

# Нейросетевой синтез текстур с трендами

Будалян Я. С.

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

2017 г.

## Введение

Задача состоит в синтезе изображений среды, которые будут содержать в себе тренд, т.е. изменение некоторой статистической характеристики. Такими трендами могут быть, например, изменение интенсивности появления частиц среды вдоль изображения, или изменение пористости среды.

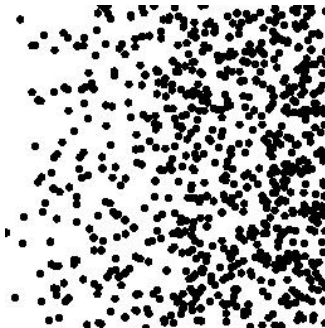


Рис.: Пример текстуры с трендом интенсивности частиц

## Математическая постановка

С математической точки зрения, задача сводится к синтезу случайного изображения  $X'$  (и построению соответствующей процедуры синтеза), принадлежащему распределению, близкому к желаемому:

$$P_{X'} \approx P_X,$$

где  $P_X$  - распределение изображений с трендами, удовлетворяющих следующим ограничениям (для упрощения задачи):

- Это монохромные изображения 256 x 256 пикселей
- Изменяющимся свойством является интенсивность появления частиц  $\lambda$
- Тренд является линейным и направлен вдоль оси  $x$ :  $\lambda = \lambda_0 + kx$

Распределение  $P_X$  задается обучающей выборкой.

# Существующие подходы к решению задачи

Есть несколько подходов к решению задач подобного рода:

- 'Классический' статистический подход
- Базовый нейросетевой подход
- Генеративные состязательные сети (GAN)

## 'Классический' статистический подход

- Вводится параметризованное семейство распределений вероятности  $P_{\theta}(x)$
- Параметры  $\theta$  находятся из обучающей выборки:

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

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

- Генерируется семпл из  $P_{\theta^*}$

Этот подход приводит к проблемам:

- Пространство параметров  $\theta$  может быть огромной размерности
- Известной параметрической модели распределения может вообще не существовать

Простой пример - синтез человеческих лиц: с помощью классического подхода эта задача не была решена с хорошим качеством.

## Базовый нейросетевой подход

- Вводится параметризованное семейство распределений вероятности  $P_{\theta}(x)$ 
  - ▶ Вводятся скрытые переменные  $V$  и функция(нейросеть) для получения  $x$  из  $V$  (фактически, классификация, развернутая в другую сторону)
- Определяются параметры распределения (т.е. обучение нейросети)
- Генерируются семплы из  $P_{\theta^*}$

Этот подход возможен, однако на практике трудноосуществим.

Изначальная задача: найти процедуру генерирования  $X'$  так, чтобы  $P_{X'} \approx P_X$ .  
 Переформулируем:

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

- Введем некоторые скрытые переменные с фиксированным распределением, например

$$V \sim U^n[-1, 1]$$

- и параметризованную процедуру генерации:

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

Переформулируем:

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

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

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

Возникает вопрос: что использовать в качестве метрики похожести двух распределений  $\rho$ , где одно из распределений задано обучающей выборкой.

- В качестве такой метрики можно использовать loss-функцию обученного классификатора:

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

где  $\mathcal{L}$  - функция потерь обученного классификатора.



- Введем две нейросети:
  - $d_{\zeta}(x)$  - классификатор для измерения расстояния, **дискриминатор**
  - $g_{\theta}(x)$  - сеть, трансформирующая  $V$  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 отличить сгенерированные изображения от реальных из обучающей выборки.

- Введем loss-функцию дискриминатора(например, logloss):

$$\begin{aligned}
 L(X, X') &= \frac{1}{2} \mathbb{E}_{x \sim X} l(d_{\zeta}(x), 1) + \frac{1}{2} \mathbb{E}_{x' \sim X'} l(d_{\zeta}(x'), 0) = \\
 &= -\frac{1}{2} (\mathbb{E}_{x \sim X} \log d_{\zeta}(x) + \mathbb{E}_{x' \sim X'} \log(1 - d_{\zeta}(x'))) = \\
 &= -\frac{1}{2} (\mathbb{E}_{x \sim X} \log d_{\zeta}(x) + \mathbb{E}_{v \sim V} \log(1 - d_{\zeta}(g_{\theta}(v)))) = \\
 &= L(\zeta, \theta)
 \end{aligned}$$

- Loss-функция обученного классификатора:

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

- Соответственно,

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

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

- Определим оптимальный дискриминатор:

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

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

# Обучение GAN

В итоге, процесс обучения принимает следующий вид:

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

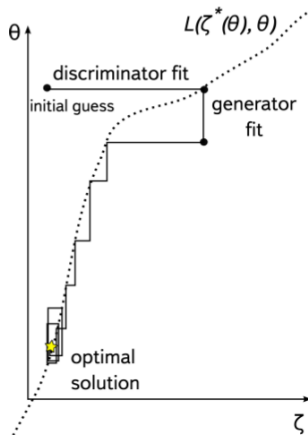


Рис.: Схематическое изображение процесса обучения GAN

## pix2pix GAN

Для решения задачи было попробовано применить модификацию GAN-сети под названием "pix2pix GAN". Общий функционал потерь для pix2pix GAN выглядит следующим образом:

$$\mathcal{L}(G, D) = \mathcal{L}_{adv}(G, D) + \eta \mathbb{E}_{s_1, s_2, r \sim p_{data}(s_1, s_2, r)} (\|r - G(s_1, s_2)\|_1)$$

$$\begin{aligned} \mathcal{L}_{adv}(G, D) = & \mathbb{E}_{s_1, s_2, r \sim p_{data}(s_1, s_2, r)} \log D(s_1, s_2, r) + \\ & + \mathbb{E}_{s_1, s_2 \sim p_{data}(s_1, s_2)} \log(1 - D(s_1, s_2, G(s_1, s_2))) \end{aligned}$$

где  $G, D$  - генератор и дискриминатор,  $(s_1, s_2, r)$  - тройка изображений (интенсивность слева, справа и реальное изображение с трендом),  $\mathbb{E}_{s_1, s_2, r \sim p_{data}(s_1, s_2, r)}$  - 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  $s_1, s_2$ .

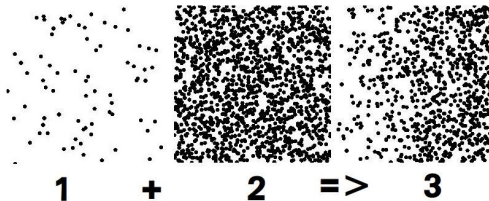


Рис.: Вход и желаемый выход нейросети

## Критерий качества

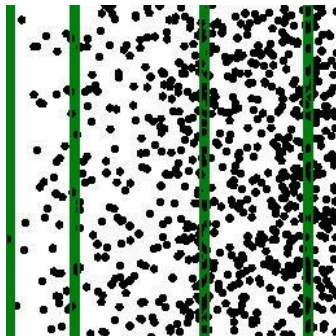
После обучения генератора, необходимо проверить, что сгенерированные им изображения действительно имеют искомые характеристики. Для этого нужно ввести специальную метрику, которая будет учитывать наличие в изображении тренда интенсивности частиц. Было решено использовать среднюю плотность черных пикселей в некотором окне, и проходить этим окном по изображению:

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

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

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

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



где  $\xi_{0k}$  -  $\xi_k$ , усредненное по примерам из обучающей выборки.

## Результаты

Было проведено обучение нейросети описанной архитектуры при различных гиперпараметрах. Обучающей выборкой был массив из 3500 троек изображений.

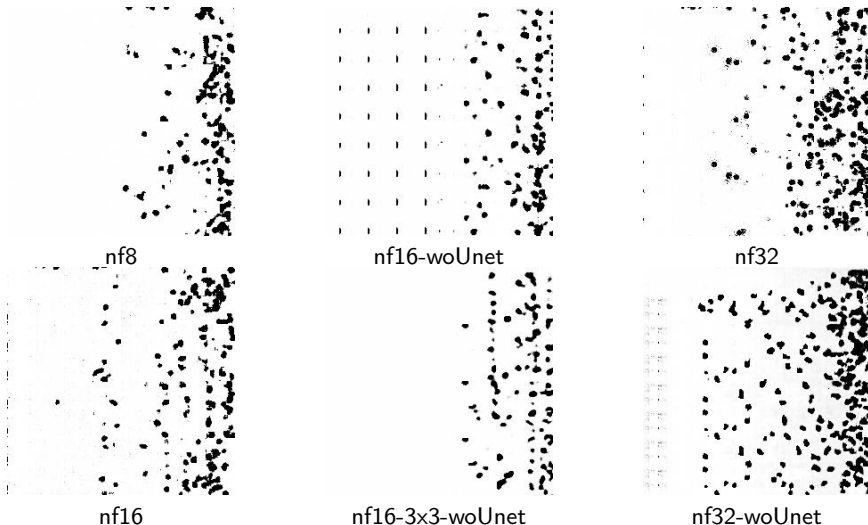
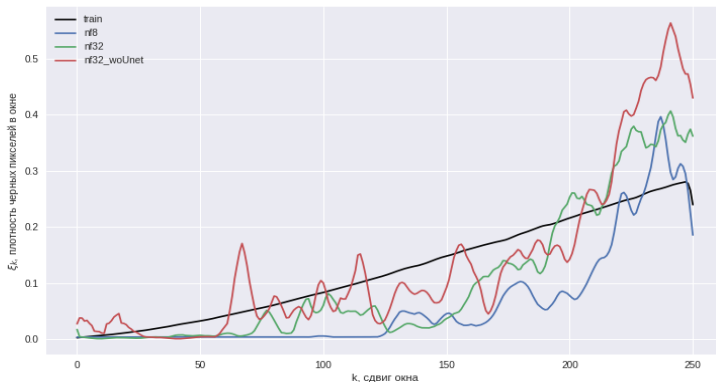


Рис.: Примеры сгенерированных текстур

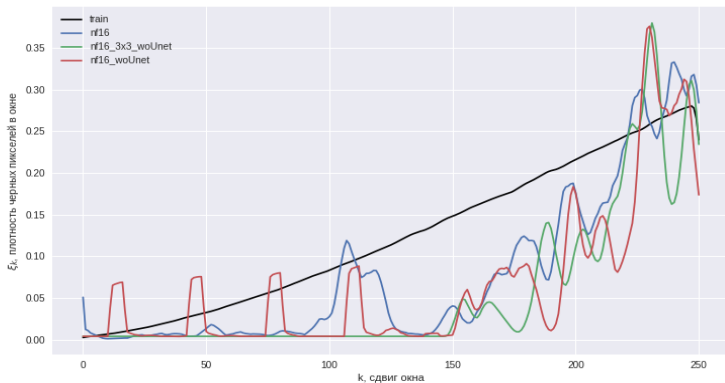
## Результаты

Для сгенерированных наборов текстур получились следующие результаты:



Сеть	Метрика
nf8	0.00825
nf32	0.00549
nf32-woUnet	0.00688

# Результаты



Сеть	Метрика
nf16	0.00606
nf16-woUnet	0.00881
nf16-3x3-woUnet	0.01034



## Заключение

Полученные результаты показывают, что в принципе нейросеть способна уловить тренд и воспроизвести его, однако на данный момент качество генерации относительно невысоко. Необходимо провести дальнейшее исследование оптимальных гиперпараметров, и, возможно, увеличить объем обучающей выборки. Также можно провести аналогичные эксперименты с другими архитектурами генераторов.