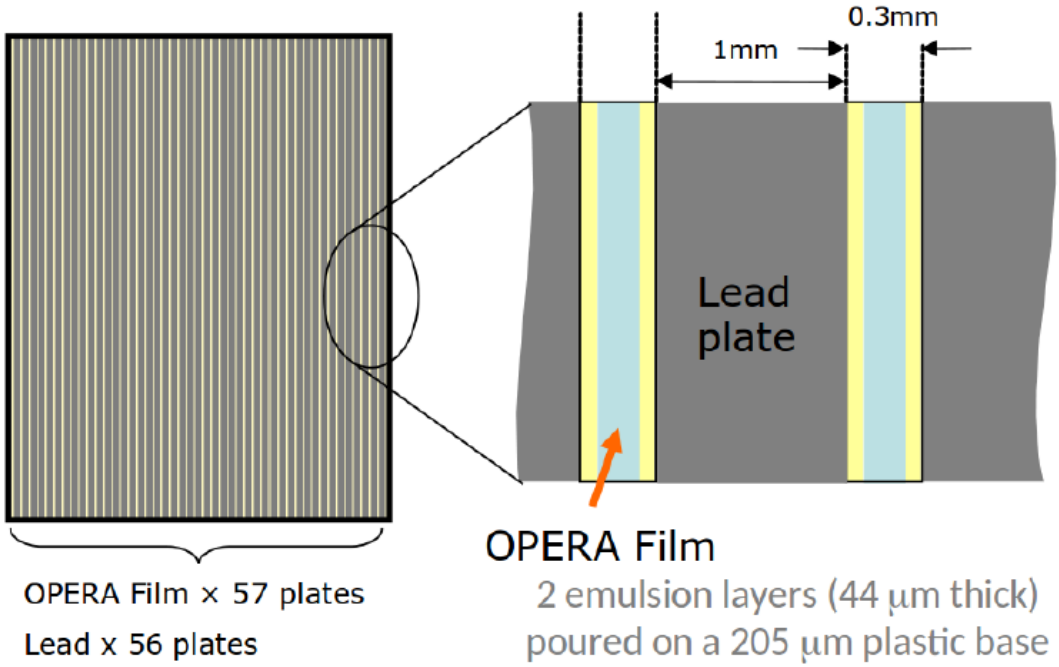


Рисунок 2 Эмульсионный кирпич в разрезе

Однако, в качестве результата наблюдается не само нейтрино, а продукты его распада. При прохождении продуктов взаимодействия через эмульсионный кирпич формируется электромагнитный ливень. По электромагнитному ливню можно восстановить характеристики частицы: энергию частицы, точку распада и направление движения частицы.

Эксперимент OPERA в значительной степени основан на детальном моделировании и анализе данных. Моделирование электромагнитного ливня используются для интерпретации результатов текущих экспериментов и оценки эффективности новых, включая модернизацию детекторов. Полное моделирование ливней в детекторе является наиболее вычислительно затратной частью всего процесса моделирования, и одна итерация может занимать несколько минут на современных распределенных высокопроизводительных платформах.

В настоящее время полное моделирование Монте-Карло в физике высоких энергий занимает 50-70% доступных вычислительных ресурсов для экспериментов, что эквивалентно миллиардам CPU часов в год [3]. И

симуляция электромагнитных ливней одна из наиболее вычислительно затратных задач, поэтому разработка альтернативного более быстрого подхода для симуляции электромагнитных ливней позволит в значительной мере сократить затраты на моделирование.

Цель данного исследования состоит в том, чтобы предложить модель глубокого обучения, чтобы обеспечить высокоточное быстрое моделирование электромагнитных ливней в эмульсионном кирпиче. Электромагнитный ливень естественным образом можно представить в виде графа, поэтому задача генерации ливней сводится к двум основным задачам: генерации графа и генерации характеристик вершин.

Для достижения данной цели были поставлены следующе задачи:

- получение из сырых данных ориентированный граф;
- изучение существующих методов для генерации графа;
- построение модели генерации данных на основе рекуррентных нейронных сетей;
- построение модели для генерации признаков вершин на основе GAN;
- валидация полученных результатов;

Данная работа имеет следующую структуру: раздел 2 приводит обзор литературы по генерации графов, раздел 3 описывает предложенный метод для генерации электромагнитных ливней, раздел 4 содержит результаты численных экспериментов, а раздел 5 завершает работу заключением и перспективами дальнейшего развития.

2 Обзор Литературы

На данный момент наиболее удобным и подходом методом для симуляции электромагнитных ливней является моделирование переноса излучения по методу Монте-Карло, когда все взаимодействия с веществом моделируются в последовательном порядке. Существует несколько работ по симуляции электромагнитных ливней методом Монте-Карло.

Во-первых, быстрая параметризация [4]. В ней авторы предлагают выразить ливень тремя функциями плотности, описывающими продольное, поперечное и азимутальное распределение энергии независимо друг от друга. Этот подход опирается на предположение о том, что этих распределений достаточно для описания пространственного распределения энергии. Качество моделирования зависит от качества параметризации ливня. Более того, было показано, что данный подход лучше всего подходит для частиц высокой энергии и менее выразителен для частиц средней и низкой энергии.

Во-вторых, группа ATLAS в 2008 году предложила подход «замороженных ливней» [5]. Основная идея состоит в том, чтобы использовать сохраненную на диск библиотеку с предварительно смоделированными ливнями, то есть перед использованием генерируется база данных ливней для разных направлений, энергий и позиций. Во время симуляции, когда частица попадает в детектор ЕМ-ливня, симулятор проверяет условия для использования библиотеки частиц: тип частицы и энергию. При соблюдении условий ливень из библиотеки выбирается. Этот метод лучше всего подходит для частиц со средней энергией.

В-третьих, метод «убийства» используется, когда мы имеем дело с частицами с низкой энергией. В этом случае частица просто игнорируется из-за низкого влияния в дальнейшем анализе, где используется моделирование

Монте-Карло. Очевидно, что этот метод применяется только для частиц с низкой энергией.

Эти подходы могут быть объединены [5], то есть частицы высокой энергии (> 10 GeV) моделируются методом быстрой параметризацией, средние энергии (<1 GeV) с использованием библиотеки с ливням и частицы с низкой энергией убиваются.

На данный момент не существует работ по генерации электромагнитных ливней с помощью нейронных сетей. Однако существует ряд работ, в которых решается задача генерации графа с помощью нейронных сетей.

Традиционные генеративные модели для графов (например, модель Барабаши-Альберта, графы Кронекера, экспоненциальные случайные графы и стохастические блочные модели [6], [7], [8], [9], [10]) спроектированы для моделирования определенного семейства графов и, следовательно, не имеют возможности непосредственно изучать генеративную модель из наблюдаемых данных.

Последние достижения в глубоких генеративных моделях, таких как вариационные автоэнкодеры (VAE) [11] и генеративные состязательные сети (GAN) [12], добились значительного прогресса в направлении генеративного моделирования для сложных областей, таких как изображения и текстовые данные. На основе этих подходов был предложен ряд моделей глубокого обучения для генерации графов.

В работах [13] и [14] предоставлен полный обзор графовых нейронных сетей (GNNs) в областях интеллектуального анализа данных и машинного обучения. Предлагается новая таксономия для разделения современных графовых нейронных сетей на различные категории. Сосредоточившись на сверточных сетях графов, также в статьях рассматриваются альтернативные архитектуры, которые недавно были разработаны; эти парадигмы обучения

включают сети внимания графа, графовые автоэнкодеры, графовые генеративные сети и пространственно-временные сети графа. Также авторы обсуждают приложения графовых нейронных сетей в различных областях.

В работе [15] авторы предлагают генерировать молекулярные графы в два этапа, используя реальные структуры молекул в качестве компонентов. Общий генеративный подход, называемый как вариационный автоэнкодер дерева переходов, сначала генерирует древовидный структурированный объект (дерево переходов), роль которого заключается в представлении грубых относительных расположений компонентов подграфа. Компоненты являются реальными химическими субструктурами, автоматически извлеченными из обучающего набора данных с использованием декомпозиции дерева, и используются в качестве строительных блоков. На втором этапе подграфы (узлы в дереве) собираются вместе в молекулярный граф. Латентное представление графа получается из нейронной сети передачи сообщений графа (Message Passing Neural Network).

Авторы работы [16], также, как и предыдущие авторы, предлагают новую вероятностную модель для генерации графа - вариационный автоэнкодер (VAE), где кодер и декодер состоит из графовых нейронный сетей (Graph Gated Neural Networks).

В работе [17] авторы предложили модель условной генерации графа, которая является вычислительно эффективной и обеспечивает прямую оптимизацию графа. Авторы предложили модель автоэнкодера, где кодировщик выполняет несколько пространственных графовых сверток на входном графе и агрегирует полученные эмбединги в одно скрытое представление графа. Декодер авторегрессионно генерирует набор непрерывных эмбедингов вершин графа, обусловленных выученным скрытым представлением графа, и восстанавливает весь граф из эмбедингов вершин.

В работе [18] авторы предлагают Graph Convolutional Policy Network (GCPN), общую модель графовой сверточной сети для генерации графа посредством обучения с подкреплением (Reinforcement Learning, RL). Авторы сформулировали задачу генерации графа как обучения агента RL, который итеративно добавляет вершины и ребра к молекулярному графу.

3 Описание метода

3.1 Данные

Рассматриваемый набор данных содержит 1000 электромагнитных ливней. Ливень представляют собой набор треков, который возникает при прохождении заряженной частицы через эмульсионный кирпич. Каждый ливень характеризуется 5 величинами:

- x, y, z – координаты распада частицы/начала ливня
- θ_x, θ_y – направление движения частицы, нормированное так, что $\theta_z = 1$

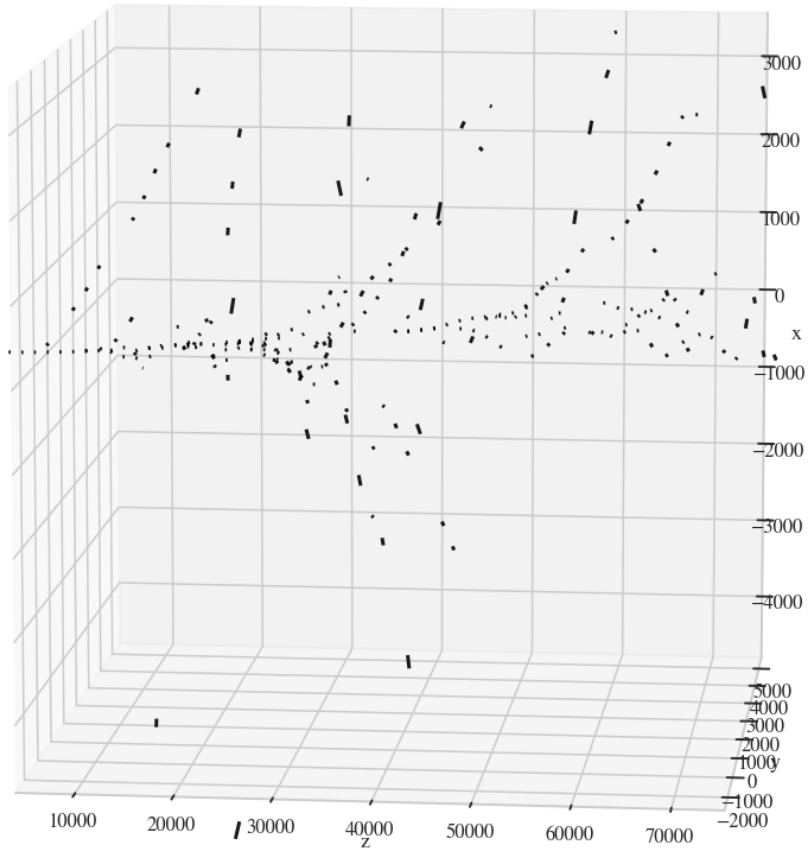


Рисунок 3 Пример одного ливня в кирпиче

Результатом предобработки исходных данных является ориентированный граф, вершины которого соответствуют трекам. В качестве

меры вероятности, что два трека соединены ребром, используется интегральное расстояние между прямыми, выходящими из треков. Данная мера определяется следующим интегралом, который можно посчитать аналитически:

$$IntDist = \int_{z_1}^{z_2} \left(\left(z(\theta_{x_2} - \theta_{x_1}) - \left(x_1 - x_2 + \theta_{x_2}(z_2 - z_1) \right) \right) + \left(z(\theta_{y_2} - \theta_{y_1}) - \left(y_1 - y_2 + \theta_{y_2}(z_2 - z_1) \right) \right) \right)^{1/2} dz$$

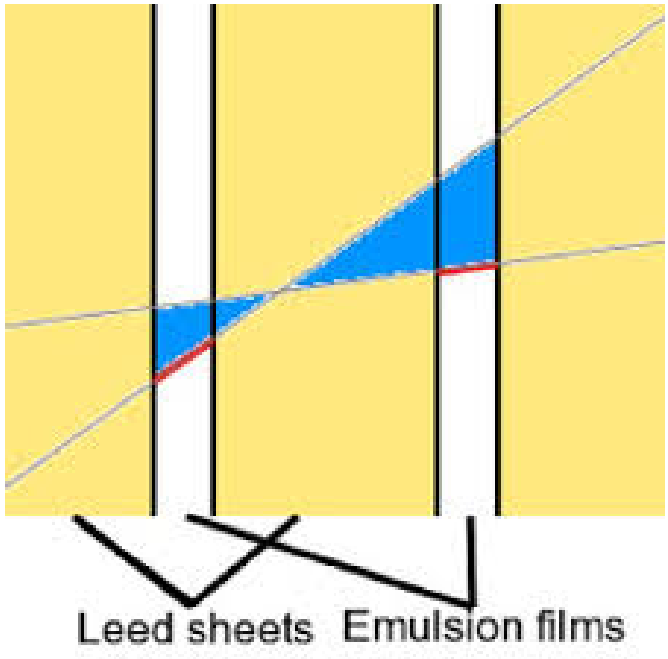


Рисунок 4 Красные линии представляют собой треки. Площадь синего сегмента является мерой сходства между двумя треками.

Если интегральное расстояние между вершинами меньше заданного порога, то вершины соединяются направленным ребром.

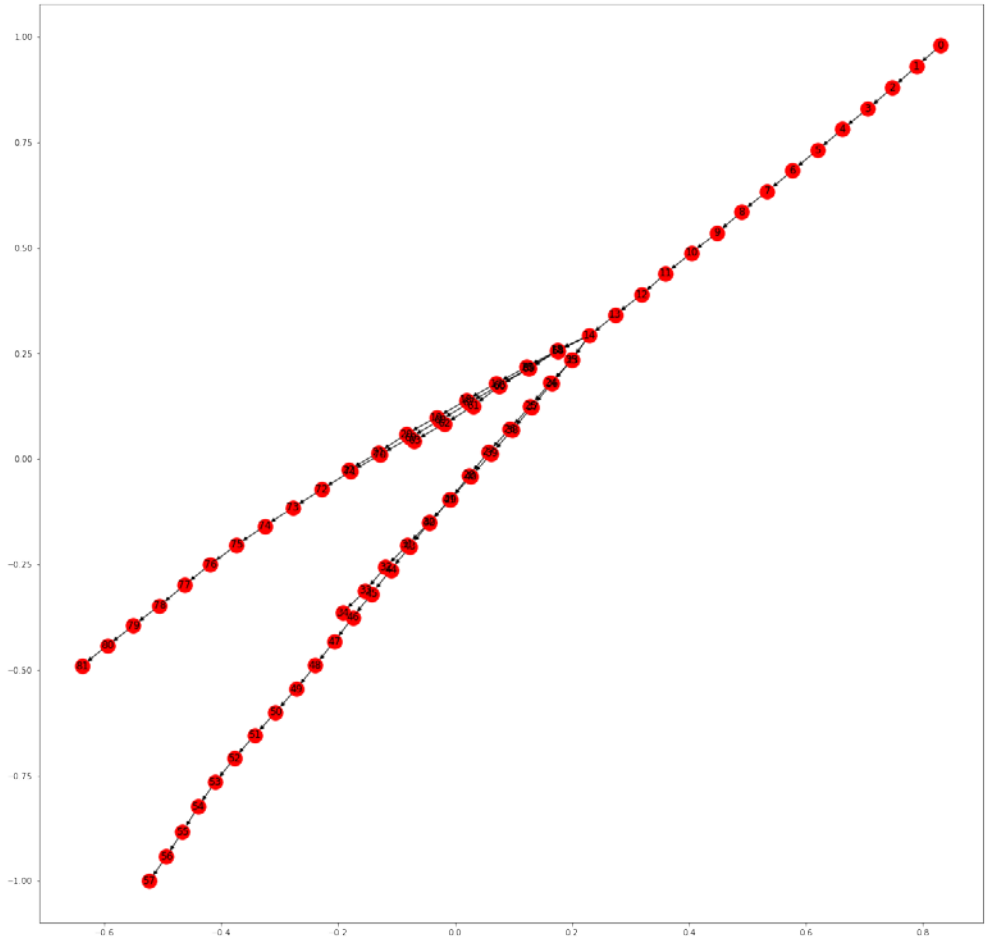


Рисунок 5 Пример ливня в виде ориентированного графа

Следовательно, задача генерации ливня сводится к задаче генерации ориентированного графа и генерации признаков вершин графа.

3.2 Ограничения для генерации графов

Моделирование и генерация графов является фундаментальной задачей и имеет важные приложения во многих областях, включая моделирование физических и социальных взаимодействий, открытие новых химических и молекулярных структур и построение графов знаний. Однако моделирование сложных распределений по графам, а затем эффективное семплирование из этих распределений является сложной задачей из-за неуникального, высокоразмерного характера графов и сложных нелокальных зависимостей, которые существуют между ребрами в данном графе.

Разработка генеративных графовых моделей имеет богатую историю, и было предложено много методов, которые могут генерировать графики на основе априорных структурных предположений о графе [19]. Однако ключевой открытой проблемой в этой области является разработка методов, которые могут непосредственно обучаться из наблюдаемого набора графов.

Ограничения, вытекающие из трех фундаментальных проблем в задаче генерации графов:

- **Большая и переменная размерность выходного пространства:** для генерации графа с n вершинами модель должна сгенерировать n^2 значений, чтобы полностью определить его структуру. Кроме того, число вершин n и максимальное количество ребер m варьируется между различными графами, и генеративная модель должна учитывать такую сложность и изменчивость размерности выходного пространства.
- **Неуникальное представление структуры графа:** граф с n вершинами может быть представлен $n!$ эквивалентными матрицами смежности, каждая из которых соответствует произвольному порядку/нумерации вершин. Обучение на всевозможных изоморфных графах, необходимое для улучшения обобщающей силы алгоритма, крайне сложно с точки зрения вычислительных ресурсов.
- **Сложные зависимости в структуре графа**: формирование ребер в графах включает сложные структурные зависимости. Например, во многих графах реального мира два узла с большей вероятностью будут связаны, если они имеют общих соседей [19]. Поэтому ребра не могут быть смоделированы как последовательность независимых событий, а должны генерироваться совместно, где каждое следующее ребро зависит от предыдущих сгенерированных ребер.

3.2 Генерация структуры графа

Для генерации структуры графа используется графовая рекуррентная нейронная сеть (Graph Recurrent Neural Network, GraphRNN) [20]. GraphRNN моделирует граф авторегрессионным (или рекуррентным) способом—последовательно добавляя новые вершины и ребра—для моделирования сложной совместной вероятности всех вершин и ребер в графе. В частности, GraphRNN можно рассматривать как иерархическую модель, где graph-level RNN поддерживает состояние графа и генерирует новые вершины, в то время как edge-level RNN генерирует ребра для каждой вновь созданной вершны. Благодаря своей авторегрессионной структуре GraphRNN может естественным образом генерировать графы разного размера.

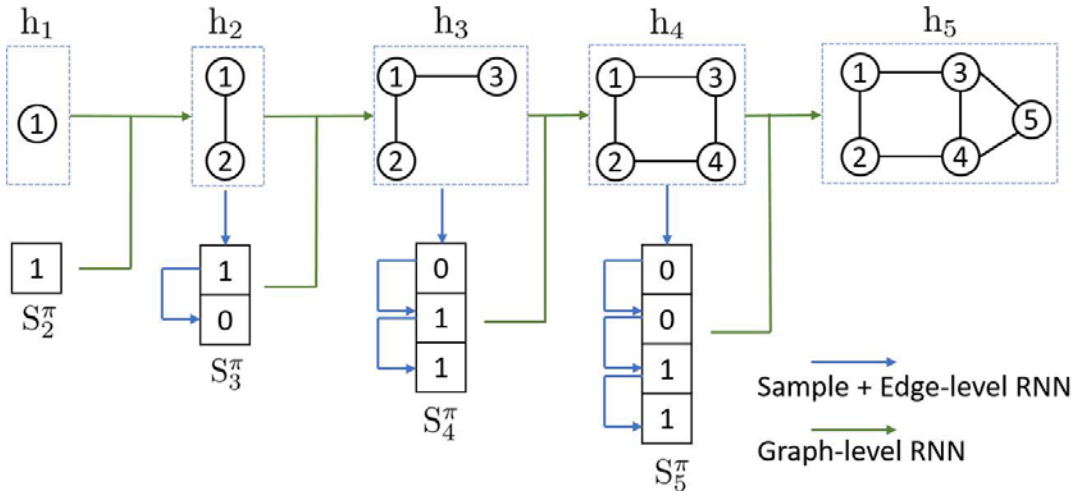


Рисунок 6 GraphRNN inference. Зеленые стрелки обозначают graph-level RNN, который кодирует вектор «состояния графа» h_i в его скрытом состоянии, обновляя прогнозируемый вектор смежности S_i для вершины $\pi(v_i)$). Синие стрелки представляют edge-level RNN, скрытое состояние которого инициализируется выходом graph-level RNN.

Для улучшения масштабируемости метода вводится обход графа в ширину (BFS) для упорядочения вершин, чтобы резко уменьшить сложность обучения по всем возможным последовательностям вершин. Обход в ширину запускается от начала ливня (первый трек). Этот подход облегчает тот факт, что графы имеют неуникальное представление, и древовидная структура, получаемая из BFS, позволяет ограничить количество предсказаний ребер, сделанных для каждой вершины во время обучения.

Описание метода

Моделируем граф как последовательность вершин: определим отображение f_S из графа в последовательность, где для графа G с n вершинами при порядке вершин π, получаем

$$S^\pi = f_S(G, \pi) = (S_1^\pi, \dots, S_n^\pi)$$

где каждый элемент $S_i^\pi \in \{0,1\}^{i-1}, i \in \{1, \dots, n\}$ - список смежности, обозначающий ребра между вершиной $\pi(v_i)$ и предыдущими вершинами в графе $\pi(v_j), j \in \{1, \dots, i-1\}$.

Алгоритм. GraphRNN inference

Вход: RNN модуль f_{RNN}, выходной слой f_{out}, вероятностное распределение P_θ, токен начала SOS, токен конца EOS, пустое скрытое представление графа h'

Выход: Графовые последовательности S^π

$S_1^\pi = SOS, h_1 = h', i = 1$

repeat

$i = i + 1$

$h_i = f_{RNN}(h_{i-1}, S_{i-1}^\pi)$ {Обновляем скрытое представление графа}

$\theta_i = f_{out}(h_i)$

$S_i^\pi \sim P_\theta$ {сэмлируем ребра для i вершины}

until $S_i^\pi = EOS$

Return $S^\pi = (S_1^\pi, \dots, S_i^\pi)$

Для GraphRNN используется следующая архитектура нейронных сетей: graph-level RNN использует 4-слойную ячейку GRU, с 256 размерным скрытым состоянием в каждом слое. В edge-level RNN используется также 4-слойная ячейка GRU с 64-мерным скрытым состоянием. Чтобы получить прогнозирование вектора смежности, edge-level RNN отображает 64-мерное скрытое состояние в 64-мерный вектор через линыйный слой с активацией ReLU, затем другой линейный слой отображает вектор в скаляр с активацией

сигмоида. Edge-level RNN инициализируется выходом сети graph-level RNN. Во время обучения используется teacher forcing как для graph-level RNN, так и для edge-level RNN, то есть мы используем истинную структуру графа, а не собственное предсказание модели во время обучения. Во время вывода модель использует свои собственные предсказания на каждом шаге генерации графа. В качестве оптимизатора используется Adam со скоростью обучения 0.001, обучение проводиться на мини-батчах. Каждый мини-батч содержит 20 графов. Для реализации GraphRNN использовался фреймворк Pytorch.

3.3 Генерация признаков вершин графа

Для генерации сигналов на графе используется генеративно-состязательная сеть (Generative Adversarial Network, GAN). GAN - структура для оценки генеративных моделей с помощью состязательного процесса, в котором мы одновременно обучаем две модели: генеративную модель G, которая оценивает распределение данных, и дискриминативную модель D, которая оценивает вероятность того, что выборка пришла из обучающих данных, а не из генеративной модели G. Процедура обучения для G заключается в максимизации вероятности ошибки D. Эта структура соответствует минимаксной игре для двух игроков [12].

Алгоритм: GAN framework

for количество эпох обучения **do**

for k шагов **do**

- Семплировать мини-батч из m элементов шума z
- Семплировать мини-батч из m элементов данных x
- Обновить веса дискриминатора, шагая стохастическим градиентным спуском:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log \mathrm{D}(x^i) + \log(1 - D(G(z^i)))]$$

end for

- Семплировать мини-батч из m элементов шума z
- Обновить веса генератора, шагая стохастическим градиентным спуском:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^i)))$$

end for

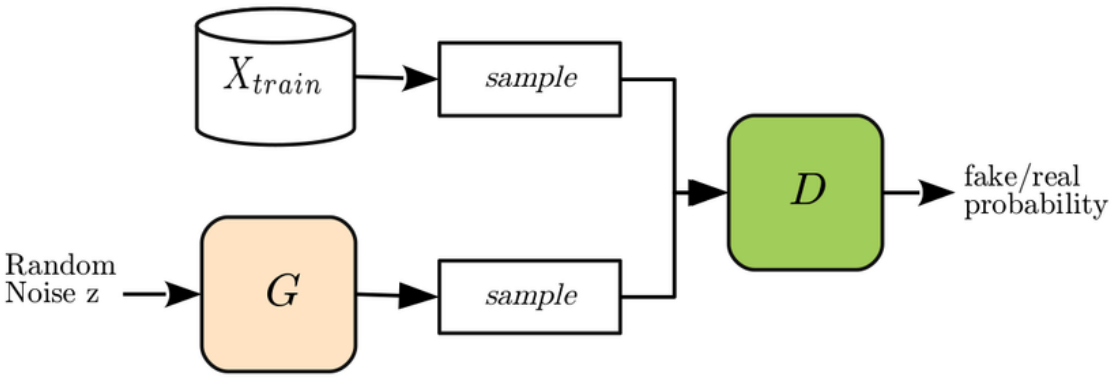


Рисунок 7 GAN фреймворк

Для генерации признаков на графе генератор и дискриминатор в GAN представляют собой графовые сверточные сети (Graph Convolution Network, GCN). Графовые сверточные сети обобщают операцию свертки из традиционных данных (изображений или сеток) для графовых данных. Идея состоит в том, чтобы выучить функцию f для создания представления вершины v_i путем агрегирования ее собственных прзнаков X_i и признаков соседей X_J, где $j \in N(v_i)$. На рис. 8 показана архитектура GCNs для представления вершин графа [13].

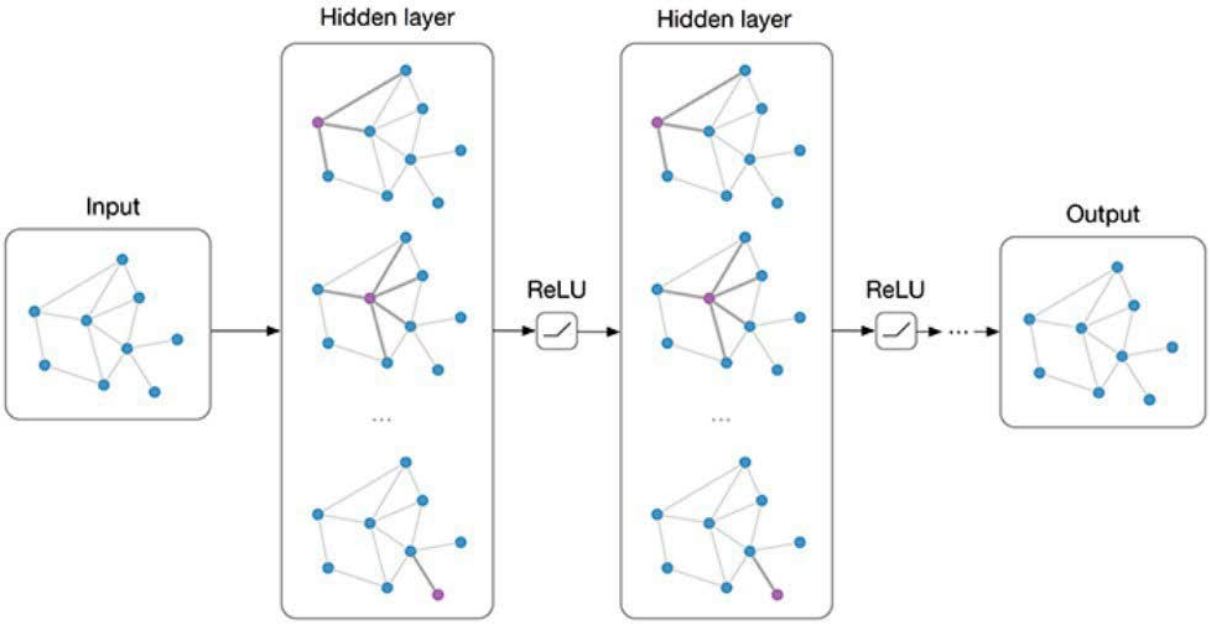


Рисунок 8 Вариант графовых сверточных сетей с несколькими слоями GCN. Слой GCN вычисляет скрытое представление каждой вершины путем агрегирования информации о признаках от ее соседей. После агрегации признаков к результирующим выходам применяется нелинейное преобразование. Путем добавления нескольких слоев, окончательное скрытое представление каждой вершины получает сообщения от дальних соседей.

Графовые сверточные сети играют центральную роль в построении многих других сложных моделей графовых нейронных сетей, включая модели на основе автоэнкодеров, генеративных моделей, пространственно-временных сетей и т. д.

Архитектура генератора

Чтобы генератор был инвариантен к сдвигам и поворотам ливня, генерируются не сами значения признаков вершин, а разница между двумя соседними вершинами $\Delta x,\ \Delta y, \Delta z,\ \Delta\theta_x,\ \Delta\theta_y$. Поэтому в качестве свертки на графе используется свертка для ребер графа EdgeConv.

EdgeConv выявляет локальную геометрическую структуру при сохранении инвариантности перестановок вершин. Вместо генерации признаков вершин непосредственно из их эмбедингов EdgeConv создает признаки ребер, описывающие отношения между вершиной и ее соседями. EdgeConv предназначен для инвариантного упорядочения соседей и, следовательно, инвариантной перестановки вершин [21].

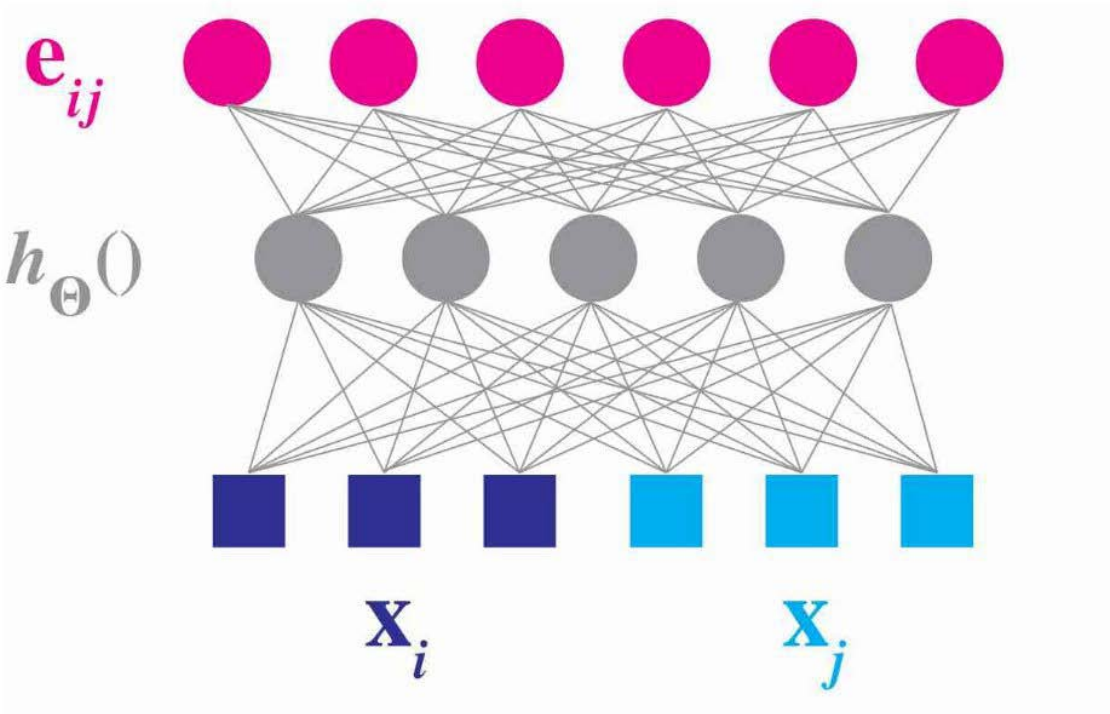


Рисунок 9 Пример вычисления признаков ребер e_{ij} из пары вершин x_i и x_j. В примере, $h_\theta()$ - полносвязанный слой.

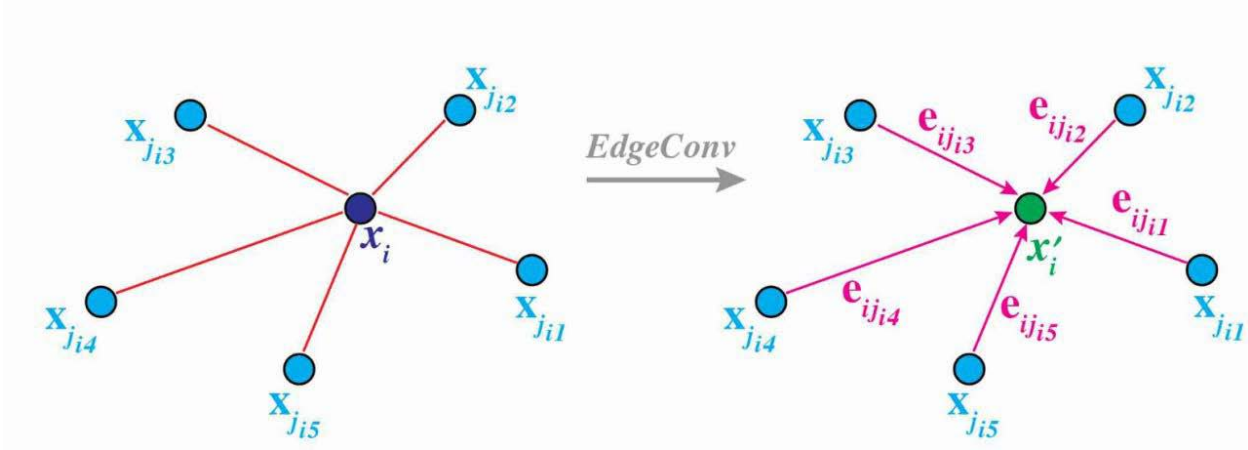


Рисунок 10 Визуализация операци EdgeConv. Выход EdgeConv вычисляется путем агрегирования признаков ребер, связанных со всеми ребрами, исходящими из каждой связанной вершины.

Свертку для ребер графа определим как

$$x_i' = \sum_{j \in N(i)} h_\theta(x_i || x_j - x_i)$$

где h_θ - нейронная сеть типа MLP.

В качестве результата генератора получаются признаки ребер графа. Поэтому сгенерированный граф восстанавливался итеративно, где признак

вершины формировался как сумма признаков вершины, входящую в нее, и признаков ребра, соединяющих их.

$$track_0 = \left(x_0, y_0, z_0, \theta_x^0, \theta_y^0\right) = (0,0,0,0,0)$$

$$\forall\ edges\ (i,j) \in G:$$

$$track_j = track_i + [\Delta x, \Delta y, \Delta z, \Delta\theta_x, \Delta\theta_y]$$

Для генератора используется следующая архитектура графовой сверточной сети: генератор содержит 5 слоев свертки на ребрах. В качестве h_θ линейный слой с 32 размерным скрытым состоянием, исключение вершин (dropout) с вероятностью 0.3 и нелинейность в виде ReLU. На вход генератору в качестве признаков приходят эмбединги вершин, полученных из GraphRNN, сконкатенированные со степенью захода вершины и степенью исхода вершины. Далее преобразованные эмбединги вершин, полученные из свертки на ребрах, передаются в линейный перцепторон с 3 полносвязными линейными слоями с нелинейность ReLU. После из выхода перцептрона формируется таблица списка ребер, где каждая строка соответствует признакам вершины, из которой ребро выходит, и признаку вершины, в которое ребро входит. Чтобы получить сгенерированную матрицу признаков вершин применяется еще один линейный слой с 5 размерным выходом. Во время обучения используется teacher forcing, то есть мы используем истинную структуру графа для генерации. В качестве оптимизатора используется RMSProp со скоростью обучения 0.0001, обучение проводиться на мини-батчах. Каждый мини-батч содержит 20 графов.

Архитектура дискриминатора

Дискриминатор решает задачу классификации на уровне графа. По обучающему набору данных (истинные ливни и сгенерированные ливни) классификация на уровне графа направлена на прогнозирование метки класса для всего графа.

Сквозное обучение (end-to-end learning) для этой задачи может быть выполнено с помощью фреймворка, который объединяет как графовые сверточные слои, так и процедуру объединения (pooling) вершн. В частности, применяя графовые сверточные слои, получаем представление с фиксированным числом измерений для каждой вершины в каждом отдельном графе. Затем мы можем получить представление всего графа через объединение вершин (pooling), которое агрегирует векторы представления всех вершин в графе. Наконец, применяя линейные слои (MLP) и слой softmax, которые обычно используются в существующих фреймворках глубокого обучения, мы можем построить сквозную структуру (end-to-end framework) для классификации графов. Пример приведен на рисунке 11.

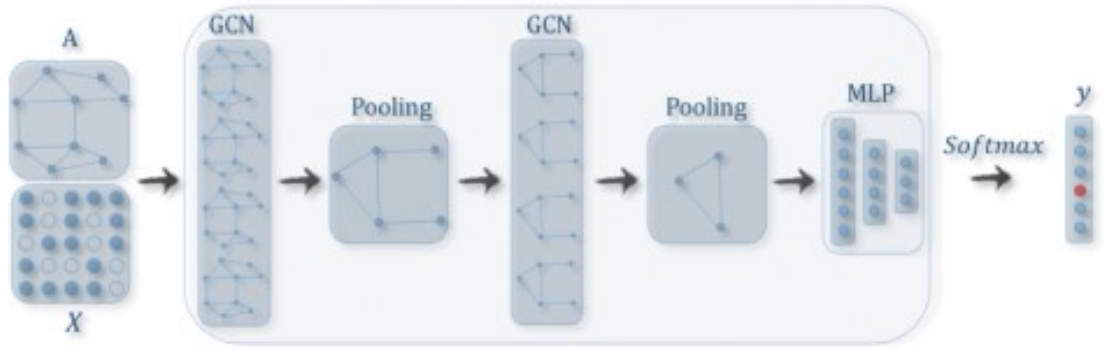


Рисунок 11 Графовые сверточные сети с модулями pooling для классификации графов. Для вычисления вероятности для каждой метки графика выходной слой представляет собой линейный слой с функцией SoftMax

В качестве свертки на графе используется GraphConv [22]:

$$x_i' = \theta x_i + \sum_{j \in N(i)} \theta x_j$$

В качестве функции объединения (pooling) используется global mean pooling:

$$r_i = \frac{1}{N_i} \sum_{n=1}^{N_i} x_n$$

Для дискриминатора используется следующая архитектура графовой сверточной сети: дискриминатор содержит 3 слоя свертки на графе с 64

скрытым представлением, слой global mean pooling, который усредняет эмбединги всех вершин в графе. После применяется полносвязный слой с 64 размерным скрытым состоянием, исключение вершин (dropout) с вероятностью 0.3 и нелинейностью в виде ReLU, и добавляется еще один линейный слой для вычисления скаляра с функцией softmax на выходе. В одной эпохе обучения GAN дискриминатор обучается 10 раз. В качестве оптимизатора используется Adam со скоростью обучения 0.001, обучение проводиться на мини-батчах. Каждый мни-батч содержит 20 графов.

Также проводились эксперименты, где в качестве свертки на графах в дискриминаторе использовался EdgeConv, чтобы сделать дискриминатор также инвариантным к сдвигам и поворотам ливня.

Генератор и дискриминатор реализованы с использованием фреймворка pytorch geometric для обучения нейронных сетей на графах [22].

Реализацию метода можно найти по ссылке: https://github.com/anastdem/MasterDegree

4 Результаты

В данной работе была обучена генеративная модель для симуляции электромагнитных ливней, состоящая из двух частей. GraphRNN – для генерации структуры данных и GAN – для генерации признаков вершин.

Финальная архитектура всей модели:

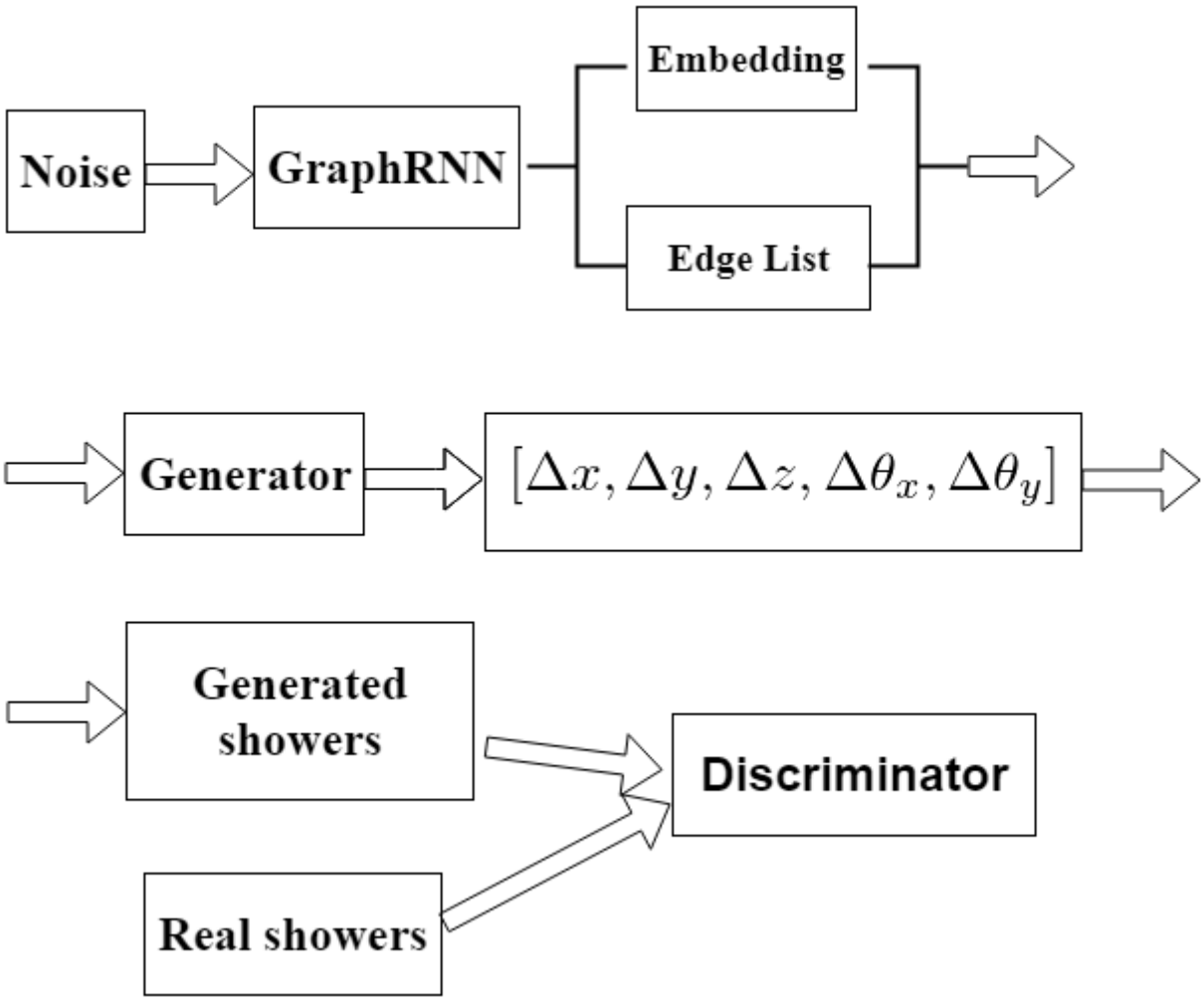


Рисунок 12 Фреймворк обучения модели

GraphRNN и GAN обучались отдельно друг от друга, так как GraphRNN сходится быстрее, и если обучать эти модели вместе, то GraphRNN может переобучиться. GraphRNN обучался 300 эпох, GAN обучался 600 эпох, также перед началом обучения GAN генератор предобучался по среднеквадратической ошибке 10 эпох.

Ниже приведены кривые обучения для обеих моделей.

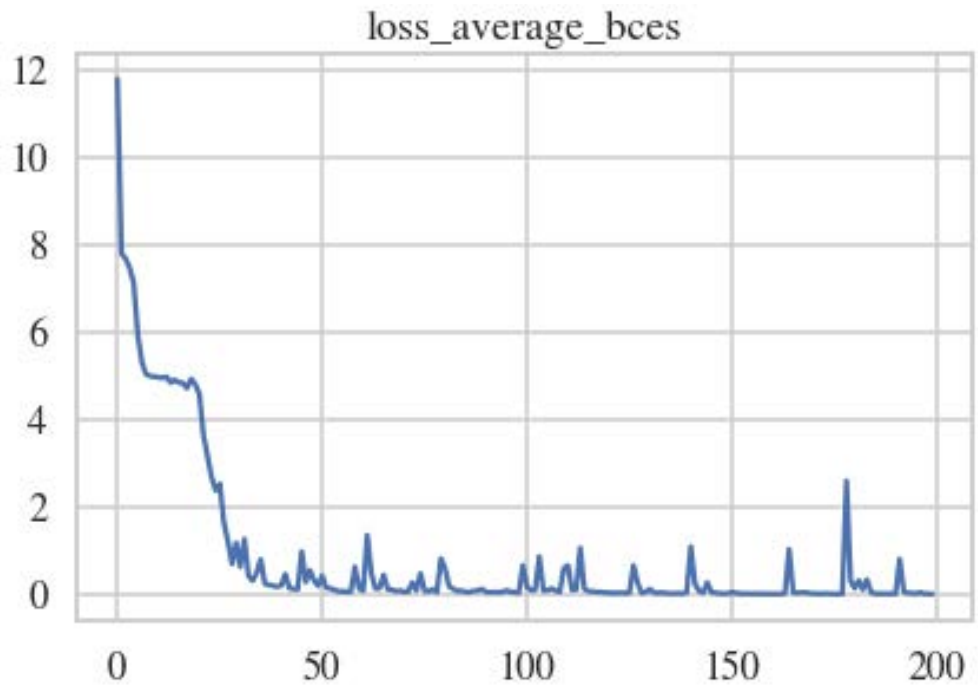


Рисунок 13 Кривая обучения GraphRNN

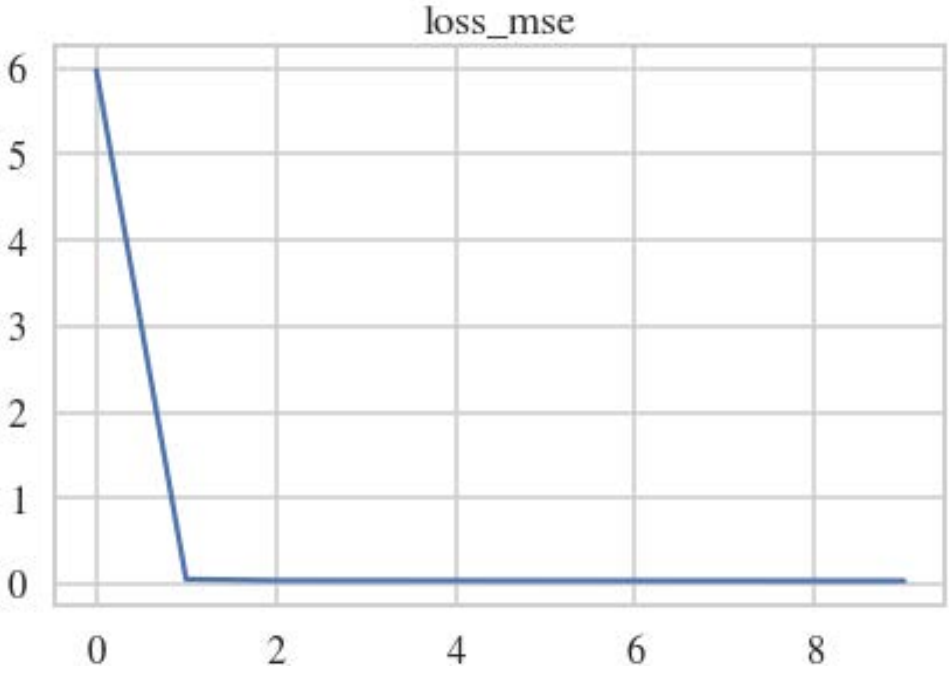


Рисунок 14 Предобучение генератора

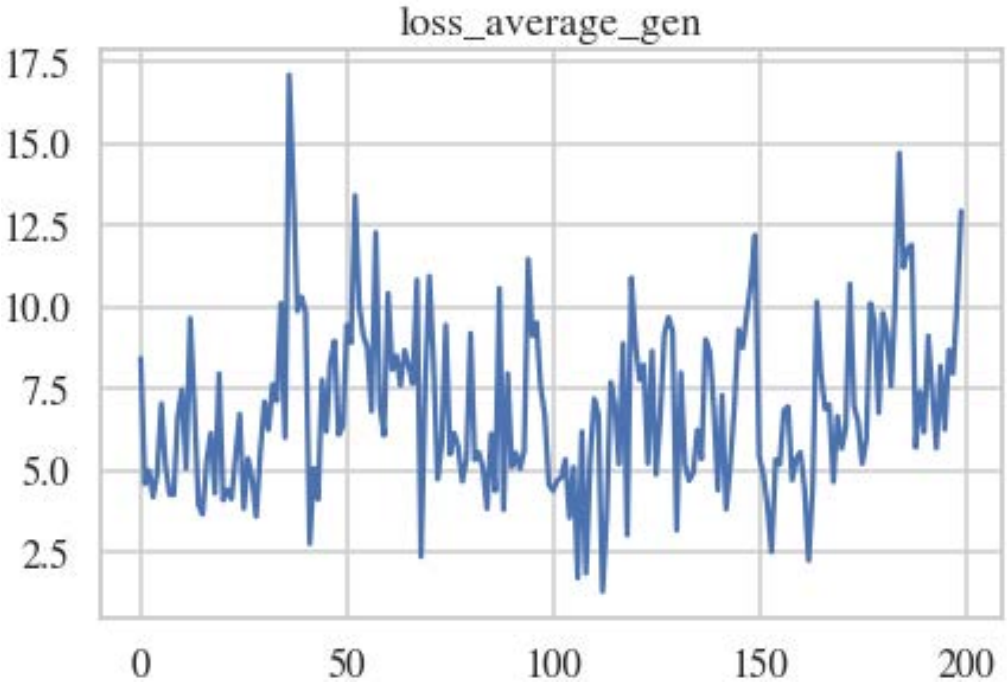


Рисунок 15 Кривая обучения генератора

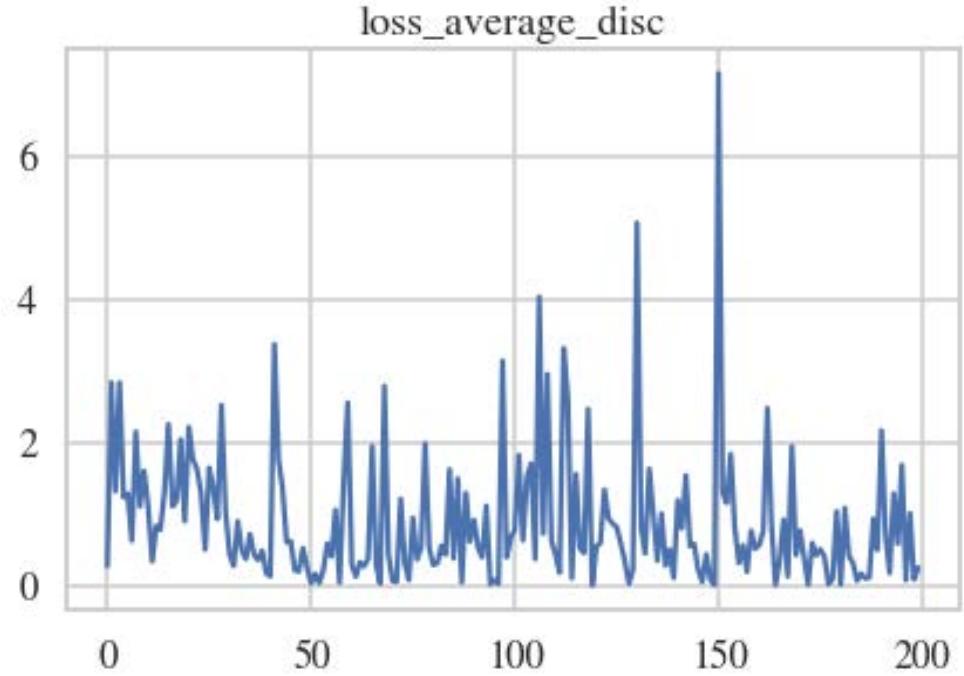
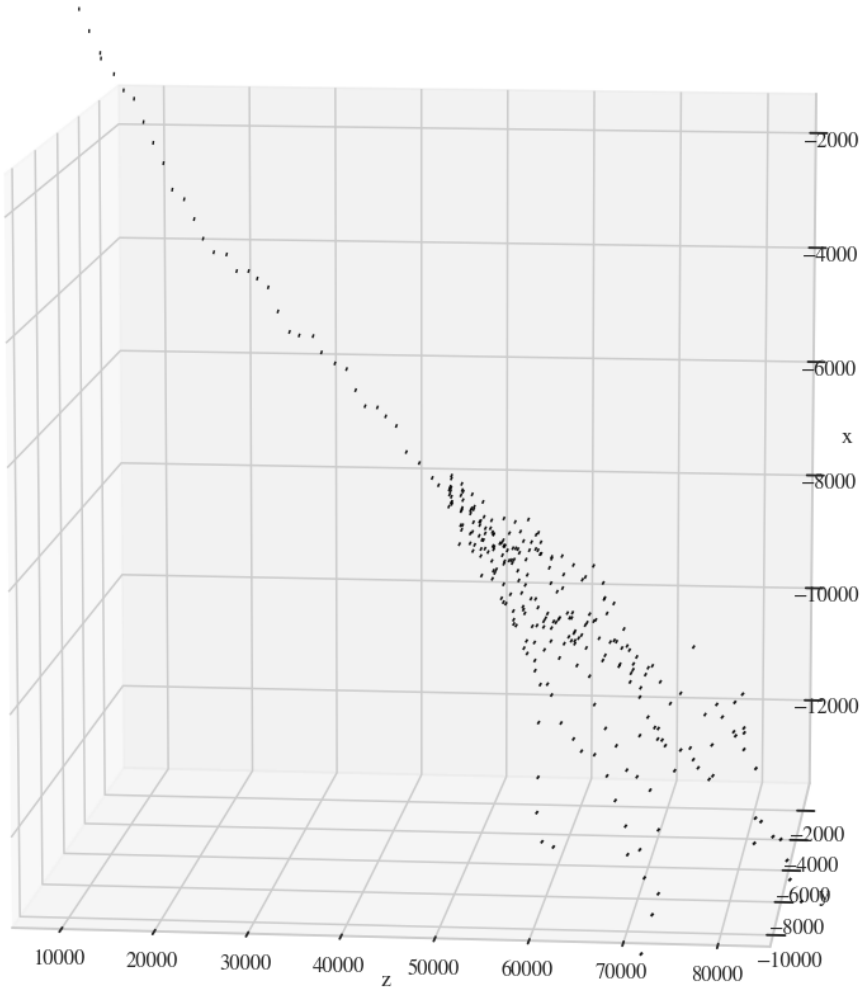


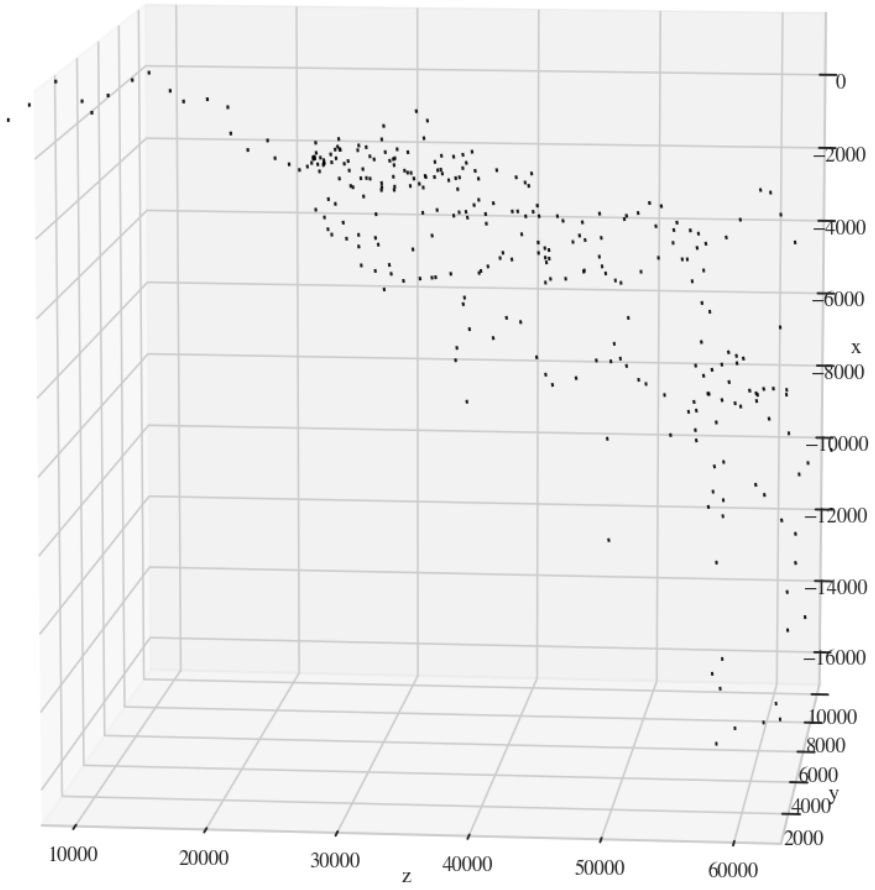
Рисунок 16 Кривая обучения дискриминатора

Оценка качества генеративных моделей является сложной задачей в целом, а в нашем случае эта оценка требует сравнения между двумя наборами графов (сгенерированные графы и тестовые графы). Универсальной и числовой метрики для оценки качества генерации

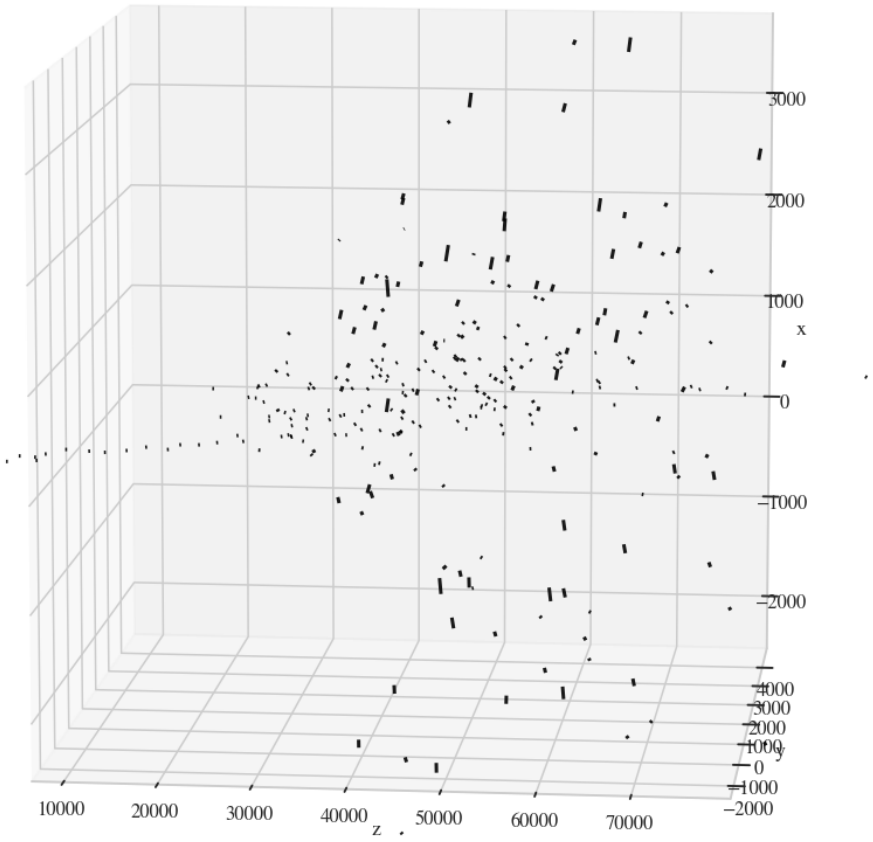
электромагнитных ливней нет. Ниже приведены несколько примеров сгенерированных электромагнитных ливней.

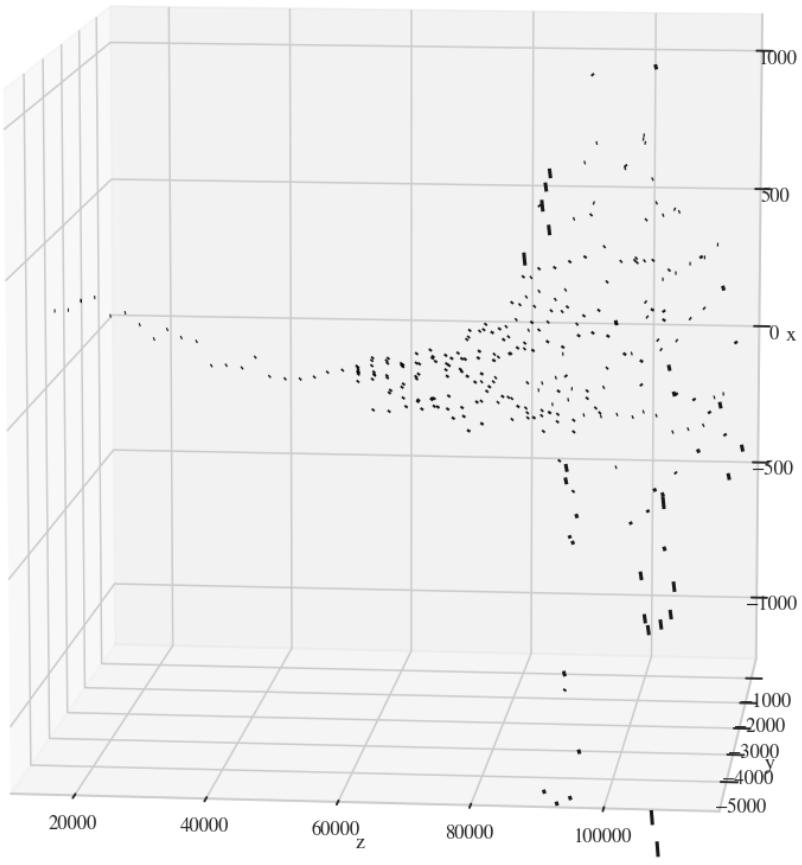
Примеры, где в архитектуре дискриминатора использовались свертки на графах GraphConv:

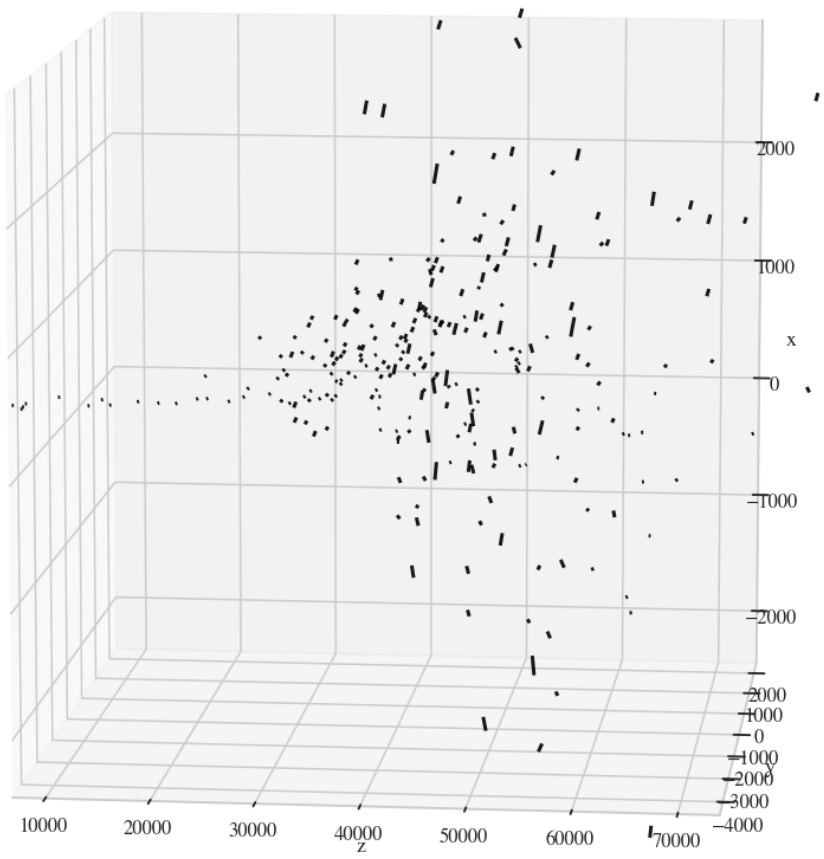


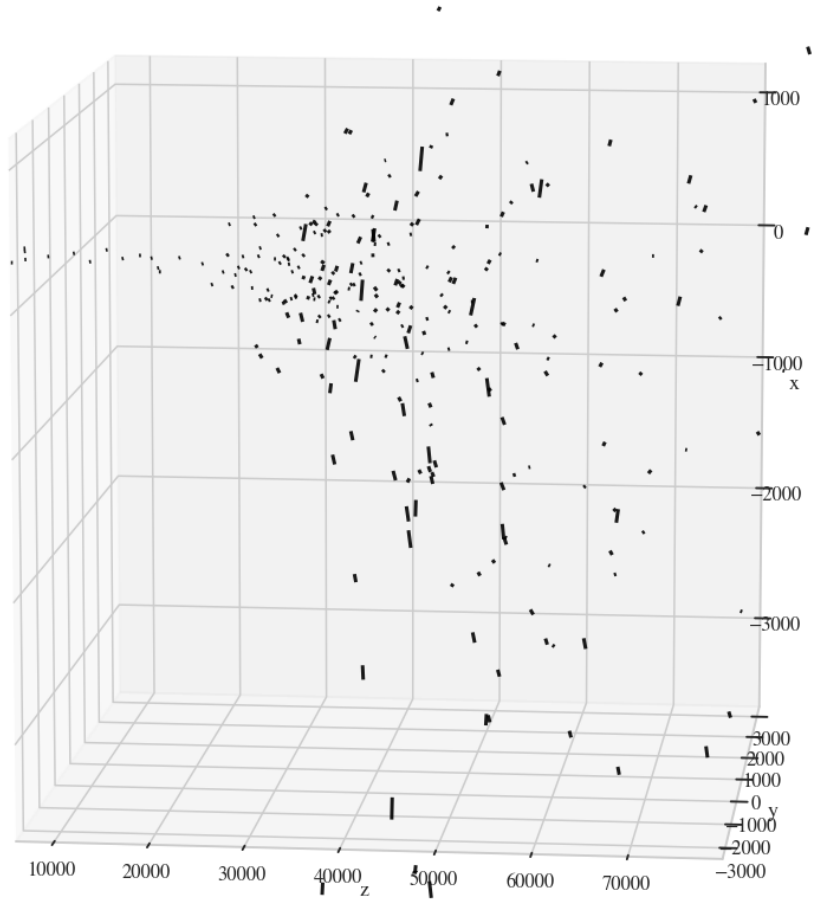


Примеры, где в архитектуре дискриминатора использовались свертки на графах EdgeConv:


3000
2000
1000
x
0
−1000
−2000
−4000
−3000
−2000
−1000
y
0
1000
−2000
10000
20000
30000
40000
50000
60000
70000
z

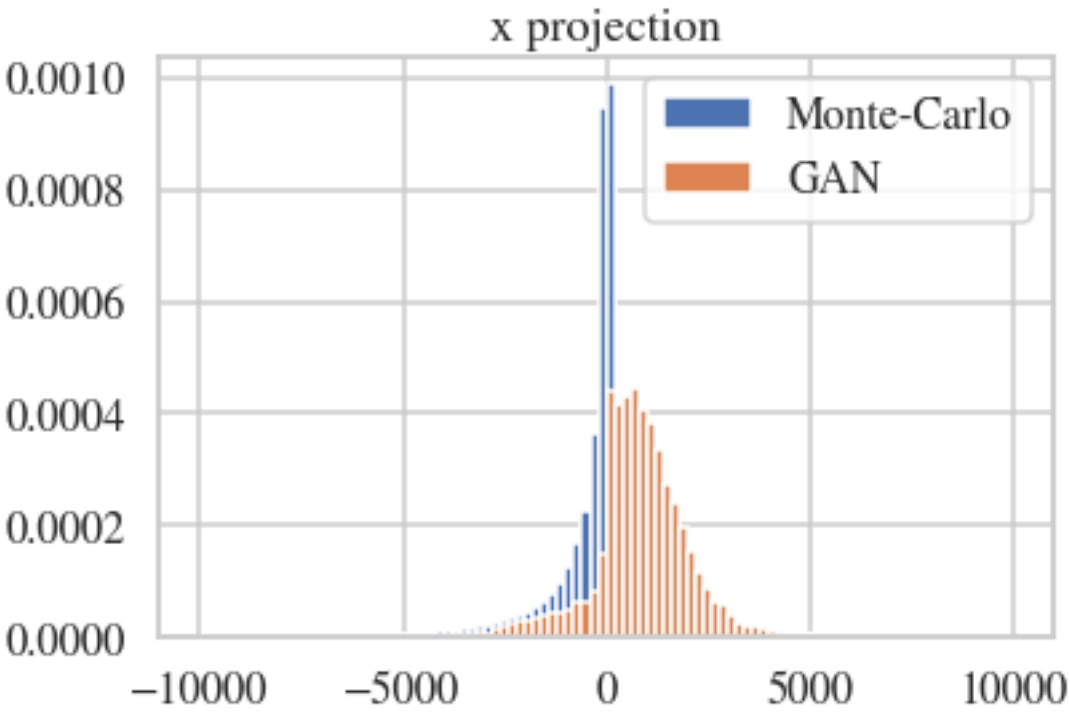

1000
500
0 x
−500
−1000
−1000
−2000
−3000
−4000
−5000
y
20000
40000
60000
80000
100000
z


2000
1000
x
0
−1000
−2000
−2000
−1000
0
−1000
−2000
−3000
−4000
y
10000
20000
30000
40000
50000
60000
70000
z

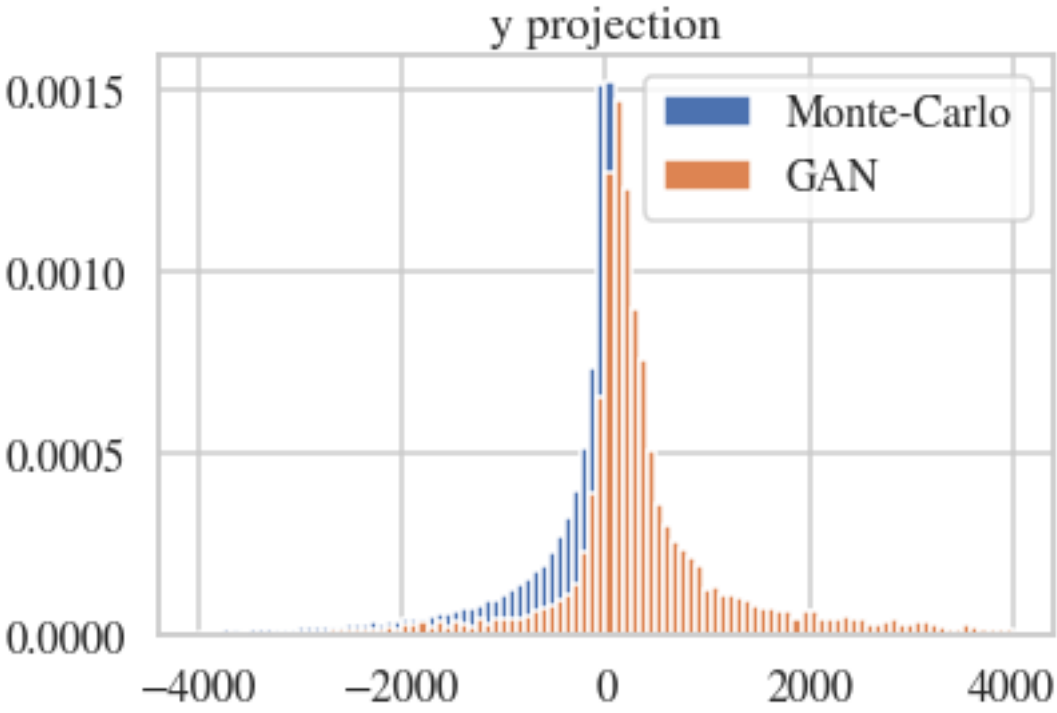

1000
0
−1000
x
−2000
−3000
−3000
−2000
−1000
0
y
−1000
−2000
−3000
10000
20000
30000
40000
50000
60000
70000
z

Как видно из примеров, модель фиксирует общую структуру графа, но ветвления ливня моделируются плохо. Также модель, где в архитектуре дискриминатора использовались свертки на графах EdgeConv, показывает лучшие результаты.

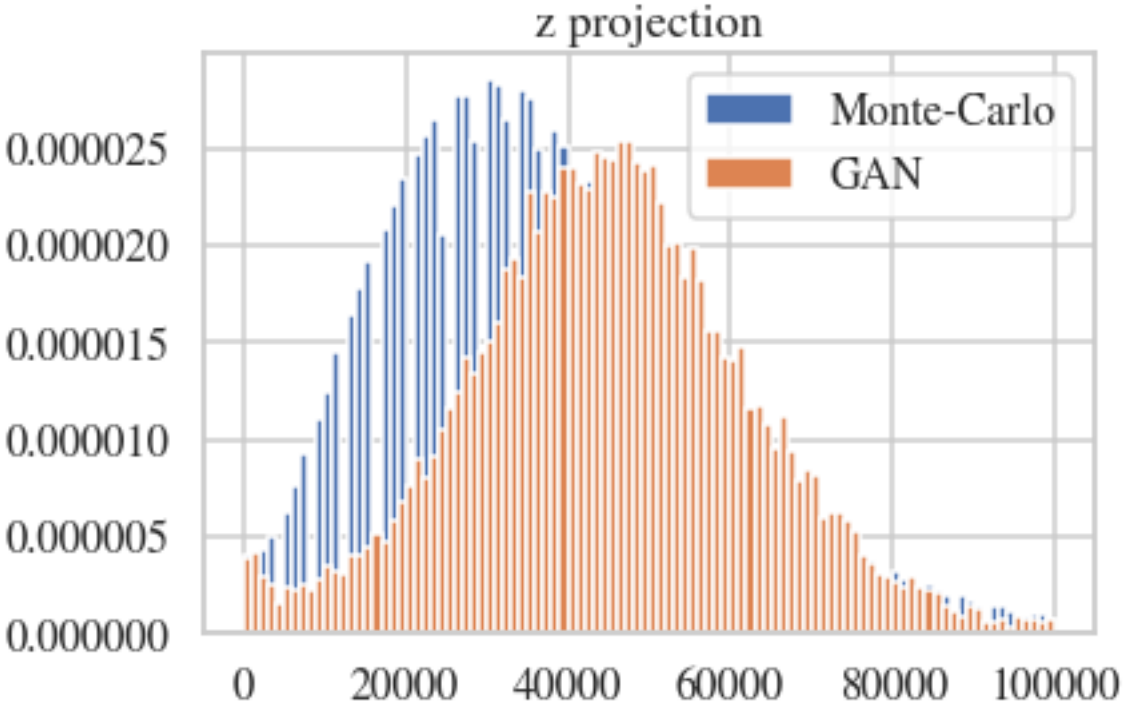
Для качественной оценки полученной модели предлагается сравнить распределения 5 характеристик электромагнитного ливня.



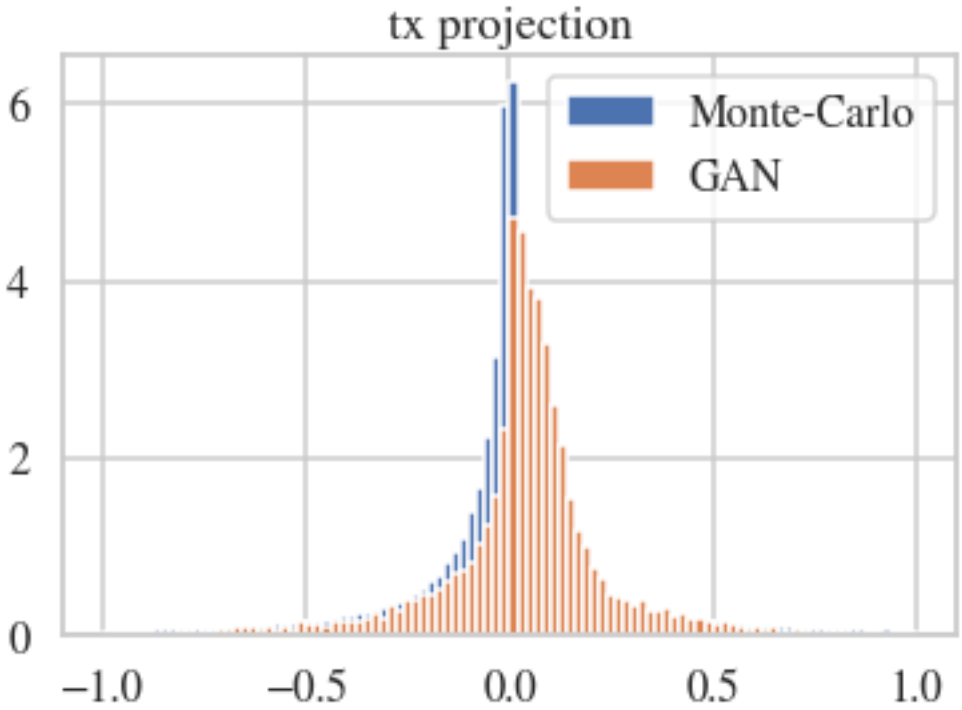
Распределение x координаты сгенерированного ливня отличается от распределения реального ливня. Реальный ливень находится в окрестности нуля, в то время как сгенерированный ливень смещен в сторону от нуля и имеет больший разброс.



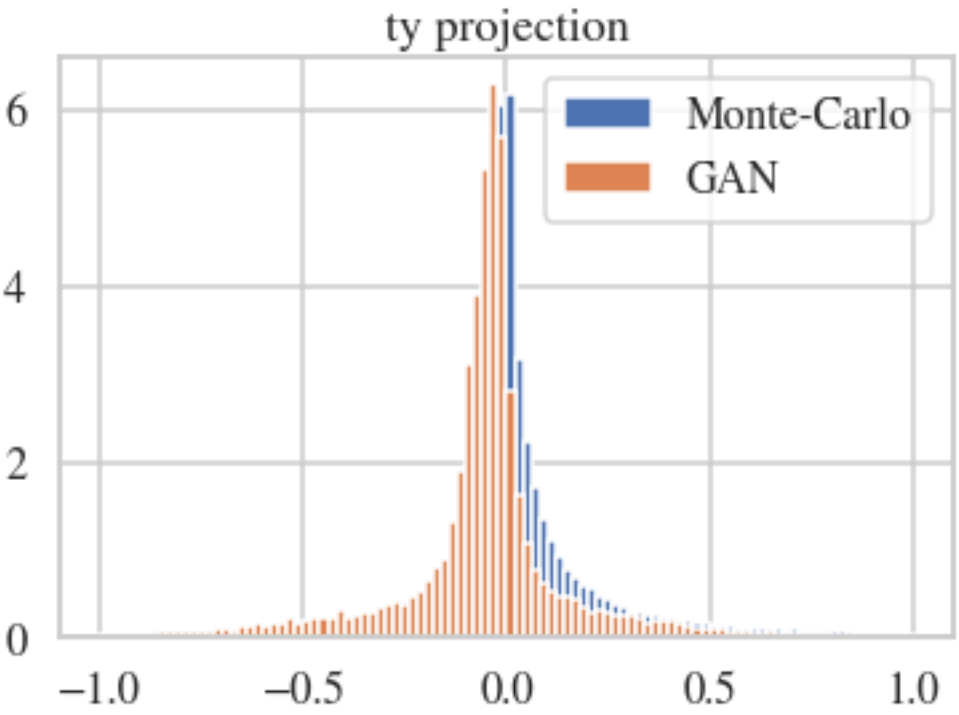
Распределение y координаты сгенерированного ливня похоже на распределения реального ливня. Распределения имеют одинаковую форму, но распределение сгенерированного ливня немного смещено в сторону.



Распределения z координаты сгенерированного ливня и реального ливня имеют одинаковую форму, но распределение сгенерированного ливня смещено в сторону от распределения реального ливня.



Распределения направления движения частицы θ_x сгенерированного ливня и реального совпадают.



Распределения направления движения частицы θ_y сгенерированного ливня и реального совпадают.

Также для оценки схожести распределений сгенерированных ливней и реальных используется статистика Колмогорова-Смирнова. Критерий Колмогорова-Смирнова используется для проверки гипотезы, что две независимые выборки принадлежат одному распределению. Чем меньше статистика Колмогорова-Смирнова, тем более схожи распределения между собой.

	x	y	z	θ_x	θ_y
K-S	0.45	0.31	0.24	0.23	0.25

5 Заключение

В данной работе разработана генеративная модель для симуляции электромагнитных ливней. Электромагнитный ливень был преобразован в граф, поэтому генерация ливня происходит в два этапа: генерация структуры графа и генерация характеристик вершин. Генерация структуры графа осуществляется с помощью рекуррентной модели GraphRNN, которая состоит из двух сетей graph-level и edge-level RNN. Graph-level RNN генерирует граф как последовательность вершин, а edge-level RNN генерирует список ребер для новой вершины. Для генерации признаков вершин графа использовалась генеративно-состязательная сеть, где архитектура генератора и дискриминатора представляет собой графовые сверточные сети.

Данная модель генерирует ливни по структуре похожие на реальные, но плохо отображает ветвление ливня, что подтверждается анализом распределений признаков сгенерированных и реальных ливней.

В качестве направления для дальнейшего исследования может быть предложена единая модель для генерации электромагнитных ливней, которая генерирует новую вершину и признаки для нее одновременно.

6 Список литературы

[1] F. Juget, «Electromagnetic shower reconstruction with emulsion,» *XIII International Conference on Calorimetry in High Energy Physics (CALOR 2008)*, p. 160, 2009.

[2] A. Rogozhnikov, "Machine learning applied to showers in the OPERA," 24 June 2017. [Online]. Available: http://arogozhnikov.github.io/2017/06/24/opera.html.

[3] Michela Paganini , Luke de Oliveira и Benjamin Nachm, «CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks,» *High Energy Physics - Experiment*, т. no.1, № 014021, p. 12, 2018.

[4] S. P. G.Grindhammer, «The Parameterized Simulation of Electromagnetic Showers in Homogeneous and Sampling Calorimeters.,» в *Int. Conf. on Monte Carlo Simulation in High Energy and Nuclear Physics*, Tallahassee, Florida, USA, 1993.

[5] E. a. a. Barberio, «Fast simulation of electromagnetic showers in the ATLAS calorimeter: Frozen showers,» в *Journal of Physics: Conference Series*, 2009.

[6] P. Erdos и A. Renyi, On random graphs I., Publicationes Mathematicae (Debrecen), 6:290–297, 1959.

[7] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos и Z. Ghahramani, Kronecker graphs: An approach to modeling networks, JMRL, 2010.

[8] R. Albert и L. Barabasi, «Statistical mechanics of complex networks,» *networks. Reviews of Modern Physics*, т. 1, № 74, p. 47, 2002.

[9] E. Airoldi, D. Blei, S. Fienberg и E. Xing, «Mixed membership stochastic blockmodels,» *JMLR*, 2008.

[10] G. Robins, P. Pattison, Y. Kalish и D. Lusher, «An introduction to exponential random graph (p*) models for social networks,» *Social Networks*, т. 2, № 29, pp. 173-191, 2007.

[11] D. P. Kingma и M. Welling, «Auto-Encoding Variational Bayes,» в *ICLR*, 2014.

[12] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville и Y. Bengio, «Generative Adversarial Networks,» в *NIPS*, 2014.

[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang и P. S. Yu, «A Comprehensive Survey on Graph Neural Networks,» 10 Mar 2019. [В Интернете]. Available: https://arxiv.org/abs/1901.00596.

[14] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li и M. Sun, «Graph Neural Networks: A Review of Methods and Applications,» 7 Mar 2019. [В Интернете]. Available: https://arxiv.org/abs/1812.08434.

[15] Wengong Jin, Regina Barzilay и Tommi Jaakkola, «Junction Tree Variational Autoencoder for Molecular Graph Generation,» в *Proceedings of the 35 th International Conference on Machine Learning*, Stockholm, Sweden, 2018.

[16] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt и Alexander L. Gaunt, «Constrained Graph Variational Autoencoders for Molecule Design,» в *32nd Conference on Neural Information Processing Systems (NeurIPS)*, Montréal, Canada, 2018.

[17] R. Assouel, M. Ahmed, M. H. Segler, A. Saffari и Y. Bengio, «DEFactor: Differentiable Edge Factorization-based Probabilistic Graph Generation,» https://arxiv.org/abs/1811.09766, 2018.

[18] J. You, B. Liu, R. Ying, V. Pande и J. Leskovec, «Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation,» в *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, Montréal, Canada, 2018.

[19] M. Newman, Networks: An Introduction, Oxford university press, 2010.

[20] J. You, R. Ying, X. Ren, W. L. Hamilton и J. Leskovec, «GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models,» в *Proceedings of the 35 th International Conference on Machine Learning, PMLR 80*, Stockholm, Sweden, 2018.

[21] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein и J. M. Solomon, «Dynamic Graph CNN for Learning on Point Clouds,» в *arXiv:1801.07829 [cs.CV]*, 2018.

[22] M. Fey и J. E. Lenssen, «Fast Graph Representation Learning with PyTorch Geometric,» в *CLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.