

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

ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ И ИНФОРМАТИКИ

БАКАЛАВРСКАЯ РАБОТА

«НЕЙРОСЕТЕВОЙ СИНТЕЗ ТЕКСТУР С ТРЕНДАМИ»

Выполнил студент

435 группы:

Будакян Я. С.

Научный руководитель:

к.т.н., доц. Грачев Е. А.

Допущена к защите

Зав. кафедрой _____

Москва

2017

Содержание

ВВЕДЕНИЕ	3
1 Постановка задачи	5
2 Нейронные сети	6
2.1 Математическая модель нейрона	7
2.2 Метод обратного распространения ошибки	8
3 Сверточные нейронные сети	9
3.1 Сверточная арифметика	9
3.2 Сверточные слои	9
4 Генеративные состязательные сети	9
4.1 Общая структура	10
4.2 Обучение GAN	11
4.3 Модификации	12
4.3.1 pix2pix GAN	12
5 Методы стохастической оптимизации	14
5.1 SGD	14
5.2 RMSprop	14
5.3 Adam	15
6 Оценка качества синтеза	15
7 Результаты	16
7.1 Выборки с реализацией одного тренда	16
7.1.1 Выборка 1	16
7.1.2 Выборка 2	18
7.1.3 Выборка 3	20
7.2 Выборки с реализацией различных трендов	22
7.2.1 Выборка 4	22
ВЫВОДЫ	22

ЗАКЛЮЧЕНИЕ	22
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	23
ПРИЛОЖЕНИЕ	25

ВВЕДЕНИЕ

В современной добыче полезных ископаемых активно применяются средства математического моделирования для симуляции процессов, происходящих в пласте. Однако, на данный момент полноценное моделирование не представляется возможным, поэтому используются различные приближенные методы. Для правильной симуляции процессов, происходящих в недрах, необходимо смоделировать саму среду, в которой эти процессы протекают. Обычно, данные о структуре среды доступны только в некотором количестве точек(скважин), в которых непосредственно идет добыча. Предлагается попытаться использовать методы машинного обучения для синтеза модели среды, похожей на природную. В рамках решения этой проблемы возникает задача синтеза текстур с трендами, например, для учета каких-либо межскважинных корреляций.

Под текстурой с трендом в данном случае понимается изображение, в котором есть изменение некоторой статистической характеристики вдоль одного из направлений. Такими характеристиками, например, могут быть изменение интенсивности появления частиц среды или пористости среды.

Базовым подходом в задаче синтеза текстур на данный момент является использование искусственных нейросетей. Однако, известные работы в области синтеза текстур с помощью искусственных нейросетей [1, 2] показывают, что у нейросетевых моделей есть проблемы с воспроизведением формы объектов, а также различных пространственно скоррелированных структур. Соответственно, целью данной работы ставится поиск нейросетевой архитектуры, способной улавливать и воспроизводить протяженные корреляции.

Для упрощения задачи, будем в дальнейшем рассматривать множество изображений с трендами, удовлетворяющее следующим ограничениям:

- Это монохромные изображения 256×256 пикселей
- Изменяющимся свойством является интенсивность появления частиц λ
- Тренд является линейным и направлен вдоль оси изображения z_1 : $\lambda = \lambda_{init} + kz_1$
- По оси z_2 остается равномерное распределение частиц

Каждый такой тренд фиксируется значениями λ_{init} и λ_{final} . Соответственно,

$$k = \frac{\lambda_{final} - \lambda_{init}}{256}$$

Пример подобного изображения с трендом приведен на (Рис. 1):

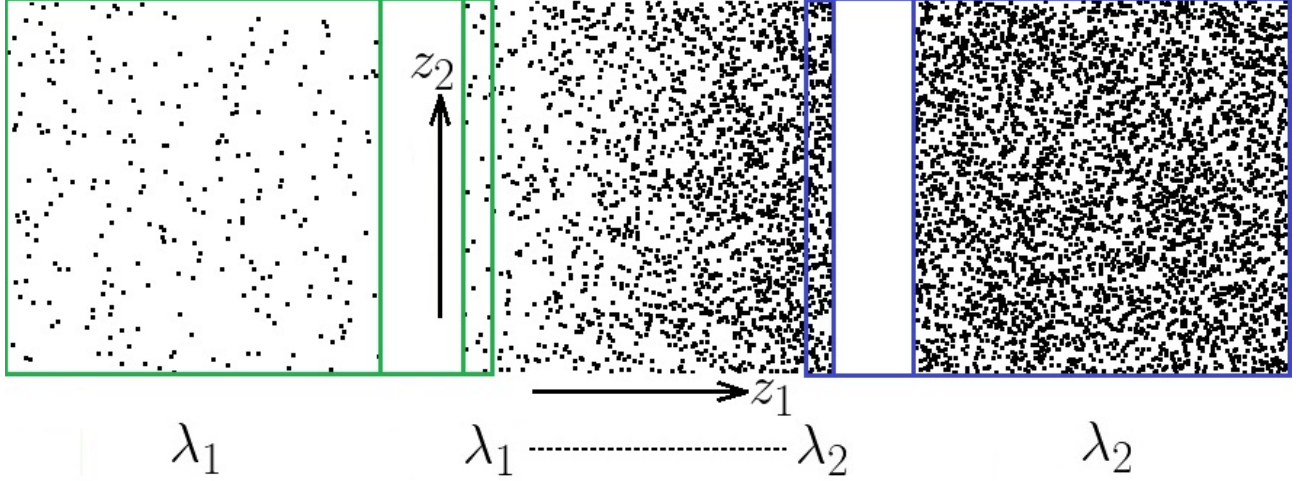


Рис. 1: Пример изображения с трендом, фиксируемого двумя изображениями

1 Постановка задачи

Математически сформулировать постановку описанной задачи можно с помощью так называемой вероятностной постановки задачи обучения [3, 4]. Рассмотрим многомерное пространство X , содержащее множество всех изображений x : $X = \{x\}$. Пусть у нас есть обучающая выборка из изображений, содержащих в себе рассматриваемое множество трендов $D = \{x_i\}$. Тогда считается, обучающая выборка изображений с трендами D задает 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, с математической точки зрения задача синтеза текстуры с трендом 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 в рассмотрение параметризованного семейства распределений вероятности и его подстройке к имеющимся данным:

- Вводится параметризованное семейство распределений вероятности $P_\theta(x)$
- Параметры 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 приводит к проблемам:

- Пространство параметров θ может быть огромной размерности
- Известной параметрической модели распределения может вообще не существовать

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) [5, 6, 7, 8], что мотивирует попытку применения нейросетей этой архитектуры в поставленной задаче.

2 Нейронные сети

В общем смысле, искусственная нейронная сеть - это математическая модель, построенная по принципу организации и функционирования биологических нейронных сетей. Она представляет из себя систему соединенных 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.

2.1 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. 2).

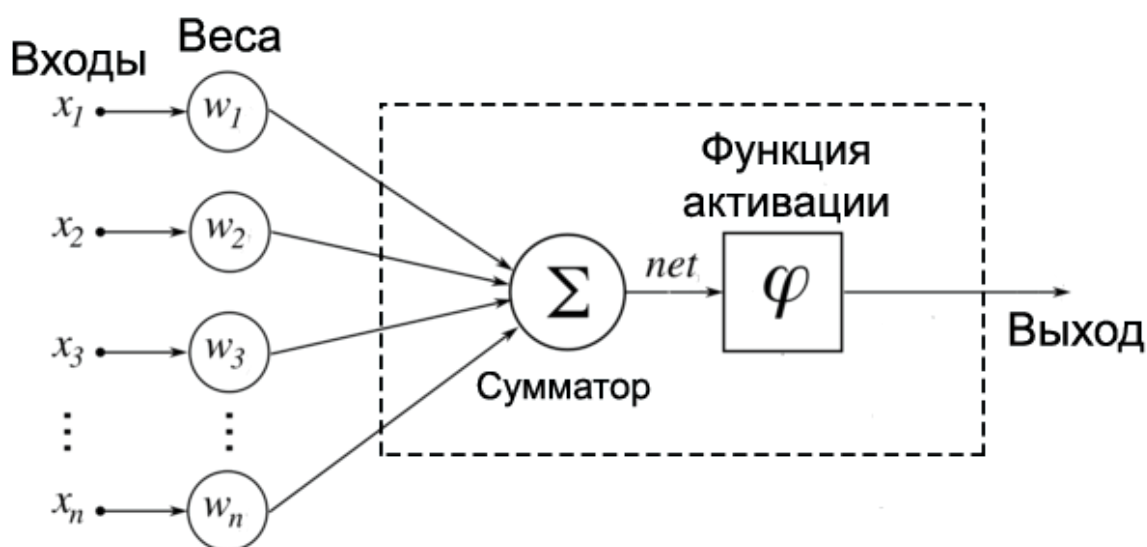


Рис. 2: Математическая модель нейрона

Функция активации может выбираться разной 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.

2.2 Метод обратного распространения ошибки

Метод обратного распространения ошибки (backpropagation) - самый широко используемый и успешный алгоритм обучения глубоких (многослойных) нейронных сетей. Суть этого 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, в стохастическом 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. Введем 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.

3 Сверточные нейронные сети

Сверточные нейронные сети - это специальная архитектура нейронной сети, нацеленная на эффективное распознавание изображений, впервые предложенная Яном Лекуном [9]. Структура такой сети имеет некоторое сходство со строением зрительной коры головного мозга.

3.1 Сверточная арифметика

3.2 Сверточные слои

4 Генеративные состязательные сети

Архитектура нейронной сети, получившая название генеративной состязательной сети (generative adversarial network - GAN), впервые была описана в 2014 году [4]. 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.

4.1 Общая структура

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

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

Введем параметризованную процедуру генерации:

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

Переформулируем:

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

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

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

Возникает вопрос: что использовать в качестве метрики похожести двух распределений ρ , где одно из распределений 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 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. произведенные гене-

ратором). Для дальнейшего рассмотрения введем функцию потерь дискриминатора(например, logloss):

$$l_1 = l(d_\zeta(x), 1) - \text{ошибка 1 рода}$$

$$l_2 = l(d_\zeta(x'), 0) - \text{ошибка 2 рода}$$

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

Функция потерь обученного классификатора:

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

Соответственно,

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

Определим оптимальный дискриминатор:

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

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

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 изображен на (Рис. 3).

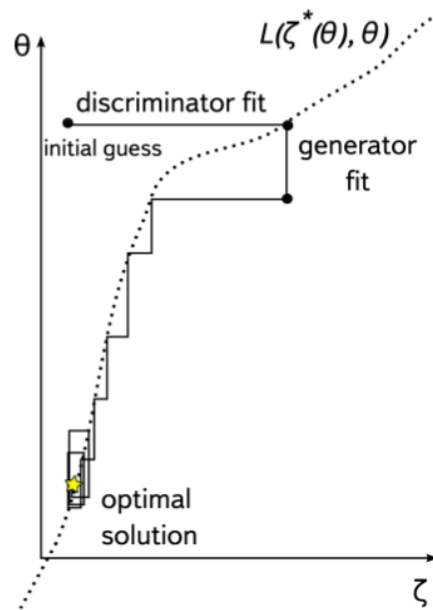


Рис. 3: Схематическое изображение процесса обучения GAN.

4.3 Модификации

4.3.1 pix2pix GAN

Для решения задачи была опробована модификация GAN под названием “pix2pix GAN” [10, 11]. Ее отличие от схемы GAN, введенной выше, состоит в том, что вместо шума на вход генератору приходят другие изображения, на которых он основывается при синтезе. Схематически ее устройство изображено на (Рис. 4).

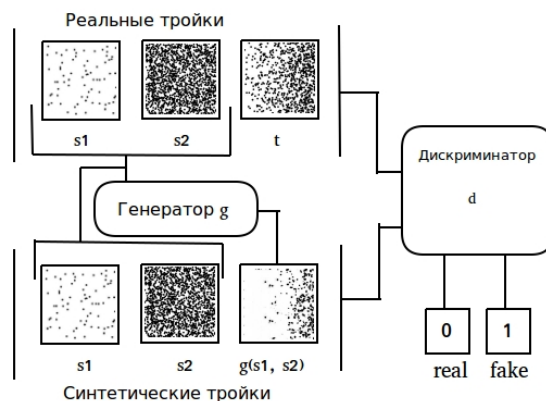


Рис. 4: Схематическое устройство сети pix2pix GAN.

Для pix2pix сети общий функционал потерь выглядит следующим образом:

$$L(G, D) = L_{adv}(G, D) + \eta L1$$

$$L1 = \mathbb{E}_{p_{data}(s_1, s_2, r)}(\| r - G(s_1, s_2) \|_1)$$

$$L_{adv}(G, D) = \mathbb{E}_{p_{data}(s_1, s_2, r)} \log D(s_1, s_2, r) + \mathbb{E}_{p_{data}(s_1, s_2)} \log(1 - D(s_1, s_2, G(s_1, s_2)))$$

где G, D - генератор и дискриминатор, (s_1, s_2, r) - тройка изображений (интенсивность слева, справа и реальное изображение с трендом), $\mathbb{E}_{p_{data}(s_1, s_2, r)}$ - мат. ожидание логарифмического правдоподобия того, что тройка изображений (s_1, s_2, r) принадлежит вероятностному распределению реальных троек $p_{data}(s_1, s_2, r)$, а $p_{data}(s_1, s_2)$ соответствует распределению реальных изображений s_1, s_2 .

В качестве генератора в [10, 11] использовалась сеть “U-Net” [12]. Основное отличие сети “U-Net” от обычной сети архитектуры “encoder-decoder” заключается в наличии прямых связей между сверточными и разверточными слоями. Использование такого типа генератора позволяло увеличить качество синтезируемых изображений. Схемы сетей типа “U-Net” и “Encoder-decoder” приведены на (Рис. 5).

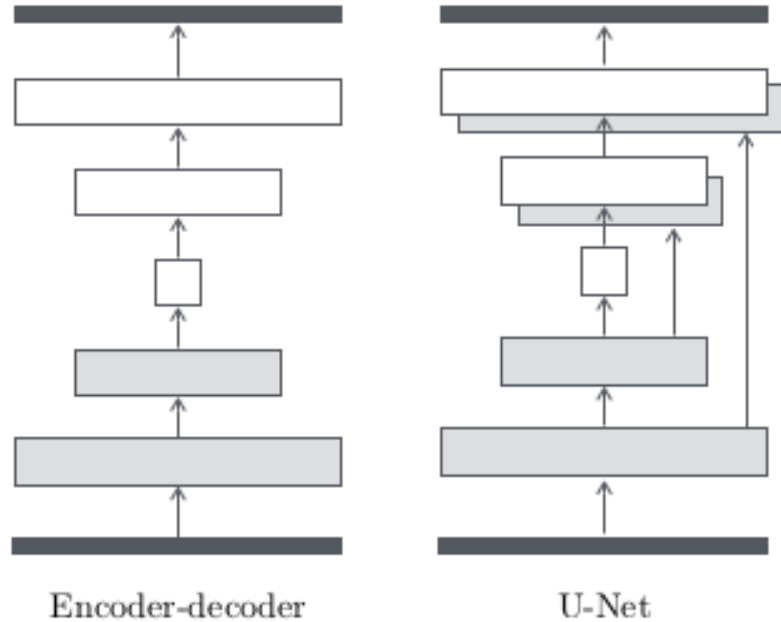


Рис. 5: Схематическое изображение нейросети-генератора.

5 Методы стохастической оптимизации

Для обучения глубоких нейросетей применяется метод обратного распространения ошибки, который позволяет рассчитать градиенты весов, и различные методы стохастической оптимизации первого порядка. В данной работе для обучения моделей применялось 2 алгоритма оптимизации под названиями “RMSprop” и “Adam”. Оба эти алгоритма являются некоторыми модификациями стохастического градиентного спуска (SGD).

5.1 SGD

Описание стохастического градиентного спуска есть в [13]. Стохастический градиентный 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.

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

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

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

5.2 RMSprop

RMSprop расшифровывается как root mean square propagation. Описание этого алгоритма дано в [14]. 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.

5.3 Adam

Adam расшифровывается как adaptive moment estimation. Описание этого алгоритма дано в [15]. Этот алгоритм, помимо перемасштабирования, использует инерцию градиента, что позволяет смягчить быстрое изменение градиента, присущее стохастическому градиентному спуску.

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

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

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

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

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

6 Оценка качества синтеза

После обучения сети, необходимо проверить, что сгенерированные ей изображения действительно имеют искомые характеристики (то есть, содержат искомый тренд). Для этого вводится специальная метрика, которая будет учитывать наличие в изображении тренда интенсивности частиц. Рассмотрим среднюю плотность черных пикселей в некотором окне ξ_k , и пройдем этим окном по изображению (Рис. 6).

$$\xi_k = \frac{1}{Hw} \sum_{i=k}^{k+w} \sum_{j=0}^H \left| \frac{x(i, j) - 255}{255} \right|,$$

$$k = \overline{1, W - w}$$

Построив график $\xi(k)$, можно увидеть, как меняется плотность пикселей и прослеживается ли тренд. В качестве метрики можно взять среднеквадратичную ошибку:

$$\xi = \frac{1}{W - w} \sum_{k=1}^{W-w} (\xi_k - \xi_{0k})^2,$$

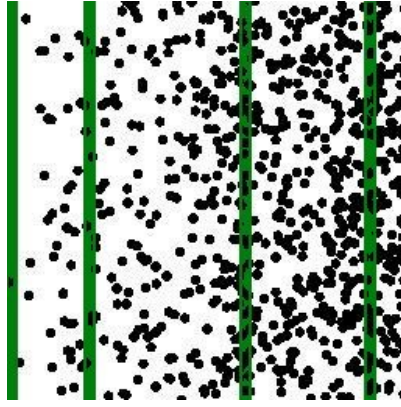


Рис. 6: Прохождение окном, W, H - размеры изображения, w - ширина окна.

где ξ_{0k} - это ξ_k , усредненное по примерам из обучающей выборки. Соответственно, чем меньше значение метрики, тем лучше тренд, присутствующий на сгенерированном изображении, приближает искомый.

7 Результаты

Описанные подходы были реализованы в виде компьютерных программ на языке Python с помощью библиотеки для построения искусственных нейронных сетей Keras [21], которая, в свою очередь, использует для расчетов библиотеку Tensorflow [22]. Используемые версии программных пакетов указаны в Приложении 1. Обучение проводилось на синтетических данных, сгенерированных самостоятельно реализованным генератором.

7.1 Выборки с реализацией одного тренда

7.1.1 Выборка 1

Выборка состояла из 3000 обучающих изображений и 500 валидационных. Частицами среды были круги диаметром 7 пикселей. Все изображения содержали в себе различные случайные реализации одного тренда интенсивности

$$\lambda(x) = 0.2 + 0.01875x : \lambda_i = 0.2, \quad \lambda_f = 5$$

Отличия в параметрах обученных нейросетей указаны в (Таб. 1). Для всех обученных моделей параметр η , отвечающий за баланс между L_{adv} и L_1 был равен 100.

Сеть	Число фильтров на первом слое	Сеть-генератор
nf8	8	U-Net
nf16	16	U-Net
nf16woU	16	Encoder-decoder
nf32	32	U-Net

Таблица 1: Отличия в параметрах моделей (Выборка 1)

Примеры синтеза, полученные с помощью нейросетей с различными параметрами, приведены в (Таб. 2).

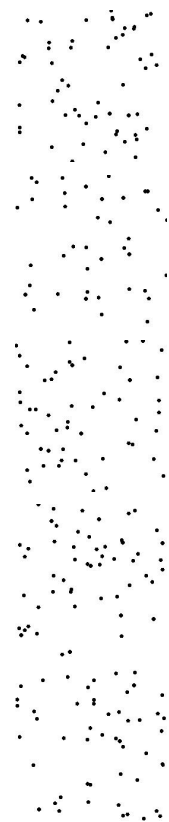


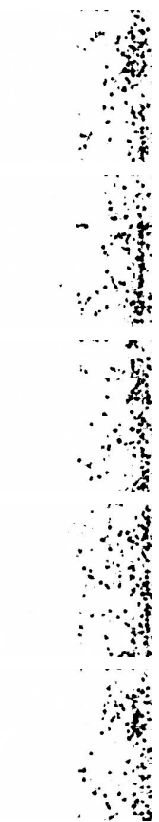
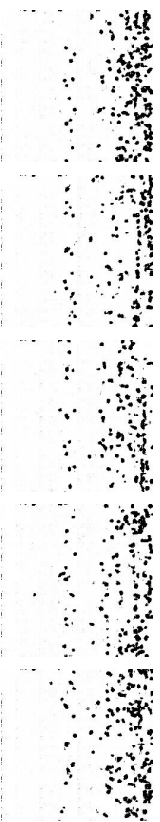

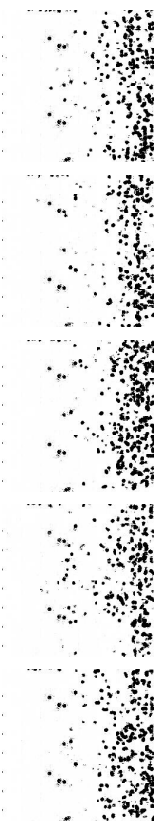
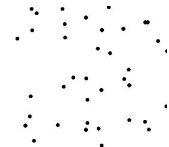



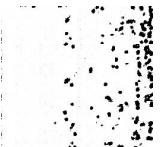










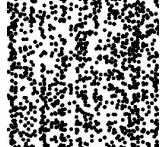



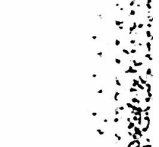

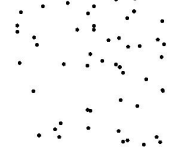






Вход 1	Вход 2	Тренд	Сеть nf8	nf16	nf16woU	nf32
						
						
						
						
						

Таблица 2: Примеры синтеза (Выборка 1)

График приближения тренда различными сетями показан на (Рис. 7). Значения введенной метрики для различных сетей указаны в (Таб. 3). Лучшей по введенной метрике оказалась сеть nf32.



Рис. 7: Аппроксимация тренда различными сетями (Выборка 1)

Сеть	Метрика
nf16woU	0.24048
nf8	0.22511
nf16	0.18844
nf32	0.14589

Таблица 3: Значения метрики для разных сетей (меньше - лучше)

7.1.2 Выборка 2

Выборка состояла из 6000 обучающих изображений и 316 валидационных. Частицами среды были квадраты стороной 3 пикселя. Все изображения содержали в себе различные случайные реализации одного тренда интенсивности

$$\lambda(x) = 1 + 0.0546875x : \lambda_i = 1, \quad \lambda_f = 15$$

Отличия в параметрах обученных нейросетей указаны в (Таб. 4). Во всех обученных моделях в качестве сети-генератора использовалась сеть “U-Net”.

Примеры синтеза, полученные с помощью нейросетей с различными параметрами, приведены в (Таб. 5).

График приближения тренда различными сетями показан на (Рис. 8).

Значения введенной метрики для различных сетей указаны в (Таб. 6).

Лучшей по введенной метрике оказалась сеть nf64e1.

Сеть	Число фильтров на первом слое	η	Метод оптимизации
nf32e5	32	5	Adam
nf64e1	64	1	RMSprop
nf64e5	64	5	RMSprop
nf64e10	65	10	RMSprop

Таблица 4: Отличия в параметрах моделей (Выборка 2)




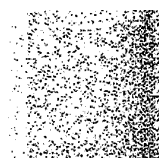
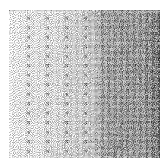

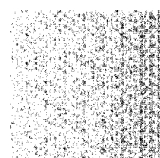

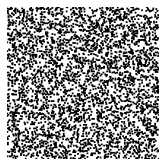


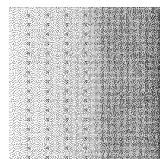
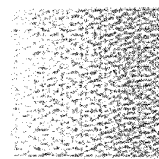
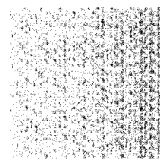

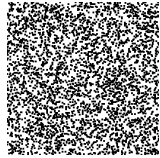
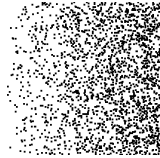
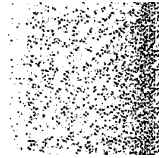
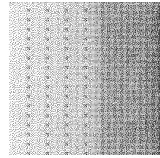
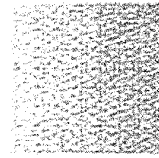
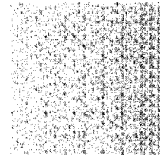
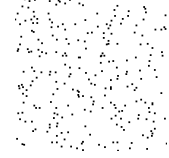



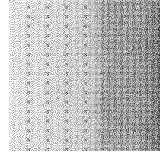
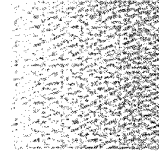
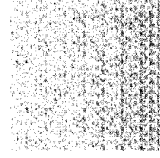
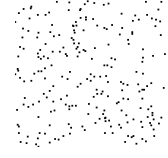



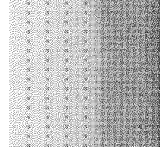
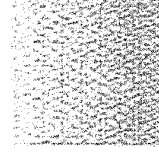
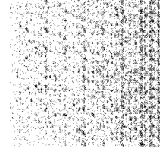
Вход 1	Вход 2	Тренд	Сеть nf32e5	nf64e1	nf64e5	nf64e10
						
						
						
						
						

Таблица 5: Примеры синтеза (Выборка 2)

Сеть	Метрика
nf64e10	0.11168
nf64e5	0.06501
nf32e5	0.04827
nf64e1	0.01393

Таблица 6: Значения метрики для разных сетей (меньше - лучше)



Рис. 8: Аппроксимация тренда различными сетями (Выборка 2)

7.1.3 Выборка 3

Выборка состояла из 6000 обучающих изображений и 316 валидационных. Частицами среды были единичные пиксели. Все изображения содержали в себе различные случайные реализации одного тренда интенсивности

$$\lambda(x) = 1 + 0.19140625x : \lambda_i = 1, \quad \lambda_f = 50$$

Отличия в параметрах обученных нейросетей указаны в (Таб. 7). Во всех обученных моделях в качестве сети-генератора использовалась сеть “U-Net”. В качестве метода оптимизации использовался Adam.

Сеть	Число фильтров на первом слое	η
nf64e1	64	1
nf64e5	64	5
nf100e0	100	0
nf100e5	100	5

Таблица 7: Отличия в параметрах моделей (Выборка 3)

Примеры синтеза, полученные с помощью нейросетей с различными параметрами, приведены в (Таб. 8).

График приближения тренда различными сетями показан на (Рис. 9).

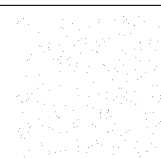
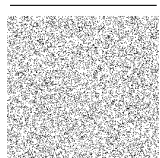
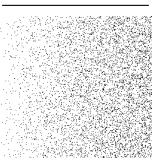

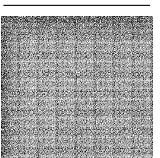
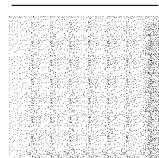
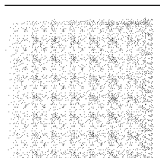

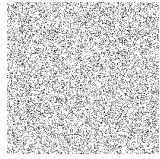
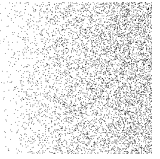
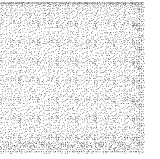
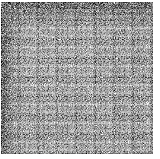
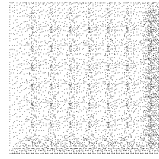
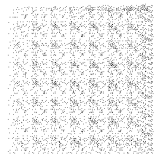


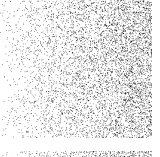
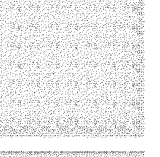
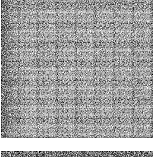
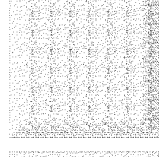
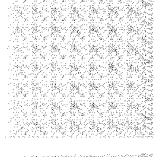

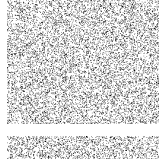
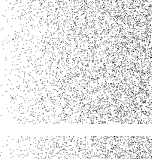

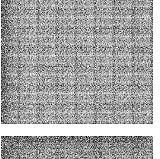
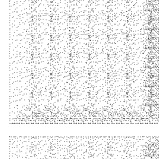
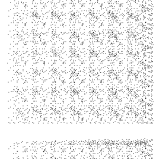

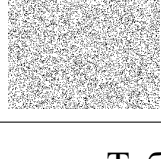
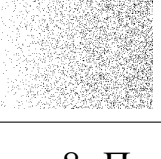
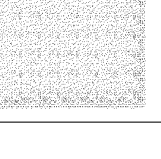
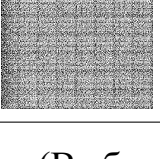
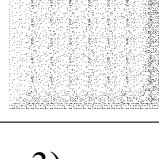
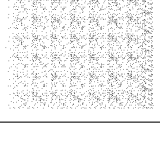
Вход 1	Вход 2	Тренд	Сеть nf64e1	nf64e5	nf100e0	nf100e5
						
						
						
						
						

Таблица 8: Примеры синтеза (Выборка 3)

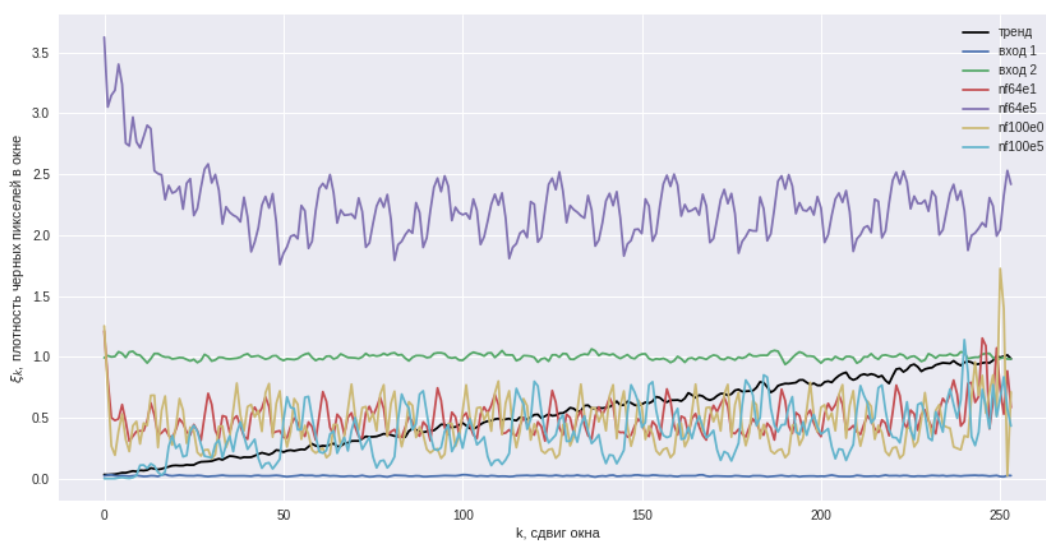


Рис. 9: Аппроксимация тренда различными сетями (Выборка 3)

Значения введенной метрики для различных сетей указаны в (Таб. 9).

Сеть	Метрика
nf64e5	3.13633
nf100e0	0.1178
nf100e5	0.08266
nf64e1	0.08139

Таблица 9: Значения метрики для разных сетей (меньше - лучше)

Лучшей по введенной метрике оказалась сеть nf64e1. Однако видно, что на этой выборке сети не смогли распознать тренд, и сошлись примерно к средней по выборке интенсивности (не учитывая сети nf64e5, явно являющейся выбросом).

7.2 Выборки с реализацией различных трендов

7.2.1 Выборка 4

Выборка состояла из 6000 обучающих изображений и 316 валидационных. Частицами среды были единичные пиксели. Все изображения содержали в себе различные случайные реализации различных трендов интенсивности $\lambda(x)$ (то есть, значения λ_{init} и λ_{final} для каждого отдельного экземпляра выбирались случайно).

тут картинки примеров из обучающей выборки

тут сгенерированные семплы

тут метрики, графики

ВЫВОДЫ

Ну вот и чем выводы от результатов отличаются...

ЗАКЛЮЧЕНИЕ

Список литературы

- [1] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge: "Texture Synthesis Using Convolutional Neural Networks"// arXiv: 1505.07376 [cs.CV], 2015.
- [2] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, Victor Lempitsky: "Texture Networks: Feed-forward Synthesis of Textures and Stylized Images"// arXiv: 1603.03417 [cs.CV], 2016.
- [3] Воронцов К. В.: "Математические методы обучения по прецедентам (теория обучения машин)".
- [4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio: "Generative Adversarial Nets"// arXiv: 1406.2661 [stat.ML], 2014.
- [5] Mehdi Mirza, Simon Osindero: "Conditional Generative Adversarial Nets"// arXiv: 1411.1784 [cs.LG], 2014.
- [6] Jon Gauthier: "Conditional generative adversarial nets for convolutional face generation Tech. rep., 2015.
- [7] Junbo Zhao, Michael Mathien, Yann LeCun: "Energy-based Generative Adversarial Networks"// arXiv: 1609.03126 [cs.LG], 2016.
- [8] David Berthelot, Thomas Schumm, Luke Metz: "BEGAN: Boundary Equilibrium Generative Adversarial Networks"// arXiv: 1703.10717 [cs.LG], 2017.
- [9] 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.
- [10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros: "Image-to-Image Translation with Conditional Adversarial Networks"// arXiv: 1611.07004 [cs.CV], 2016.

- [11] 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.
- [12] Olaf Ronneberger, Philipp Fischer, Thomas Brox: "U-Net: Convolutional Networks for Biomedical Image Segmentation"// arXiv: 1505.04597 [cs.CV], 2015.
- [13] Amari, Shunichi: "A theory of adaptive pattern classifiers"// Electronic Computers, IEEE Transactions on 3, ctp. 299-307, 1967.
- [14] 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.
- [15] Diederik P. Kingma, Jimmy Lei Ba: "Adam: A method for stochastic optimization"// arXiv:1412.6980 [cs.LG], 2014.
- [16] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge: "A Neural Algorithm of Artistic Style"// arXiv: 1508.06576 [cs.CV], 2015.
- [17] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, Koray Kavukcuoglu: "Conditional Image Generation with PixelCNN Decoders"// arXiv: 1606.05328 [cs.CV], 2016.
- [18] Augustus Odena, Christopher Olah, Jonathon Shlens: "Conditional Image Synthesis with Auxiliary Classifier GANs"// arXiv: 1610.09585 [stat.ML], 2016.
- [19] Tejas D. Kulkarni, Will Whitney, Pushmeet Kohli, Joshua B. Tenenbaum: "Deep Convolutional Inverse Graphics Network"// arXiv: 1503.03167 [cs.CV], 2015.
- [20] Chuan Li, Michael Wand: "Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis"// arXiv: 1601.04589 [cs.CV], 2016.
- [21] François Chollet: Keras, 2015. Software available from <http://keras.io/>.

- [22] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng: "TensorFlow: Large-scale machine learning on heterogeneous systems 2015. Software available from <http://tensorflow.org/>.

ПРИЛОЖЕНИЕ

Приложение 1. Используемые версии пакетов

- Python 3.6
- Tensorflow 1.1.0
- Keras 2.0.4

Приложение 2. Еще что-нибудь

Еще что-нибудь