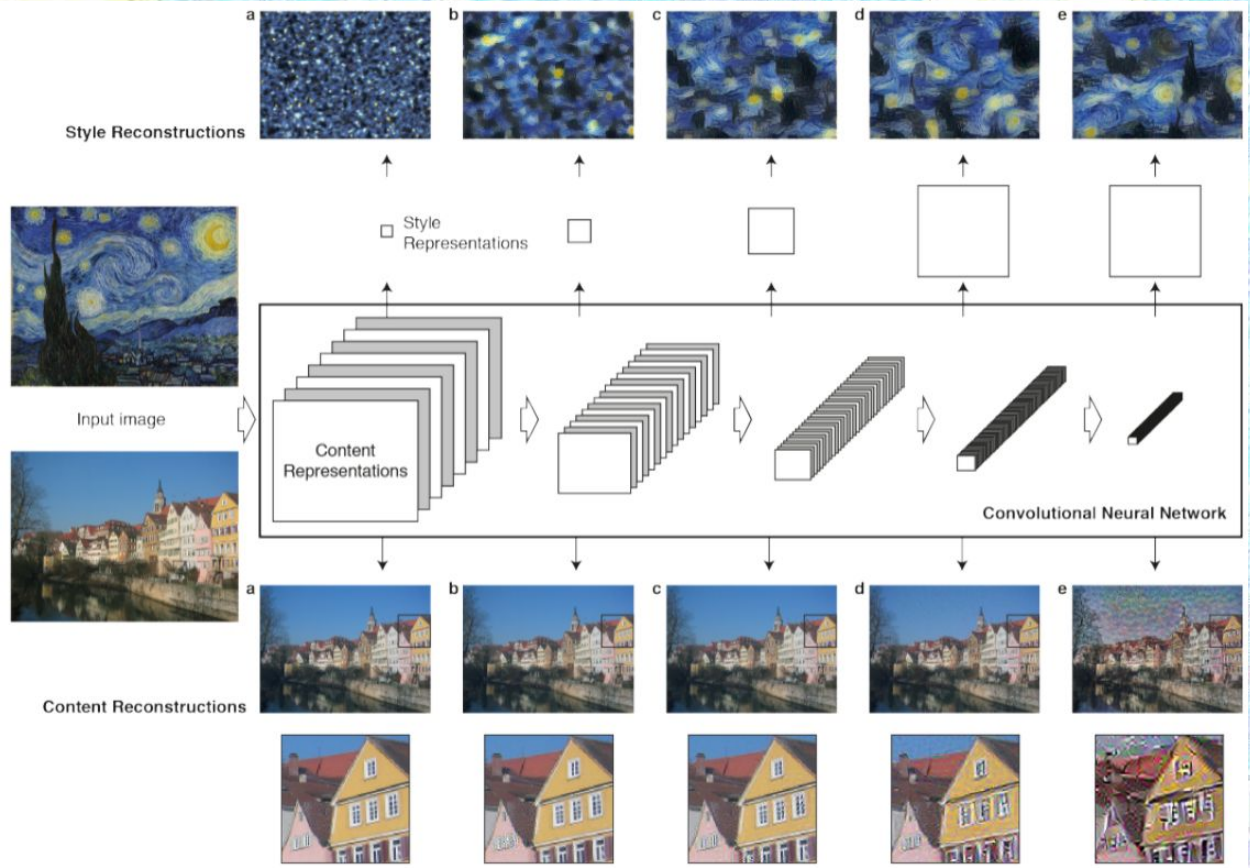

ARTIFICIAL ART

A Neural Algorithm of Artistic Style

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge
2015

- Первая работа, где продемонстрирован качественный перенос стиля.
- Показано, что сверточная сеть может разделить стиль и содержание.
- Хорошее качество, но алгоритм медленный и требует много памяти.

CNN может разделить стиль и содержание



CNN может разделить стиль и содержание

- CNN на основе VGG-Network, 16 сверточных слоев и 5 слоев average pooling.
- **Content Reconstructions**: восстановление картинки на основе ответов текущего слоя. Менее глубокие слои “знают” изображение почти идеально, более глубокие теряют детальную информацию, но сохраняют знание об объектах и их расположении.
- **Style Reconstructions**: на основе ответов отдельных слоев CNN было построено новое признаковое пространство. Тогда, комбинируя ответы разных слоев, можно получить информацию о стиле, но не о содержании исходного изображения.

Content Reconstructions

Градиентный спуск от белого шума до тех пор, пока его ответы не становятся похожи на ответы исходного изображения на данном сверточном слое. Для вычисления градиента используется обратное распространение ошибки.

p - исходное изображение, x - сгенерированное, l - текущий слой, P^l и F^l - соответствующие ответы слоя, $F^l \in \mathcal{R}^{N_l \times M_l}$

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^l - P_{ij}^l\right)^2$$

$$\frac{\partial \mathcal{L}_{content}}{\partial F_{ij}^l} = \begin{cases} \left(F^l - P^l\right)_{ij} & \text{if } F_{ij}^l > 0 \\ 0 & \text{if } F_{ij}^l < 0 \end{cases}$$

Style Reconstructions

Тоже градиентный спуск от белого шума, но оптимизируя не по ответам слоя, а по матрице Грама, построенной на них: $G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$

Для вычисления градиента используется обратное распространение ошибки.

a - исходное изображение, x - сгенерированное, l - текущий слой, A^l и G^l - соответствующие матрицы Грама.

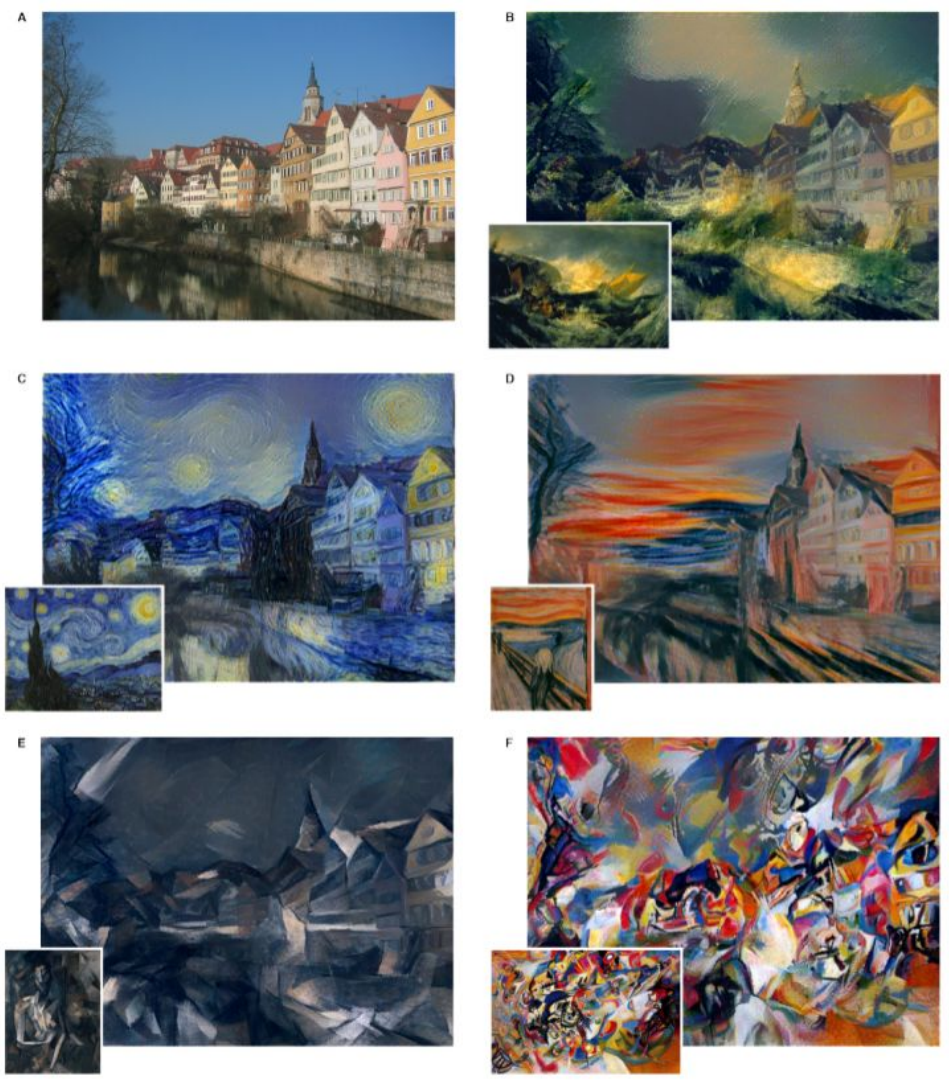
$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - A^l_{ij}\right)^2$$

$$\frac{\partial E_l}{\partial F^l_{ij}} = \begin{cases} \frac{1}{N_l^2 M_l^2} \left((F^l)^{\mathrm{T}} \left(G^l - A^l\right)\right)_{ji} & \text{if } F^l_{ij} > 0 \\ 0 & \text{if } F^l_{ij} < 0 \end{cases}$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

Content & Style

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$



Content & Style: ratio α/β

Два подхода в генерировании изображений

- **descriptive method** (описательный метод) - сеть используется для получения статистики по изображению. Таким образом задача сводится к случайному выбору из изображений, удовлетворяющих этой статистике. Использовался в прошлой статье.
- **generative approach** (генеративный подход) - сеть принимает на вход случайное состояние и прямым проходом вычислений получает на выходе изображение.

Texture Networks: Feed-forward Synthesis of Textures and Stylized Images

Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, Victor Lempitsky

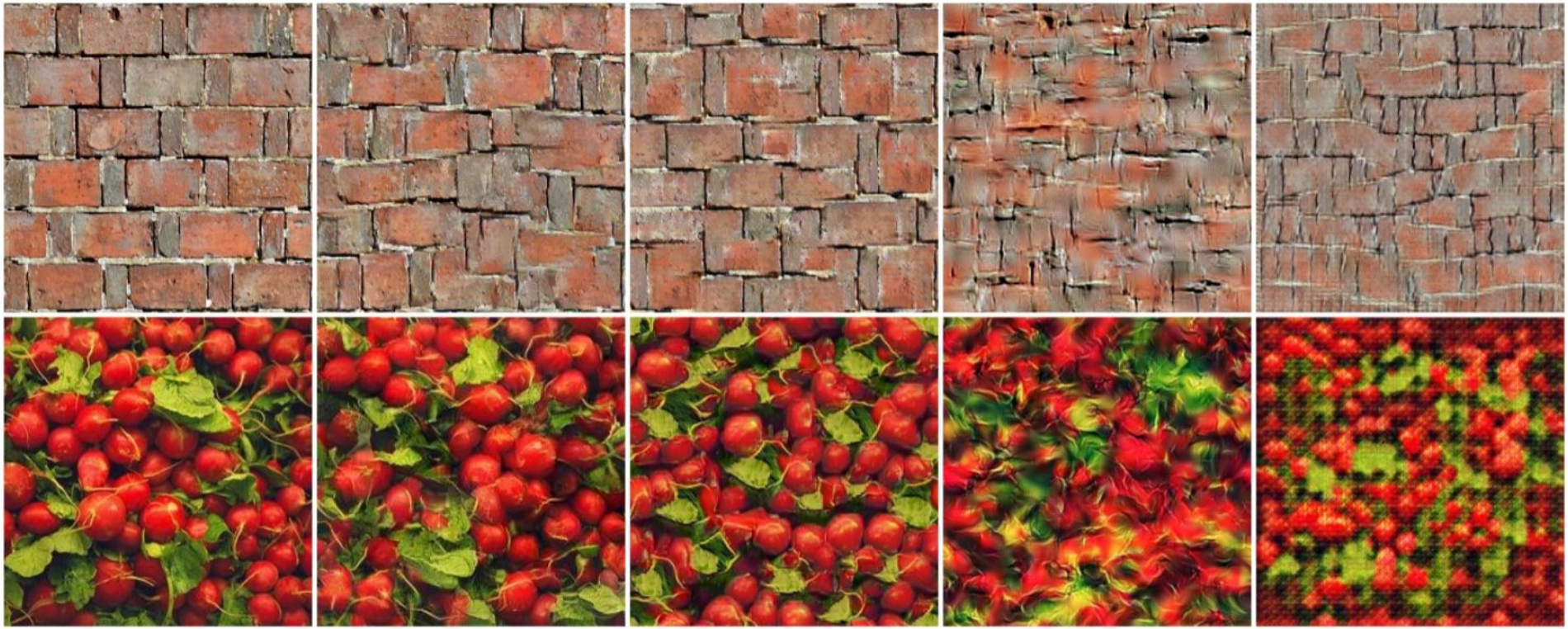
2016

Вклад статьи:

- Первая работа, показывающая, что генеративный подход может добиться сравнимого качества с описательным методом.
- Предложили генеративную модель, на два порядка быстрее и на порядок более эффективную по памяти, чем описательный метод.
- Предложили новую архитектуру, хорошо подходящую для такого рода задач.

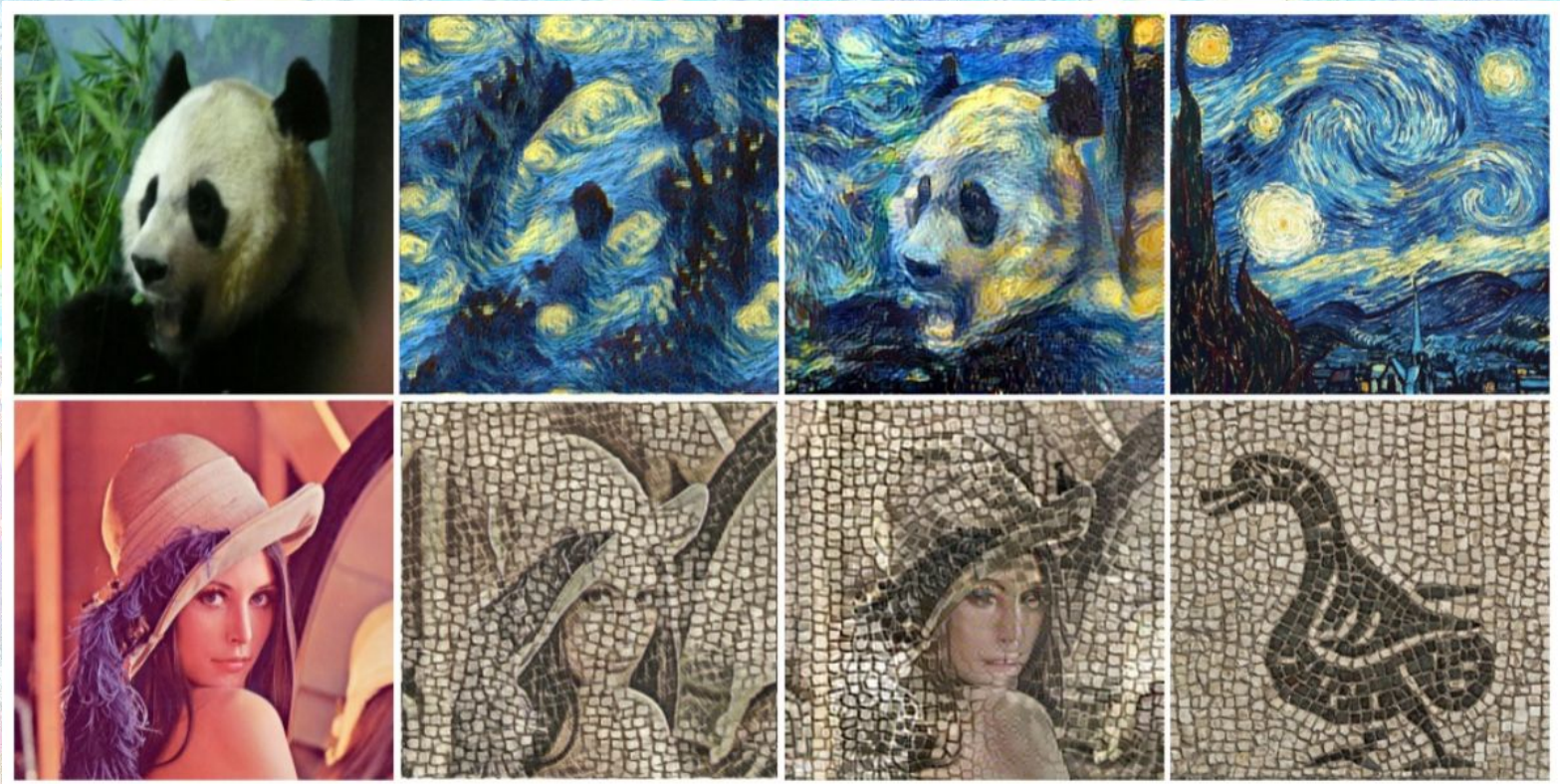
Сравнение работы методов: текстура

Сравнение работы методов: текстура

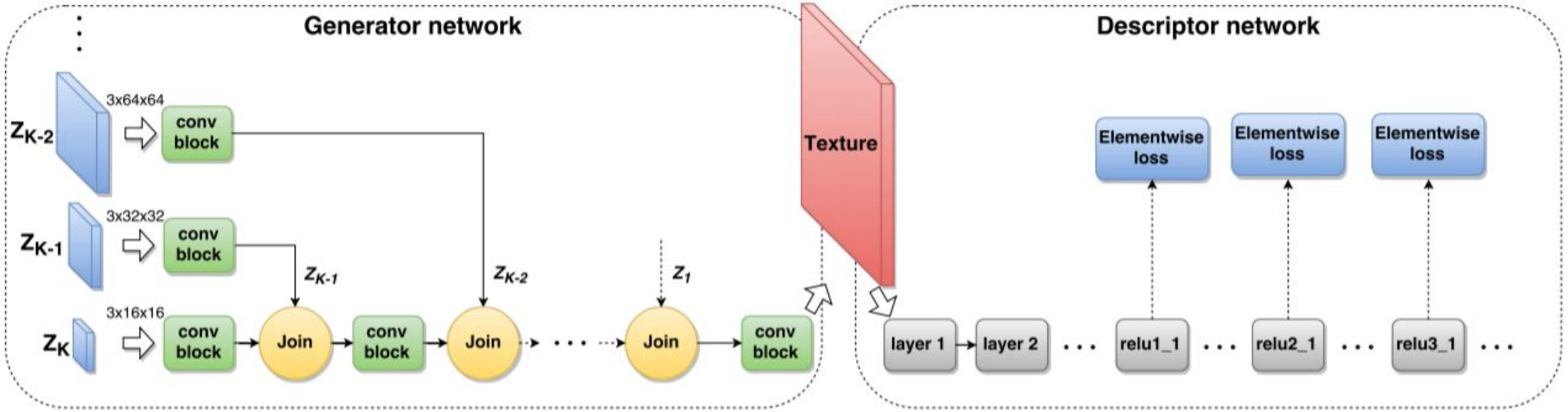


Input | Gatys et al. | Texture nets (ours) | Portilla, Simoncelli | DCGAN

Сравнение работы методов: перенос стиля



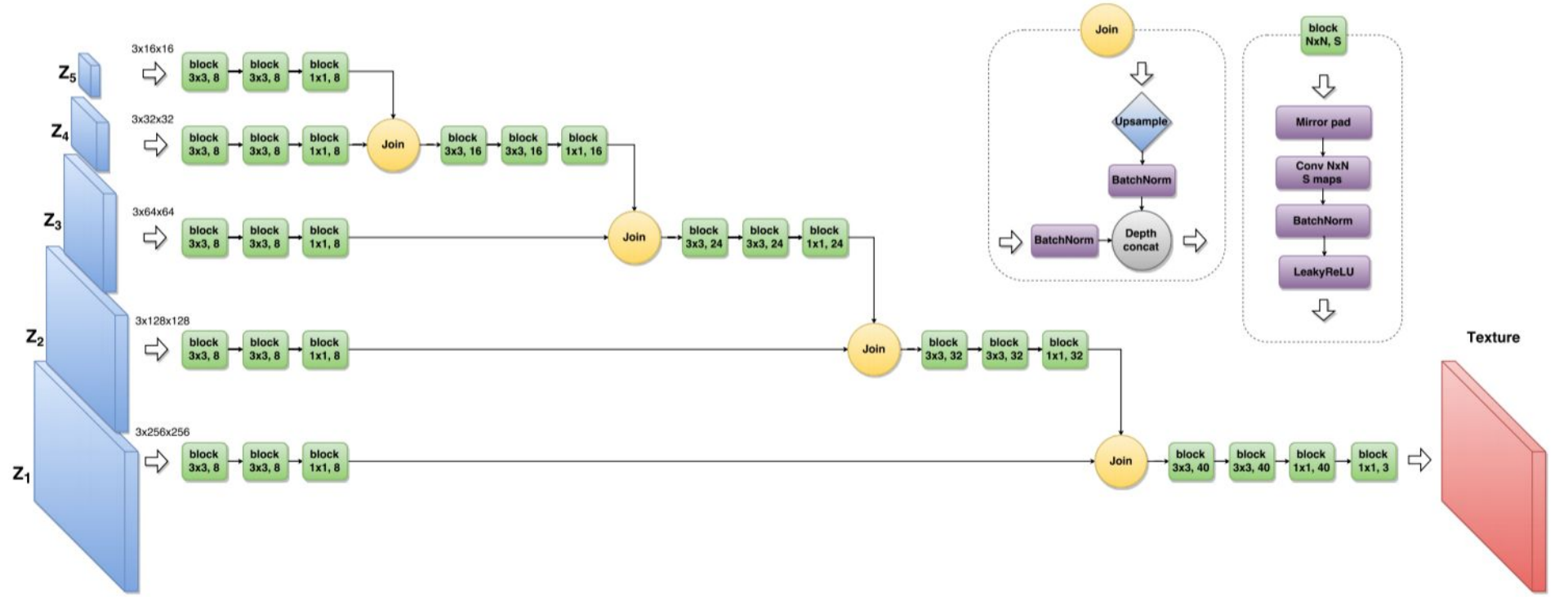
Архитектура: обзор



Архитектура: детали

- Референсная текстура - тензор $\mathbb{R}^{M \times M \times 3}$
- Случайный шум из равномерного распределения $\mathbf{z}_i \in \mathbb{R}^{\frac{M}{2^i} \times \frac{M}{2^i}}, i = 1, 2, \ldots, K$
- M = 256, K = 5 для текстуры и K = 6 для переноса стиля
- Сверточный блок состоит из трех слоев, с фильтрами 3x3, 3x3 и 1x1
- Функция активизации: ReLU
- Batch normalization после каждого сверточного слоя и до concatenation layers
- В случае переноса стиля, появляется дополнительный вход y - изображение, к которому надо применить стиль

Архитектура: конкретная модель



Loss functions

Texture loss:

$$\mathcal{L}_T(\mathbf{x}; \mathbf{x_0}) = \sum_{l \in L_T} \|G^l(\mathbf{x}) - G^l(\mathbf{x_0})\|_2^2$$

$$G^l_{ij}(\mathbf{x}) = \langle F^l_i(\mathbf{x}), F^l_j(\mathbf{x}) \rangle$$

Content loss:

$$\mathcal{L}_C(\mathbf{x}; \mathbf{y}) = \sum_{l \in L_C} \sum_{i=1}^{N_l} \|F^l_i(\mathbf{x}) - F^l_i(\mathbf{y})\|_2^2$$

Для генерации текстуры используется только texture loss, для переноса стиля - взвешенная сумма texture loss по подмножеству слоев и content loss

Обучение

Оптимизируется с помощью стохастического градиентного спуска.

- Вычисление generator network: $\mathbf{x}_k = \mathbf{g}(\mathbf{z}_k, \theta)$.
- Вычисление descriptor network: $G^l(\mathbf{x}_k), l \in L_T$.
- Считаем loss
- Обратное распространение ошибки для подсчета градиента
- Обновляем параметры

Параметры для texture synthesis:

$$\theta_{\mathbf{x_0}} = \underset{\theta}{\operatorname{argmin}} \; E_{\mathbf{z} \sim \mathcal{Z}} \left[\mathcal{L}_T \left(\mathbf{g}(\mathbf{z}; \theta), \mathbf{x_0} \right) \right]$$

Параметры для style transfer:

$$\theta_{\mathbf{x_0}} = \underset{\theta}{\operatorname{argmin}} \; E_{\mathbf{z} \sim \mathcal{Z};\, \mathbf{y} \sim \mathcal{Y}} \left[\mathcal{L}_T \left(\mathbf{g}(\mathbf{y}, \mathbf{z}; \theta), \mathbf{x_0} \right) + \alpha \, \mathcal{L}_C \left(\mathbf{g}(\mathbf{y}, \mathbf{z}; \theta), \mathbf{y} \right) \right]$$

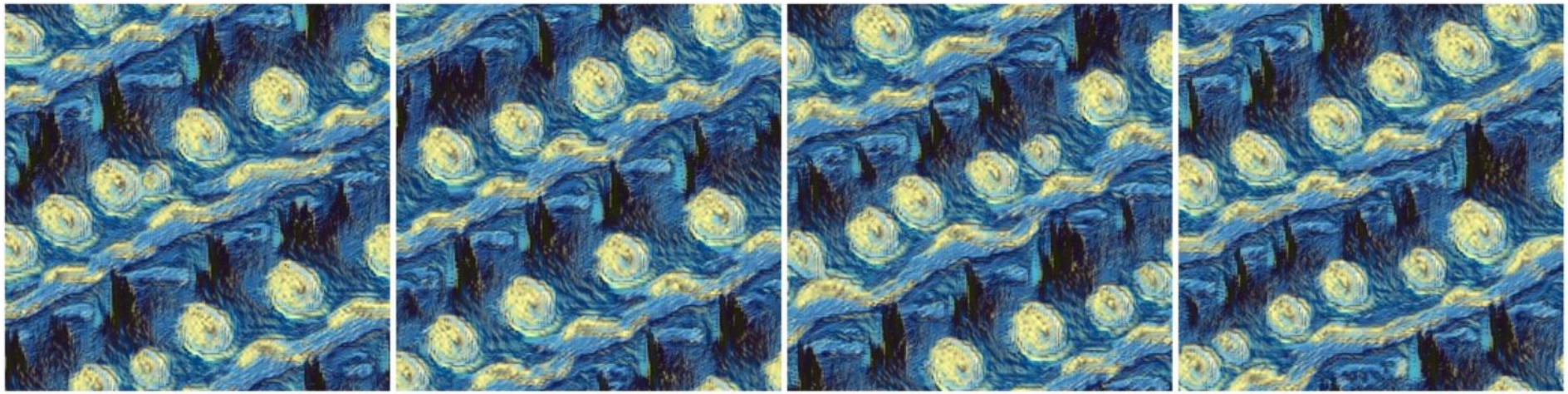
Технические детали

- Веса generator network инициализируются с помощью Xavier’s method
- Training: Torch7’s implementation of Adam, 2000 iteration
- Learning rate: изначально 0.1, начиная с итерации 1000 каждые 200 итераций уменьшается в 0.7
- Batch size: 16
- VGG-19, для texture loss использовались слои {relu1_1, relu2_1, relu3_1, relu4_1, relu5_1}, для content loss: {relu4_2}
- Полное обучение занимает 2 часа на NVIDIA Tesla K40
- ~20ms на оценку loss (Gatys et al.: 10s)
- 170MB на генерацию 256x256 изображения (Gatys et al.: 1100MB)

Масштабирование шума

$k = 0.01$ $k = 0.1$ $k = 1$ $k = 10$

Переобучение



Ссылки

- https://arxiv.org/abs/1508.06576 A Neural Algorithm of Artistic Style
- https://arxiv.org/abs/1603.03417 Texture Networks: Feed-forward Synthesis of Textures and Stylized Images