ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В.ЛОМОНОСОВА»

ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ И ИНФОРМАТИКИ

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

«НЕЙРОСЕТЕВАЯ РЕКОНСТРУКЦИЯ ПОРИСТЫХ ТЕЛ С СОХРАНЕНИЕМ ТОПОЛОГИЧЕСКИХ И СТАТИСТИЧЕСКИХ СВОЙСТВ ОБРАЗЦА»

Выполнил студент
235М группы:
Будакян Я. С.
____

Научный руководитель:
к.т.н., доц. Грачев Е. А.
____

Допущена к защите
Зав. кафедрой ____________

Москва

2019

Содержание

ВВЕДЕНИЕ

В настоящее время для эффективной добычи полезных ископаемых (в частности, нефти и газа) широко применяется математическое моделирование для симуляции процессов переноса, происходящих в пласте. Однако, прямое моделирование невозможно как минимум из-за неполноты данных о среде, в которой эти процессы протекают (данных о структуре и геометрии пласта). Эти данные доступны только в некотором количестве точек (скважин), из которых забирают пробу среды - керн. Керны представляют собой очень небольшой объем среды; если же говорить о данных компьютерной томографии керна, то она доступна на ещё более мелких масштабах - порядка микрометров. Получение дополнительных данных (например, новых кернов, компьютерной томографии) связано с большими затратами. При этом, многие эксперименты в реальности с керном можно провести только один раз, поскольку они необратимым образом влияют на керн. Таким образом возникает проблема “апскейлинга” - правдоподобного переноса знаний о локальной структуре среды на большие масштабы, а так же проблема переиспользования керна для проведения разных экспериментов.

В последнее время исследуется возможность применения методов машинного обучения для решения задач в этой области. Одним из возможных подходов является реконструкция новых моделей геометрии среды на основе реальных образцов с использованием искусственных нейронных сетей. Керны на микромасштабе являются пористой средой, поэтому наличие устойчивого алгоритма реконструкции образцов пористой среды позволит:

- попробовать провести апскейлинг (реконструировать среду в большем размере, чем оригинальный доступный образец)

- проводить статистические эксперименты (моделировать процессы не только на оригинальном образце, но и на его реконструкциях, получая таким образом распределение а, не значение в одной точке)

Под образцом пористой среды в данном случае понимаются данные компьютерной томографии керна - трёхмерное бинарное изображение.

Широко известны подходы к задачам синтеза двухмерных изображений с помощью искусственных нейросетей [**?**, 3], что мотивирует попробовать применить ИНС для задач реконструкции 3D-изображений.

Математически сформулировать постановку такой задачи реконструкции можно с помощью так называемой вероятностной постановки задачи обучения [**?**, **?**].

Рассмотрим многомерное пространство X, содержащее множество всех трёхмерных бинарных изображений x: $X = \{x\}$. Пусть у нас есть обучающая выборка из изображений, содержащих в себе рассматриваемое множество интересующих нас образцов $D = \{x_i\}$. Тогда считается, что обучающая выборка изображений D задаёт в этом пространстве вероятностное распределение $P_X : X \longrightarrow [0, 1]$, устроенное таким образом, что точки, соответствующие изображениям из выборки, имеют высокую вероятность, а остальные - низкую. Таким образом задача реконструкции образца пористой среды сводится к синтезу случайного изображения x', принадлежащего распределению, близкому к задаваемому обучающей выборкой:

$$P_{X'} \approx P_X, \quad x' \sim X'$$

“Классический” статистический подход к решению подобного рода задач заключается в введении параметризированного семейства распределений вероятности и его подстройке на имеющихся данных:

- Вводится семейство распределений вероятности $P_\theta(x)$ с параметром θ
- Параметр θ находятся из обучающей выборки:

$$\mathcal{L}_\theta(D) = \prod_{x \in D} P_\theta(x)$$

$$\theta^* = \arg\max_\theta \mathcal{L}_\theta(D)$$

- Генерируется объект (изображение) из распределения P_{θ^*}

Этот подход подвержен проблемам:

- Пространство параметров θ может быть огромной размерности
- Для сложных случаев невозможно априорно задать модель распределения

Простой пример объекта со сложным пространством параметров - человеческое лицо. Задачу генерации изображения реалистичного человеческого лица долгое время не могли решить с удовлетворительным качеством. Однако последние достижения в области искусственных нейронных сетей привели к существенному улучшению качества генеративных моделей самого разнообразного типа. В частности, впечатляющие результаты были достигнуты с помощью генеративных состязательных сетей (GAN) [**?**, **?**, **?**, **?**], что мотивирует попытку применения нейросетей этой архитектуры в поставленной задаче.

Постановка задачи

Для достижения обозначенной во введении цели, поставить задачу работы можно так:

- Реализовать модифицированные для реконструкции образцов пористых сред архитектуры нейронных сетей
- Провести вычислительные эксперименты, связанные с обучением нейросетей (то есть, с решением задач многопараметрической оптимизации)
- Реконструировать с помощью обученных нейросетей новые образцы и провести их верификацию

Верификация реконструированных образцов основана на сохранении их топологических и статистических характеристик, а именно:

- 4 первых функционала Минковского:
  - Объем
  - Площадь поверхности
  - Средняя кривизна
  - Характеристика Эйлера-Пуанкаре
- Двухточечная корреляционная функция

Обзор

TODO

1 Нейронные сети

ИНС - искусственная нейронная сеть - это математическая модель, построенная по принципу организации и функционирования биологических нейронных сетей. Она представляет собой систему соединённых простых блоков - искусственных нейронов, каждый из которых имеет входы и выходы для взаимодействия с другими нейронами. Главное преимущество нейронных сетей перед традиционными алгоритмами в том, что они обучаются на некотором наборе данных, а не программируются в классическом смысле этого понятия. Процесс обучения заключается в нахождении оптимальных весовых коэффициентов между нейронами. С математической точки зрения, процесс обучения - это задача многопараметрической нелинейной оптимизации.

1.1 Математическая модель нейрона

Одиночный нейрон обычно представляет собой взвешенный сумматор с нелинейной функцией активации на выходе:

$$x_{out} = \phi(\vec{w} \cdot \vec{x}_{in}),$$

где $\vec{w}$ - вектор весовых коэффициентов связей, $\vec{x}_{in}$ - входной вектор, ϕ - нелинейная функция активации (Рис. 1).

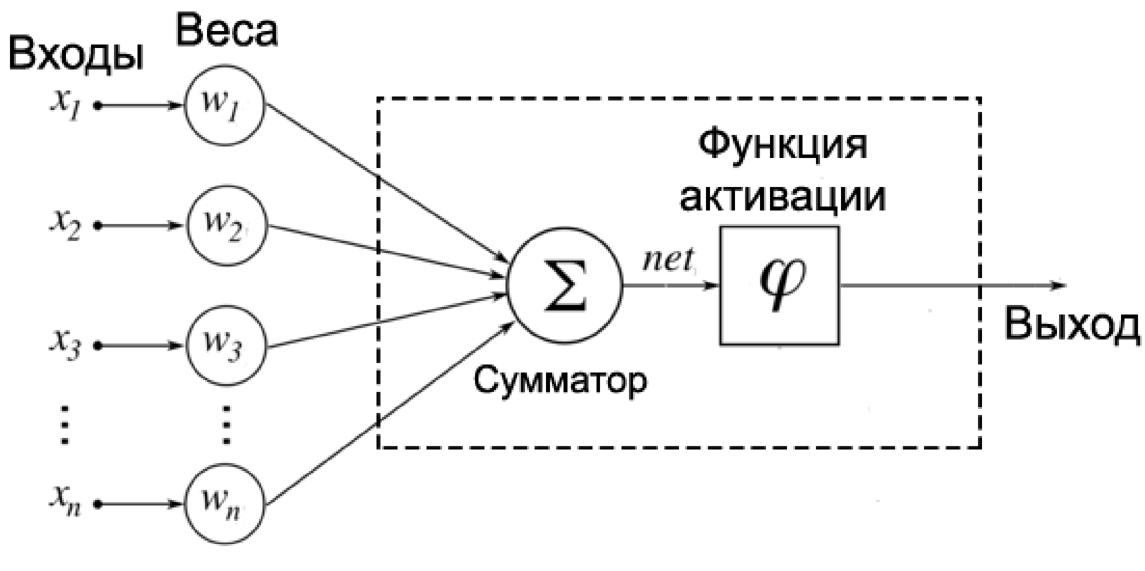


Рис. 1: Математическая модель нейрона

Функции активации могут выбираться разными в зависимости от задачи.

Наиболее часто используемые функции:

- Сигмоида (логистическая функция)

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Гиперболический тангенс

- ReLU

$$ReLU(x) = \max(0, x)$$

- softmax

$$\sigma(\vec{x})_j = \frac{e^{x_j}}{\sum_{k=1}^{N} e^{z_k}}$$

Множество таких нейронов объединяется в сеть и обучается каким-либо методом оптимизации.

1.2 Метод обратного распространения ошибки

Метод обратного распространения ошибки (backrpropagation) - самый широко используемый и успешный алгоритм обучения глубоких (многослойных) нейронных сетей. Суть метода заключается в распространении сигналов ошибки от выходов сети к ее входам в обратном к распространению сигнала в сети направлении. Это позволяет вычислить производные ошибки по весам сети, которые потом можно использовать в любом градиентном алгоритме оптимизации (например, градиентном спуске).

Обозначим множество входов сети как $\{x_1, \ldots, x_n\}$, множество выходов - O, w_{ij} - вес, присвоенный ребру, соединяющему i-й и j-й узлы, y_k - известные (правильные) ответы, o_i - выход i-го узла. Введём функцию ошибки (например, сумма квадратов расстояний):

$$L(\vec{x}, W) = \frac{1}{2} \sum_{k \in O} (y_k - o_k)^2,$$

где $W = \{w_{ij}\}$ - матрица весовых коэффициентов

Рассмотрим сначала нейроны последнего слоя. Весовой коэффициент w_{ij} влияет на выход сети как часть суммы $S_j = \sum_i w_{ij}x_i$. Соответственно,

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial S_j}\frac{\partial S_j}{\partial w_{ij}} = x_i\frac{\partial L}{\partial S_j}$$

Аналогично, S_j влияет на общую ошибку только в рамках выхода j-го узла o_j, поэтому

$$\frac{\partial L}{\partial S_j} = \frac{\partial L}{\partial o_j}\frac{\partial o_j}{\partial S_j} = \left(\frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k\in Out}(y_k - o_k)^2\right)\left(\frac{\partial \phi(S)}{\partial S}\bigg|_{S=S_j}\right)$$

Если узел j не находится на последнем слое, то у него есть набор связей с нейронами следующего слоя. Обозначим их множество как K_j. Тогда

$$\frac{\partial L}{\partial S_j} = \sum_{k\in K_j}\frac{\partial L}{\partial S_k}\frac{\partial S_k}{\partial S_j}$$

$$\frac{\partial S_k}{\partial S_j} = \frac{\partial S_k}{\partial o_j}\frac{\partial o_j}{\partial S_j} = w_{jk}\frac{\partial o_j}{\partial S_j}$$

$\frac{\partial L}{\partial S_k}$ - аналогичная поправка, но для нейрона следующего слоя. В итоге, получены выражения для производных ошибки по весам для нейронов выходного слоя, а аналогичные производные для нейронов внутренних слоев выражены через нейроны следующих слоев. Это и есть процесс обратного распространения ошибки - градиенты ошибки по весам вычисляются последовательно, начиная с выходного слоя и заканчивая первым.

1.3 Сверточные нейронные сети

Сверточные нейронные сети (CNN - convolutional neural networks) - это специальная архитектура нейронной сети, нацеленная на эффективное распознавание изображений, впервые предложенная Яном Лекуном [**?**]. Структура такой сети имеет некоторое сходство со строением зрительной коры головного мозга. Свое название CNN получили из-за наличия сверточных слоев, в которых каждый фрагмент изображения умножается на ядро свертки, полученный результат суммируется и записывается в аналогичную позицию выходного изображения. Одно отдельное ядро свертки обычно интерпретируют как кодирование какого-либо признака изображения. При этом сами ядра выучиваются сетью самостоятельно, а не закладываются человеком. В CNN чередуются сверточные и субдискретезирующие слои, таким образом более глубокие сверточные слои могут выделять абстрактные детали изображения, вплоть до общих понятий, таких как “кошка”, “собака”, и т.п. На данный момент CNN считаются базовым нейросетевым подходом при работе с изображениями.

1.4 Генеративные состязательные сети

Архитектура нейронной сети, получившая название генеративной состязательной сети (generative adversarial network - GAN), впервые была описана в 2014 году [**?**]. За последнее время сети такого типа добились больших успехов в задачах синтеза объектов из сложных распределений. Этим объясняется мотивация попытки применения данной архитектуры для решения поставленной задачи.

1.4.1 Общая структура

Переформулируем изначальную задачу нахождения такой процеруды синтеза X', что $P_{X'} \approx P_X$:

$$\rho(P_{X'}, P_X) \longrightarrow \min_{P_{X'}}$$

Введём параметризированную процедуру генерации:

$$X' = g_\theta(\cdot)$$

Получаем:

$$\rho(P_{X'}, P_X) \longrightarrow \min_{P_{X'}}$$

$$\rho(g_\theta(\cdot), P_X) \longrightarrow \min_{g_\theta(\cdot)}$$

$$\rho(g_\theta(\cdot), P_X) \longrightarrow \min_{\theta}$$

Возникает вопрос: что использовать в качестве метрики похожести двух распределений ρ, где одно задано обучающей выборкой. В качестве такой метрики можно использовать функцию потерь обученного классификатора, потому что естественно предположить, что чем чаще ошибается обученный классификатор, тем больше одно распределение похоже на другое. Тогда задача примет вид:

$$\rho(P_{X'}, P_X) \longrightarrow \min \Leftrightarrow L \longrightarrow \max,$$

где L - функция потерь обученного классификатора. Соответственно, можно ввести две нейросети:

- $d_\zeta(x)$ - классификатор для измерения расстояния, “дискриминатор”
- $g_\theta(x)$ - сеть, трансформирующая шум в элементы множества X', “генератор”

Суть использования двух сетей состоит в том, что они обучаются совместно, конкурируя друг с другом: генератор пытается имитировать целевое распределение, а дискриминатор пытается классифицировать поступающие от генератора и из обучающей выборки изображения на 2 класса: реальные (из изначального распределения P_X) и ложные (из $P_{X'}$, т.е. синтезированные генератором). Для дальнейшего рассмотрения введём функцию потерь дискриминатора (BCE - binary cross-entropy, logloss):

$$l_1 = l(d_\zeta(x), 1)$$

$$l_2 = l(d_\zeta(x'), 0)$$

$$L(X, X') = \frac{1}{2}\mathbb{E}_X l_1 + \frac{1}{2}\mathbb{E}_{X'} l_2 = -\frac{1}{2}(\mathbb{E}_X \log d_\zeta(x) + \mathbb{E}_{X'} \log(1 - d_\zeta(x'))) =$$
$$= -\frac{1}{2}(\mathbb{E}_X \log d_\zeta(x) + \mathbb{E}_V \log(1 - d_\zeta(g_\theta(v)))) = L(\zeta, \theta).$$

Функция потерь обученного классификатора:

$$L^*(\theta) = \min_\zeta L(\zeta, \theta)$$

Соответственно,

$$\min_\zeta L(\zeta, \theta) \longrightarrow \max_\theta$$

$$\theta^* = \arg\max_\theta \left[\min_\zeta L(\zeta, \theta)\right]$$

Определим оптимальный дискриминатор:

$$d_\theta^* = d_{\zeta^*(\theta)}$$

$$\zeta^*(\theta) = \arg\min_\zeta L(\zeta, \theta)$$

1.4.2 Обучение GAN

Итак, задача обучения GAN свелась к нахождению

$$\theta^* = \arg\max_\theta \left[\min_\zeta L(\zeta, \theta)\right]$$

В итоге, процесс обучения принимает следующий вид:

- Обучаем дискриминатор при фиксированном генераторе
- Обучаем генератор при фиксированном дискриминаторе
- Повторяем до сходимости параметров обеих моделей

Описанный процесс схематично изображён на (Рис. 2).

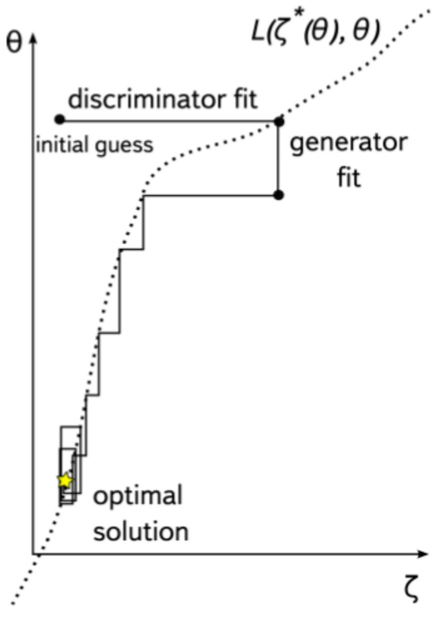
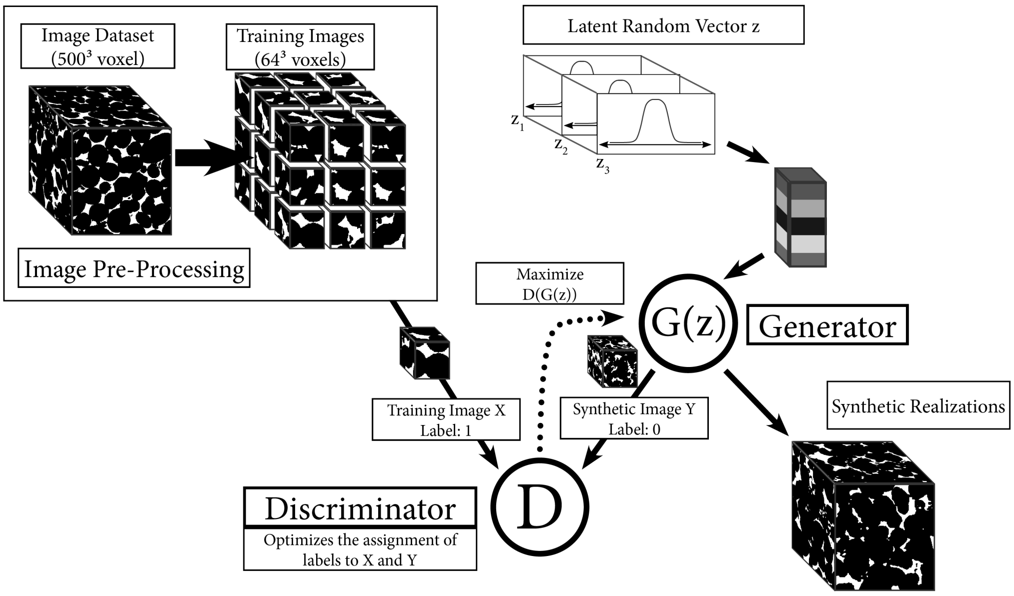


Рис. 2: Схематическое изображение процесса обучения GAN

1.4.3 Модификации “DCGAN” и “3D-DCGAN”

TODO



Ее обучение сводится к минимизации функционала:

$$\min_{\theta}\max_{\zeta} \mathbb{E}_{x\sim p_{data}} \log D_{\zeta}(x) + \mathbb{E}_{z\sim p_z} \log(1 - D_{\zeta}(G_{\theta}(z)))$$

2 Градиентные методы оптимизации

Для обучения глубоких нейросетей применяется метод обратного распространения ошибки, который позволяет рассчитать градиенты весов, и различные алгоритмы стохастической оптимизации. В данной работе для обучения моделей применялось два алгоритма оптимизации под названиями “RMSprop” и “Adam”, являющихся некоторыми модификациями стохастического градиентного спуска (SGD).

2.1 Градиентный спуск

TODO

2.2 Стохастический градиентный спуск

Описание стохастического градиентного спуска есть в [**?**]. Стохастический градиентый спуск обновляет каждый параметр путем вычитания градиента целевой функции по соответствующему параметру и умножая его на гиперпараметр λ - шаг обучения. Градиент целевой функции подсчитывается не на всем наборе данных, а на его случайном подмножестве.

$$i \sim \mathcal{U}\{1, 2, ..., n\}$$

$$\theta_{t+1} = \theta_t - \lambda \nabla L_i(\theta_t),$$

где L_i - целевая функция, вычисленная на i-ой части данных (mini-batch), индекс i выбирается случайно.

2.2.1 RMSprop

RMSprop (root mean square propagation) описан в [**?**]. Идея заключается в том, что для каждого параметра градиент перемасштабируется, учитывая прошлые значения градиентов для этого параметра. Производится это путем деления градиента на нормировочный множитель, который является корнем из среднего квадрата градиентов.

$$g_{t+1} = \gamma g_t + (1 - \gamma)(\nabla L_i(\theta_t))^2$$

$$\theta_{t+1} = \theta_t - \frac{\lambda \nabla L_i(\theta_t)}{\sqrt{g_{t+1} + \epsilon}}$$

ϵ - это небольшая константа, введенная для численной стабильности.

2.2.2 Adam

Adam (adaptive moment estimation, описан в [**?**]). Этот алгоритм, помимо перемасшабирования, использует инерцию градиента, что позволяет смягчить быстрое изменение градиента, присущее стохастическому градиентному спуску.

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) \nabla L_i(\theta_t)$$

$$v_{t+1} = \beta_2 g_t + (1 - \beta_2)(\nabla L_i(\theta_t))^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\lambda \hat{m}_t}{\sqrt{v_t + \epsilon}}$$

Авторы статьи [**?**] пишут, что этот алгоритм достаточно устойчив к неоптимальному выбору параметров, поэтому во многих статьях в начале для обучения пробуют применить именно Adam.

3 Верификация

Верификация синтеза

3.1 Функционалы Минковского

Функционалы Минковского для трехмерных тел вводятся следующим образом:

- $V = M_0 = \int_X dV$
- $S = M_1 = \frac{1}{3}\int_{\delta X} dS$
- $B = M_2 = \frac{1}{6}\int_{\delta X} \left(\frac{1}{R_1} + \frac{1}{R_2}\right) dS$
- $\xi = M_3 = \frac{1}{3}\int_{\delta X} \frac{1}{R_1 R_2} dS$

+ теория

3.1.1 Расчет для дискретного случая

Алгоритм расчета функционалов Минковского в дискретном случае: []

3.2 Двухточечная корреляционная функция

+ теория

4 Результаты вычислительных экспериментов

Архитектуры, описанные в пункте 1.4.3, а так же модификация процедуры обучения, описанная в 1.4.4 были реализованы в виде комплекса программ на языке Python с помощью библиотеки для глубокого обучения PyTorch. Использованные версии программных пакетов указаны в Приложении 1. Обучение проводилось на наборе данных Berea.

4.1 Набор данных Berea

Исходные данные - изображение компьютерной томографии песчаника, объёмом 400^3 вокселей, бинарно сегментированная на породу и поры, в формате TIFF. Для обучения сетей из этого кубика был вырезан набор кубиков размером 64^3 вокселей, с перекрытием в 16 вокселей, они и представили собой обучающую выборку. Оригинальный образец размером 400^3 вокселей представлен на (Рис. 3). Некоторые примеры из получившейся обучающей выборки представлены в (Таб. 1).

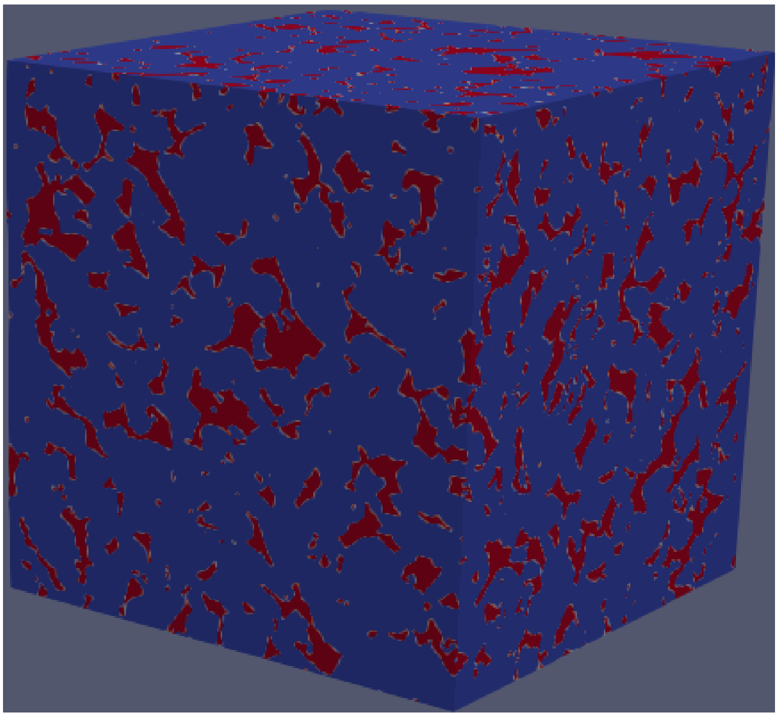


Рис. 3: Оригинальный образец

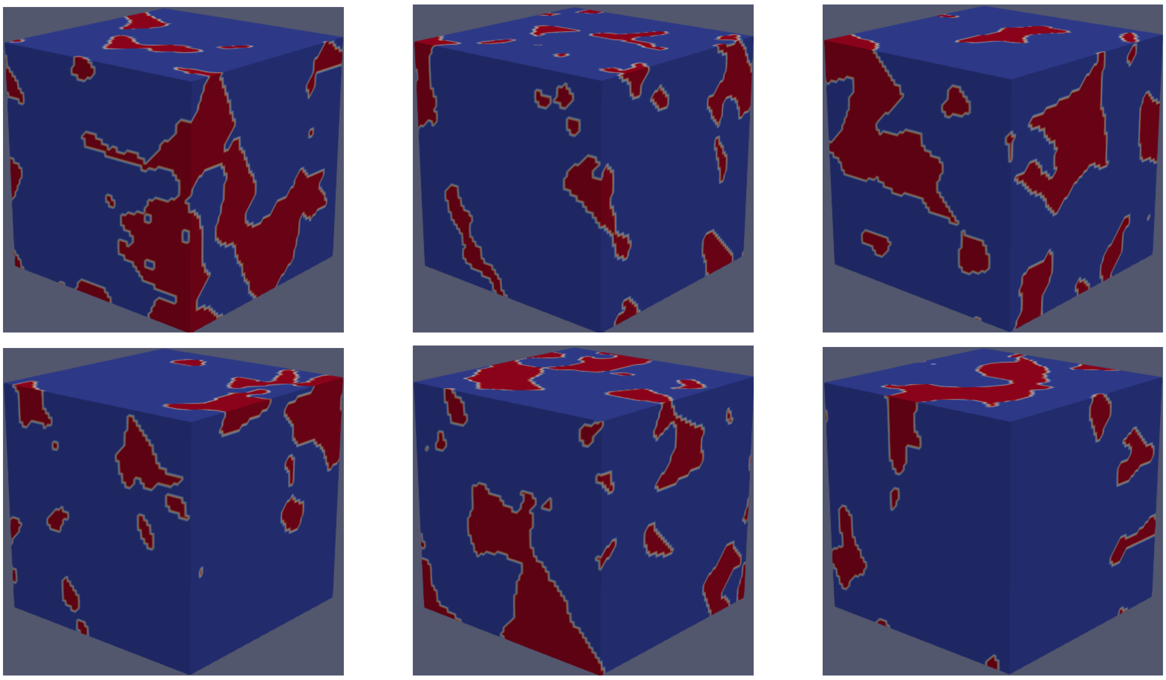


Таблица 1: Примеры из обучающей выборки

4.2 Обучение нейросети

Было проведено обучение нейросети со следующими параметрами (число сверточных фильтров указано для первого слоя):

Число фильтров G, D	Размерность z	Размер пакета	Начальный LR
64, 32	512	64	2e-4

Таблица 2: Гиперпараметры нейросети и процесса обучения

Графики функций ошибок сетей генератора и дискриминатора, а также графики сходимостей функционалов Минковского в процессе обучения приведены на (Рис. 4, 5, 6, 7, 8).

4.3 Реконструкции размера 64^3

4.3.1 Примеры синтеза

Примеры реконструкции размера 64^3, приведены в (Таб. 3).

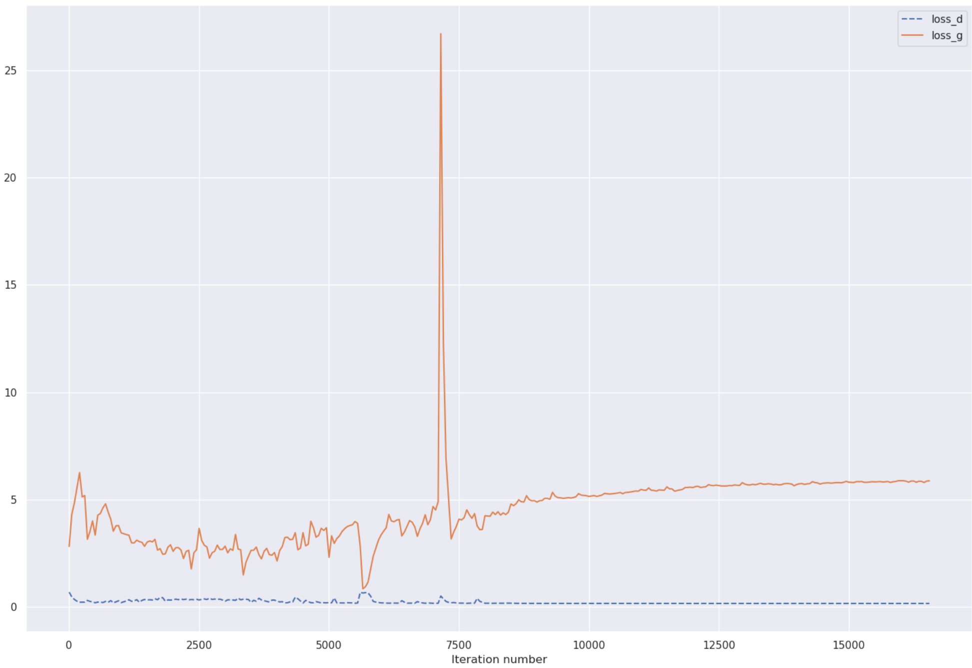


Рис. 4: График функций ошибок сетей дискриминатора и генератора

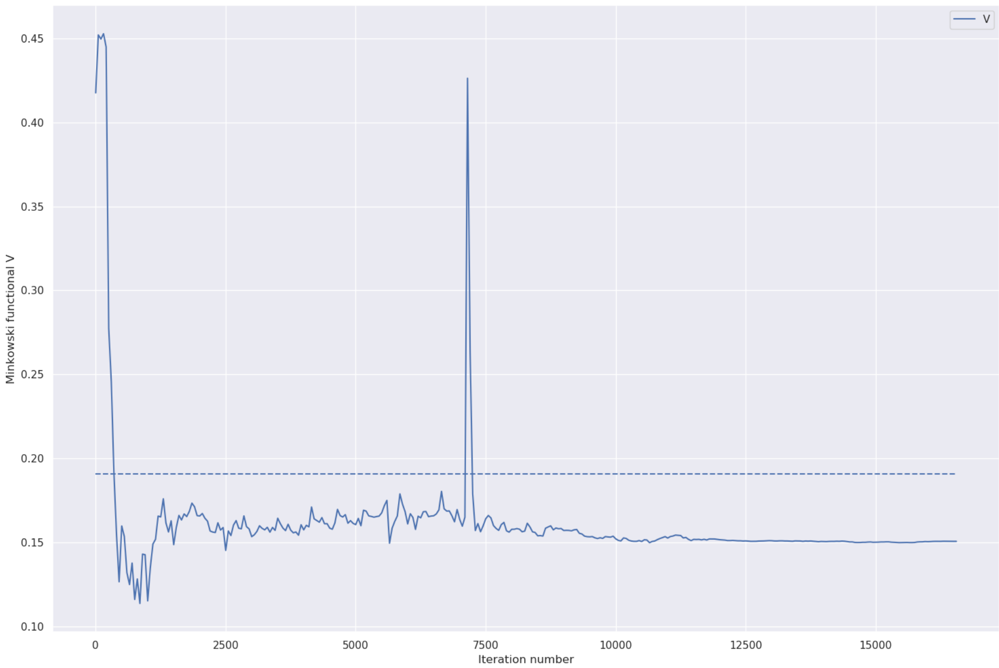


Рис. 5: График сходимости функционала Минковского V

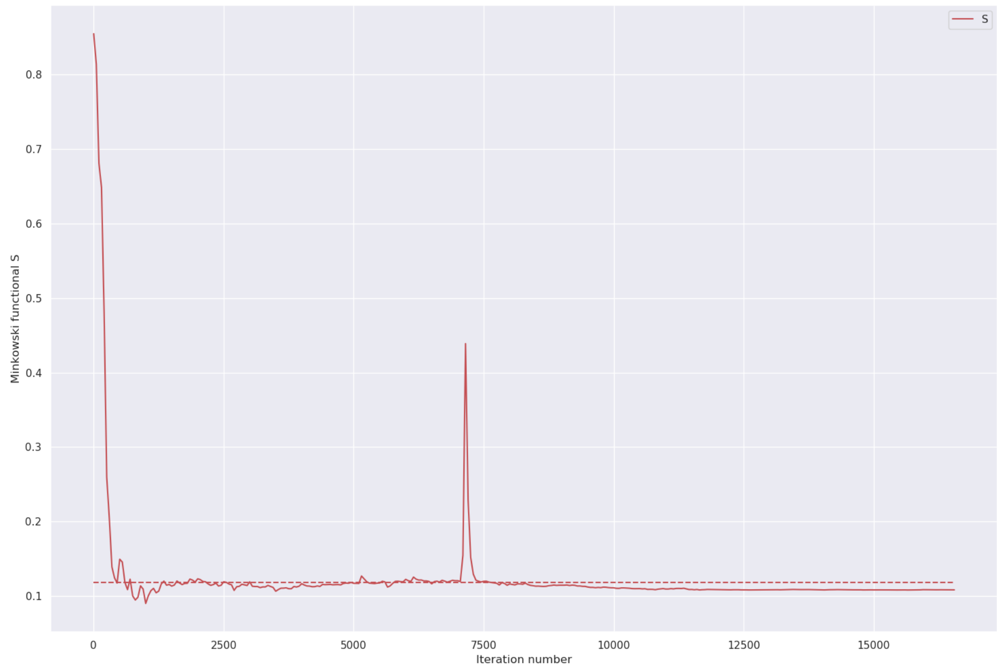


Рис. 6: График сходимости функционала Минковского S

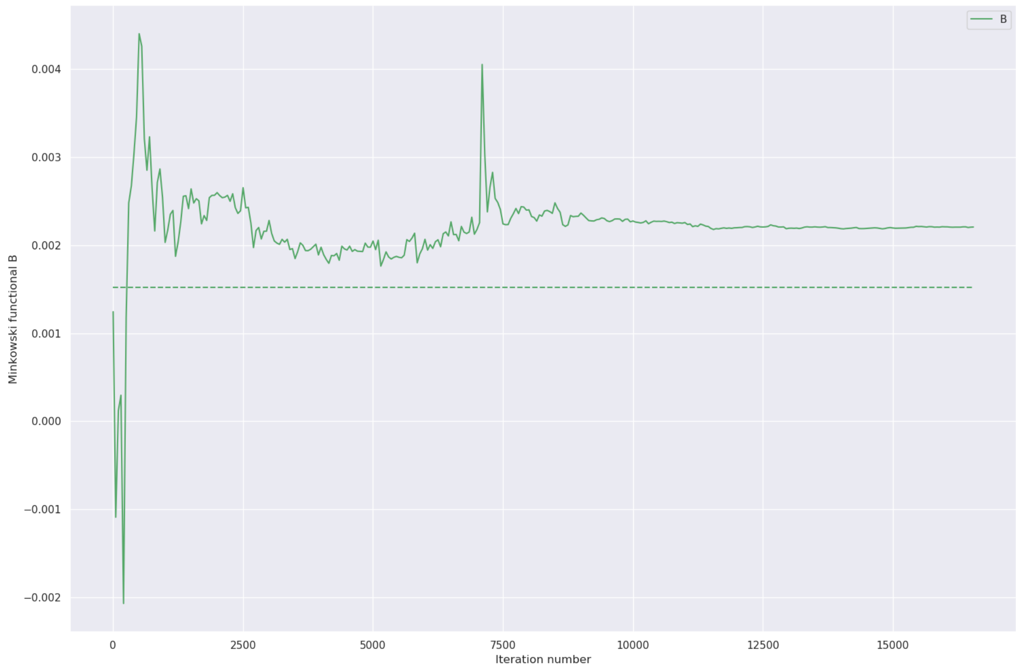


Рис. 7: График сходимости функционала Минковского B

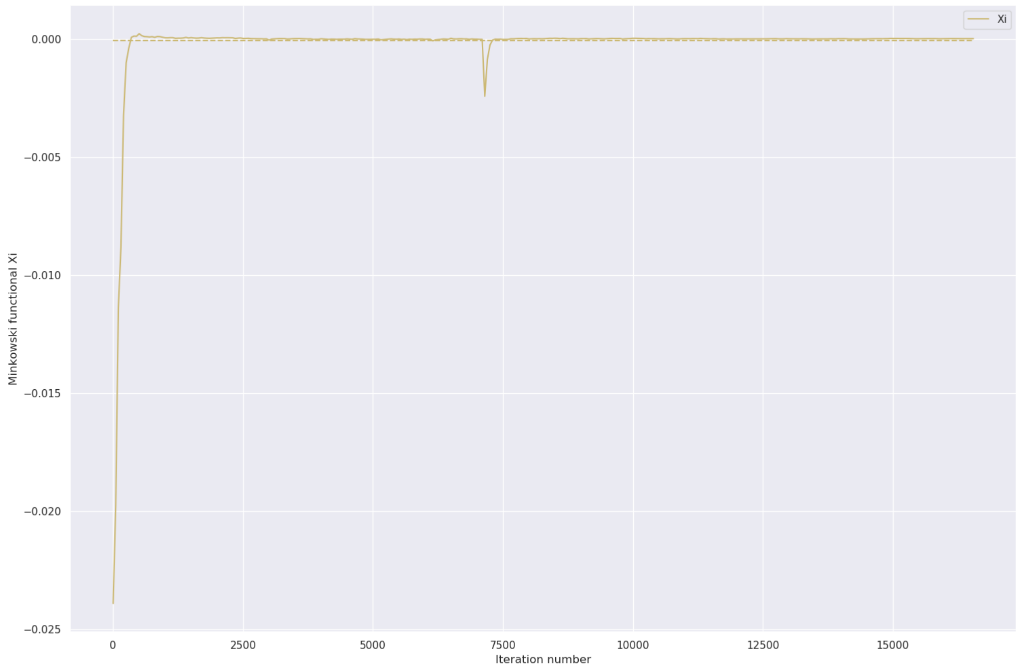


Рис. 8: График сходимости функционала Минковского ξ

4.3.2 Анализ реконструкций

Было реконструировано 100 образцов размера 64^3. На основе этого набора были получены распределения интересующих функционалов Минковского. Так же, используя предоставленную[1] сеть, было реконструировано 1000 образцов того же размера для сравнения распределений функционалов. Графики полученных распределений приведены на (Рис. 9, 10, 11, 12)

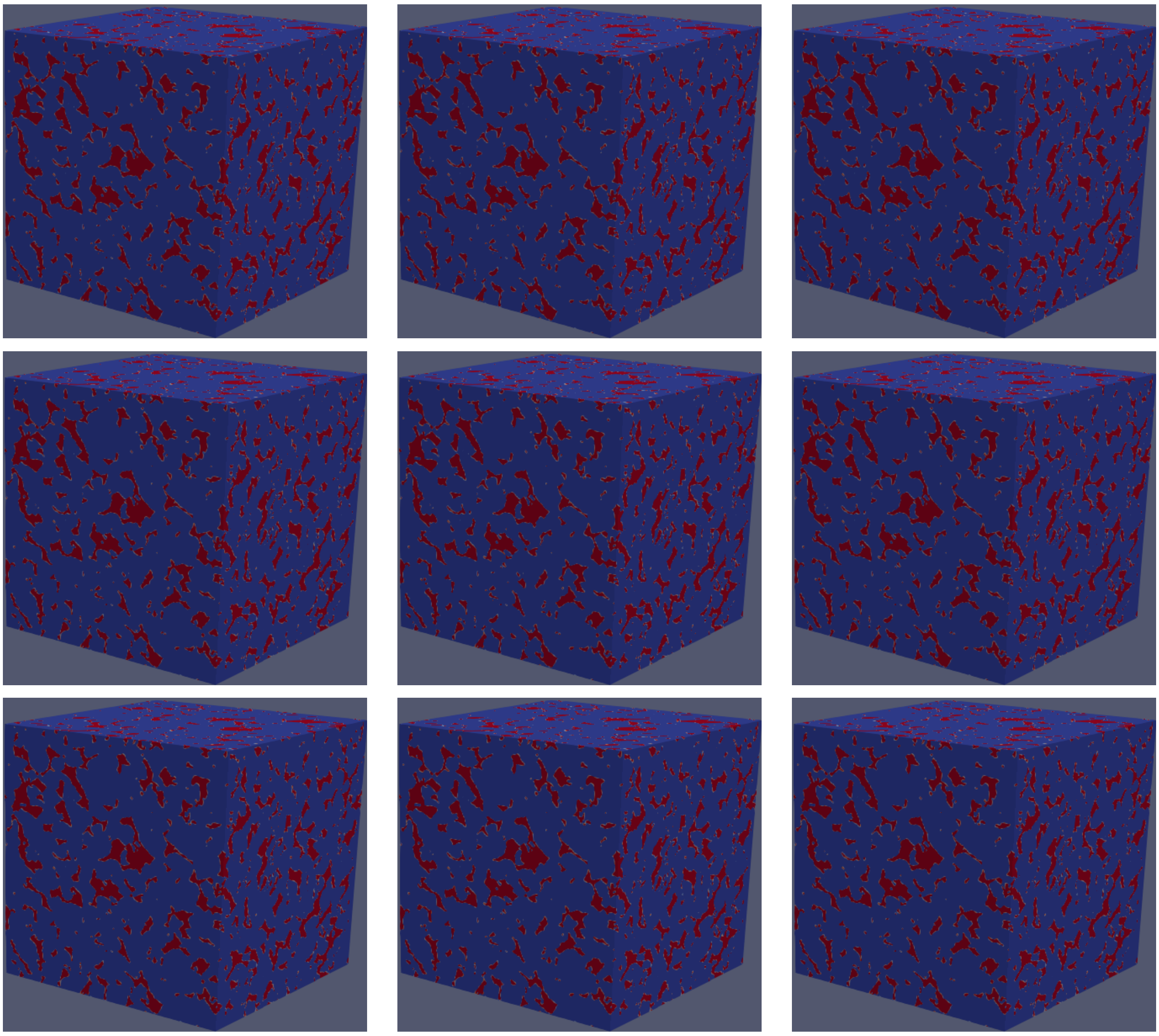


Таблица 3: Примеры реконструкции 64x64x64

Обученная сеть

Предоставленная[1] сеть

Рис. 9: Распределения функционала Минковского V

Обученная сеть

Предоставленная[1] сеть

Рис. 10: Распределения функционала Минковского S

Обученная сеть

Предоставленная[1] сеть

Рис. 11: Распределения функционала Минковского B

Обученная сеть Предоставленная[1] сеть

Рис. 12: Распределения функционала Минковского ξ

ВЫВОДЫ

TODO

ЗАКЛЮЧЕНИЕ

TODO

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Lukas Mosser, Olivier Dubrule, Martin J. Blunt: “Reconstruction of three-dimensional porous media using generative adversarial neural networks” // arXiv: 1704.03225 [cs.CV], 2017

[2] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge: “Texture Synthesis Using Convolutional Neural Networks” // arXiv: 1505.07376 [cs.CV], 2015.

[3] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, Victor Lempitsky: “Texture Networks: Feed-forward Synthesis of Textures and Stylized Images” // arXiv: 1603.03417 [cs.CV], 2016.

[4] Воронцов К. В.: “Математические методы обучения по прецедентам (теория обучения машин)”.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bign Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio: “Generative Adversarial Nets” // arXiv: 1406.2661 [stat.ML], 2014.

[6] Mehdi Mirza, Simon Osindero: “Conditional Generative Adversarial Nets” // arXiv: 1411.1784 [cs.LG], 2014.

[7] Jon Gauthier: “Conditional generative adversarial nets for convolutional face generation”, Tech. rep., 2015.

[8] Junbo Zhao, Michael Mathien, Yann LeCun: “Energy-based Generative Adversarial Networks” // arXiv: 1609.03126 [cs.LG], 2016.

[9] David Berthelot, Thomas Schumm, Luke Metz: “BEGAN: Boundary Equilibrium Generative Adversarial Networks” // arXiv: 1703.10717 [cs.LG], 2017.

[10] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: “Backpropagation applied to handwritten zip code recognition” // Neural Comput. 1(4), 541–551, 1989.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros: “Image-to-Image Translation with Conditional Adversarial Networks” // arXiv: 1611.07004 [cs.CV], 2016.

[12] Pedro Costa, Adrian Galdran, Maria Inês Meyer, Michael David Abràmoff, Meindert Niemeijer, Ana Maria Mendonça, Aurélio Campilho: “Towards Adversarial Retinal Image Synthesis” // arXiv: 1701.08974 [cs.CV], 2017.

[13] Olaf Ronneberger, Philipp Fischer, Thomas Brox: “U-Net: Convolutional Networks for Biomedical Image Segmentation” // arXiv: 1505.04597 [cs.CV], 2015.

[14] Amari, Shunichi: “A theory of adaptive pattern classifiers” // Electronic Computers, IEEE Transactions on 3, стр. 299-307, 1967.

[15] Tieleman, Tijmen, Geoffrey Hinton: “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude” // Coursera: Neural Networks for Machine Learning 4: 2, 2012.

[16] Diederik P. Kingma, Jimmy Lei Ba: “Adam: A method for stochastic optimization” // arXiv:1412.6980 [cs.LG], 2014.

[17] Martin Arjovsky, Soumith Chintala, Léon Bottou : “Wasserstein GAN” // arXiv: 1701.07875 [stat.ML], 2017

ПРИЛОЖЕНИЕ

Приложение 1. Использованные версии пакетов

- Python 3.7.3
- PyTorch 1.0.1
- ignite

Приложение 2. Дополнительные примеры генерации

КАРТИНКИ