Федеральное государственное автономное образовательное учреждение
высшего профессионального образования

**«Национальный исследовательский университет
Высшая школа экономики»**

Факультет компьютерных наук

Основная образовательная программа

«Прикладная математика и информатика»

КУРСОВАЯ РАБОТА

на тему

Построение Generative Adversarial Network для быстрой генерации откликов калориметров детекторов элементарных частиц в ЦЕРН

Выполнил студент группы БПМИ151 3 курса

Наумов Антон Иванович

Научный руководитель:

Старший научный сотрудник,
Ратников Фёдор Дмитриевич

Москва 2018

Содержание

1 О проекте 2

2 Введение 3

3 Данные 4

4 Generative Adversarial Network (GAN) 5

5 Построение первого GAN для нашей задачи 6

6 Анализ полученных результатов 7

7 Сравнение энергий 8

8 Последующие вариации GAN 9

9 Результаты 11

10 Заключение 13

1 О проекте

Точное моделирование взаимодействия элементарных частиц и распространения через материю - первостепенная задача для продвижения исследований и точных измерений в областях ядерной физики и физики элементарных частиц. Наиболее вычислительно дорогой шаг в симуляции типичного эксперимента в Большом Адронном Коллайдере (LHC) - это точное моделирование всего комплекса физических процессов, которые определяют эволюцию ливней частиц внутри калориметра.

Основная задача проекта - реализовать более дешёвый вычислительно метод симуляции электромагнитных ливней в поперечно сегментированном калориметре, что позволило бы получить значительную экономию вычислительных мощностей при симуляции эксперементов в LHC, по сравнению с полной симуляцией ливней.

В качестве данного метода будет использована генеративно-состязательная нейросеть (Generative Adversarial Network - GAN), моделирующуая ливни в калориметре.

2 Введение

Физические программы всех экспериментов на базе LHC в значительной степени зависят от точного моделирования всех аспектов реконструкции событий и анализа данных. Смоделированные столкновения частиц, распады и взаимодействия с материей используются для интерпретации результатов текущих экспериментов и оценки работы последующих. Современные симуляции способны точно моделировать геометрию детектора и физические процессы, на размерах вплоть до 10^{-20} м. Эти процессы, которые включают ядерные и атомные взаимодействия, такие как ионизация, а также сильные, слабые и электромагнитные процессы, изменяют состояние входящих частиц при их распространении и взаимодействии со слоями материала в различных компонентах детектора. Методы обнаружения, такие как калориметрия используют эти физические взаимодействия для обнаружения составляющих и измерения энергии частиц, таких как фотоны, электроны и адроны, через их взаимодействие с сотнями тысяч компонентов детектора. В результате взаимодействия с калориметром, образуется каскад (ливень) вторичных частиц и их энергия собирается и преобразуется в электрические сигналы.

Физическое полное моделирование ливней частиц в калориметрах является самой вычислительно-затратной частью всего процесса моделирования, и может занимать по несколько минут на событие на современных распределенных высокопроизводительных платформах. Получение физических результатов часто ограничивается отсутствием адекватного по вычислительным затратам метода моделирования Монте-Карло (MC), а увеличение числа событий на промежуток времени (luminocity) на LHC будет только усугубляют проблему. В настоящее время полная симуляция MC занимает $50-70\%$ вычислительных мощностей, использующихся на эксперименты, что эквивалентно миллиардам процессорных часов в год.

Актуальность и значимость решения проблемы этапа моделирования событий в калориметре повлекла за собой развитие приближенных, но быстрых алгоритмов моделирования для смягчения его вычислительной сложности. Быстрые методы моделирования основаны на параметризованных ливнях для флуктуаций и справочных таблиц для низких энергий взаимодействия. Для многих задач эти методы оказались достаточными. Однако для анализа детальной структуры ливней для распознования частиц или для более точной оценки энергии и направления этих упрощённых методов может оказаться недостаточно.

Мы исользуем метод Deep Learning для обеспечения высокой точности и быстрого моделирования ливней частиц в электромагнитных калориметрах. [2, 3] Идея GAN, использующияся в данной работе, уже хорошо показала себя в различных областях, таких как космология, онкология и физика конденсированного состояния.

3 Данные

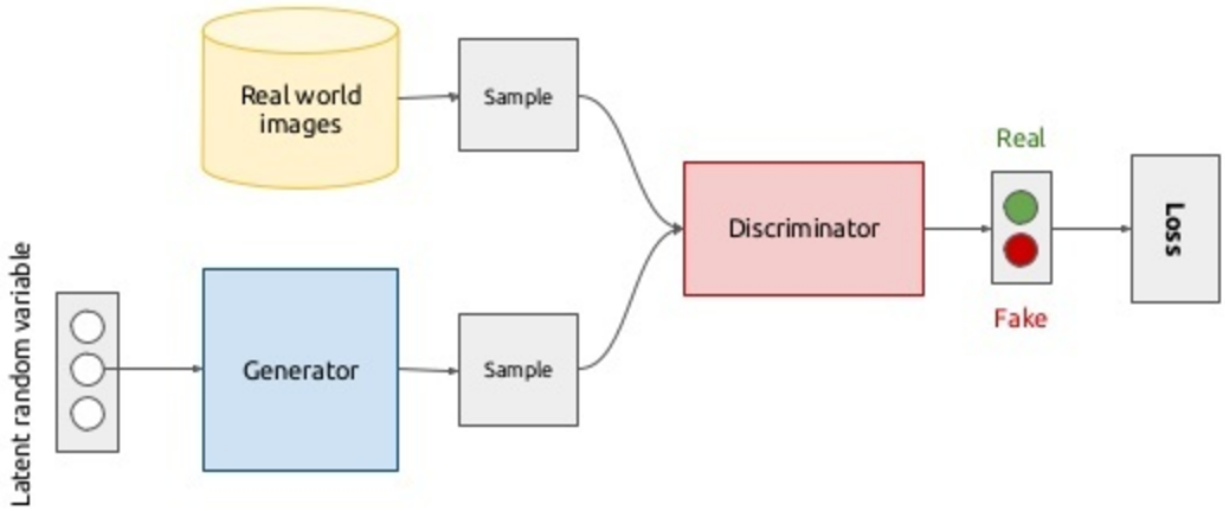
Данные откликов частиц на поверхности калориметра по сути представляют из себя изображения 30x30. Основная идея проекта состоит в том, чтобы вместо полного физического моделирования, которое требует значительных затрат по времени, попытаться сразу генерировать изображения, похожие на реальные.

4 Generative Adversarial Network (GAN)

GANs (генеративно-состязательные нейросети) используются для задачи генерации данных, для которых очень сложно (если вообще возможно) задать явную функцию потерь, оценивающую качество сгенерированной модели. Основная идея GAN состоит в том, чтобы учить нейросеть генерировать данные из шума, не задавая явно функцию потерь полученных данных относительно реальных, а, вместо этого, ввести ещё одну нейросеть, называемую Дискриминатором (D), которая будет оценивать качество сгенерированных первой нейросетью, называемой Генератором (G), данных. Они будут обучаться вместе, в результате чего дискриминатор будет всё лучше отличать сгенерированные данные от реальных, а генератор будет выдавать всё более правдоподобные данные.

Всё происходящее в ходе обучения GAN будет являтся minmax игрой с нулевой суммой, где двумя игроками будут являтся две нейросети: G и D.

Качественно: Дискриминатор получает на вход данные (либо реальные, либо созданные генератором - выбирается случайно) и должен сказать являются данные реальными или нет (вероятность того что данные взяты из обучающей выборки, а не созданы генератором), тогда как генератор преобразует некоторый полученный на вход шум в данные, пытаясь сделать так, чтобы дискриминатор поверил ему и сказал что это реальные данные.



Строго:

- Реальные данные (обучающая выборка) $X = \{x_1, x_2, ..., x_m\}$ из реального распределения p_{real}.
- Шум $Z = \{z_1, z_2, ..., z_m\}$ из шумового распределения p_{noise}
- Две многослойные нейросети D и G с параметрами θ_D, θ_G соответственно
- Генератор G получает на вход шум $Z \to G(Z, \theta_G) \in p_{gen}$
- Дискриминатор D получает на вход либо реальные данные ($\in p_{real}$), либо сгенерированные ($\in p_{gen}$) и выдаёт вероятность того что данные принадлежат $p_{real} \to D(x, \theta_D) = \mathbb{P}(x \in p_{real})$
- Игра $\to \min\limits_{G} \max\limits_{D} V(D, G) = \mathbb{E}_{x \in p_{real}}[\log(D(x))] + \mathbb{E}_{z \in p_{noise}}[\log(1 - D(G(z)))]$
- Функция потерь G: $L_G = -\log(D(G(Z))$
- Функция потерь D: $L_D = -\log(D(X)) - \log(1 - D(G(Z)))$
- В статье, в которой презентуется GAN, доказывается что p_{gen} сходится к p_{real} [1]

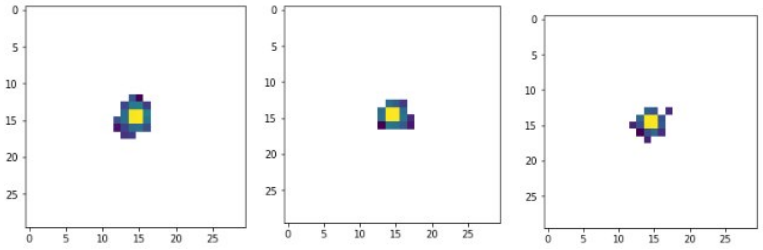
Но несмотря на множество положительных моментов в использовании GAN у него есть несколько проблем:

1. Плохая сходимость
2. Возможность остановки до сходимости, если дискриминатор и генератор окажутся плохо сбалансированы между собой
3. Mode collapse - если распределение p_{real} многомодальное, то GAN опишет лишь несколько мод и не сможет сойтись к нему полностью

5 Построение первого GAN для нашей задачи

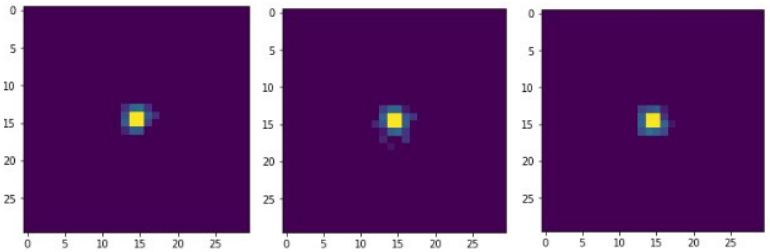
Как ранее уже упоминалось, данные - 10000 изображений 30x30 и моей задачей является создание GAN для генерации подобных изображений.

Распаковал данные, они представляют из себя подобные изображения:



Данные в том виде, что они есть сейчас представляются не очень хорошими для обучения на них GAN, т.к. содержат значения $-\infty$, поэтому прежде чем перейти к построению GAN данные были приведены к более удобному виду: тут могут быть разные варианты того, какую функцию взять для ограничения, я же решил использовать $f(\bullet) = \log(1 + \exp(\bullet))$. Это простое преобразование заменяет все $-\infty$ на 0, а у всех остальных данных незначительно увеличивает значения, не меняя отношения $\{>, <, =\}$ между ячейками. Так же очень легко строится обратное преобразование к начальным данным без потерь $f^{-1}(\bullet) = \log(\exp(\bullet) - 1)$.

Новые изображения:



Дальше строим собственно сам генератор и дискриминатор. В рамках проекта был освоен на базовом уровне tensorflow и keras, так же сделанно несколько небольших сторонних работ, чтобы лучше разобраться в том, как с ними работать, а так же научиться базовой работе с нейросетями.

Так как мы работаем с изображениями, то в качестве структуры генератора и дискриминатора были сделаны многослойные нейросети, состоящие преимущественно из свёрточных (Conv2D для дискриминатора) и развёрточных (Deconv2D для генератора) слоёв с функиями активации relu. Генератор начинается с полносвязного слоя из шума в изображение 7*7 и завершается одним свёрточным слоем с активацией tanh. Дискриминатор включает в себя два пулинга (MaxPool2D) и последние два слоя - это полносвязные слои с активациями tanh и softmax соответственно.

Функции потерь заданы как описано ранее, только дискриминатору ещё добавлена регуляризация.

В качестве оптимизатора взят градиентный спуск.

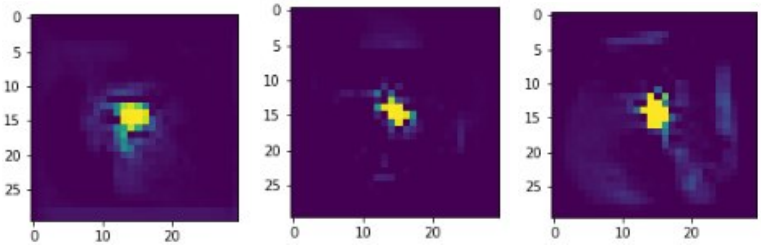
Процедура обучения и все промежуточные вещи (вроде создания шума для генератора из нормального распределения, мини-батч данных, вывод изображений и вывод сравнения ответов дискриминатора на реальных и сгенерированных данных) вынесены в отдельные функции, в которые передаются параметры для настройки процесса обучения.

6 Анализ полученных результатов

Первые же запуски GAN показали себя не очень удачно. Было очевидно, что дискриминатор достаточно сильно превосходит генератор и довольно быстро обучается на то, чтобы однозначно отделять реальные изображения от сгенрированных, в результате чего процесс обучения прекращался.

В качестве попытки справится с этой проблемой я производил отдельный подбор числа повторений обучения генератора и дискриминатора за одну эпоху, пытаясь так сделать генератор сильнее и стабилизировать обучение.

При определённых параметрах результаты уже визуально начинали походить на задачу, но качество всё ещё было недостаточно высоким:



7 Сравнение энергий

В момент, когда мы начинаем получать какие-то результаты (в нашем случае финальные изображения), хочется научиться оценивать качество полученных результатов и самого полученного генератора объективно, относительно нашей выборки, а не субъективно, с точки зрения дискриминатора.

Если посмотреть на изображения выборки, то можно увидеть что почти вся информация содержится в центральной части изображения. Предлагается рассматривать распределения суммы центральных пикселей изображения. Пусть $p_2 = \{$центральный квадрат 2:2$\}, p_4 = \{$центральный квадрат 4:4$\}, p_6 = \{$центральный квадрат 6:6$\}$ - множества пикселей, тогда:

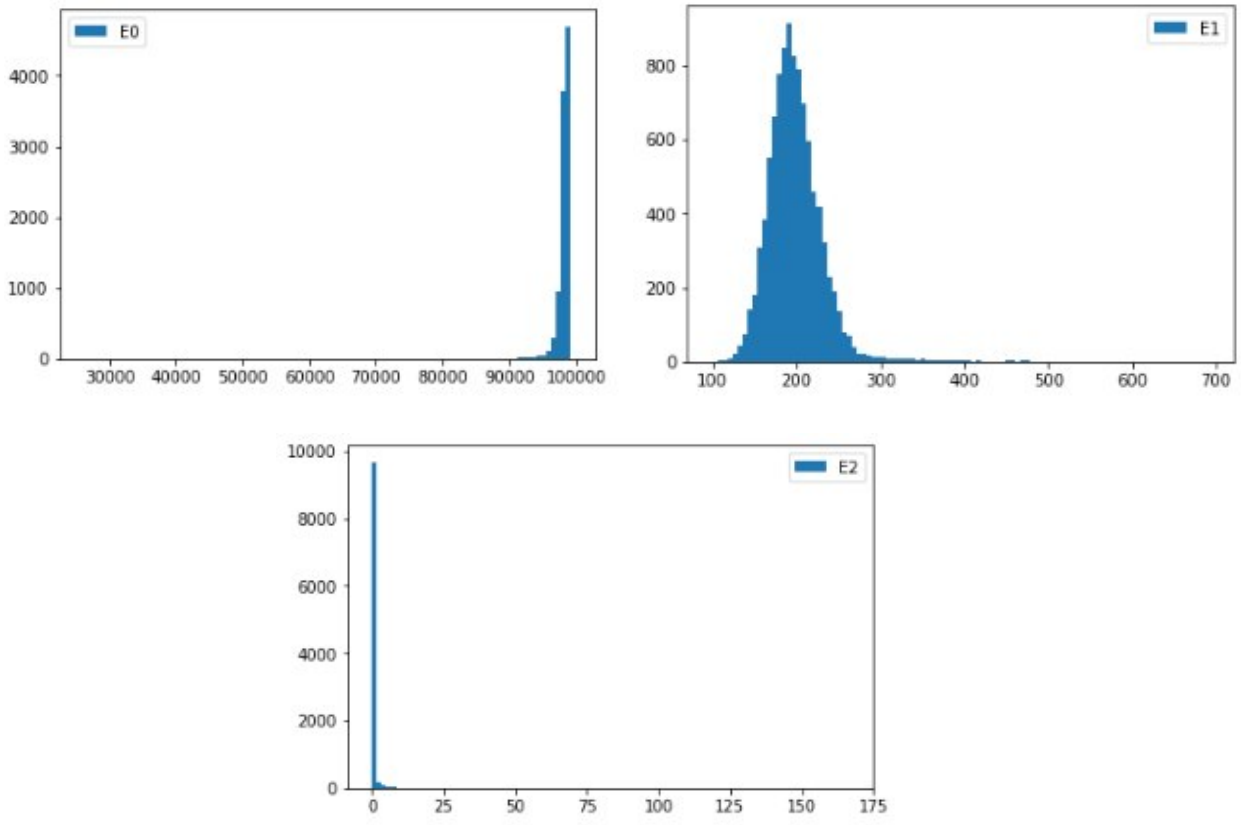
$$E_0 = \sum_{p_2}(\exp(\bullet))$$

$$E_1 = \sum_{p_4 \setminus p_2}(\exp(\bullet))$$

$$E_2 = \sum_{p_6 \setminus p_4}(\exp(\bullet))$$

- энергии распределений центральных пикселей $((2:2), (4:4 \setminus 2:2), (6:6 \setminus 4:4)$ соответственно)

Сравнение распределений энергий E_0, E_1, E_2 в реальных и сгенерированных данных - хорошая независимая метрика качества полученного генератора. Делать это проще всего графически. Так выглядят реальные распределения энергий:



8 Последующие вариации GAN

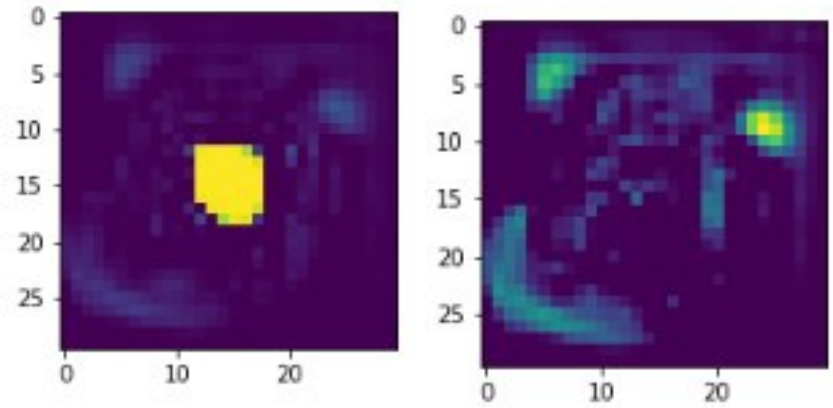
В попытках решить возникшую на первых подходах проблему были опробованы различные структуры генератора и дискриминатора. Разные параметры обучения и оптимизаторы. Попробованы другие функции, убирающие $(-\infty)$ из данных.

Распишу подробнее про наиболее результативное:

1. Так как дискриминатор всегда выигрывал, то очевидно что часть проблемы заключалась в нём, поэтому разумным решением оказалось ослабление структуры дискриминатора. Чисто уменьшение числа слоёв и весов привело к тому, что дискриминатору стало сложнее настраиваться, за счёт чего дискриминатор, хоть и продолжил выигрывать в большинстве ситуаций, но стал это делать на десятки тысяч эпох позже, а за генератор же за это время успевает больше обучится и, как следствие, результаты стали лучше.

2. В ходе работы время от времени на этапе обучения возникали "взрывы" генератора. Когда в течение одной серии из 50 эпох он переставал выдавать хоть что-либо похожее на требуемое изображение, хотя до этого постоянно выдавал изображения всё более похожие на данные. После этого генератор достаточно быстро проигрывал Вероятнее всего, это возникало в следствие слишком большого learning rate, из-за чего на очередном шаге оптимизации он "перескакивал" через оптимум. Уменьшение learning rate помогло исправить эту проблему в большинстве случаев.

   Пример "взрыва" генератора. Слева сгенерированное изображение до "взрыва" (на 7800 эпохе), а справа после (на 7850 эпохе):

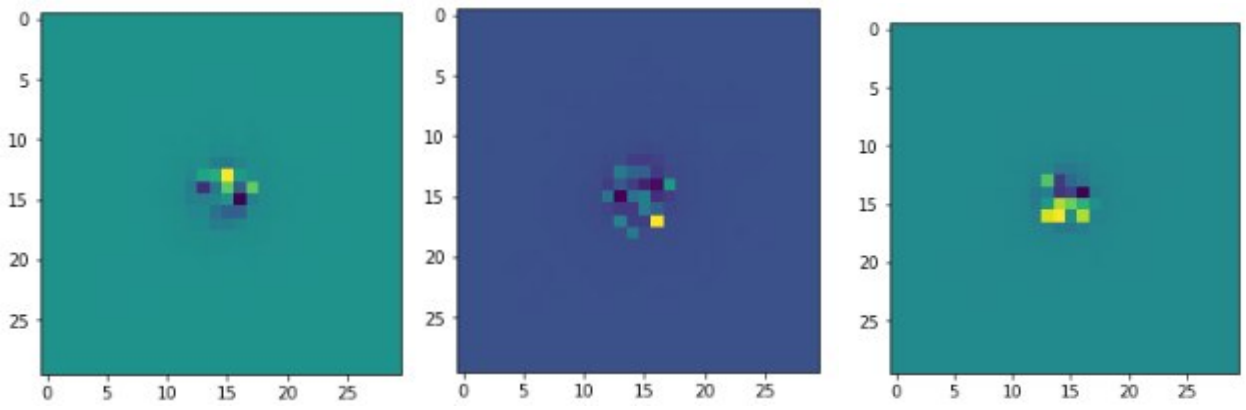


3. Следующим решением, давшим положительные результаты, оказалась смена функции, которая применяется к данным предварительно. В этот раз я решил предобработать данные иначе, а именно нормализовать exp от данных:

$$f(\bullet) = \frac{\exp(\bullet) - \mathbb{E}[\exp(\bullet)]}{\sqrt{\mathbb{D}[\exp(\bullet)]}}$$

$$f^{-1}(\bullet) = \log(\bullet * \sqrt{\mathbb{D}[\exp(\bullet)]} + \mathbb{E}[\exp(\bullet)])$$

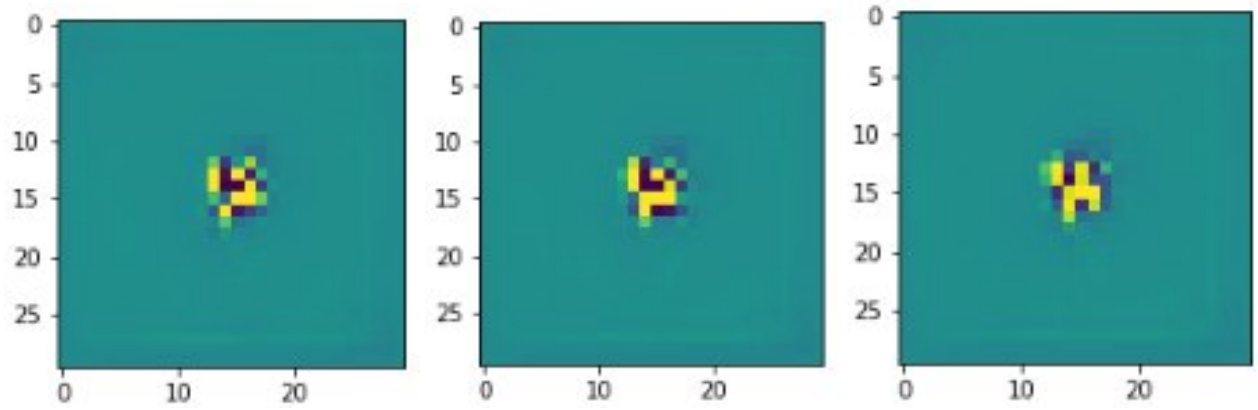
   Новые картинки для генерации выглядят так:



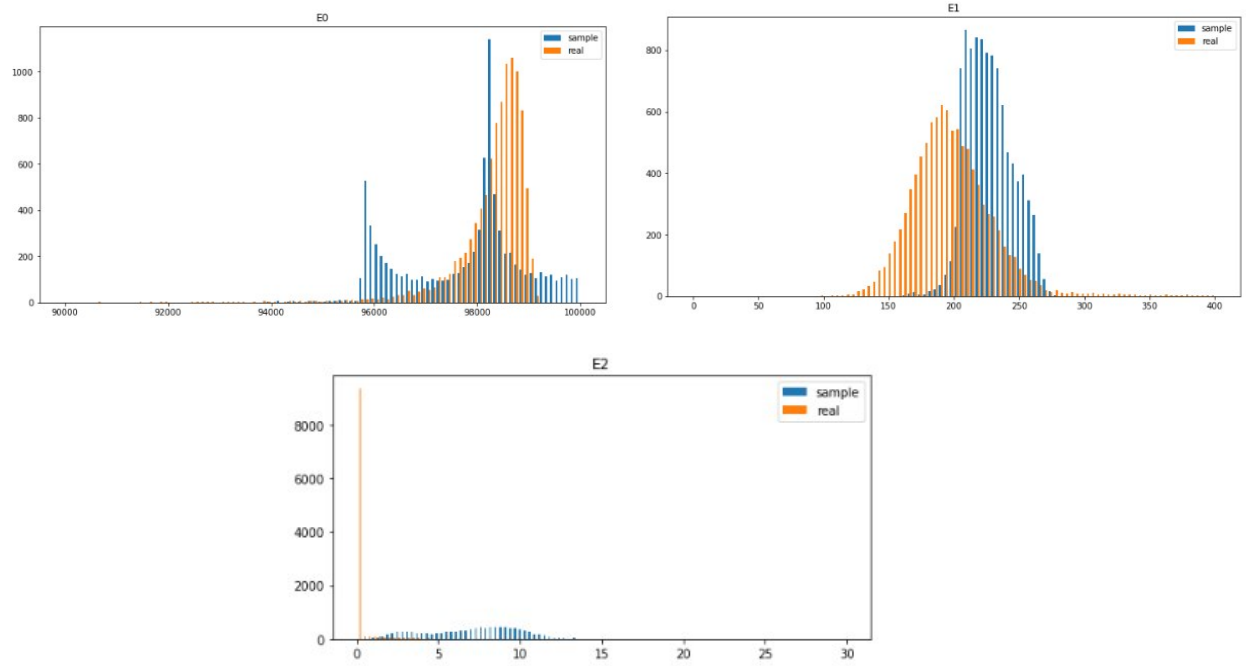
4. На новых данных я стал использовать Adam Optimizer вместо Gradient Descent и немного изменил функции потерь.

9 Результаты

Изображения, созданные генератором:

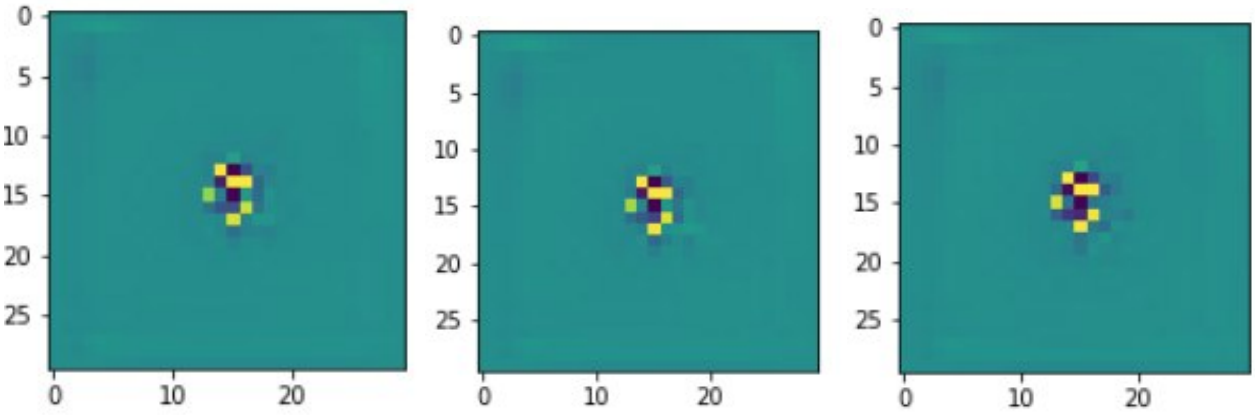


Графики сравнения энергий реальных и сгенерированных данных:

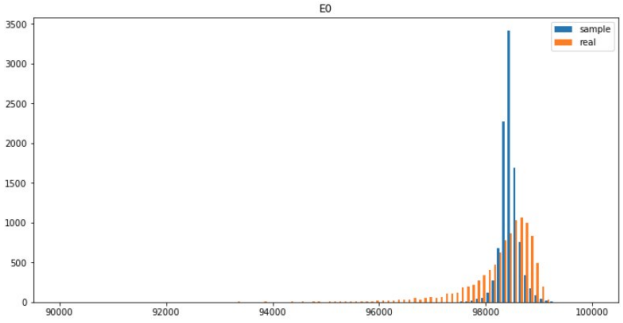


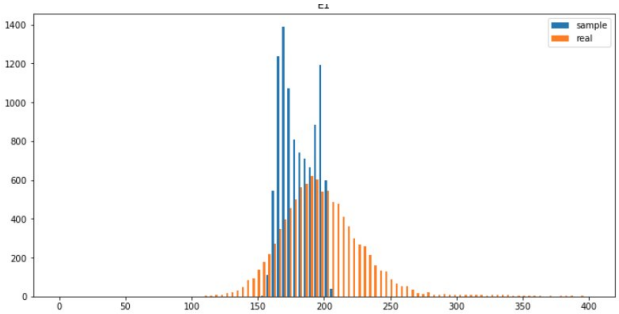
Результаты ещё одного теста:

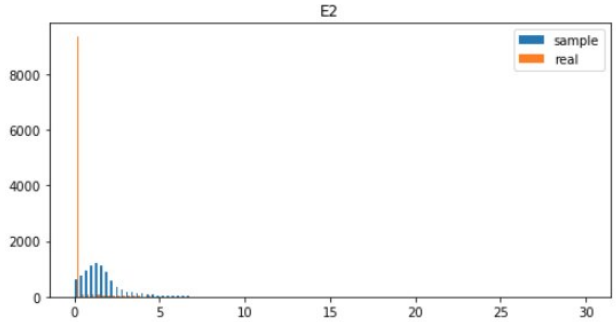
Изображения, созданные генератором:



Графики сравнения энергий реальных и сгенерированных данных:


E0
sample
real
3500
3000
2500
2000
1500
1000
500
0
90000
92000
94000
96000
98000
100000


sample
real
1400
1200
1000
800
600
400
200
0
0
50
100
150
200
250
300
350
400


E2
sample
real
8000
6000
4000
2000
0
0
5
10
15
20
25
30

10 Заключение

В данной работе были рассмотрены GAN для генерации ливней частиц в электромагнитных калориметрах. Насколько видно из результатов работы, GAN действительно преуспел в поставленной задаче и генерирует весьма качественные изображения при относительно небольших затратах ресурсов и времени. Однако, подобное решение ещё далеко от идеала. Сейчас появляется всё больше различных модификаций GAN и, может быть, какие-то из них смогут решить данную задачу ещё лучше.

В будущих работах можно попробовать различные модификации GAN, такие как Conditional GAN, Wasserstein GAN и пр.

Всё описанное в работе → https://github.com/Any0019/CaloGAN

Список литературы

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. 10 Jun 2014. URL: https://arxiv.org/pdf/1406.2661.pdf

[2] Michela Paganini, Luke de Oliveira and Benjamin Nachman. Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multi-Layer Calorimeters. 21 Dec 2017. URL: https://arxiv.org/pdf/1705.02355.pdf

[3] Michela Paganini, Luke de Oliveira and Benjamin Nachman. CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks. 21 Dec 2017. URL: https://arxiv.org/pdf/1712.10321.pdf