Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Физико-механический институт

Работа допущена к защите
Руководитель ОП
К. Н. Козлов
«___» _____ 2022 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА
ПОСТРОЕНИЕ ЛАНДШАФТНЫХ КАРТ НА ОСНОВЕ ГЕНЕРАТИВНЫХ НЕЙРОННЫХ СЕТЕЙ

по направлению подготовки 01.03.02 Прикладная математика и информатика
Направленность (профиль) 01.03.02_02 Системное программирование

Выполнил
студент гр. 5030102/80201 Игнатьев Д. Д.

Руководитель
Доцент ВШ ПМиВФ
кандидат физико-математических наук Беляев С. Ю.

Ассистент ВШ ПМиВФ Савчук Д. А.

Санкт-Петербург – 2022

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

ФИЗИКО-МЕХАНИЧЕСКИЙ ИНСТИТУТ

УТВЕРЖДАЮ
Руководитель ОП
_____ К. Н. Козлов
“__” _____ 20__ г.

ЗАДАНИЕ

На выполнение выпускной квалификационной работы

студенту Игнатьеву Даниилу Дмитриевичу 5030102/80201

1. Тема работы: Построение ландшафтных карт на основе генеративных нейронных сетей.
2. Срок сдачи студентом законченной работы: 01.06.2022
3. Исходные данные по работе: Карта высот поверхности Земли. Карта была разделена на квадратные изображения с разным разрешением от 64x64 до 256x256 пикселей с перекрытием и без. Таким образом было образовано 4 датасета для исследования генеративных нейросетей, размер датасетов от 3000 до 12000 изображений.
4. Содержание работы:
    - Разработка подхода генерации набора данных для обучения.
    - Сравнительный анализ работы двух архитектур генеративной нейросети по результатам обучения на исходных датасетах.
    - Оценка и визуализация результатов.
5. Консультанты по работе: Даниил Александрович Савчук
6. Дата выдачи задания: 04.11.2022

Руководитель ВКР _____ С. Ю. Беляев

Задание принял к исполнению: 04.11.2022

Студент _____ Д. Д. Игнатьев

РЕФЕРАТ

На 31 с., 27 рисунков, 2 таблицы

КЛЮЧЕВЫЕ СЛОВА: HEIGHT MAP, TERRAIN GENERATION, CNN, GAN, SGAN, U-NET, КАРТА ВЫСОТ, КОНКУРИРУЮЩИЕ НЕЙРОННЫЕ СЕТИ, ГЕНЕРАТОР, ДИСКРИМИНАТОР, СИНТЕЗ ИЗОБРАЖЕНИЙ, ГЕНЕРАЦИЯ ЛАНДШАФТА.

Тема выпускной квалификационной работы: «Построение ландшафтных карт на основе генеративных нейронных сетей».

Данная работа посвящена исследованию применимости нейронных сетей в сфере генерации ландшафтов, подобных земным.

Область применения – синтез близких к реалистичным ландшафтов.

Методы исследования – анализ, эксперимент, тестирование, сравнение.

Задачи, которые решались в ходе исследования:

1. Поиск исходных данных и генерация обучающего набора.

2. Разработка двух различных архитектур нейронных сетей.

3. Обучение реализованных алгоритмов на различных наборах данных.

4. Оценка и визуализация результатов обучения нейронных сетей.

Работа основывалась на статьях со схожими исследованиями и матераилах из сети Интернет.

Программирование и анализ проводился с помощью интерактивной облачной среды Google Colab. Задание и обучение нейросетей осуществлено на языке Python с использованием открытой библиотеки для машинного обучения Keras.

Программа, подготавливающая обучающий набор данных реализована на языке программирования C++ с использованием открытой библиотеки для работы с изображениями stb_image.

Визуализация результатов выполнена в межплатформенной среде разработки компьютерных игр Unity.

В результате исследования были запрограммированы и обучены две нейронные сети, выявлены их достоинства и недостатки, получены и визуализированы в трехмерные модели результаты генерации нейронных сетей.

ABSTRACT

31 pages, 27 figures, 2 tables

KEYWORDS: HEIGHT MAP, TERRAIN GENERATION, CNN, GAN, SGAN, U-NET.

The topic of the final qualification work: "Construction of landscape maps based on generative neural networks."

This work is devoted to the study of the applicability of neural networks in the field of generating landscapes like the earth.

The scope of application is the synthesis of landscapes close to realistic.

Research methods - analysis, experiment, testing, comparison.

Tasks that were solved during the study:

1. Search for initial data and generation of a training set.

2. Development of two different architectures of neural networks.

3. Training of implemented algorithms on various data sets.

4. Evaluation and visualization of learning outcomes of neural networks.

The work was based on articles from the Internet.

Programming and analysis was carried out using the interactive cloud environment Google Colab. The task and training of neural networks was carried out in Python using the open library for machine learning Keras.

The program that prepares the training dataset is implemented in the C++ programming language using the stb_image open library for working with images.

Visualization of the results was carried out in the cross-platform environment for the development of computer games Unity.

As a result of the study, two neural networks were programmed and trained, their advantages and disadvantages were identified, and the results of generating neural networks were obtained and visualized in three-dimensional models.

СОДЕРЖАНИЕ

1. Введение

На сегодняшний день потребность в создании реалистичных ландшафтов в кинематографической и игровой промышленностях становится все острее. Модели рельефа становятся больше, требования к их качеству выше. Однако наиболее популярными в использовании методы процедурной генерации являются строго настроенными и мало расширяемыми. Следовательно, алгоритмы нуждаются в частой доработке, а сами продукты их работы, по мере увеличения размеров, требуют все больше вмешательства левел-дизайнеров. Таким образом, повышение качества, разнообразия и размера модели ландшафта влечет к большим затратам.

Решением данной проблемы должен стать новый инструмент, способный к расширению и слабо привязанный к типу ландшафта. Целью данной бакалаврской работы является исследование применимости нейронных сетей для генерации ландшафтных карт.

2. Обзор литературы

Основную роль в поиске нужных архитектур нейронных сетей и принципов их работы выполняли статьи авторства Ian J. Goodfellow [3], Sheng-Yu Wang [7] и Sakshi Indolia [9]. Под их воздействием была выбрана модель, являющаяся композицией двух нейронных сетей, известной как генеративно-состязательная сеть.

Более подробно эту область удалось благодаря статьям Nikolay Jetchev [6] и Ryan Spick [8]. В статьях описан процесс поиска и подготовки данных, реализация и обучение нейронных сетей, а также результаты проведенных экспериментов, что было весьма сподручным в проведении исследования.

3. Основная часть

3.1 Искусственные нейронные сети: основные понятия, классификация

Искусственными нейронными сетями (далее «нейронная сеть» или «нейросеть») называются вычислительные структуры, моделирующие биологические процессы, схожие с процессами человеческого мозга. Это параллельные и распределенные системы, имеющие возможность обучения с помощью анализа положительных и отрицательных воздействий.

Алгоритмов машинного обучения на данный момент известно достаточно много. Нейронные сети являются одним из их подмножества. Однако, в последние годы они показали удивительные результаты, чем привлекли всеобщее внимание.

Основной структурной единицей нейронной сети является искусственный нейрон. Множество таких нейронов взаимодействует между собой. На рисунке №1 представлена схема искусственного нейрона:

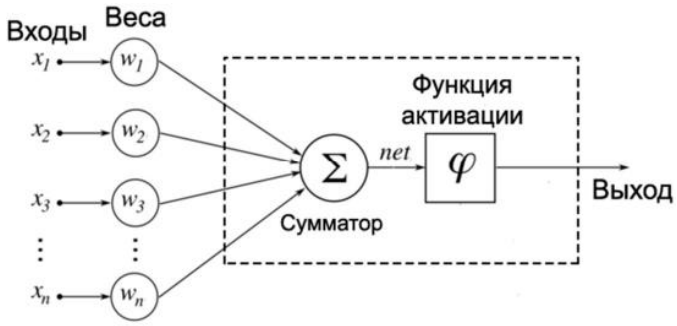

Рис. 1 Схема искусственного нейрона

Как видно из схемы, у нейрона может быть несколько входных связей, через которые подаются различные сигналы. Он преобразует их посредством активационной функции и передает информацию другим нейронам.

Другими словами, искусственный нейрон — это функция $R^n \rightarrow R$, которая преобразует несколько входных параметров в один выходной. Своей эффективностью нейронные сети обязаны двум составляющим:

1) возможности параллельной обработки больших объемов информации;
2) способности обучаться, т. е. выделять признаки или создавать обобщение, или, иными словами, способности получать обоснованный результат для данных, не принимавших участия в процессе обучения.

Задача нейросети – найти достаточно хорошую функцию, аппроксимирующую обучающую выборку, при условии, чтобы эта функция обобщалась и на неизвестные алгоритму данные. Отсюда следуют два ключевых понятия: функция ошибки и регуляризация.

3.1.1 Функция ошибки

Задача функции ошибки предельно проста. Оптимизация функции ошибки, а именно, минимизация её значения, влечет улучшение точности ответов нейросети. В качестве алгоритма оптимизации используется метод градиентного спуска (и его многочисленные варианты улучшения), суть которого заключается в поэтапном изменении значений параметров функции на значение, пропорциональное градиенту в точке.

Роль функции ошибки часто выполняет средняя квадратичная ошибка, в которой минимизируется среднее квадратов отклонений предсказанных значений от истинных:

$$RSS(\omega) = \frac{1}{N}\sum_{i=1}^{N}(y_i - x_i w)^2$$

Подобная функция ошибки используется в задачах регрессии. Так же в задачах регрессии могут использоваться различные её модификации, такие как средняя квадратичная логарифмическая ошибка $MSLE(\omega) = \frac{1}{N}\sum_{i=1}^{N}(\log(1 + y_i) - \log(x_i w + 1))^2$,

которую можно использовать в случае, когда целевое значение обучающей выборки имеет большой разброс, и при обучении подобные случаи нет необходимости налагать слишком высокие штрафы. В задачах же классификации используется функция кросс-энтропии, которая бывает как бинарной, в случае выбора между двумя возможными классами, так и мульти-классовой:

$$J = -\frac{1}{N}\sum_{i=1}^{N}(L_i \log(S_i))$$

Где L_i- элемент вектора значения в one-hot кодировке, в которой длина вектора равна количеству классов; для правильного класса проставлена 1, для всех остальных – 0. S_i – результат работы алгоритма, а именно, вероятность, с которой алгоритм относит результат обработки входящих данных к i – ому классу. Вектор S_i- в большинстве случае – это результат работы функции softmax, которая используется в паре с кроссэнтропией в качестве функции ошибки.

Бинарная классификация является частным случаем кросс-энтропии и имеет вид:

$$J = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(y_i^*) + (1 - y_i) \log(1 - y_i^*)).$$

Альтернативой функции кросс-энтропии в случае бинарной классификации может быть так называемая hinge loss: $max(0, 1 - y_i)$. Для ее использования целевые значения должны принадлежать множеству {-1,1}. Однако, использование этой функции не всегда может привести к хорошему результату относительно кросс-энтропии.

3.1.2 Функция активации

Еще одним важным понятием нейронных сетей является функция активации – нелинейная функция, которая применяется нейроном для получения выходного значения. Существуют несколько функций, которые на данный момент используются в большей части решений на основе нейронных сетей. Они продемонстрированы на рисунке №2 в порядке их описания.

$$\text{Binary step: } f(x) = \begin{cases} 0, x \leq 0 \\ 1, x > 0 \end{cases}$$

Эта функция является пороговой. Она хорошо работает для бинарной классификации, но слабо применима к нейросетям с большим числом нейронов или небинарной классификацией.

$$\text{linear: } f(x) = x$$

Эта функция пропорциональна входящему значению и, в отличии от предыдущей, возвращает не булевые значения, что хорошо сказывается на мульти-классовой классификации.

Однако из-за того, что производная равна константе, невозможно использовать метод обратного распространения ошибки. Следовательно, при обновлении весов невозможно распознать, улучшается ли эмпирический риск на текущем шаге или нет.

$$\text{sigmoid: } f(x) = \frac{1}{1+e^{-x}}$$

Функция является гладкой и лежит в диапазоне (-1, 1). Поэтому её часто используют в задачах классификации. Помимо этого, значения функции для $|x| > 2$ прижимаются к асимптотам, что позволяет делать четкие предсказания и, что более важно, нормализовать выходные значения. Однако, её производная крайне мала во всех точках, за исключением сравнительно небольшого промежутка. Это усложняет процесс обучения с помощью градиентного спуска. А в нейронных сетях с большим количеством слоев приведет к проблеме затухающего градиента.

$$\text{tanh: } f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Гиперболический тангенс схож с сигмоидой и является ее скорректированным видом.

$$tanh(x) = 2 * sigma(2x) - 1$$

Он применяется чаще в случаях, когда нет необходимости в нормализации. Это основывается на том, что область определения данной функции активации центрирована относительно нуля. Следовательно, снимается ограничение при подсчете градиента для перемещения в определенном направлении. Кроме того, производная гиперболического тангенса принимает большие значения вблизи нуля, давая большую амплитуду градиентному спуску, а значит, и более быструю сходимость.

$$\text{ReLu: } f(x) = \begin{cases} 0, x \leq 0 \\ x, x > 0 \end{cases}$$

Rectified Linear Unit — одна из наиболее распространенных функций активации при глубоком обучении. Она имеет сходство с линейной функции, но не перенимает её проблемы. Для нее очень просто считается производная, но существует проблема «умирающего ReLu»: из-за того, что для отрицательного аргумента производная равна нулю, градиент тоже будет равен нулю. А следовательно, веса не будут меняться и нейросеть не будет обучаться.

$$\text{LeakyRelu: } f(x) = \begin{cases} ax, x \leq 0 \\ x, x > 0 \end{cases},$$

где a – настраиваемый коэффициент. Чаще всего $a = 0.01$.

Чтобы бороться с проблемой «умирающих нейронов», была создана модификация ReLu. Для этого добавляют «утечку», дающие ненулевые значения для отрицательных параметров. На практике результат не сильно улучшается относительно обычного ReLu.

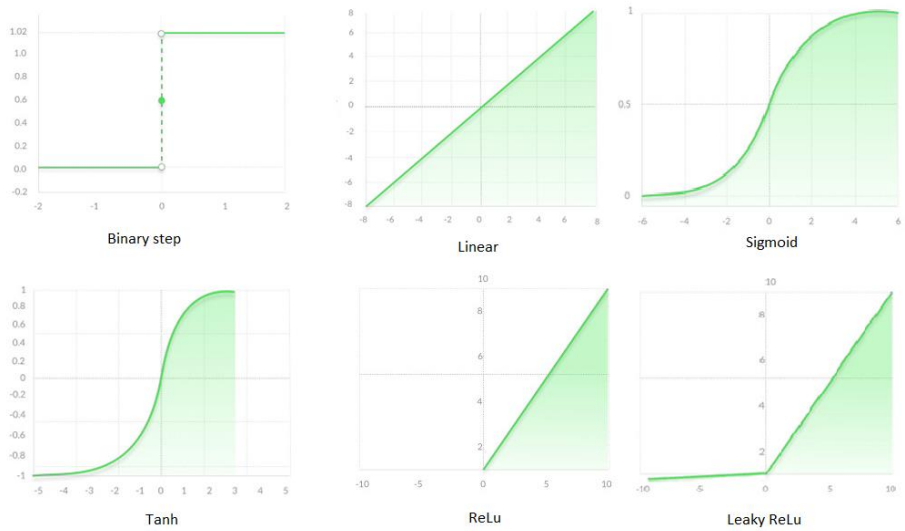

Рис. 2 Графики функций активации

3.1.3 Обратное распространение ошибки

Рассмотрим процесс обучения нейронной сети с заранее известными результатами в обучаемом наборе. Он состоит из двух компонент: прямого прохода и обратного. В прямом проходе по входным данным формируется выходной вектор и вычисляется его ошибка относительно истинных результатов из датасета. На этом моменте начинается обратный проход. На обратном проходе полученная ошибка распространяется от выхода сети к ее входам: каждый выходной нейрон $y_k, k = 1, 2, \ldots$ получает целевое значение — то выходное значение, которое является правильным для данного входного сигнала, и вычисляет ошибку σ_k, так же вычисляет величину, на которую изменится вес связи $w_{i,j}$. Помимо этого, находит величину корректировки смещения $\Delta w_{i,j}$ и посылает нейронам в предыдущем слое. Каждый скрытый нейрон предыдущего слоя суммирует входящие ошибки от нейронов в последующего слоя и вычисляет величину ошибки, умножая полученное значение на производную активационной функции, а так же вычисляет величину, на которую изменится вес связи.

3.1.4 Генеративно-состязательная сеть

Одной из значимых проблем алгоритмов машинного обучения с учителем является нехватка данных. Для построения рельефа, похожего на земной, нужно обучать нейросеть на существующих данных поверхности земли, но их количество существенно ограничено. Поэтому за основу была выбрана генеративно-состязательная сеть (GAN), способная генерировать новые данные для обучения.

Основной их идеей GAN является наличие двух нейросетей, работающих в паре: генератор и дискриминатор. Дискриминатор на вход получает данные из двух источников: заранее подготовленной выборки с размеченными данными, а также вывод генератора. Основной

задачей дискриминатора является определение того, является ли входящее сообщение реальным, либо оно поступило из генератора. Таким образом выходом дискриминатора будет являться булевое значение. Соответственно, задача оптимизации дискриминатора – это задача минимизации его ошибки. Генератор же в свою очередь из случайного вектора генерирует данные, которые должны быть неправильно классифицированы дискриминатором. Данная схема изображена на рисунке №3.

Рис. 3 Модель GAN

Чтобы определить распределение данных, выдаваемых генератором $p_G(\mathrm{x})$, сперва вводится вектор шумов $p_Z(\mathrm{z})$, затем вводится дифференцируемая функция отображения $G(\mathrm{z}, \theta_G)$, представляющая собой многоуровневый перцептрон с параметров θ_g. Так же определяется второй перцептрон $D(\mathrm{z}, \theta_D)$, областью значений которого является скаляр. D(x) представляет собой вероятность того, что x принадлежит реальным данным, а не порожденными генератором p_G. Таким образом, выведем функцию ошибки:

$$min_G\, max_D\, V(D,G) = E[\log D(x)] + E[\log(1 - D(G(z)))]$$

Где $E[\log D(x)]$ - кросс-энтропия ответов дискриминатора в случае реальных данных, а $E[\log(1 - D(G(z)))]$ - в случае сгенерированных.

Другими словами, задача дискриминатора – это максимизировать данную функцию ошибки, генератора – минимизировать. Можно заметить, что в случае минимизации выражения по функции генератора, первое слагаемое никак от генератора не зависит; таким образом оно может быть опущено. Однако, как показывает практика, минимизации $E[\log(1 - D(G(z)))]$ недостаточно для успешного обучения генератора в виду того, что на ранних этапах обучения, когда дискриминатор с довольно большой уверенностью может отличить изображения от произвольного шума, $\log(1 - D(G(z)))$ будет насыщаться. Решением будет обучение генератора путем максимизации $\log(D(G(z)))$ вместо минимизации $\log(1 - D(G(z)))$. Данная функция позволяет получить гораздо больший градиент на начальных этапах обучения.

Суммируя все вышесказанное, итоговый итеративный алгоритм обучения генеративно-состязательной сети по небольшим наборам (батчам) будет иметь следующий вид:

- На каждой итерации выбирается k элементов выборки $(x_1, x_2, \dots, x_k)$, а также генерируется k произвольных векторов определенной длины $(z_1, z_2, \dots, z_k)$, которые будут использованы в качестве входных параметров сети генератора.

- Для всех x_i, а так же z_i вычисляются $D(x_i)$, а так же $D(G(z_i))$
- Обновляется D, учитывая полученное значение градиента:
- $\nabla \frac{1}{k} \sum_{i=1}^{k} (\log D(x_i) + \log(1 - D(G(z_i))))$
- Данные операции, обновляющие дискриминатор, в большинстве случаев производятся несколько раз, прежде чем обновить генератор.
- Обновляем генератор по вновь сгенерированным k векторам шумов:

$$\nabla \frac{1}{k} \sum_{i=1}^{k} \log(1 - D(G(z_i)))$$

Несмотря на относительную простоту алгоритма, обучение генеративных сетей – крайне сложный и нестабильный процесс. Задача нахождения точки равновесия в данной минимакс игре во многом зависит от того, чему необходимо научить GAN.

В рамках данной работы была поставлена задача реализации генеративно-состязательных сетей, позволяющих синтезировать достоверные изображения в разрешениях 64x64, 128x128 и 256x256 пикселей. Будет проведен анализ ряда существующих успешных подходов и рекомендаций, на основе всего этого будет построен агрегированный алгоритм и проанализированы полученные результаты. Решение должно представлять собой разумный компромисс между качеством полученных изображений и скоростью обучения, однако первостепенной целью оптимизации будет являться качество изображений, получаемых на выходе работы сети.

3.1.5 Сверточные нейронные сети

Сверточная нейронная сеть известна уже достаточно давно. Она была спроектирована для задач классификации данных в графическом представлении.

Свое название сверточная нейронная сеть берет из названия операции – свертки. Для операции необходим сверточный фильтр (ядро свертки), который является матрицей небольшого размера с некоторыми весами. Он с некоторым шагом движется по двумерной матрице входных данных, совершая поэлементное умножение. Результатом операции является число, равное сумме полученных элементов. После чего пиксель, к которому была применена свертка, принимает полученное значение. Пример операции свертки представлен на рисунке №.

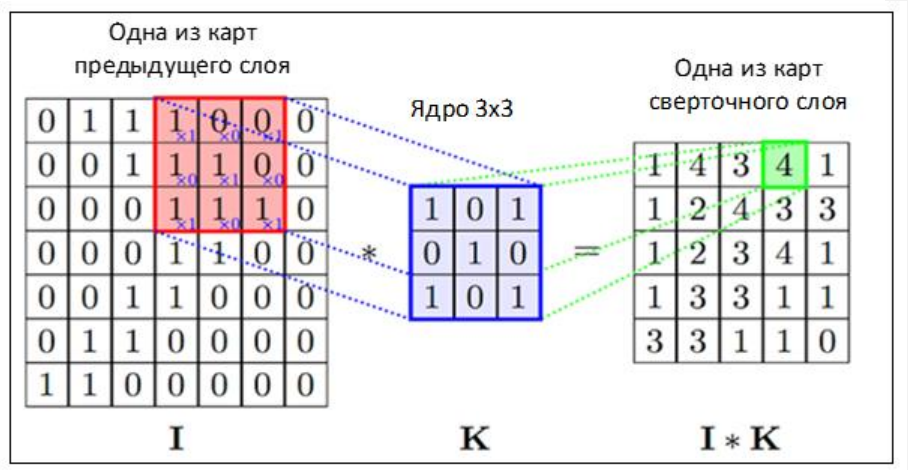

Рис. 4 Пример операции свертки

После слоя свертки должен идти слой подвыборки, так же известной как пулинг и субдискретизация. Он представляет собой нелинейное уплотнение карты признаков, при этом группа пикселей (обычно размера 2×2) уплотняется до одного пикселя, проходя нелинейное преобразование. Наиболее употребительна при этом функция максимума, однако так же используются функции минимума и среднего. Преобразования затрагивают непересекающиеся прямоугольники или квадраты, каждый из которых ужимается в один пиксель. Операция пулинга продемонстрирована на рисунке №5. С её помощью можно не только в разы уменьшить объём изображения, но и выявить больше признаков. Логику пулинга можно описать так: если на предыдущей операции свёртки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно сжимается до менее подробного, что позволяет убрать уже найденные признаки и найти новые.

Рис. 5 Пример работы max pooling

Сверточные сети являются удачным усреднением между биологически правдоподобными сетями и обычными многослойными персептронами, что делает их ключевой технологией Deep Learning.

3.1.6 Вариант модификации Spatial GAN

Spatial GAN – это одна из модификаций классической DCGAN. Её ключевая особенность заключается в том, что на вход генератору подается тензор Z, размера LxMxD. Далее генератор постепенно приводит его к размеру HxWxC, равному размеру изображений в датасете и передает дискриминатору, который производит обратные действия с картинкой, а именно, сворачивает к размеру LxMx1. Графически удобно представить модель в виде рисунка №6.

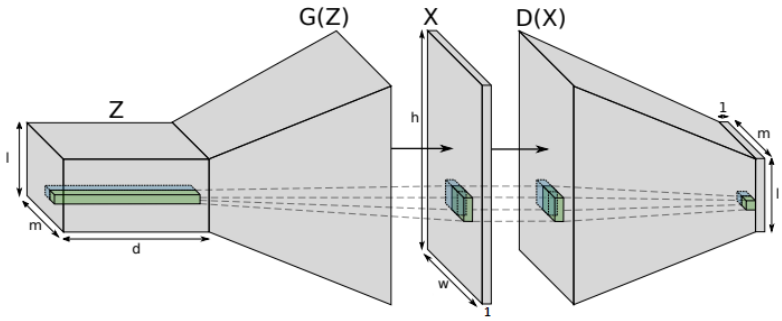

Рис. 6 Диаграмма SGAN

3.1.7 Вариант модификации U-net

U-net состоит из кодера, который понижает разрешение входного изображения с помощью сверточных слоев вплоть до того момента, когда изображение станет одномерным вектором, и декодера, повышающего дискретизацию изображения до необходимых размеров. Пропускные соединения, обозначенные пунктирными стрелками между соответствующими уровнями кодера и декодера, облегчают обучение, предоставляя важную информацию более низкого уровня от кодера к декодеру. Диаграмма модели представлена на рисунке №7.

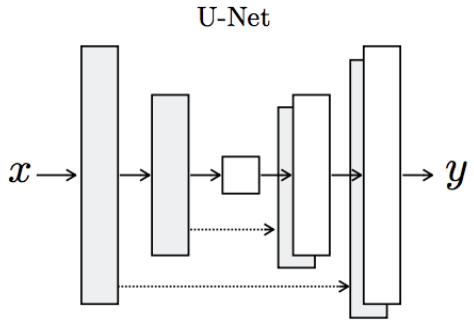

Рис. 7 Диаграмма U-Net

3.2 Разработка архитектур сетей для генерации изображений

Выбранные архитектуры базировались на статьях Nikolay Jetchev и Ryan Spick, описывающих SGAN, а так же Emmanouil Panagiotou, в которой демонстрируется U-net. Таким образом, исследование включало в себя построение двух различных архитектур и их сравнение.

Стоит заметить, что одной из распространенных проблем при обучении GAN является «Mode collapse». Заключается она в том, что генератор на протяжении многих итераций синтезирует одинаковое изображение из-за запоздания в обучении дискриминатора. Следовательно, структурная сложность дискриминатора должна быть меньше или равна сложности генератора.

Демонстрация реализованной модели SGAN (рис. №8, 9):

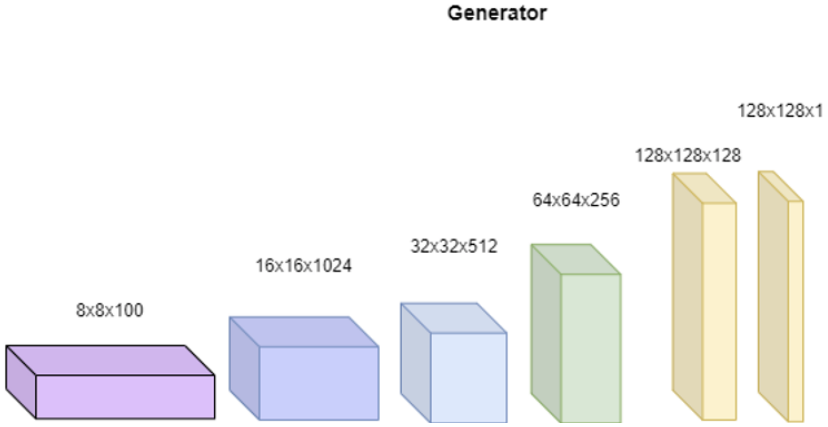

Рис. 8 Реализованная модель генератора SGAN

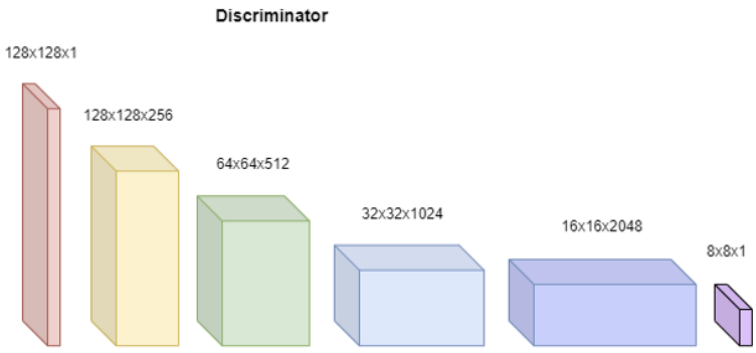

Рис. 9 Реализованная модель дискриминатора SGAN

Демонстрация реализованной модели U-net (рис. №10, 11):

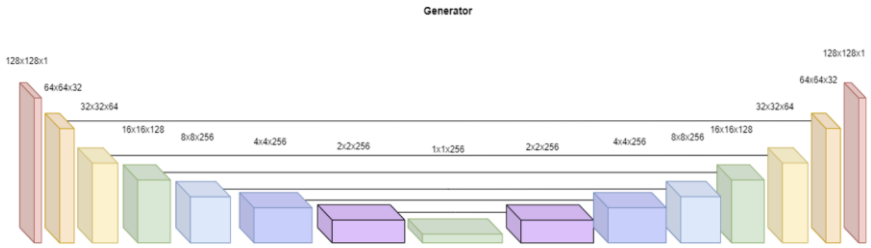

Рис. 10 Реализованная модель генератора U-net

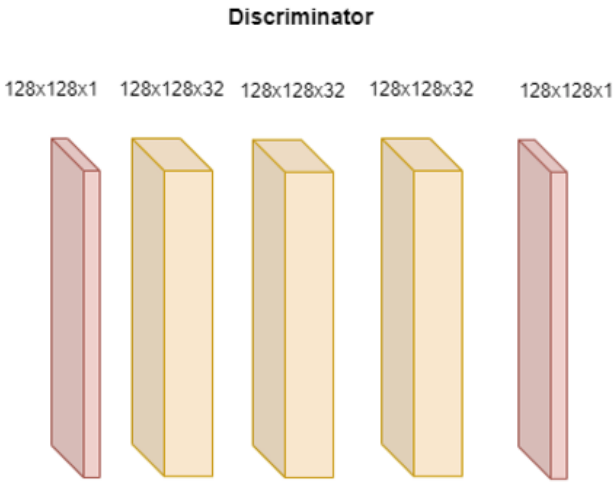

Рис. 11 Реализованная модель дискриминатора U-net

3.3 Подготовка обучающего набора

Исходными данными для датасета является карта высот Земли с сайта NASA (рис. №12). Карта высот имеет разрешение 21600x10800 пикселей, что позволяет её раздробить на изображения формата 256x256, 128x128 или 64x64.

Рис. 12 Карта высот Земли

Однако для формирования обучающего набора этого недостаточно. Необходимо избавиться от абсолютно или преимущественно черных изображений, размытых и растянутых изображений, так как они могут приведут к неправильному обучению нейронной сети. Для этого использовалось несколько «фильтров»:

- Первый, наиболее простой, считает среднюю интенсивность пикселей изображения и отсеивает те, интенсивность которых ниже определенного порога. Порог был найден экспериментально и равняется 25.
- Второй фильтр считает максимальную разность интенсивностей пикселей изображения и переводит их в метрическую систему. Аналогично предыдущему отсеивает картинки, максимальный перепад высот которых не превосходит 10 метров.
- Третий фильтр строит гистограмму градиентов изображения (HOG) по восьми возможным направлениям и отсеивает, если более 60% градиентов сонаправлены.

Для удобства и предстоящих исследований реализовано выявление изображений с рельефом типа «горы» и сохранение их в отдельную директорию на основе вышеописаных фильтров.

Генератор датасета работает по принципу «скользящего окна». Однако если сдвинуть начальную точку на половину размера окна по одной, либо по двум осям, получится новое изображение для датасета. Возможность смещения начальной точки реализована в общем виде, что позволяет генерировать большее количество данных.

3.4 Реализация нейронных сетей и эксперименты

Далее будут представлены реализации нейронных сетей, достигшие наибольшего успеха в генерации карт высот.

3.4.1 Реализация и эксперименты сети SGAN

Обучение происходило при следующих параметрах:

Размер батча = 32

Optimizer = Adam

Скорость обучения = 0.003

Регуляризация L2 = 1e-5

Размер фильтров дискриминатора: (5, 5)

Размер фильтров генератора: (7, 7)

Визуализация архитектуры посредством библиотечной функции plot_model() на рисунках № 13 и №14:

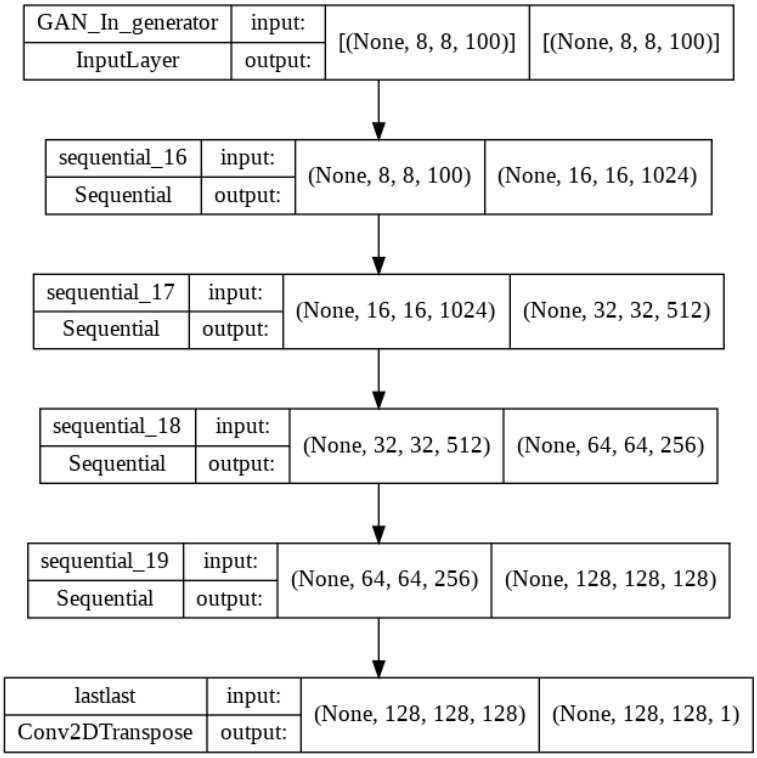

Рис. 13 Архитектура генератора

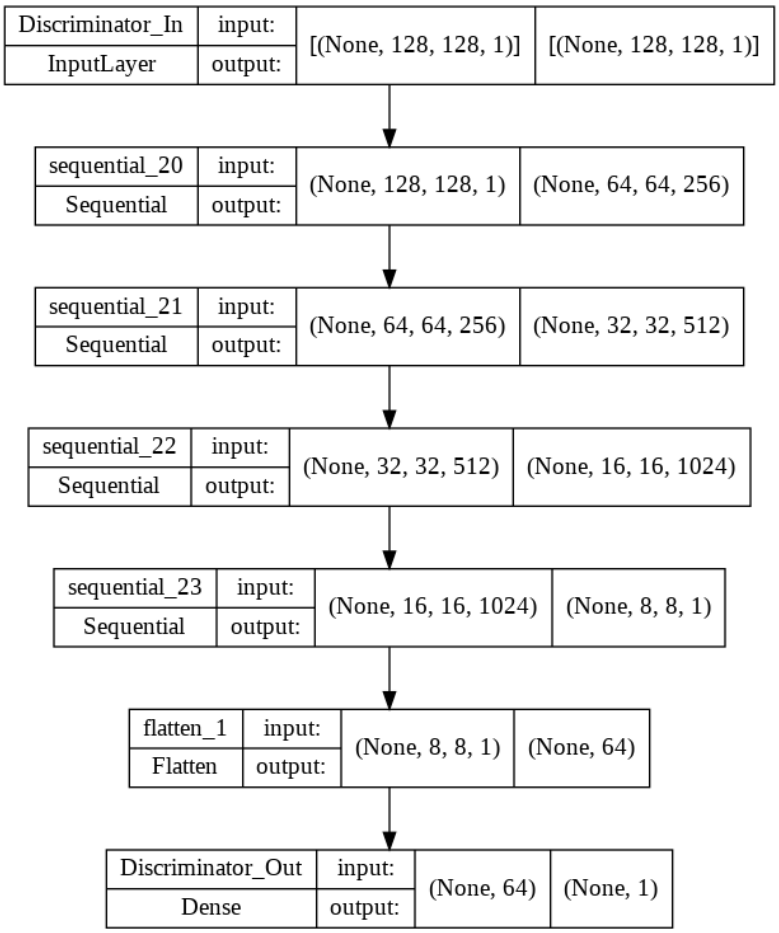

Рис. 14 Архитектура дискриминатора

Пример сгенерированных нейросетью карт высот:

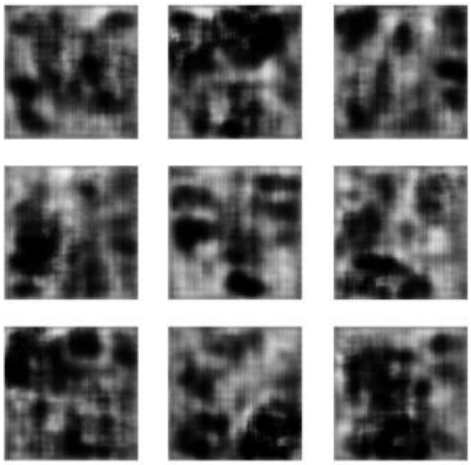

Рис. 15 Сгенерированные SGAN изображения на 8800 итерации

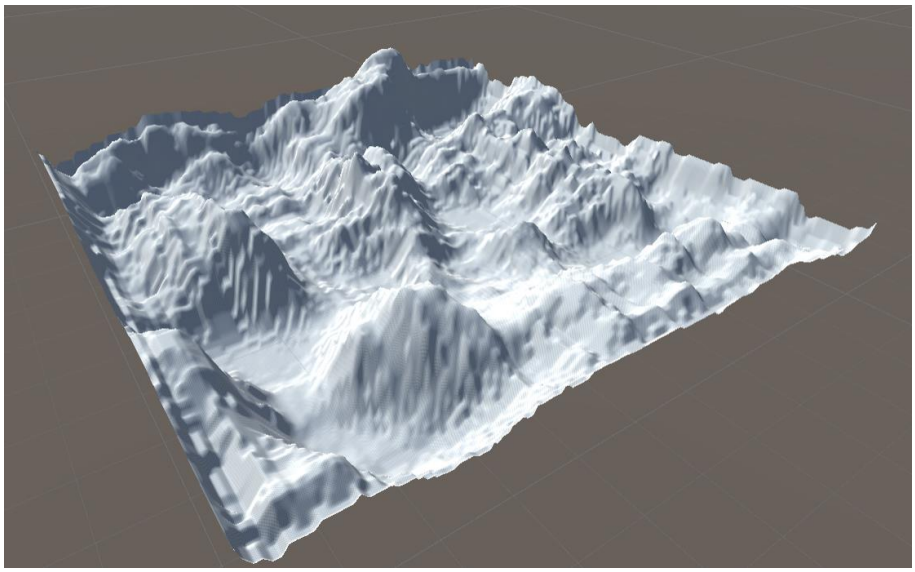

Рис. 16 Визуализация первого изображения из рисунка №15

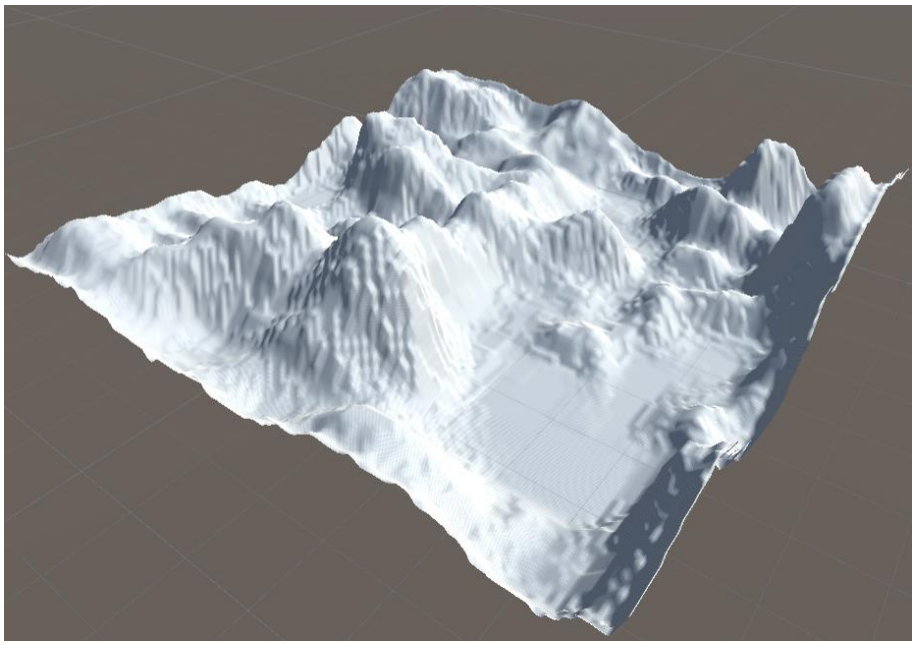

Рис. 17 Визуализация второго изображения из рисунка №15

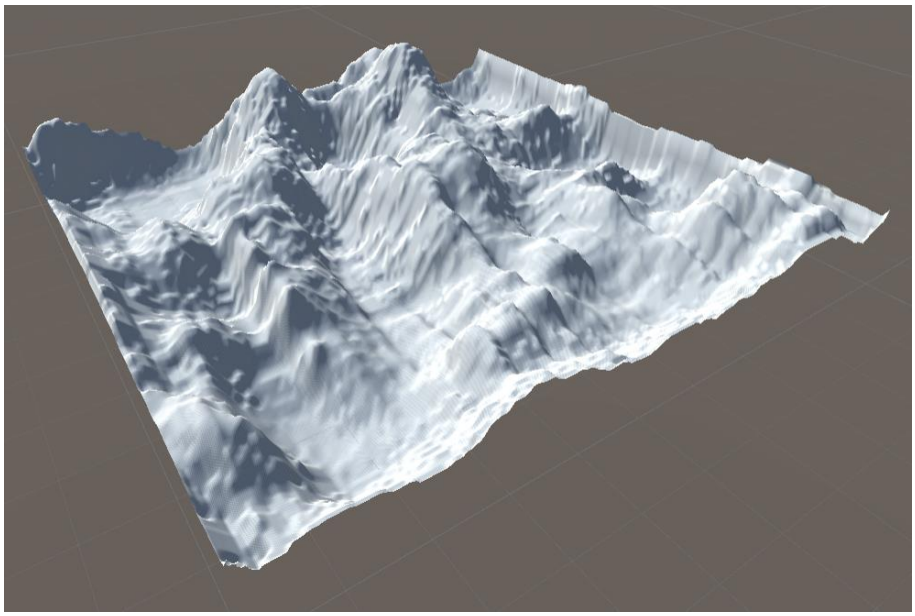

Рис. 18 Визуализация третьего изображения из рисунка №15

3.4.2 Реализация и эксперименты сети U-net

Обучение происходило при следующих параметрах:

Размер батча = 32

Optimizer = Adam

Скорость обучения = 0.002

Регуляризация L2 = 1e-5

Размер фильтров дискриминатора: (5, 5)

Размер фильтров генератора: (7, 7)

Визуализация архитектуры посредством библиотечной функции plot_model() на рисунках №19 и №20:

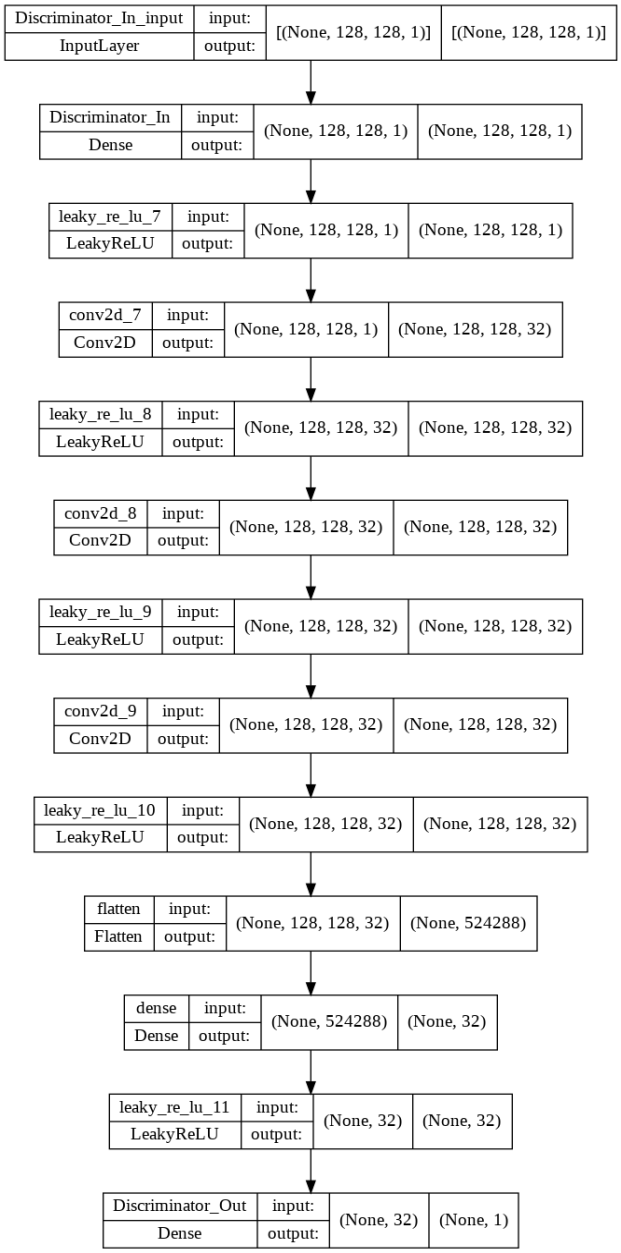

Рис. 19 Архитектура дискриминатора

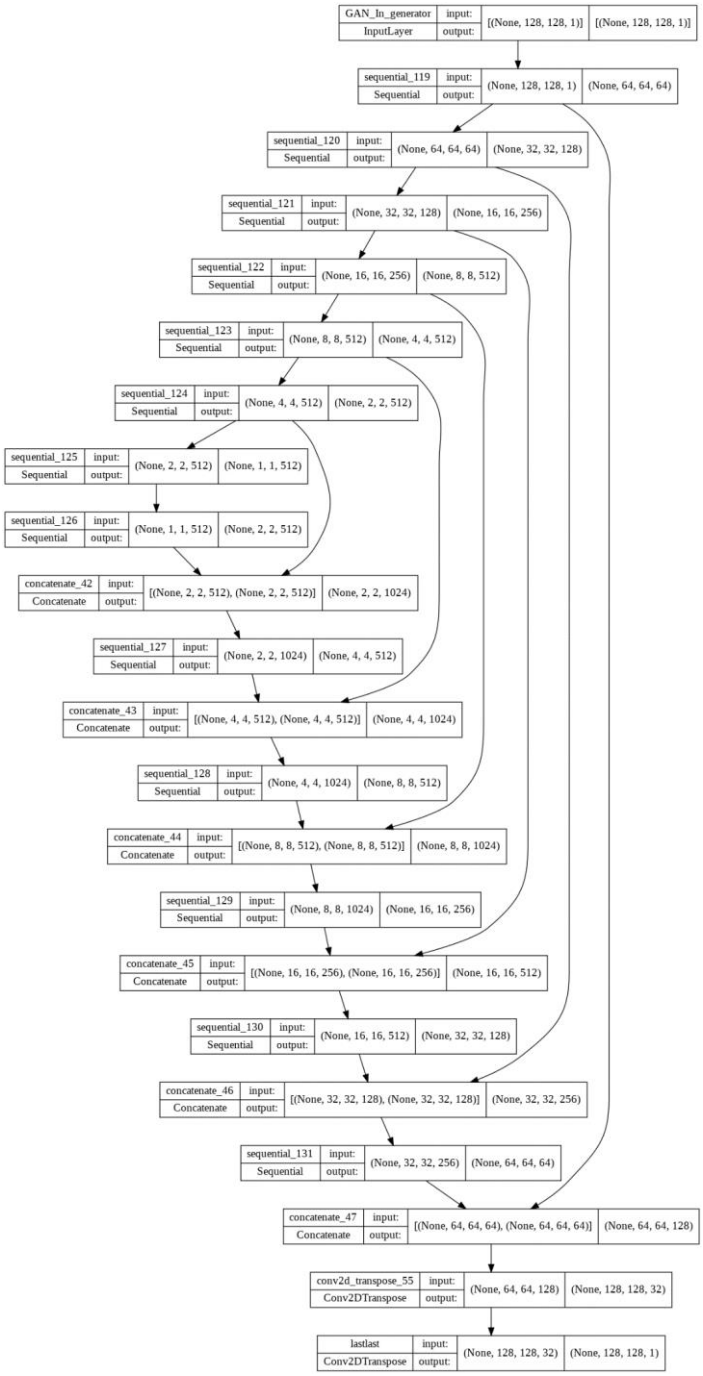

Рис. 20 Архитектура генератора

Пример сгенерированных нейросетью карт высот.

Результат был получен на 4300 итерации.

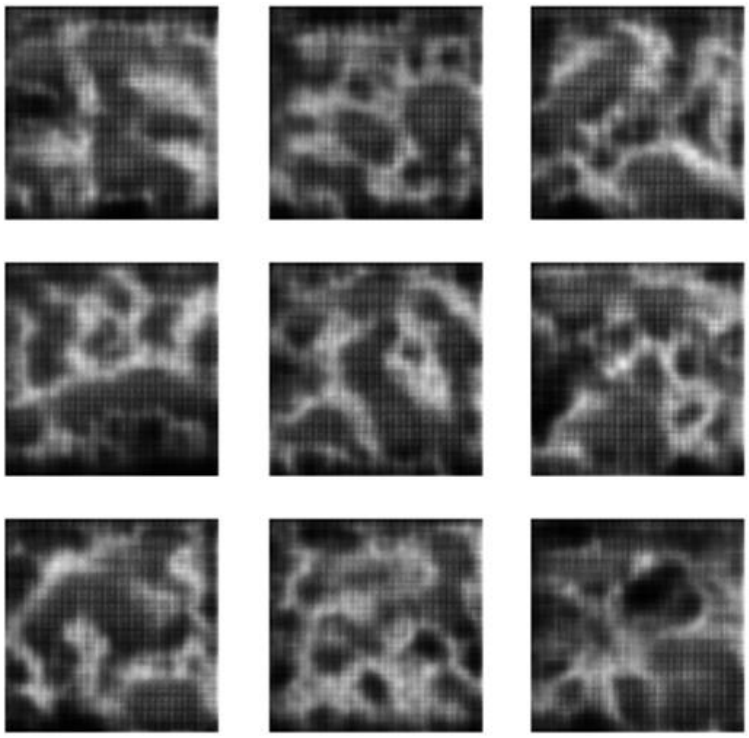

Рис. 21 Сгенерированные U-net -изображения на 4300 итерации

На сгенерированных картинках замечены помехи в виде сеточной структуры. От них можно избавиться, применив размывающий фильтр с ядром размера (5, 5). Далее на рисунках будут продемонстрированы трехмерные модели без постобработки и с постобработкой.

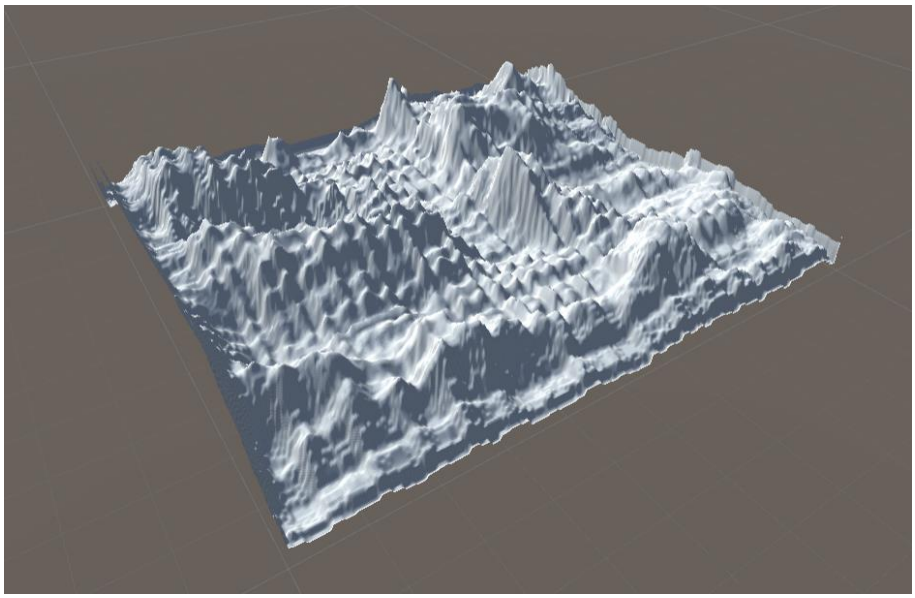

Рис. 22 Визуализация первого изображения из рисунка №21

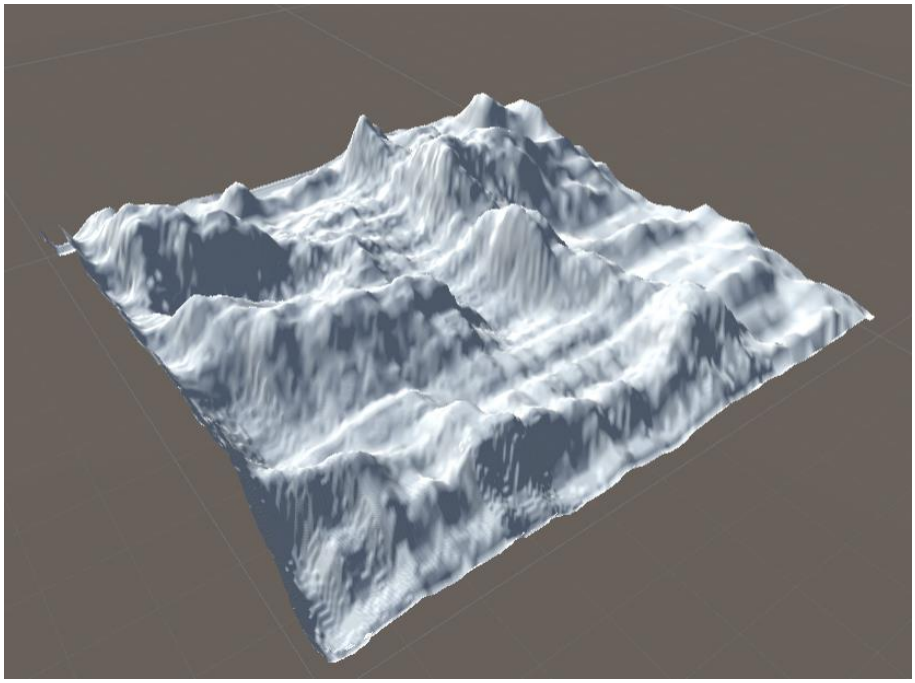

Рис. 23 Визуализация первого изображения из рисунка №21 с применением сглаживающего фильтра

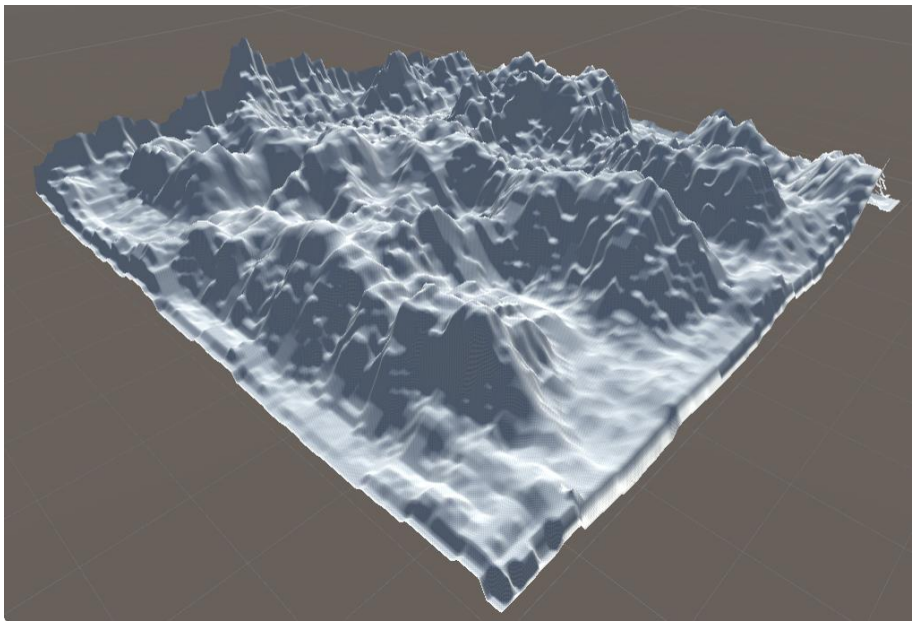

Рис. 24 Визуализация второго изображения из рисунка №21

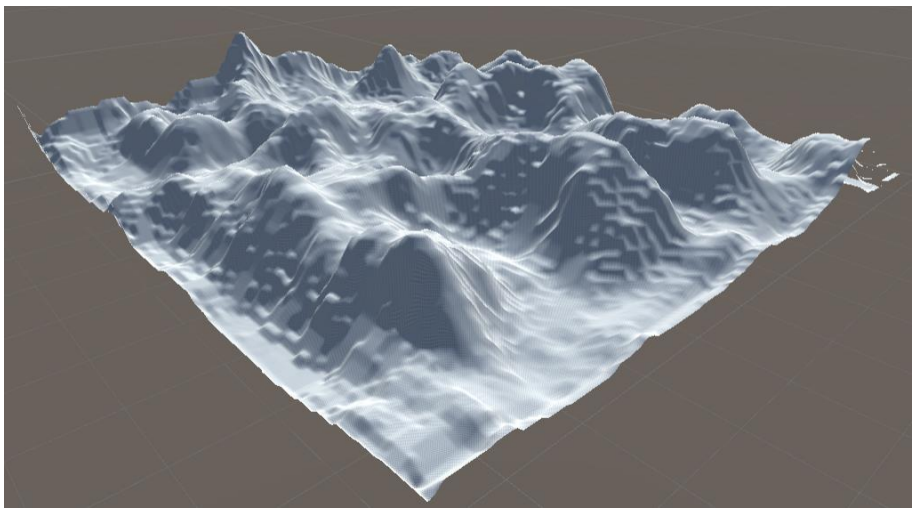

Рис. 25 Визуализация второго изображения из рисунка №21 с применением сглаживающего фильтра

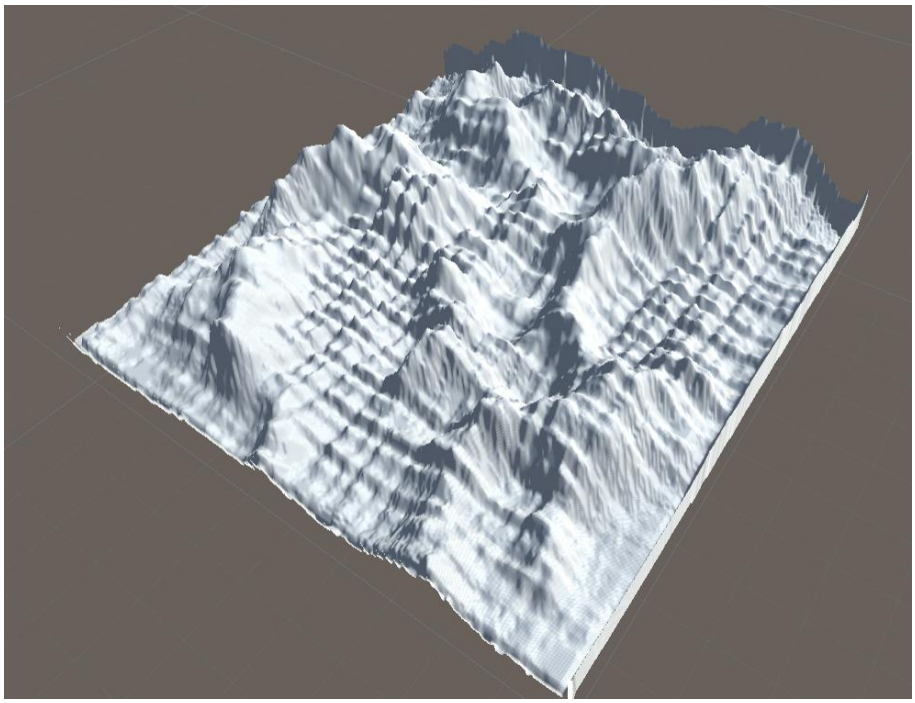

Рис. 26 Визуализация третьего изображения из рисунка №21

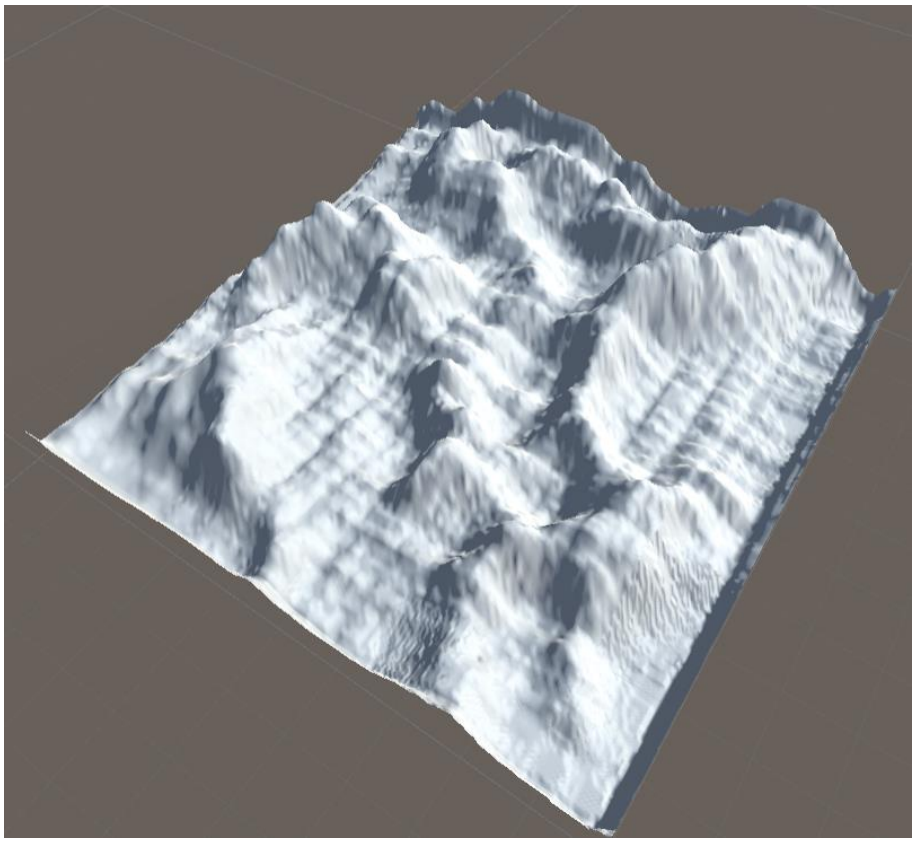

Рис. 27 Визуализация третьего изображения из рисунка №21 с применением сглаживающего фильтра

3.5 Сравнение реализованных подходов

В ходе экспериментов были собраны основные параметры нейронных сетей. Они приведены в таблицах №й и №2. Было выяснено, что U-net показывает себя лучше по всем параметрам, кроме наличия дефектов. С увеличением генерируемого изображения её затраты памяти и времени на одну итерацию не растут так же быстро, как у SGAN. Более того, во время проведения исследования было замечено, что U-net гораздо устойчивее к переобучению.

Из приведенных соображений можно сделать вывод, что для генерации карт высот рекомендуется использовать модель U-net.

SGAN			
Размер изображений в датасете	Время обучения для 1 эпохи (сек)	Затраченная видеопамять для обучения (ГБ)	Затраченная память обученной нейросети (МБ)
64x64	0,26	5,81 (21m)	90,9
128x128	0,55	6,06 (23m)	108,8
256x256	3,53	15,61(29m)	180,9

Таблица №1. Параметры SGAN.

U-net			
Размер изображений в датасете	Время обучения для 1 эпохи (сек)	Затраченная видеопамять для обучения (ГБ)	Затраченная память обученной нейросети (МБ)
64x64	0,24	8,06 (50m)	223,8
128x128	0,556	8,56 (82m)	448,5
256x256	1,38	8,56 (128.5m)	823,9

Таблица №2. Параметры U-net.

4. Заключение

В ходе работы были реализованы две модели, задачей которых была генерация карт высот. Обе реализации с задачей справились. В основном, их различие заключается в затраченных на обучение ресурсах и затраченном времени на реализацию и обучение, так как результаты почти одинаковы.

Следующим этапом исследования должна быть модификация, направленная на построение определенных типов ландшафта. На этом шаге должна возникнуть сложность создания обучающих наборов для разных видов рельефа: данные необходимо будет разделять и, скорее всего, используя только карты высот, этого добиться не получится. Возможное решение – использовать цветную или заранее размеченную карту.

По полученным результатам можно сделать вывод, что нейронные сети вполне подходят для построения ландшафтных карт. И даже текущие наработки могут быть использованы в игровой индустрии.

5. Список использованных источников

1. Emmanouil Panagiotou, Eleni Charou. 2020. Procedural 3D Terrain Generation using Generative Adversarial Networks . [Электронный ресурс] arXiv preprint https://arxiv.org/pdf/2010.06411.pdf (2020).
2. Generation of Synthetic Elevation Models and Realistic Surface Images of River Deltas and Coastal Terrains Using cGANs. 2021. [Электронный ресурс] https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9310217

3. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. 2014. [Электронный ресурс] Generative Adversarial Nets. arXiv preprint https://arxiv.org/pdf/1406.2661.pdf (2014).
4. Keras documentation [Электронный ресурс] https://keras.io/api/
5. NASA Earth height map [Электронный ресурс] https://visibleearth.nasa.gov/images/73934/topography (2005).
6. Nikolay Jetchev, Urs Bergmann and Roland Vollgraf. 2016. Texture synthesis with spatial generative adversarial networks. [Электронный ресурс] arXiv preprint https://arxiv.org/pdf/1611.08207.pdf (2016).
7. Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, Alexei A. Efros. 2020. CNN-generated images are surprisingly easy to spot... for now. [Электронный ресурс]
8. https://openaccess.thecvf.com/content_CVPR_2020/papers/Wang_CNN-Generated_Images_Are_Surprisingly_Easy_to_Spot..._for_Now_CVPR_2020_paper.pdf(2020)
9. Ryan Spick, Alfred Walke. 2019. Realistic and Textured Terrain Generation using GANs [Электронный ресурс] https://eprints.whiterose.ac.uk/153088/1/Real_world_Textured_terrain_generation_using_GANs_1_.pdf (2019)
10. Sakshi Indolia, Anil Kumar Goswami, S.P. Mishra, Pooja Asopaa. 2018. Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. [Электронный ресурс] https://www.sciencedirect.com/science/article/pii/S1877050918308019
11. Библиотека для работы с изображениями C++. [Электронный ресурс] https://github.com/nothings/stb/blob/master/stb_image.h
12. Кроссплатформенный игровой движок Unity. [Электронный ресурс] https://unity.com/ru