

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

ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ И ИНФОРМАТИКИ

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

**«РЕКОНСТРУКЦИЯ ТРЁХМЕРНЫХ ПОРИСТЫХ СРЕД С ИСПОЛЬЗОВАНИЕМ
ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ»**

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	3
1 Нейронные сети	8
1.1 Математическая модель нейрона	8
1.2 Метод обратного распространения ошибки	9
1.3 Сверточные нейронные сети	10
1.4 Генеративные состязательные сети	11
1.4.1 Общая структура	11
1.4.2 Обучение GAN	13
1.4.3 Модификации “DCGAN” и “3D-DCGAN”	14
1.4.4 Модификация процедуры обучения	15
2 Градиентные методы оптимизации	17
2.1 Градиентный спуск	17
2.2 Стохастический градиентный спуск	17
2.2.1 RMSprop	18
2.2.2 Adam	18
3 Верификация	19
3.1 Функционалы Минковского	19
3.2 Двухточечная функция вероятности	20
4 Результаты вычислительного эксперимента	21
4.1 Набор данных Berea	21
4.2 Обучение нейросети	22
4.3 Реконструкции размера 64^3	22
4.3.1 Примеры	22
4.3.2 Анализ	23
4.4 Реконструкции размера 216^3	23
4.4.1 Примеры	23
4.4.2 Анализ	24
4.5 Реконструкции размера 360^3	24
4.5.1 Примеры	24
4.5.2 Анализ	24

ВЫВОДЫ	37
ЗАКЛЮЧЕНИЕ	38
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	39
ПРИЛОЖЕНИЕ	41

ВВЕДЕНИЕ

Сегодня для эффективной добычи полезных ископаемых (в частности, нефти и газа) широко применяется математическое моделирование для симуляции процессов переноса, происходящих в пласте. При этом, прямое моделирование невозможно как минимум из-за неполноты данных о среде, в которой эти процессы протекают (знаний 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 необратимо 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:

- проводить статистические эксперименты (моделировать процессы не только на оригинальном образце, но и на его реконструкциях, получая таким образом распределение a , не значение в одной точке)
- попробовать провести апскейлинг (реконструировать среду в большем размере, чем оригиналный доступный образец)

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 [4, 5], 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 [6, 7].

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 $P_X : X \rightarrow [0, 1]$, 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 параметризованного семейства распределений вероятности и его подстройке на имеющихся данных:

- Вводится семейство распределений вероятности $P_\theta(x)$ с параметром θ
- Параметр θ находится 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) [8, 9, 10, 11], что мотивирует попытку применения нейросетей этой архитектуры в поставленной задаче.

Постановка задачи

Для достижения обозначенных целей, поставить задачу данной 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:

- Характеристики, зависящие от 4-х первых функционалов Минковского:
 - Объем V
 - Площадь поверхности S
 - Средняя кривизна B
 - Характеристика Эйлера-Пуанкаре ξ
- Радиальная двухточечная функция вероятности $S_2(r)$

Подробное описание процедур верификации и используемых характеристик приведено в разделе 3.

Предшествующие работы

Рассмотренный подход к задаче уже исследовался [1]. В этой статье авторы используют архитектуры DCGAN [3] и 3D-GAN [2] для решения задачи реконструкции пористых сред. Один из своих экспериментов авторы проводили на тех же данных компьютерной томографии, что и в этой работе, что позволяет сравнивать полученные результаты. Основное отличие данной работы от [1] заключается модификации процедуры обучения сетей для автоматического управления одним из гиперпараметров обучения - шага обучения (learning rate, LR). Более подробно модификация описана в разделе 1.4.4.

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).

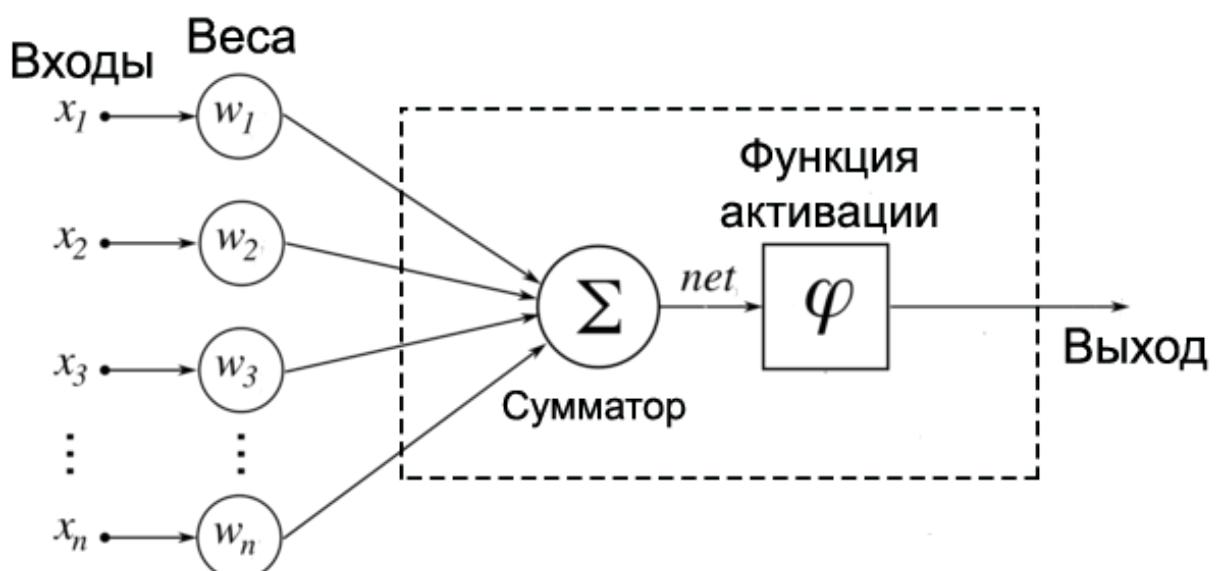


Рис. 1: Математическая модель нейрона

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 подобных нейронов соединяется в сеть и обучается с помощью метода обратного распространения ошибки вкупе с каким-либо методом численной оптимизации.

1.2 Метод обратного распространения ошибки

Метод обратного распространения ошибки (backpropagation) - 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 функционала ошибки по весам сети, которые потом можно использовать в любом градиентном алгоритме оптимизации (например, градиентном спуске).

Обозначим множество входов сети как $\{x_1, \dots, x_n\}$, 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

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

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 [12]. 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 интерпретиру-

ют как кодирование какого-либо признака изображения. При этом сами ядра выучиваются сетью самостоятельно, а не закладываются человеком. В CNN чередуются сверточные и субдискретизирующие 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 являются наиболее популярным алгоритмом машинного обучения 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 [7]. 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 процедуры синтеза X' такой что $P_{X'} \approx P_X$:

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

Введём параметризованную процедуру генерации:

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

Получаем:

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

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

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

В данном случае ρ выступает некоторой метрикой похожести двух вероятностных распределений. Идея GAN заключается в том, что в качестве такой метрики используется функционал ошибки другой нейросети - обученного

классификатора: чем чаще ошибается обученный классификатор, тем больше одно распределение похоже на другое. Тогда задача принимает вид:

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

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

- $d_\zeta(x)$ - классификатор для оценки ρ , “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 классифицирует поступающие от генератора и из обучающей выборки изображения на 2 класса: реальные (из изначально-го распределения P_X) и ложные (из $P_{X'}$, т.е. синтезированные генератором). Сигнал от дискриминатора возвращается в генератор и используется для обучения генератора “обману” дискриминатора. Для дальнейшего рассмотрения введём функцию потерь дискриминатора (BCE - binary cross-entropy, logloss):

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

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

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

Функция потерь обученного классификатора:

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

Соответственно,

$$\begin{aligned} \min_{\zeta} L(\zeta, \theta) &\rightarrow \max_{\theta} \\ \theta^* &= \arg \max_{\theta} \left[\min_{\zeta} L(\zeta, \theta) \right] \end{aligned}$$

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 оптимальных весов генератора

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

Процесс обучения такой системы сетей принимает итеративный вид:

- Обучается дискриминатор при фиксированном генераторе
- Обучается генератор при фиксированном дискриминаторе
- Повторяется до сходимости параметров обеих моделей

Описанный процесс схематично изображён на (Рис. 2).

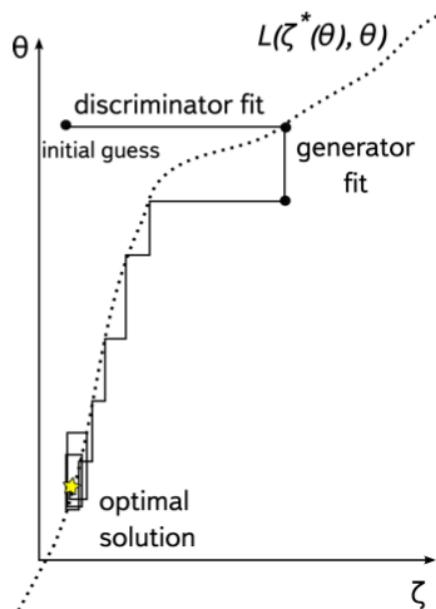


Рис. 2: Схематическое изображение процесса обучения GAN

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

Модификация архитектуры GAN под названием DCGAN [3] была предложена в 2015 году и совершила прорыв в области синтеза изображений. Суть архитектуры заключается в нескольких принципах построения сетей и их обучения. Во-первых, использование свёрточных слоев и пакетной нормализации (batch normalization) как в генераторе, так и дискриминаторе. В генераторе, при этом, используются специальные обратные свёрточные слои (transposed convolution, deconvolution). Во-вторых, использование специальной функции активации в дискриминаторе - *LeakyReLU*:

$$\text{LeakyReLU}(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

Все эти идеи позволяют стабилизировать сложный процесс обучения GAN и увеличить качество синтезируемых объектов.

Архитектура 3D-DCGAN совмещает в себе идеи [3] и [2]. Основное отличие от обычной DCGAN состоит в том, что вместо двухмерных свёрток во всех свёрточных слоях сетей используются трёхмерные, что и позволяет реконструировать трёхмерные среды.

Авторы [1] используют 3D-DCGAN для реконструкции пористых сред. Схема процесса обучения показана на (Рис. 3)

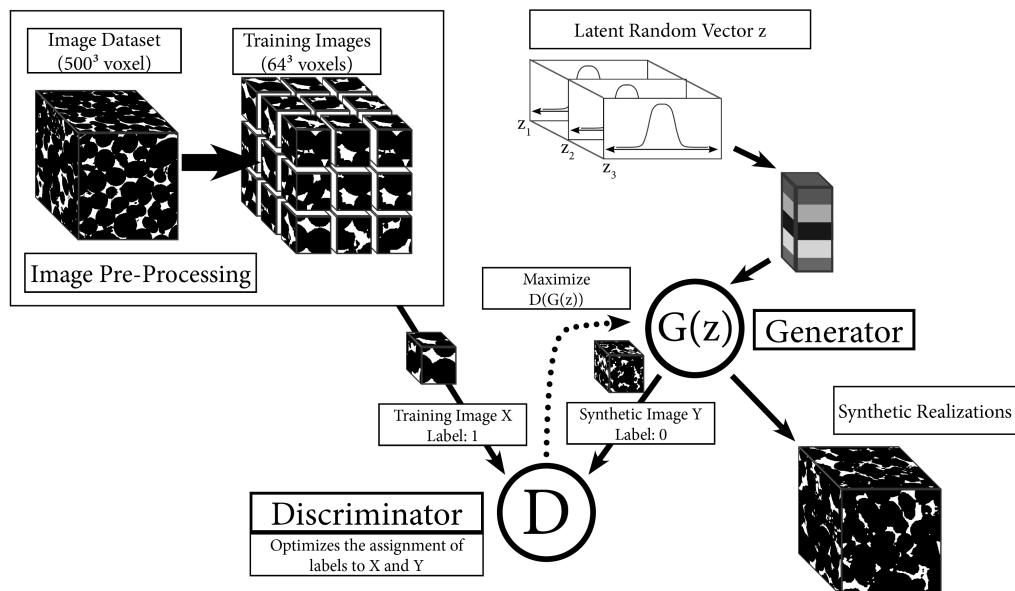


Рис. 3: Схема процесса обучения

Обучение 3D-GAN сводится к минимизации функционала:

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

1.4.4 Модификация процедуры обучения

Сеть, использованная в этой работе, полностью повторяет архитектуру сетей [1], однако в процедуру обучения сетей внесена модификация.

Скорость обучения - один из наиболее важных гиперпараметров (параметров, которые задаются человеком, а не оптимизируются в ходе процесса обучения). Её значение может радикально влиять на весь процесс сходимости сети [20]. В [1] авторы вручную управляют значением скорости обучения, изменяя её в зависимости от рассчитанных функционалов Минковского и визуальной структуры реконструируемых образцов. Это видно на графике профиля обучения сетей, приведённом в [1] (Рис. 4):

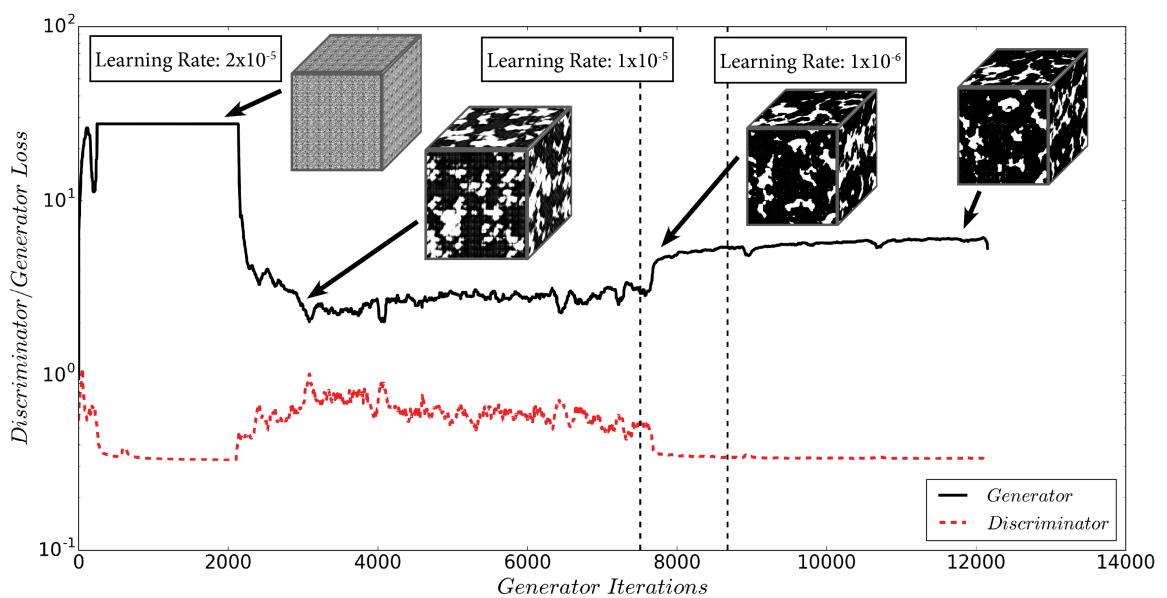


Рис. 4: Схема процесса обучения [1]

В этой работе исследуется применение модификации процедуры обучения,

которая состоит в том, что:

- Значения функционалов Минковского рассчитываются для реконструированных образцов на каждой итерации процесса обучения, что позволяет отслеживать текущее состояние сети-генератора с точки зрения совпадения топологических характеристик образцов с желаемыми;
- Это позволяет после каждого шага обучения оценить невязку Δ_i :

$$\Delta_i = |B_i - B_{berea}| + |\xi_i - \xi_{berea}|,$$

где B_i , ξ_i - это значения соответствующих функционалов Минковского для реконструированного образца с помощью генератора на текущем шаге обучения, а B_{berea} и ξ_{berea} - значения этих функционалов для исходного образца.

Если значение Δ_i не меняется более 3000 итераций, то скорость обучения уменьшается в 10 раз.

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 применялся алгоритм оптимизации “Adam”, являющийся модификацией стохастического градиентного спуска (SGD).

2.1 Градиентный спуск

Обычный градиентный спуск - простой итеративный алгоритм численной оптимизации. Для минимизации целевой функции численно рассчитывается ее градиент в этой точке, после чего совершается шаг в сторону антиградиента, т.е. из текущего значения параметра вычитается градиент целевой функции по данному параметру, умноженный на гиперпараметр λ - шаг обучения.

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

где L - целевая функция.

2.2 Стохастический градиентный спуск

Описание стохастического градиентного спуска есть в [16]. Стохастический градиентный спуск отличается от обычного тем, что градиент целевой функции подсчитывается не на всем наборе данных, а на его случайном подмножестве. Это широко используется при обучении нейронных сетей, поскольку высчитывание полного градиента функции ошибки по всем параметрам используя всю обучающую выборку приводит к слишком высоким временным затратам.

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

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

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

2.2.1 RMSprop

RMSprop (root mean square propagation) описан в [17]. Идея заключается в том, что для каждого параметра градиент перескалируется, учитывая прошлые значения градиентов для этого параметра. Производится это с помощью деления градиента на нормировочный множитель - корень из среднего квадрата градиентов.

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

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

где ϵ - это небольшая константа, введенная для численной стабильности.

2.2.2 Adam

Adam (adaptive moment estimation, описан в [18]). Этот алгоритм, помимо перескалирования, использует инерцию градиента, что позволяет смягчить быстрое изменение градиента, присущее стохастическому градиентному спуску.

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

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

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

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

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

Авторы статьи [18] пишут, что этот алгоритм достаточно устойчив к неоптимальному выбору параметров, поэтому во многих статьях в начале для обучения пробуют применить именно Adam.

3 Верификация

Для того, чтобы оценить “правдоподобие” реконструкций, необходима процедура верификации. В данной работе верификация заключается в проверке сохранения топологических и статистических свойств реконструированных образцов по сравнению с обучающими - четырёх функционалов Минковского и двухточечной корреляционной функции.

3.1 Функционалы Минковского

Функционалы Минковского для трёхмерных изображений в общем случае вводятся следующим образом:

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

Реконструкции являются бинарными трёхмерными изображениями, содержащими две фазы - среду и поры (пустоту). В дальнейшем все расчёты функционалов приведены для фазы пор. Также, поскольку абсолютные значения функционалов зависят от размера реконструкции, в дальнейшем рассматриваются их значения, нормированные на общий объем реконструкции. В этом случае, определения интересующих характеристик можно переписать следующим образом:

- $V = \frac{1}{V_{all}} \int_{V_{pore}} dV = \frac{V_{pore}}{V_{all}}$
- $S = \frac{1}{V_{all}} \int_{\delta V_{pore}} dS$
- $B = \frac{1}{V_{all}} \int_{\delta V_{pore}} \left(\frac{1}{R_1} + \frac{1}{R_2} \right) dS$

$$\bullet \quad \xi = \frac{1}{V_{all}} \int_{\delta V_{pore}} \frac{1}{R_1 R_2} dS,$$

где δV_{pore} - граница двух фаз (среды и пор). С точки зрения интерпретации, характеристики V, S, B, ξ соответствуют пористости, удельной площади поверхности, удельной кривизны поверхности и числу Эйлера-Пуанкаре (некоторая характеристика связности).

Поскольку реконструкции представляют собой дискретизированные бинарные изображения, для расчёта функционалов можно не численно вычислять интегралы, а воспользоваться эффективным алгоритмом для дискретного случая [21].

3.2 Двухточечная функция вероятности

Статистические характеристики пористой среды можно выразить с помощью двухточечной функции вероятности. Предполагая стационарность, эта функция совпадает с нецентрированной ковариацией [22, 1]:

$$S_2(r) = P(\mathbf{x} \in P, \mathbf{x} + \mathbf{r} \in P), \quad \mathbf{x}, \mathbf{r} \in \mathbb{R}^d$$

P - это вероятность того, что две точки, отстоящие друг от друга на вектор \mathbf{r} , принадлежат одной фазе (обе являются порами). Для расчёта радиальной функции вероятности сначала рассчитываются соответствующие функции вдоль x, y, z , а потом усредняются.

4 Результаты вычислительного эксперимента

Архитектуры нейронных сетей, а также модификация процедуры обучения, описанные в пунктах 1.4.3 и 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. Разрешение томограммы равно 3 микрометрам. Для обучения сетей из этого кубика был вырезан набор кубиков размером 64^3 вокселей, с перекрытием в 16 вокселей (всего 10648 различных образцов размера 64^3), они и представили собой обучающую выборку. Оригинальный образец размером 400^3 вокселей представлен на (Рис. 5). Некоторые примеры из получившейся обучающей выборки представлены в (Таб. 1).

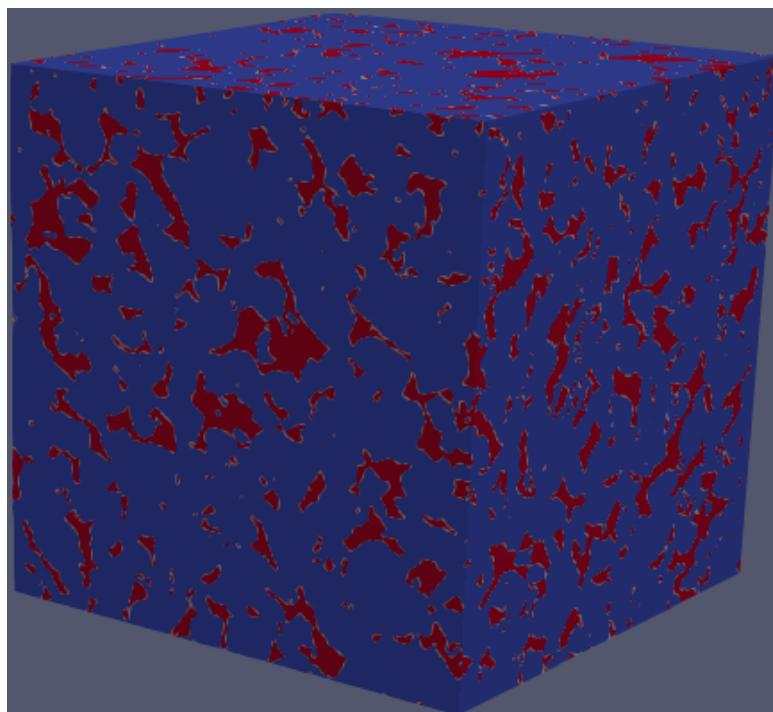


Рис. 5: Оригинальный образец

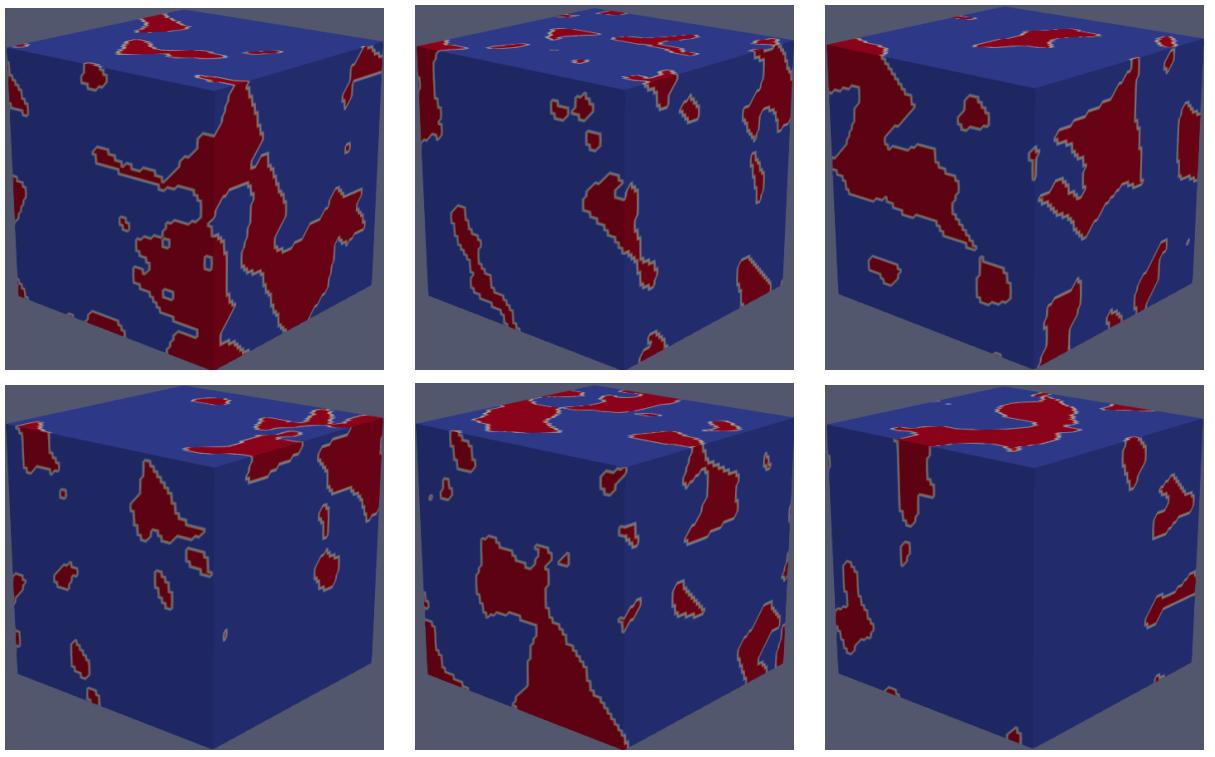


Таблица 1: Примеры из обучающей выборки

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

Итоговые архитектуры использованных нейронных сетей генератора и дискриминатора указаны в (Таб. 3, 4)

Графики функций ошибок сетей генератора и дискриминатора, а также графики сходимостей функционалов Минковского в процессе обучения приведены на (Рис. 6, 7, 8, 9, 10).

4.3 Реконструкции размера 64^3

4.3.1 Примеры

Примеры реконструкции размера 64^3 , приведены в (Таб. 5). Дополнительные примеры реконструкций приведены в Приложении 2 (Таб. 8).

Слой	Размер ядра	Размерность выхода	Кол-во параметров
0_ConvTranspose3d	[256, 512, 4, 4, 4]	[1, 256, 4, 4, 4]	8 388 610
1_BatchNorm3d	[256]	[1, 256, 4, 4, 4]	512
2_ReLU	-	[1, 256, 4, 4, 4]	-
3_ConvTranspose3d	[128, 256, 4, 4, 4]	[1, 128, 8, 8, 8]	2 097 150
4_BatchNorm3d	[128]	[1, 128, 8, 8, 8]	256
5_ReLU	-	[1, 128, 8, 8, 8]	-
6_ConvTranspose3d	[64, 128, 4, 4, 4]	[1, 64, 16, 16, 16]	524 290
7_BatchNorm3d	[64]	[1, 64, 16, 16, 16]	128
8_ReLU	-	[1, 64, 16, 16, 16]	-
9_ConvTranspose3d	[32, 64, 4, 4, 4]	[1, 32, 32, 32, 32]	131 070
10_BatchNorm3d	[32]	[1, 32, 32, 32, 32]	64
11_ReLU	-	[1, 32, 32, 32, 32]	-
12_ConvTranspose3d	[1, 32, 4, 4, 4]	[1, 1, 64, 64, 64]	2 050
13_Tanh	-	[1, 1, 64, 64, 64]	-

Таблица 3: Архитектура генератора

4.3.2 Анализ

Было реконструировано 1000 образцов размера 64^3 . На основе этого набора были получены распределения интересующих функционалов Минковского. Так же, используя предоставленную обученную сеть [1], было реконструировано 1000 образцов того же размера для сравнения распределений функционалов. Графики полученных распределений (вместе с соответствующим распределением на обучающей выборке) приведены на (Рис. 11, 12, 13, 14). Была рассчитана двухточечная функция вероятности для реконструированных образцов, образцов, полученных с помощью предобученной сети [1] и образцов из обучающей выборки. Её график приведён на (Рис. 15).

4.4 Реконструкции размера 216^3

4.4.1 Примеры

Примеры реконструкции размера 216^3 , приведены в (Таб. 6). Дополнительные примеры реконструкций приведены в Приложении 2 (Таб. 9).

Слой	Размер ядра	Размерность выхода	Кол-во параметров
0_Conv3d	[1, 32, 4, 4, 4]	[1, 32, 32, 32, 32]	2 050
1_LeakyReLU	-	[1, 32, 32, 32, 32]	-
2_Conv3d	[32, 64, 4, 4, 4]	[1, 64, 16, 16, 16]	131 070
3_BatchNorm3d	[64]	[1, 64, 16, 16, 16]	128
4_LeakyReLU	-	[1, 64, 16, 16, 16]	-
5_Conv3d	[64, 128, 4, 4, 4]	[1, 128, 8, 8, 8]	524 290
6_BatchNorm3d	[128]	[1, 128, 8, 8, 8]	256
7_LeakyReLU	-	[1, 64, 16, 16, 16]	-
8_Conv3d	[128, 256, 4, 4, 4]	[1, 256, 4, 4, 4]	2 097 150
9_BatchNorm3d	[256]	[1, 256, 4, 4, 4]	512
10_LeakyReLU	-	[1, 256, 4, 4, 4]	-
11_Conv3d	[256, 1, 4, 4, 4]	[1, 1, 1, 1, 1]	16 380
12_Sigmoid	-	[1, 1, 1, 1, 1]	-

Таблица 4: Архитектура дискриминатора

4.4.2 Анализ

Было реконструировано 500 образцов размера 216^3 . На основе этого набора были получены распределения интересующих функционалов Минковского. Так же, используя предоставленную обученную сеть [1], было реконструировано 500 образцов того же размера для сравнения распределений функционалов. Графики полученных распределений приведены на (Рис. 16, 17, 18, 19). Была рассчитана двухточечная функция вероятности для реконструированных образцов, образцов, полученных с помощью предобученной сети [1] и образцов из обучающей выборки. Её график приведён на (Рис. 20).

4.5 Реконструкции размера 360^3

4.5.1 Примеры

Примеры реконструкции размера 360^3 , приведены в (Таб. 7). Дополнительные примеры реконструкций приведены в Приложении 2 (Таб. 10).

4.5.2 Анализ

Было реконструировано 500 образцов размера 360^3 . На основе этого набора были получены распределения интересующих функционалов Минковского. Так же, используя предоставленную обученную сеть [1], было реконструиро-

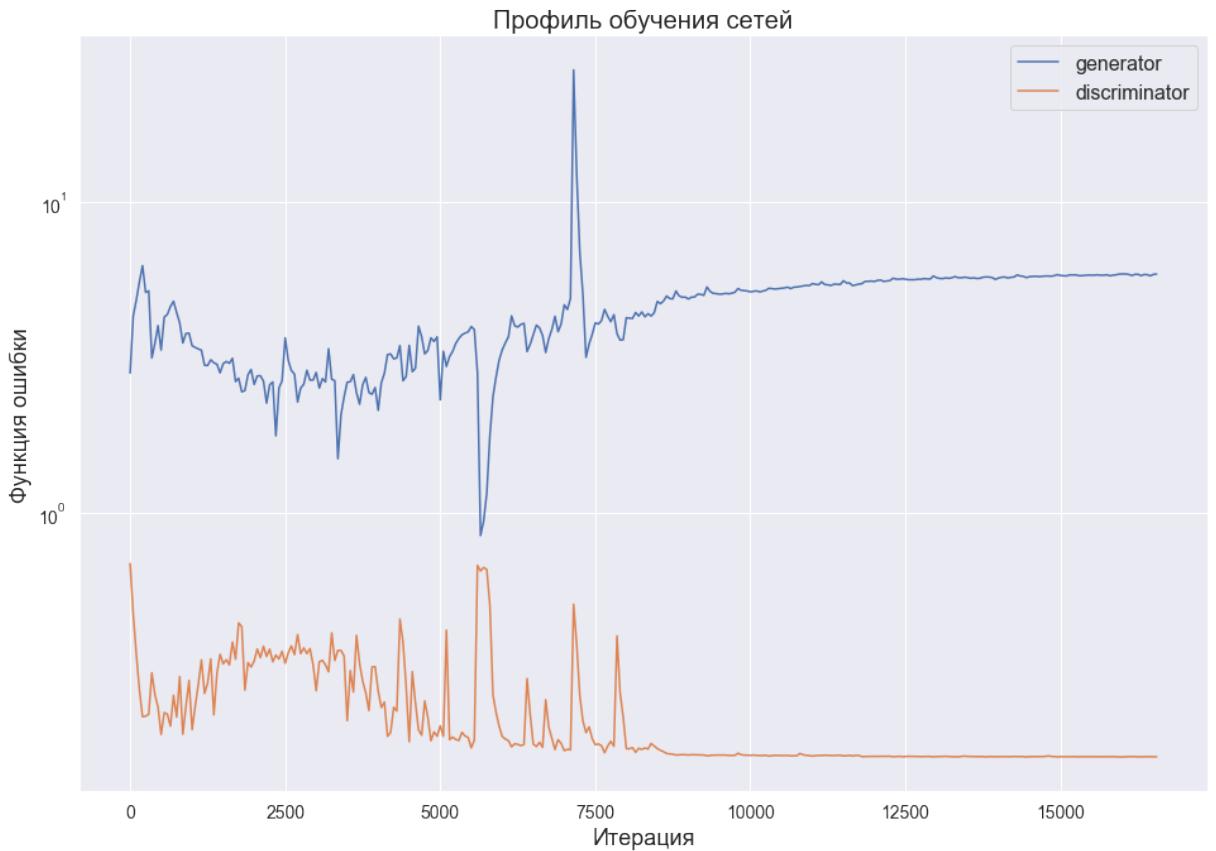


Рис. 6: График функций ошибок сетей дискриминатора и генератора

вано 1000 образцов того же размера для сравнения распределений функционалов. Графики полученных распределений приведены на (Рис. 21, 22, 23, 24). Была рассчитана двухточечная функция вероятности для реконструированных образцов, образцов, полученных с помощью предобученной сети [1] и образцов из обучающей выборки. Её график приведён на (Рис. 25).

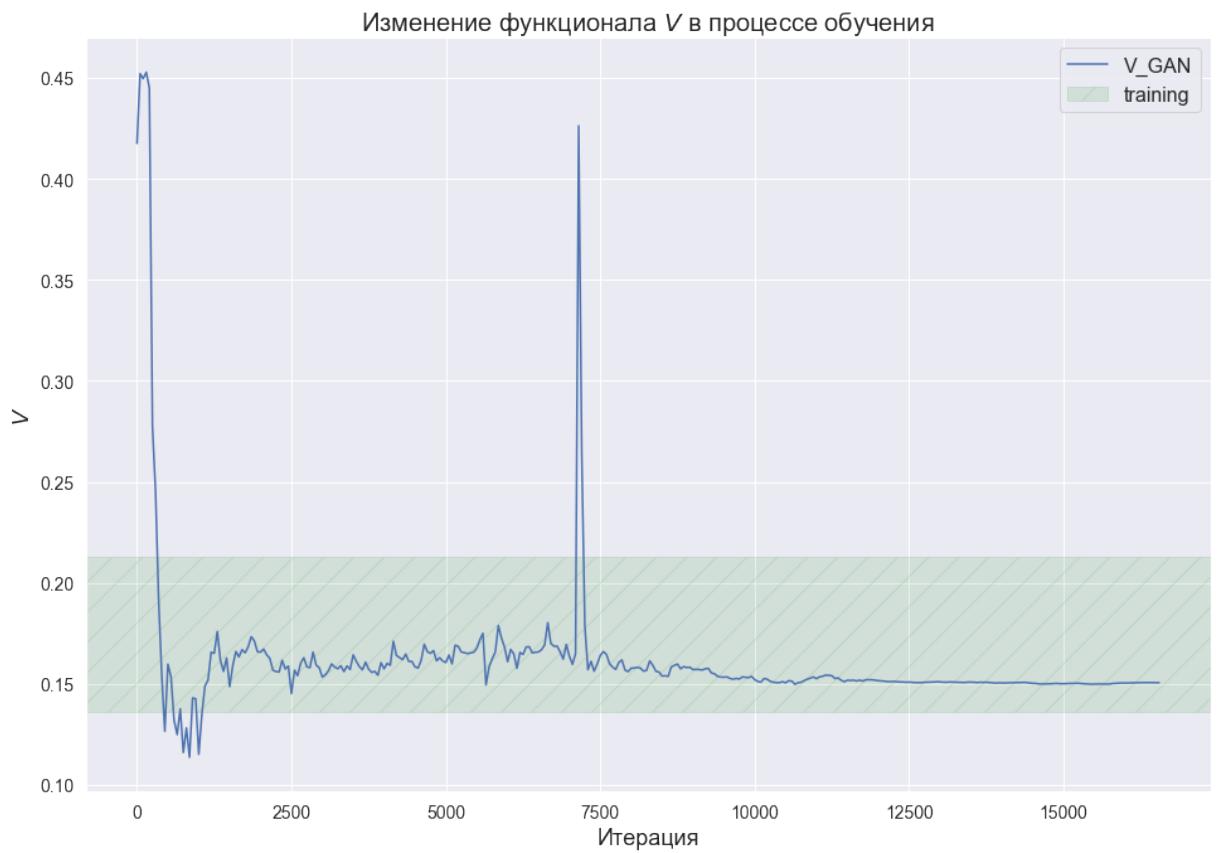


Рис. 7: График сходимости функционала Минковского V

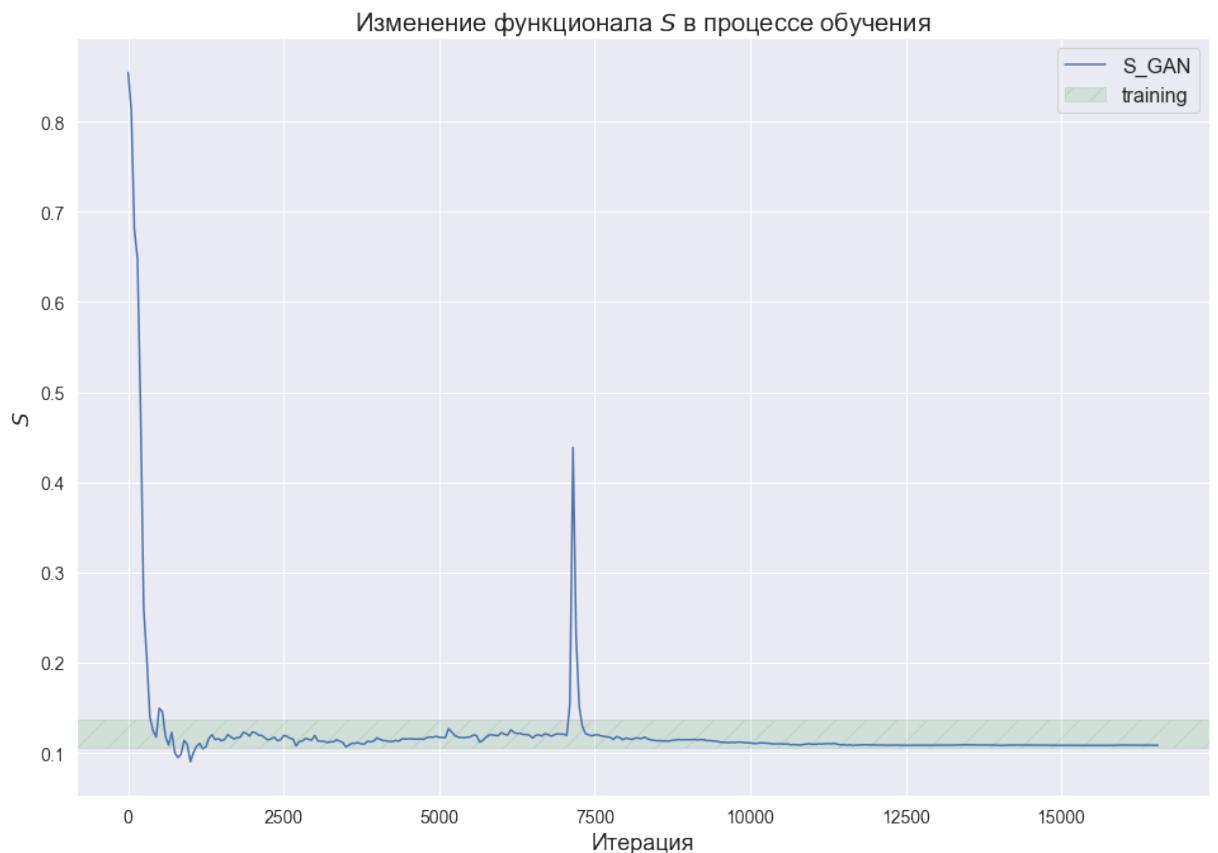


Рис. 8: График сходимости функционала Минковского S



Рис. 9: График сходимости функционала Минковского B



Рис. 10: График сходимости функционала Минковского ξ

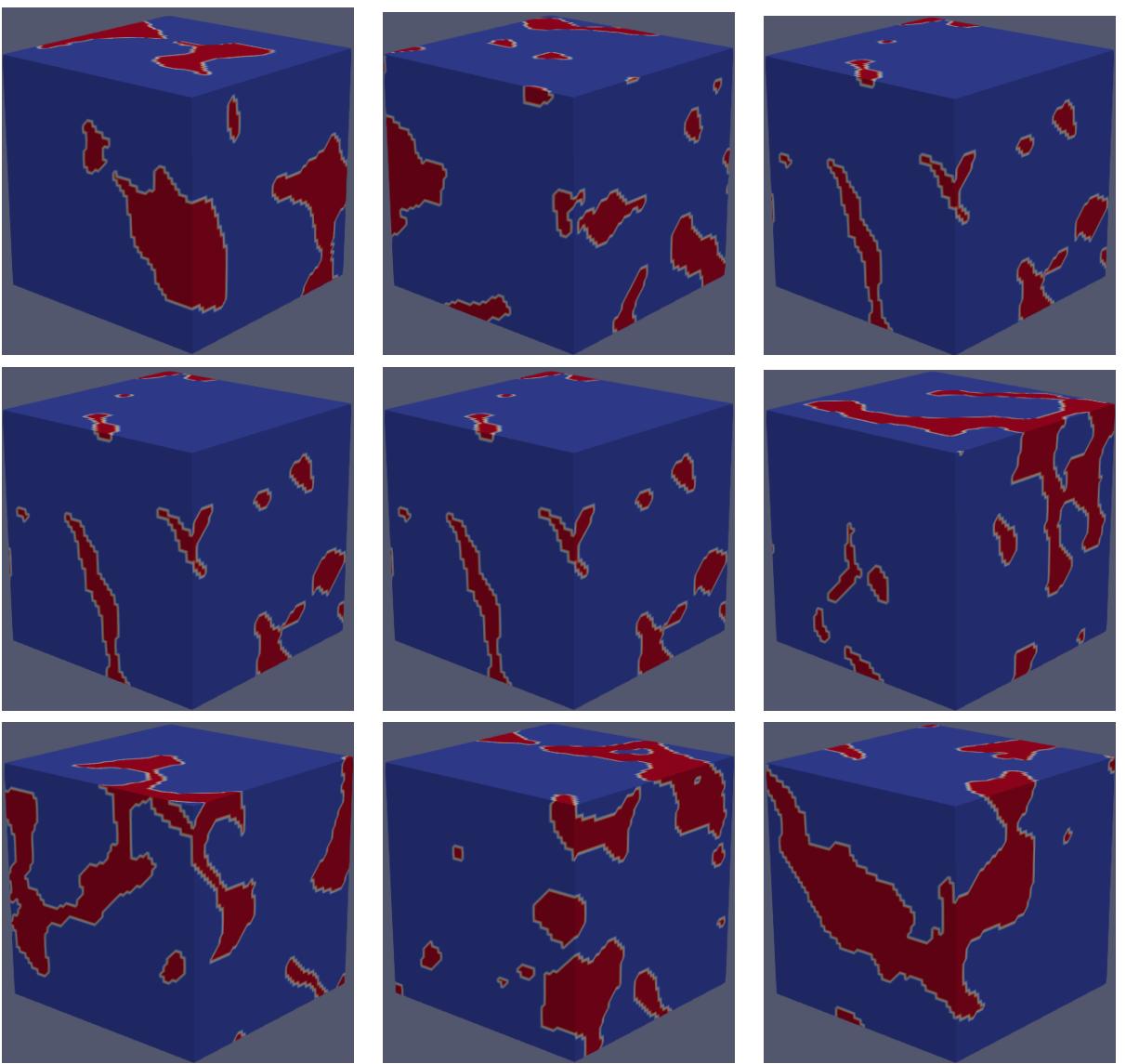
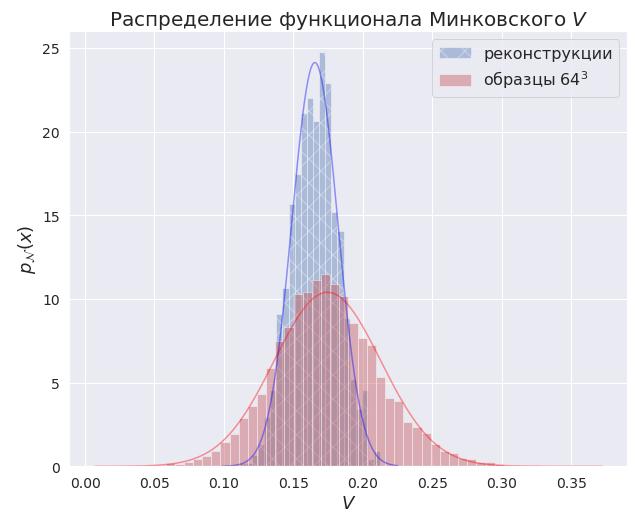


Таблица 5: Примеры реконструкции $64 \times 64 \times 64$



Обученная сеть



Mosser et al. [1]

Рис. 11: Распределения функционала Минковского V для реконструкций размера 64^3

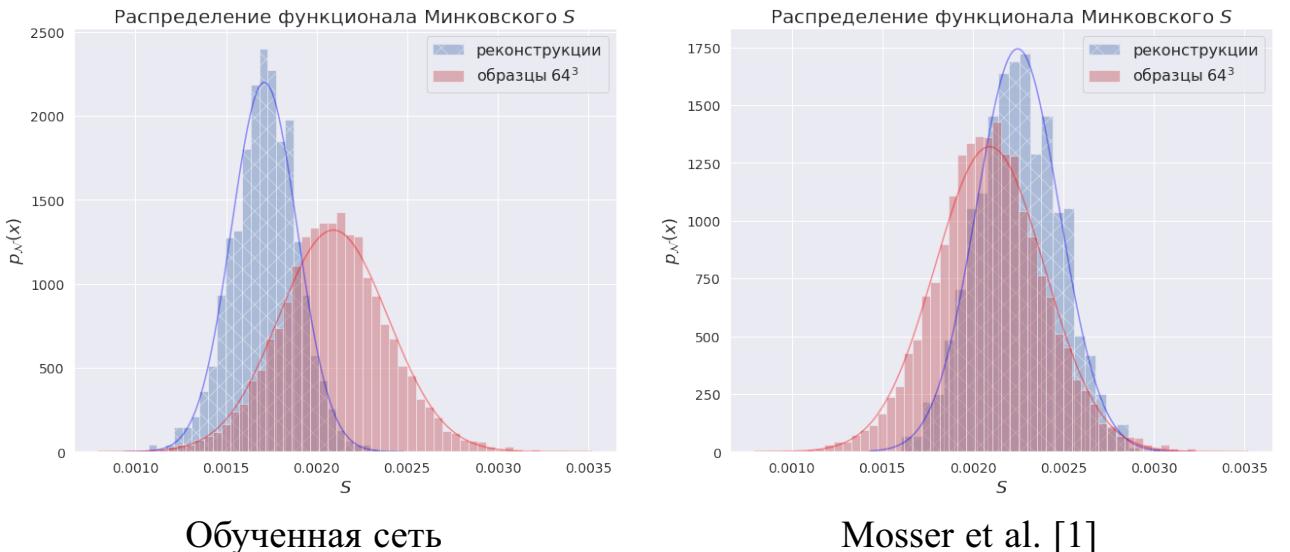


Рис. 12: Распределения функционала Минковского S для реконструкций размера 64^3

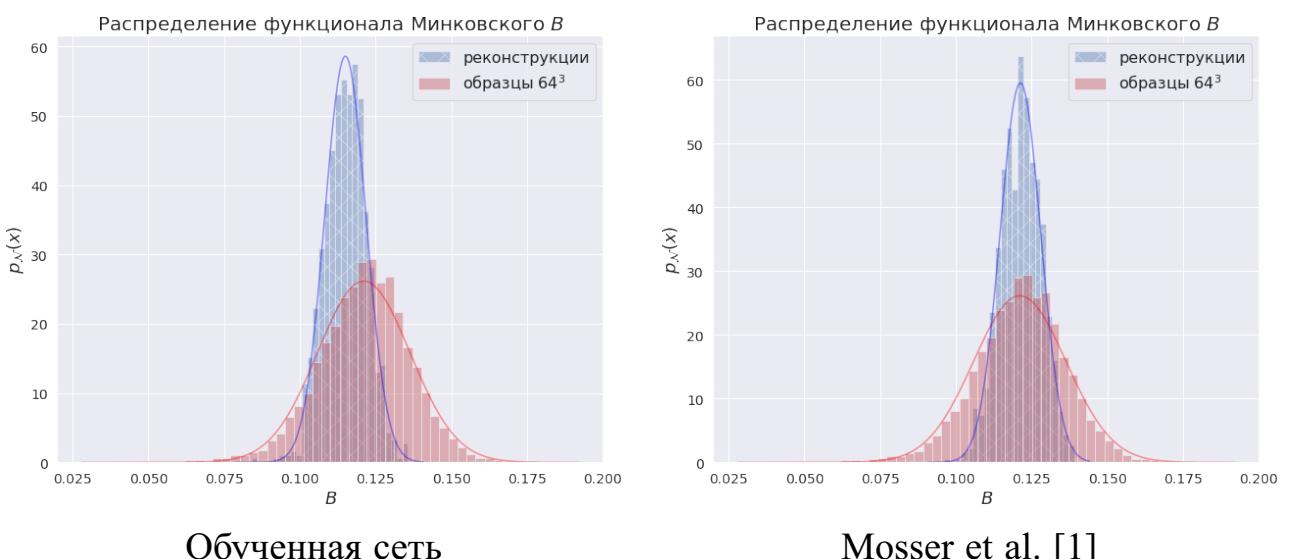


Рис. 13: Распределения функционала Минковского B для реконструкций размера 64^3

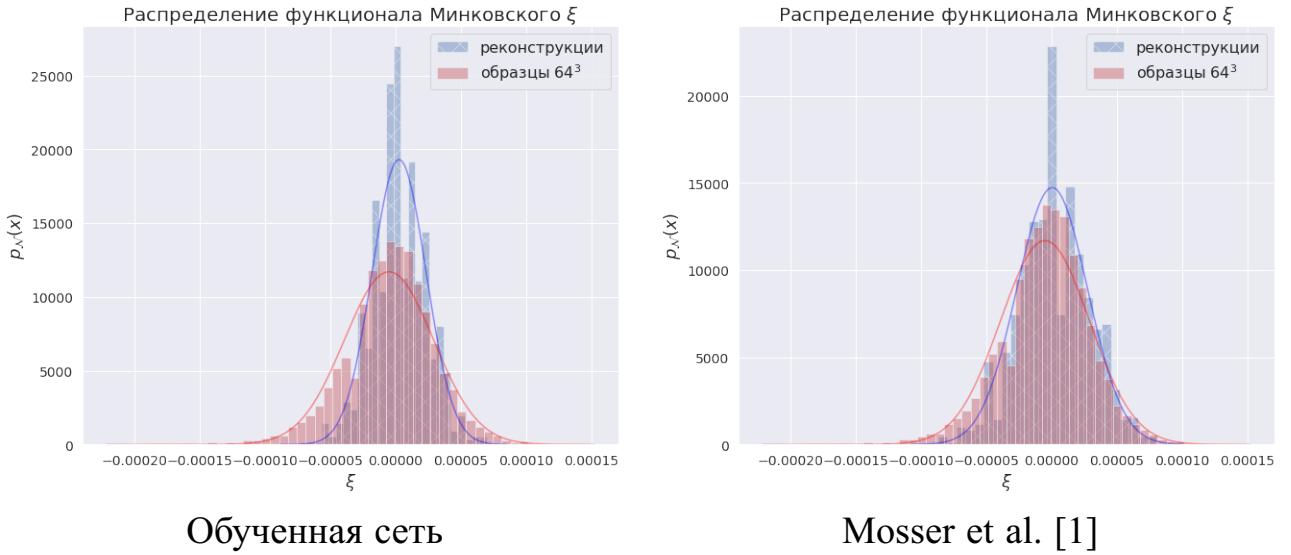


Рис. 14: Распределения функционала Минковского ξ для реконструкций размера 64^3

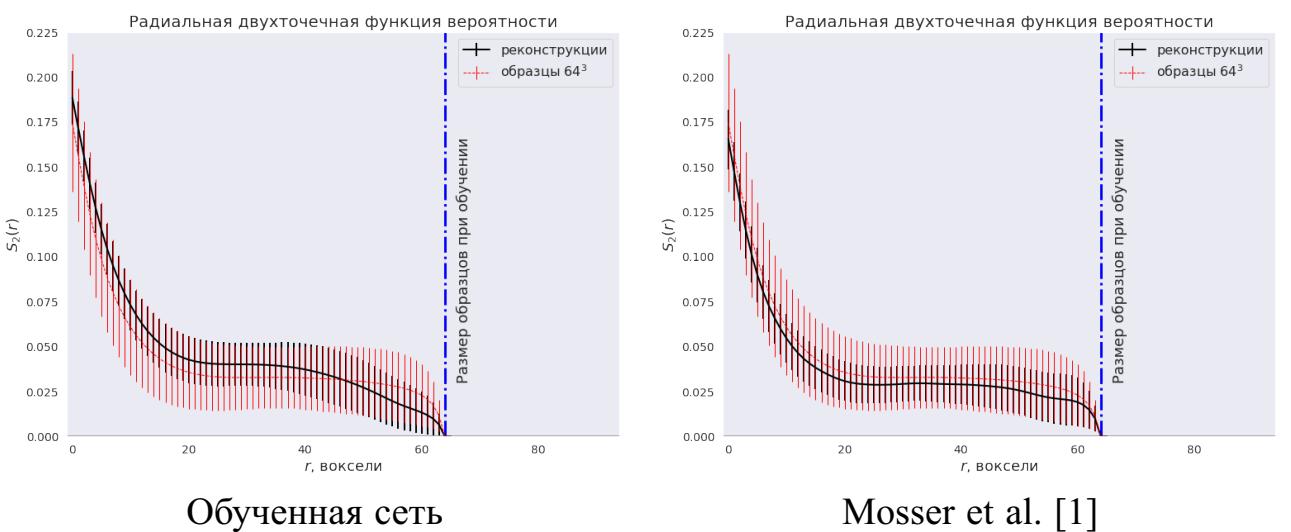


Рис. 15: Двухточечная функция вероятности для реконструкций размера 64^3

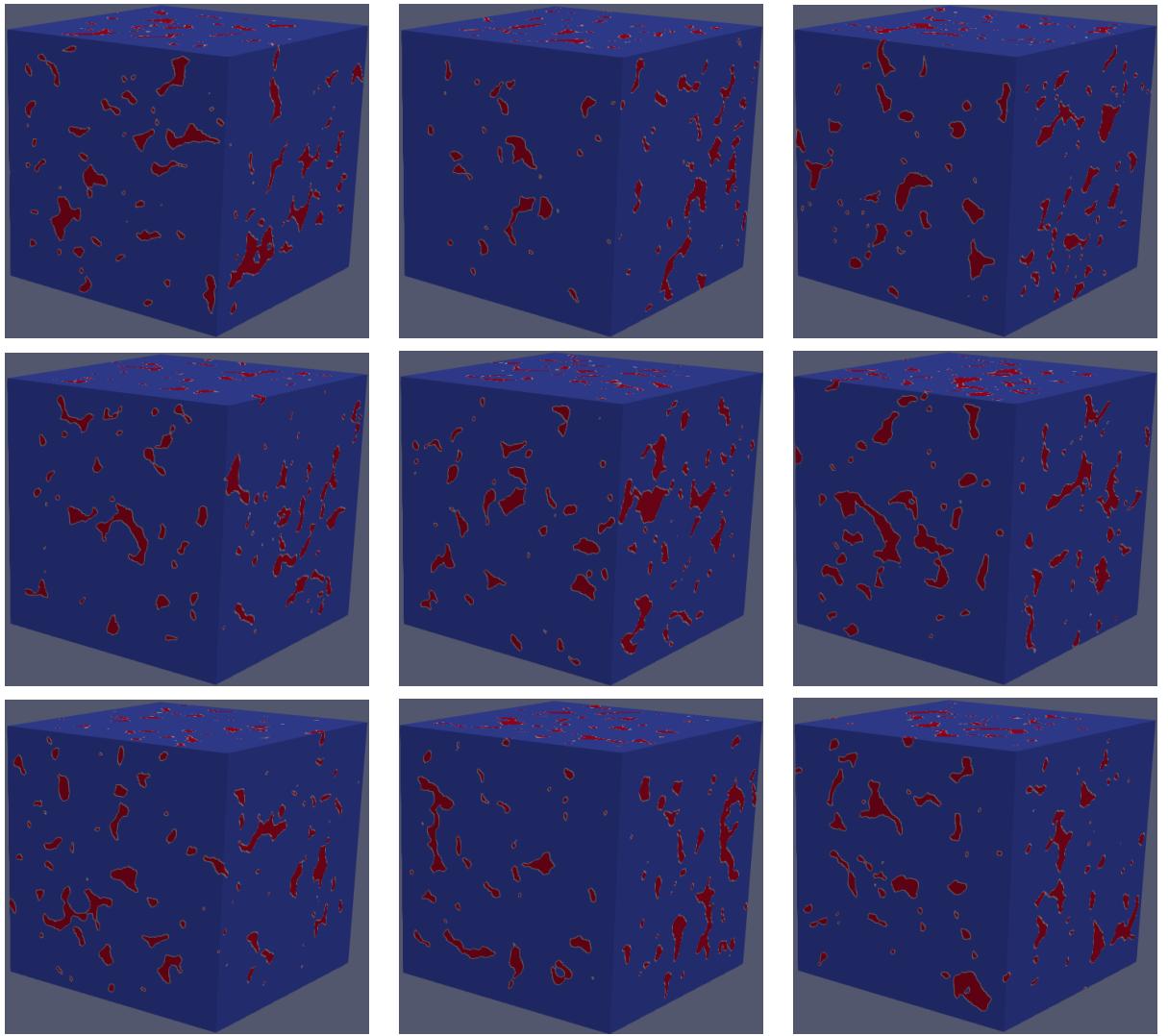
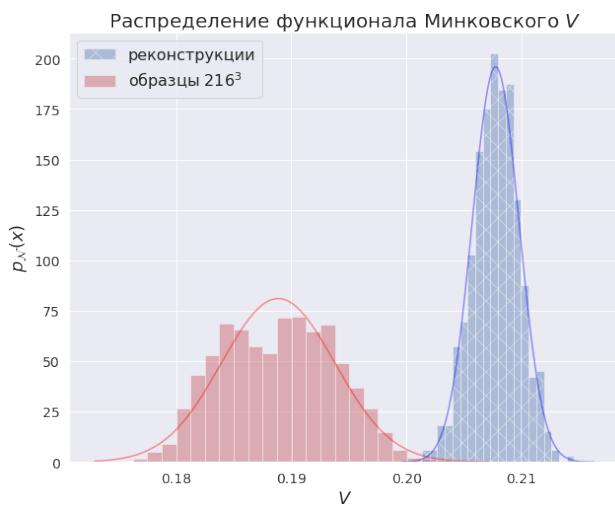
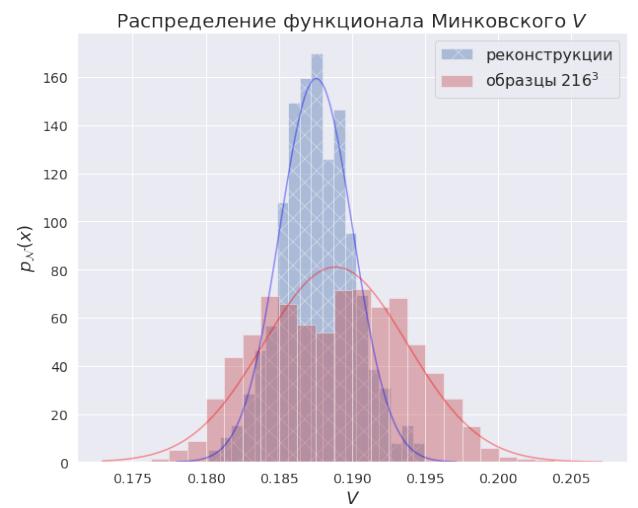


Таблица 6: Примеры реконструкции $216 \times 216 \times 216$



Обученная сеть



Mosser et al. [1]

Рис. 16: Распределения функционала Минковского V для реконструкций размера 216^3

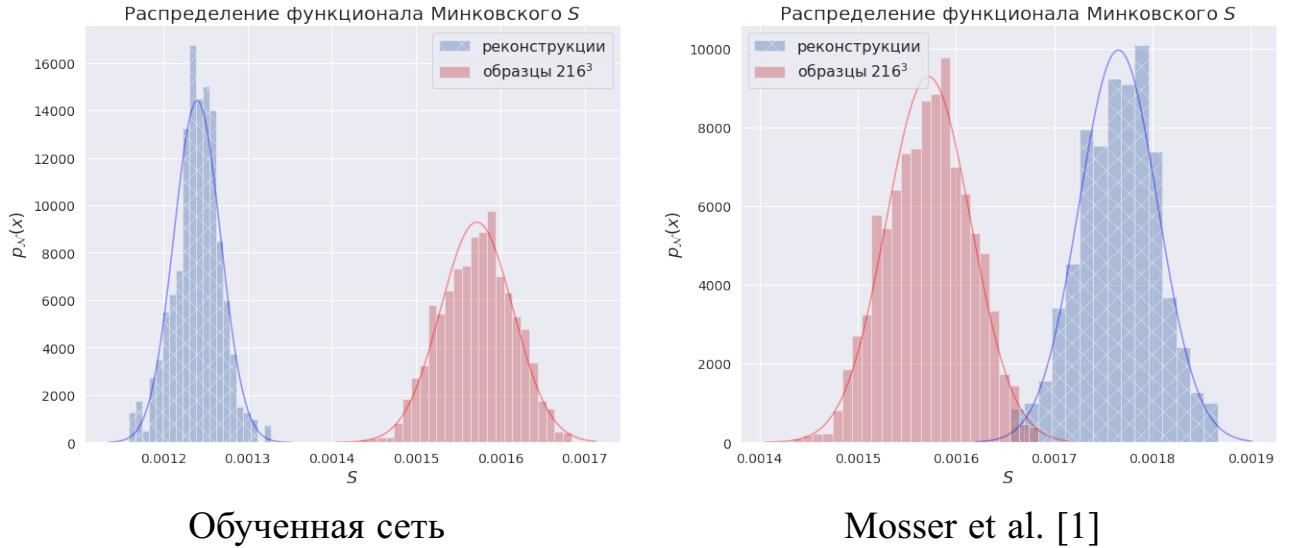


Рис. 17: Распределения функционала Минковского S для реконструкций размера 216^3

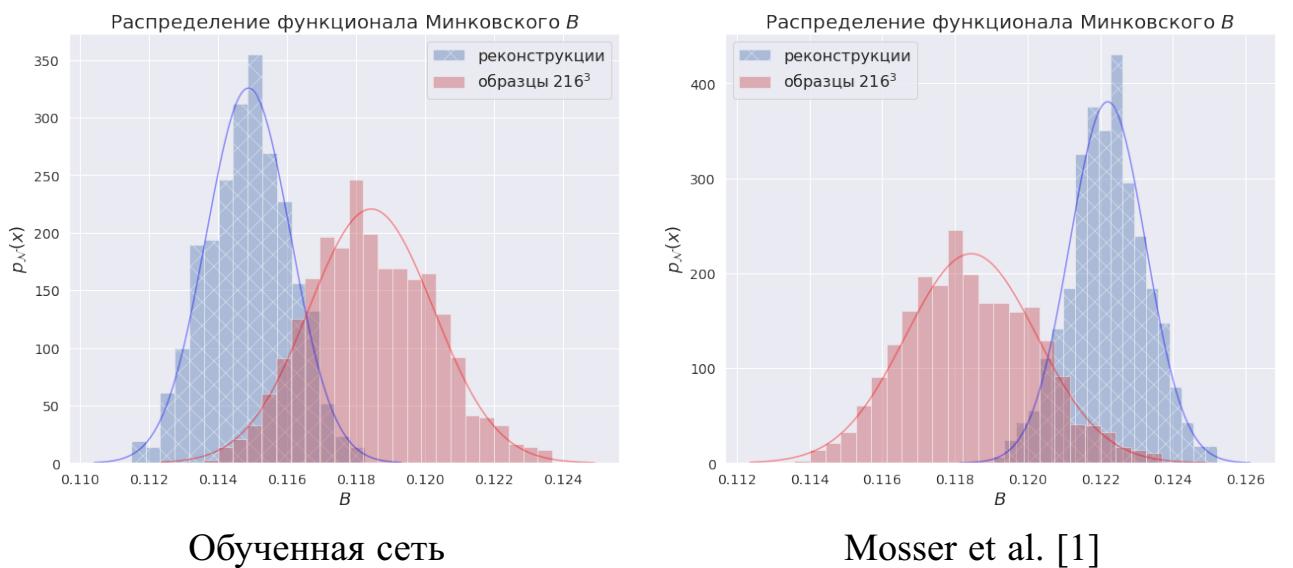


Рис. 18: Распределения функционала Минковского B для реконструкций размера 216^3

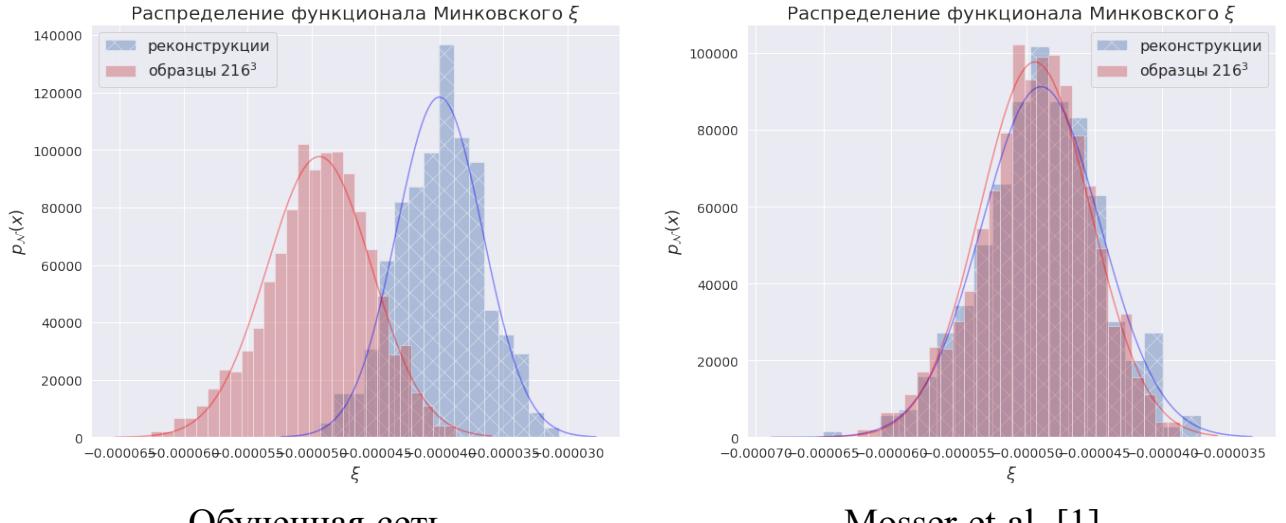


Рис. 19: Распределения функционала Минковского ξ для реконструкций размера 216^3

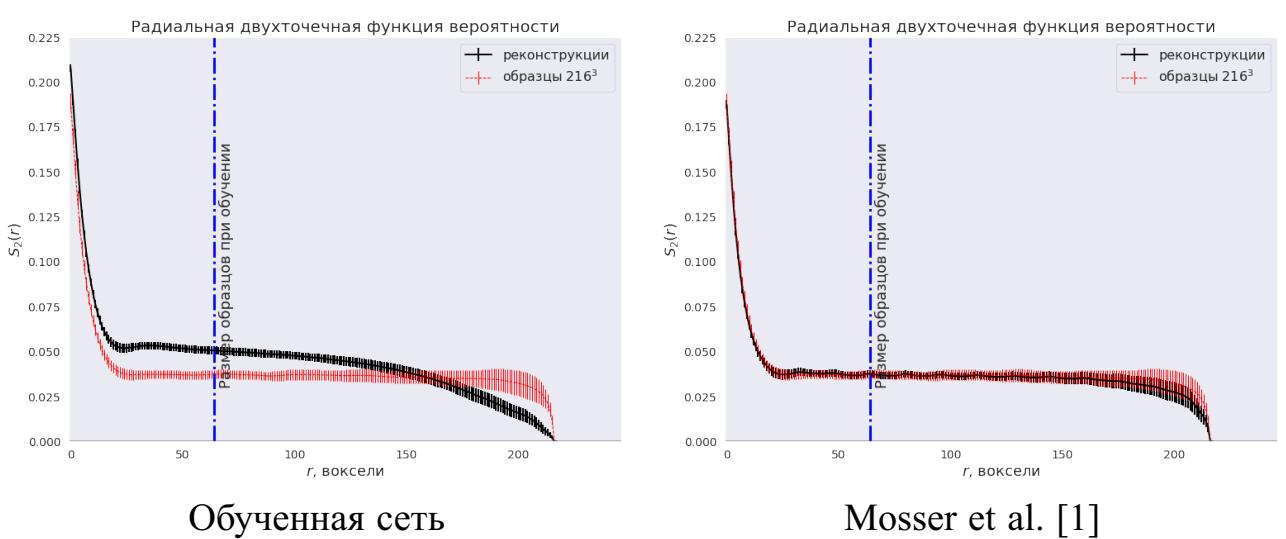


Рис. 20: Двухточечная функция вероятности для реконструкций размера 216^3

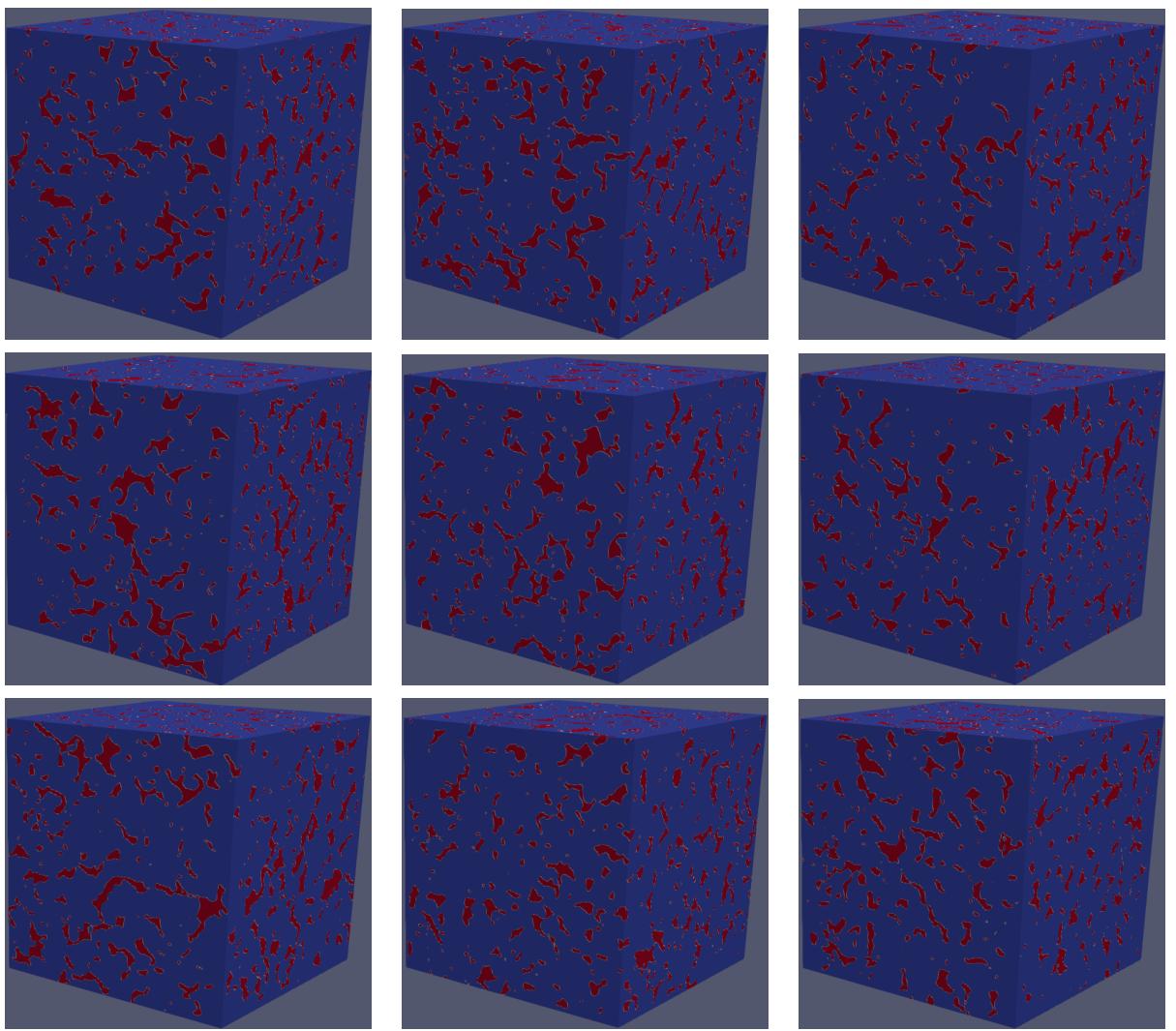
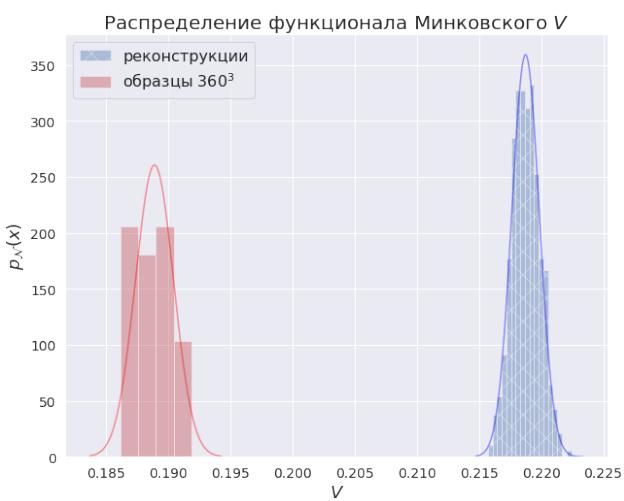
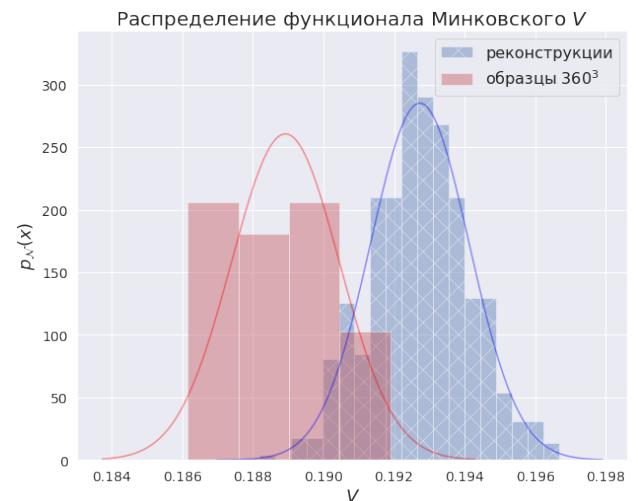


Таблица 7: Примеры реконструкции $360 \times 360 \times 360$



Обученная сеть



Mosser et al. [1]

Рис. 21: Распределения функционала Минковского V для реконструкций размера 360^3

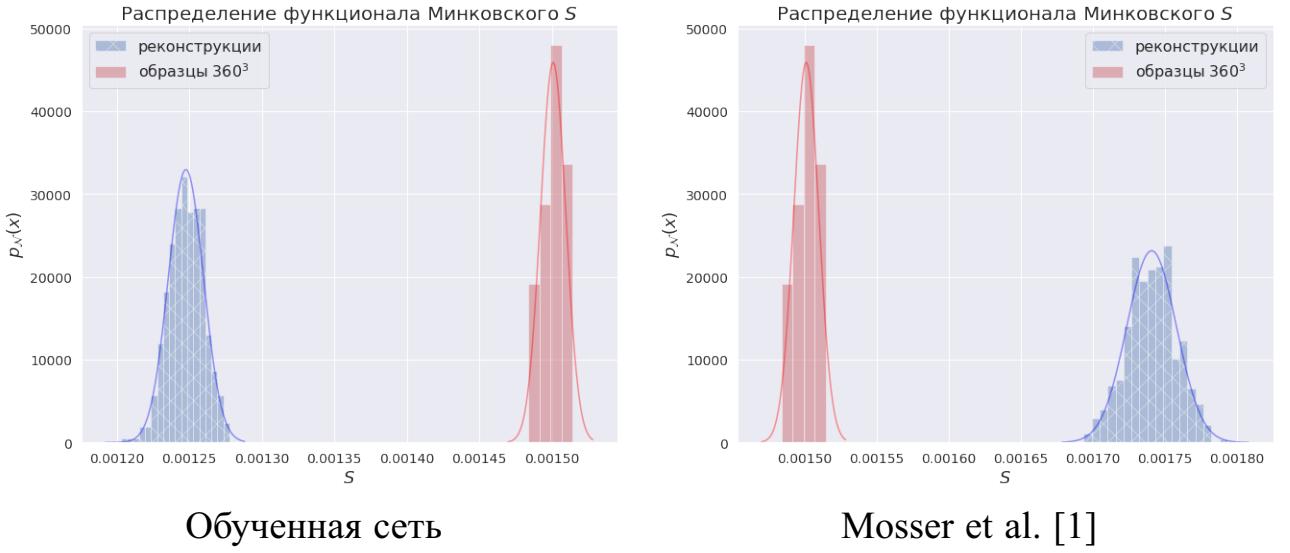


Рис. 22: Распределения функционала Минковского S для реконструкций размера 360^3

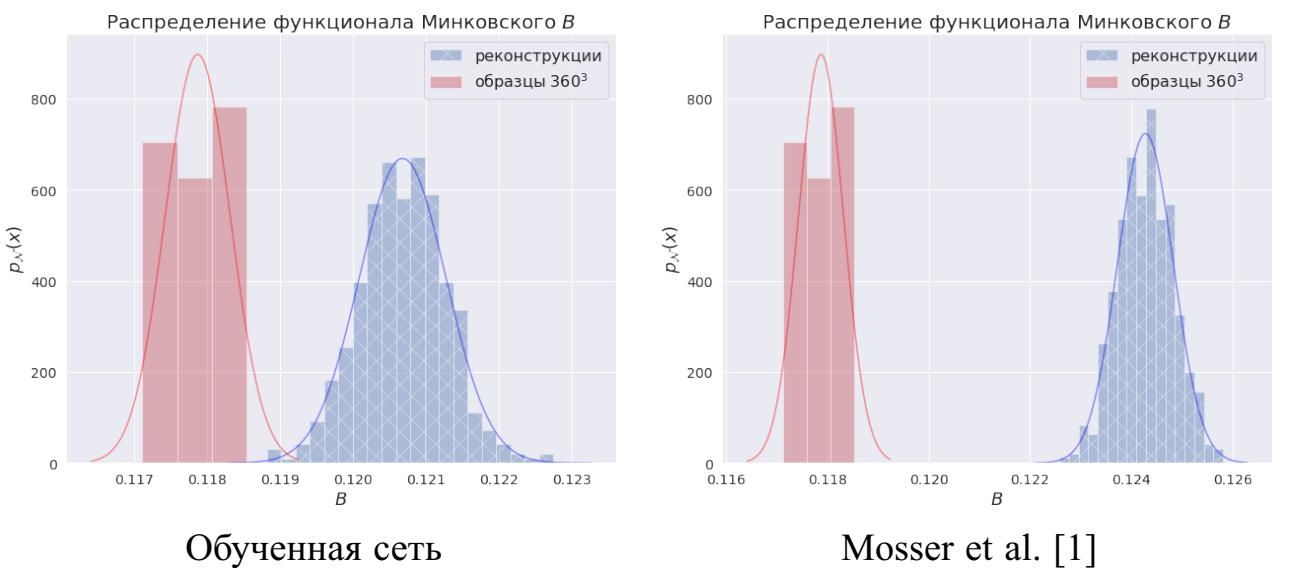


Рис. 23: Распределения функционала Минковского B для реконструкций размера 360^3

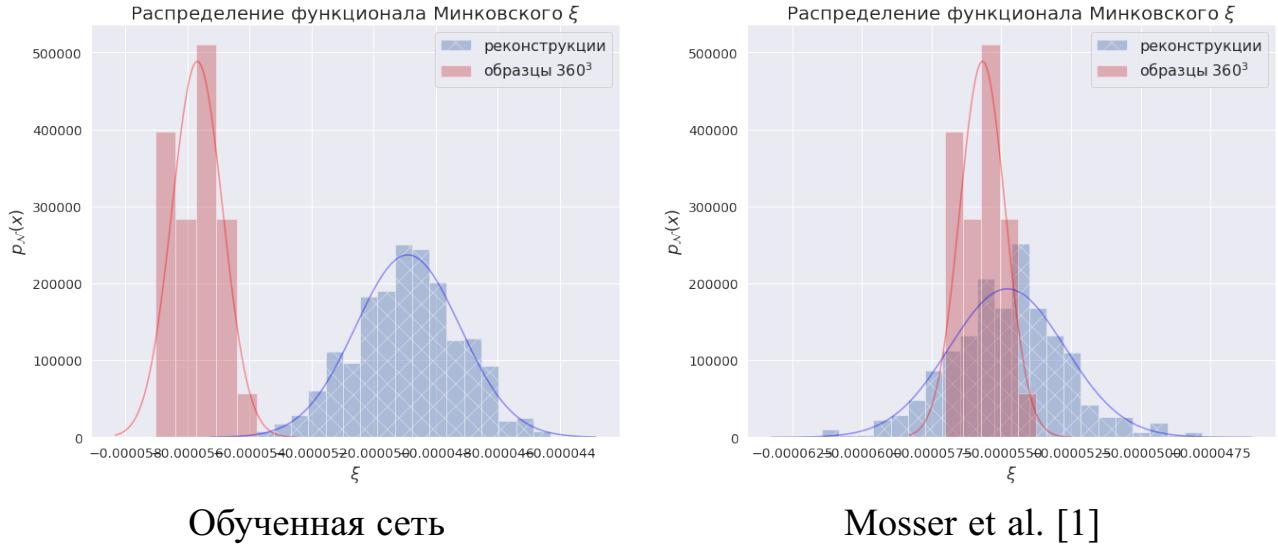


Рис. 24: Распределения функционала Минковского ξ для реконструкций размера 360^3

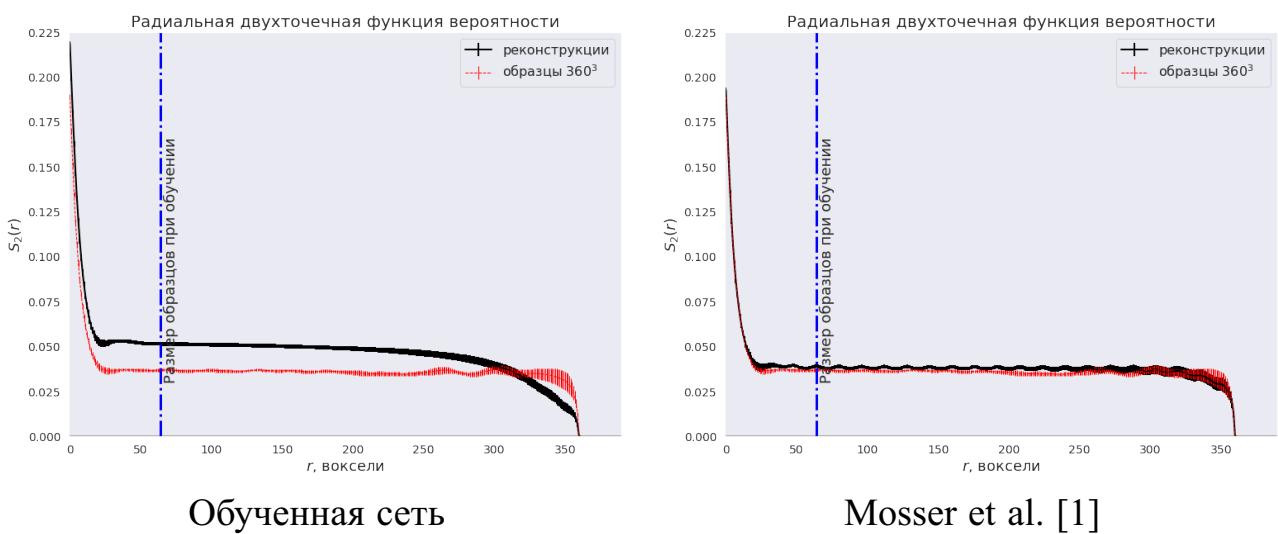


Рис. 25: Двухточечная функция вероятности для реконструкций размера 360^3

ВЫВОДЫ

Графики сходимости функционалов во время обучения (Рис. 7, 8, 9, 10) показывают, что на размере реконструкций 64^3 , на которых и обучается модель, они хорошо сходятся и остаются в зелёной зоне (разброс значений функционалов в обучающей выборке). Хорошее качество реконструкций 64^3 подтверждает и анализ распределений выборки реконструированных образцов (Рис. 11, 12, 13, 14, 15). Это даёт уверенность в том, что модели вообще обучаемы в автоматическом режиме, без ручного контроля шага обучения.

Анализ реконструкций размеров 216^3 и 360^3 показывает менее впечатляющие результаты. Видно, что сеть, обученная с модификацией процедуры обучения теряет обобщающую способность - распределения функционалов оказываются смещены относительно оригинальных распределений, в то время как сеть [1] всё ещё показывает хорошее совпадение. Причём, сравнивая графики для 216^3 с графиками для 360^3 видно, что чем больше размер реконструкции выходит за размер обучаемых образцов, тем сильнее наблюдается смещение распределений.

ЗАКЛЮЧЕНИЕ

Была разработана модификация процедуры обучения нейронных сетей для реконструкции трёхмерных пористых сред, реализован комплекс программ и проведены вычислительные эксперименты. По результатам экспериментов с помощью обученных в их ходе сетей были реконструирована выборка образцов для проведения верификации с помощью сравнения статистических и топологических характеристик выборки с характеристиками изначального образца. Также было проведено сравнение с предыдущей работой в этой области [1]. По результатам экспериментов было выявлено, что внесённая в процесс обучения модификация не нарушает ход обучения (реконструкции размера, близкого к обучающему показывают хорошее качество), однако задачу апскейлинга сеть начинает решать хуже, проигрывая в обобщающей способности сети из работы [1], которая была обучения при ручном контроле параметра шага обучения.

В будущих работах возможно проведение дополнительных исследований на тему улучшения качества работы предложенного подхода в задаче апскейлинга. Также, интересной темой для развития исследования является попытка использования новых достижений в области генеративных состязательных сетей в рамках задачи реконструкции пористых сред.

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] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, Joshua B. Tenenbaum: “Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling” // arXiv: 1610.07584 [cs.CV], 2016.
- [3] Alec Radford, Luke Metz, Soumith Chintala: “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks” // arXiv: 1511.06434 [cs.LG], 2015
- [4] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge: “Texture Synthesis Using Convolutional Neural Networks” // arXiv: 1505.07376 [cs.CV], 2015.
- [5] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, Victor Lempitsky: “Texture Networks: Feed-forward Synthesis of Textures and Stylized Images” // arXiv: 1603.03417 [cs.CV], 2016.
- [6] Воронцов К. В.: “Математические методы обучения по прецедентам (теория обучения машин)”.
- [7] 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.
- [8] Mehdi Mirza, Simon Osindero: “Conditional Generative Adversarial Nets” // arXiv: 1411.1784 [cs.LG], 2014.
- [9] Jon Gauthier: “Conditional generative adversarial nets for convolutional face generation”, Tech. rep., 2015.
- [10] Junbo Zhao, Michael Mathien, Yann LeCun: “Energy-based Generative Adversarial Networks” // arXiv: 1609.03126 [cs.LG], 2016.
- [11] David Berthelot, Thomas Schumm, Luke Metz: “BEGAN: Boundary Equilibrium Generative Adversarial Networks” // arXiv: 1703.10717 [cs.LG], 2017.

- [12] 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.
- [13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros: “Image-to-Image Translation with Conditional Adversarial Networks” // arXiv: 1611.07004 [cs.CV], 2016.
- [14] 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.
- [15] Olaf Ronneberger, Philipp Fischer, Thomas Brox: “U-Net: Convolutional Networks for Biomedical Image Segmentation” // arXiv: 1505.04597 [cs.CV], 2015.
- [16] Amari, Shunichi: “A theory of adaptive pattern classifiers” // Electronic Computers, IEEE Transactions on 3, ctp. 299-307, 1967.
- [17] 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.
- [18] Diederik P. Kingma, Jimmy Lei Ba: “Adam: A method for stochastic optimization” // arXiv:1412.6980 [cs.LG], 2014.
- [19] Martin Arjovsky, Soumith Chintala, Léon Bottou: “Wasserstein GAN” // arXiv: 1701.07875 [stat.ML], 2017.
- [20] Leslie N. Smith: “A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay” // arXiv: 1803.09820 [cs.LG], 2018
- [21] I. Blasquez, J. F. Poiradeau: “Efficient processing of Minkowski functionals on a 3D binary image using binary decision diagrams”. // Journal of WSCG. 2003, vol. 11, no. 1-3.
- [22] G. Matheron: “Elements pour une Theorie des MilieuxPoreux” // Masson, Paris, 1971.

ПРИЛОЖЕНИЕ

Приложение 1. Использованные версии основных пакетов

- python=3.7.3
- cudatoolkit=10.0.130
- pytorch=1.1.0
- ignite=0.2.0
- h5py=2.9.0
- numpy=1.16.3
- scipy=1.2.1
- tifffile=0.15.1

Приложение 2. Дополнительные примеры реконструкций

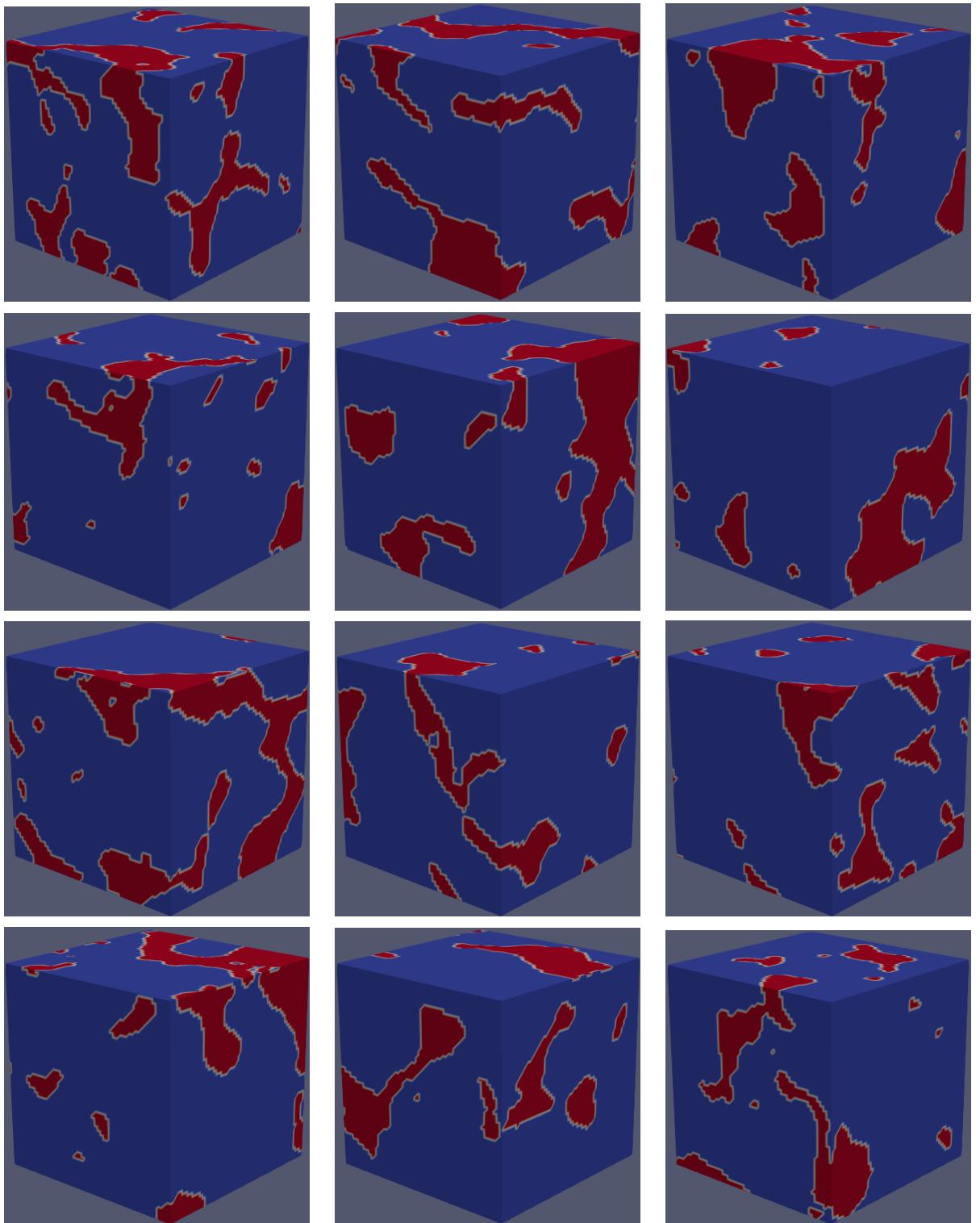


Таблица 8: Дополнительные реконструкции $64 \times 64 \times 64$

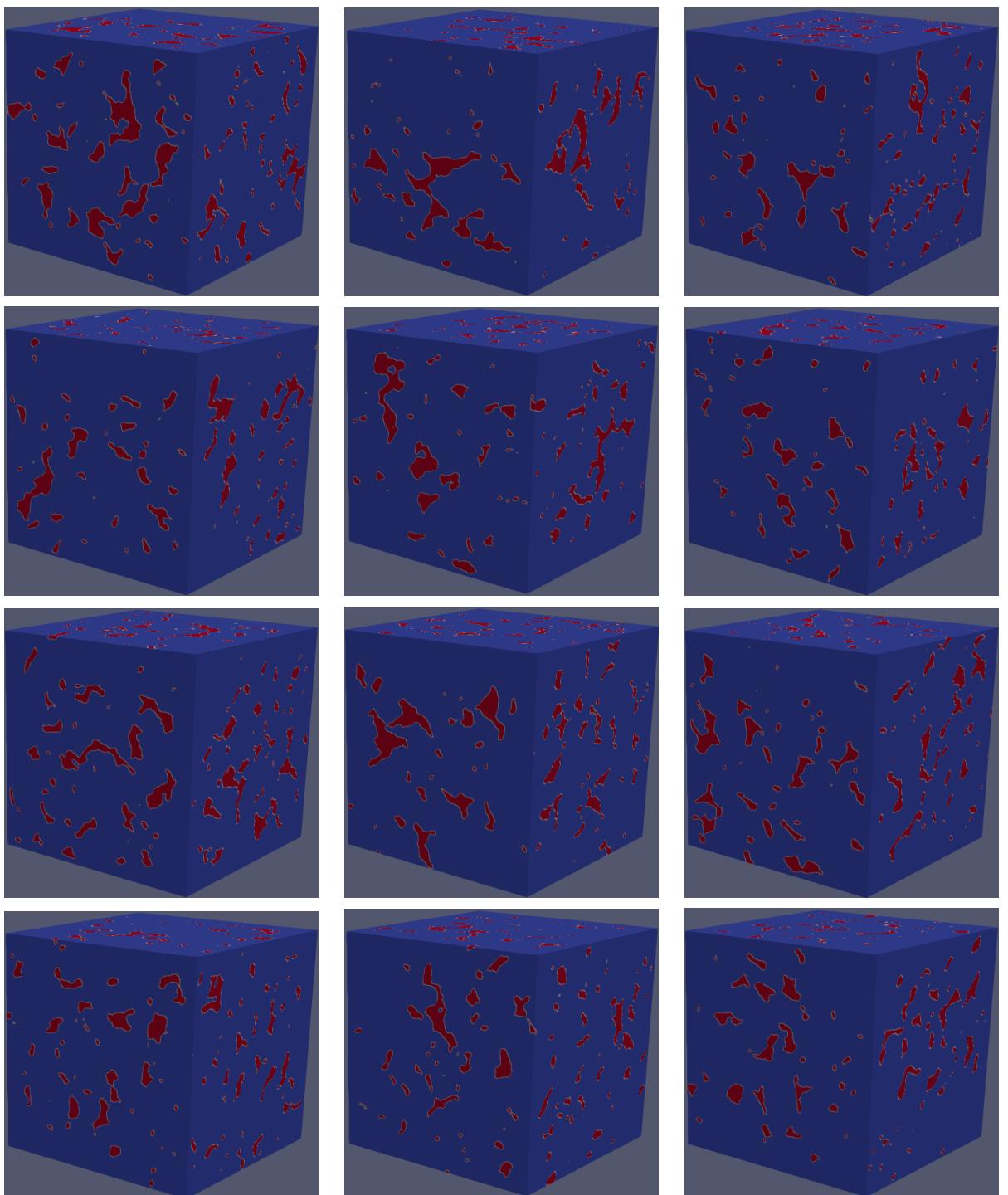


Таблица 9: Дополнительные реконструкции 216x216x216

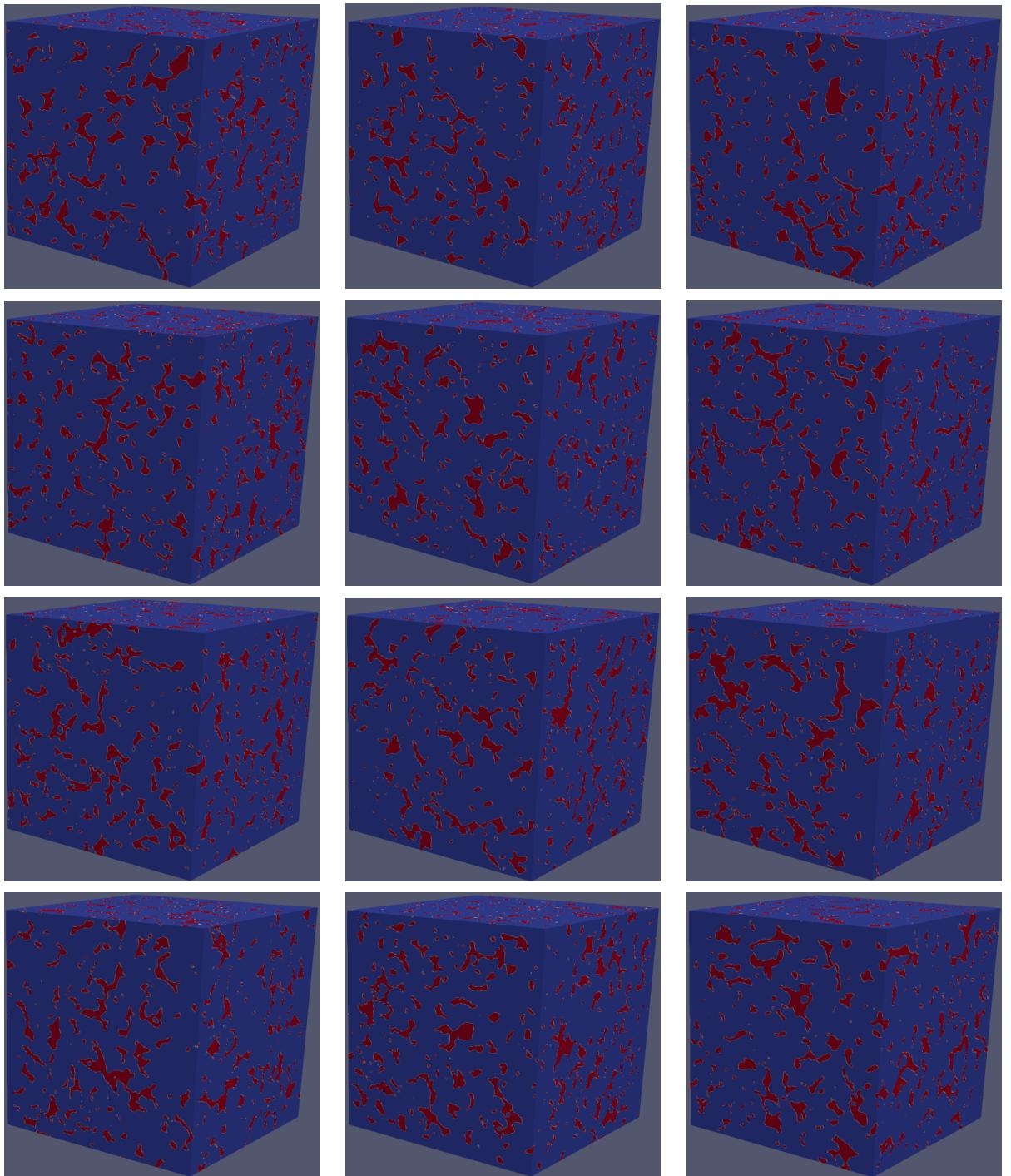


Таблица 10: Дополнительные реконструкции 360x360x360