Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра математических методов прогнозирования

Николаев Сергей Владимирович

Совершенствование метода автоматического переноса стиля изображений

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:

к.ф.-м.н.

В. В. Китов

Москва, 2018

Содержание

1 Введение

Рисование является одним из видов искусства. Перерисовывание изображения в определенном стиле требует мастерства художника и много времени. В последнее время широкое распространение в различных задачах получили сверточные нейронные сети. Одним из их приложений является нейросетевой автоматический перенос стиля изображений. Основной идеей данного подхода является то, что признаки, полученные с помощью сверточной нейронной сети, можно использовать для выделения содержательной и стилевой составляющих изображения. Алгоритм позволяет создавать новые изображения, совмещающие содержание фотографии со стилем работ известных художников. В ходе развития нейросетевого переноса стиля возникли различные задачи, решение которых позволило бы значительно улучшить качество алгоритма. Первой рассматриваемой задачей является контроль масштаба переносимого стиля. В литературе эту задачу часто называют задачей контроля размера художественной кисти. Вторая задача состоит в моделировании одной нейронной сетью нескольких стилей. Для каждой из этих задач по отдельности ранее были предложены подходы к решению. В данной работе предлагается совместить возможности генеративной сети, осуществляющей стилизацию, чтобы она могла как стилизовать под разные, в том числе неизвестные на этапе обучения, стили, так и контролировать масштаб накладываемого стиля в непрерывной шкале. К преимуществам предлагаемого метода перед существующими способами контроля масштаба переносимого стиля можно отнести меньшее требование по количеству используемой памяти, а также более высокую скорость обучения.

В разделе 2 показывается, как процесс перенос стиля изображения можно свести к задаче оптимизации. В 3.4 описаны существующие решения для обучения многостилевых нейронных сетей. Раздел 3.5 посвящен методам контроля размера художественной кисти. Описание предлагаемого способа контроля размера кисти в многостилевой сети содержится в разделе 4. В разделе 5 описываются проведенные эксперименты, а также проводится сравнение предлагаемого метода с существующими решениями.

2 Постановка задачи

В задаче переноса стиля на вход алгоритма подаются два изображения:

- Изображение $x_{content}$, в котором нас интересует содержание (content image). Обычно им является фотография;
- Изображение x_{style}, в котором нас интересует художественный стиль (style image). Под стилем обычно понимают цветовое наполнение, текстуры и т.д. Как правило, в качестве таких изображений выступают работы известных художников.

Для получения количественной оценки близости двух изображений по содержанию и стилю используют функции потерь $\mathcal{L}^{content}(x, y)$ и $\mathcal{L}^{style}(x, y)$ соответственно. Определение этих функций потерь содержится в секции 3.2. Тогда задачу переноса стиля изображения можно свести к нахождению изображения, являющегося решением следующей задачи оптимизации:

$$\mathcal{L}^{content}(x_{content}, y) + \alpha\mathcal{L}^{style}(x_{style}, y) \to \min_{y}, \tag{1}$$

где $\alpha \in \mathbb{R}^+$ — параметр, определяющий относительную важность переноса стиля x_{style} перед сохранением содержания $x_{content}$. Чем меньше α, тем больший вес имеет сохранение содержания изображения $x_{content}$.

Одной из подзадач переноса стиля является контроль размера художественной кисти. В этом случае вводят параметр $k \in \mathbb{K}$, где $\mathbb{K}$ — множество моделируемых размеров кисти. От этого параметра зависит размер кисти на стилизованном изображении y. Учесть это можно с помощью добавления соответствующего параметра в функцию потерь, что приводит к следующей задаче оптимизации:

$$\mathcal{L}^{content}(x_{content}, y, k) + \alpha\mathcal{L}^{style}(x_{style}, y, k) \to \min_{y} \tag{2}$$

3 Обзор литературы

3.1 Сверточные нейронные сети

В задачах обработки и классификации изображений большую популярность имеют сверточные нейронные сети (convolutional neural networks, CNN). К причинам такой популярности можно отнести их универсальность вкупе с высоким качеством работы. Основная идея архитектуры CNN — последовательное применение к изображению обучаемых фильтров. Пусть мы имеем тензор $X \in \mathbb{R}^{C_{in} \times H \times W}$, обычно называемый картой признаков. Например, цифровое изображение с цветовой моделью RGB. Тогда результатом применения фильтра $w \in \mathbb{R}^{C_{in} \times h \times w}$ к тензору X будет матрица Y, элементы которой вычисляются следующим образом:

$$Y(i,j) = \sum_{c=1}^{C_{in}} \sum_{m=1}^{h} \sum_{k=1}^{w} X(c,\ i - \frac{h}{2} + m,\ j - \frac{w}{2} + k)\ w(c,m,k) \tag{3}$$

Как правило, в одном слое сверточной нейронной сети используют несколько фильтров, так что выходом слоя с C_{out} фильтрами будет тензор $Y \in \mathbb{R}^{C_{out} \times H_{out} \times W_{out}}$. Это говорит о том, что для сверточной нейронной сети имеется возможность получить изображение в качестве выхода. Для уменьшения размерности признаков можно использовать субдискретизирующие слои (subsample layers). Примером такого слоя является max pooling слой, в котором прямоугольная область входной карты признаков заменяется на максимальное значение признака в этой области.

Типичная архитектура сверточной нейронной сети представляет собой последовательно соединенные блоки, состоящие из сверточного и субдискретизирующего слоев. Между блоками обычно добавляют нелинейное преобразование. Популярными функциями активации являются:

- $ReLU(x) = max(x, 0)$;
- $\sigma(x) = \frac{1}{1+\exp(-x)}$.

Последовательное применение таких блоков позволяет увеличить область восприятия (receptive field) признаков. Область восприятия признака можно определить как множество пикселей во входном изображении, оказывающих влияние на значение этого признака. На Рис. 1 проиллюстрировано данное понятие, а также показано,

каким образом возрастает область восприятия. На Рис. 2 приведен пример архитектуры сверточной нейронной сети.

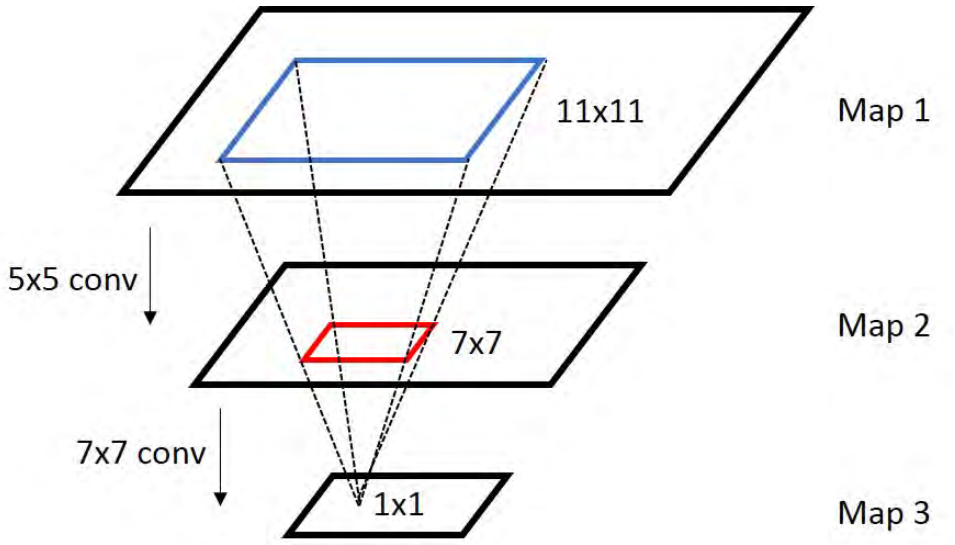


Рис. 1: Пример возрастания области восприятия в сверточных нейронных сетях. Значение признака из третьей карты признаков содержит в себе информацию об области 11×11 исходного изображения.

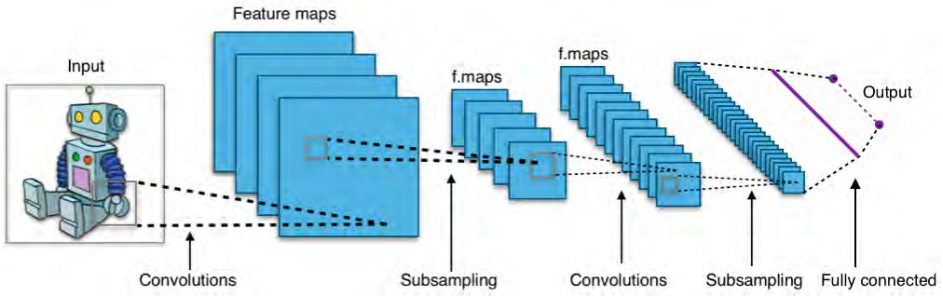


Рис. 2: Типичная архитектура сверточной нейронной сети

3.2 Функции потерь в переносе стиля

Важную роль в переносе стиля играет выбор функций потерь $\mathcal{L}^{content}(x, y)$ и $\mathcal{L}^{style}(x, y)$, так как именно от формализации „близости“ двух изображений по содержанию и стилю зависят свойства, которыми будет обладать результат стилизации.

В качестве кандидата для оценки близости по содержанию можно рассматривать квадрат L_2 нормы от разности итогового и исходного изображений $||y - x_{content}||^2$. Однако такая функция потерь не всегда соотносится с тем, как человек воспринимает

сходство по содержанию. Например, пусть одно изображение получено из другого с помощью сдвига пикселей на одну позицию. Человек в этом случае не заметит разницы, в то время как функция потерь будет принимать далекие от нуля значения.

Альтернативой является получение признакового описания изображения в пространстве, в котором евклидова норма лучше описывает близость объектов по содержанию. Распространенным способом получения высокоуровневых признаков для изображений является использование предобученных сверточных нейронных сетей. Данный способ хорошо зарекомендовал себя на практике. Использование промежуточных значений предобученной на задаче классификации сверточной нейронной сети в качестве признаков для другого алгоритма довольно успешно применялось до этого в задачах классификации и регрессии ([1], [2]). В работе [3] для переноса стиля было предложено использовать признаки, полученные с помощью сверточной нейронной сети архитектуры VGG [4]. В данной работе было показано, что выходы сверточных слоев VGG содержат в себе высокоуровневую информацию о содержимом изображения. С увеличением количества слоев признаки становятся более чувствительными к истинному содержимому изображения, при этом становясь инвариантными к тому, как оно выглядит. Таким образом, более глубокие слои сети содержат информацию о содержимом в терминах объектов и их расположений, при этом теряя информацию о значениях конкретных пикселей. На основе этих выводов в [3] предлагают задать функции потерь на основе векторизованного выхода каждого слоя VGG.

Пусть $F_l(x) \in \mathbb{R}^{H_l W_l \times C_l}$ — выход l-го слоя VGG, где C_l — количество каналов на l-ом слое, H_l и W_l — высота и ширина карты признаков на l-ом слое соответственно. Тогда функция потерь, отвечающая за сохранение содержания $x_{content}$, определяется следующим образом:

$$\mathcal{L}_l^{content}(x_{content}, y) = \frac{1}{C_l H_l W_l} \left|\left|F_l(x_{content}) - F_l(y)\right|\right|_F^2, \tag{4}$$

где $||\cdot||_F$ — норма Фробениуса.

Для нахождения близости по стилю в [3] предлагают использовать матрицу Грама, вычисляемую по следующей формуле:

$$G_l(x) = \frac{1}{H_l W_l} F_l(x)^T F_l(x) \in \mathbb{R}^{C_l \times C_l} \tag{5}$$

Если интерпретировать $F_l(x)$ как матрицу, состоящую из объектов размерности C_l, то матрица Грама будет равна нецентрированной матрице ковариаций. Таким образом, матрица Грама содержит информацию о том, какие каналы карты признаков зависят друг от друга. Тогда функция потерь, отвечающая за перенос стиля x_{style}, определяется следующим образом:

$$\mathcal{L}_l^{style}(x_{style}, y) = \frac{1}{C_l^2} ||G_l(x_{style}) - G_l(y)||_F^2 \tag{6}$$

Такая функция потерь хорошо показывает себя на практике, однако теоретическая обоснованность её применения была неочевидной. В работе [5] доказывается, что минимизация такой функции потерь эквивалентна минимизации максимального среднего расхождения (maximum mean discrepancy) с полиномиальным ядром второго порядка. Из этого следует, что в задаче переноса стиля с функцией потерь (6) решается задача выравнивания распределений признаков между генерируемым и стилевым изображениями.

Итоговая функция потерь является взвешенной суммой функций потерь для каждого слоя:

$$\mathcal{L}^{total}(y, x_{content}, x_{style}) = \sum_l \beta_l^{content} \mathcal{L}_l^{content}(x_{content}, y) + \alpha \sum_l \beta_l^{style} \mathcal{L}_l^{style}(x_{style}, y) \tag{7}$$

Обычно для подсчета $\mathcal{L}^{content}(x_{content}, y)$ используют один слой сети VGG, а для $\mathcal{L}_l^{style}(x_{style}, y)$ — разные слои с подобранными весами.

При решении задачи оптимизации (1) с функцией потерь (7) на стилизованном изображении могут возникать искажения, связанные с несоответствием соседних пикселей друг другу. Например, переход цвета может быть слишком резким. Для борьбы с такой проблемой в работе [6] предлагают регуляризовать общую вариацию (total variation regularization). Для этого в функцию потерь добавляют следующее слагаемое:

$$\mathcal{L}^{TV}(y) = \gamma \sum_{i=1}^{H-1} \sum_{j=1}^{W-1} \left(||y_{i+1,j} - y_{i,j}||^2 + ||y_{i,j+1} - y_{i,j}||^2 \right), \tag{8}$$

где γ — параметр, отвечающий за силу регуляризации, W и H — ширина и высота изображения соответственно.

3.3 Использование генеративной сети для переноса стиля

В работе [3], описанной в секции 3.2, оптимизация ведется по пикселям итогового изображения. Это означает, что решать задачу оптимизации нужно столько раз, сколько пар изображений у нас есть. При этом задача оптимизации решается методом обратного распространения ошибки [7] с проходом вперед и назад по довольно глубокой сети VGG, поэтому время работы алгоритма является существенным недостатком данного подхода. При этом были основания полагать, что процессы переноса одного стиля на разные изображения имеют схожую функциональную зависимость $f_{style}(x_{content})$ В работах [6] и [8] предлагается параметризовать стилизацию изображения с помощью генеративной сверточной нейронной сети, обучаемой для конкретного изображения стиля. Общая схема обучения генеративной сети в работе [6] изображена на Рис. 3. В этой схеме используются две сверточные нейронные сети. Обучаемой является генеративная сеть (Image Transform Net), принимающая на вход изображение $x_{content}$ и возвращающая стилизованное изображение $f_W(x_{content})$. Предобученная сеть VGG используется только для подсчета функции потерь и остается постоянной на всем протяжении обучения. Для обучения в данном подходе необходима выборка изображений $\{x_{content,i}\}_{i=1}^N$. Обучение генеративной сети сводится к следующей задаче оптимизации:

$$\sum_{i=1}^{N} \big(\mathcal{L}^{content}(x_{content,i}, f_W(x_{content,i})) + \alpha \mathcal{L}^{style}(x_{style}, f_W(x_{content,i})) + \\ + \gamma \mathcal{L}^{TV}(f_W(x_{content,i}))\big) \to \min_W \tag{9}$$

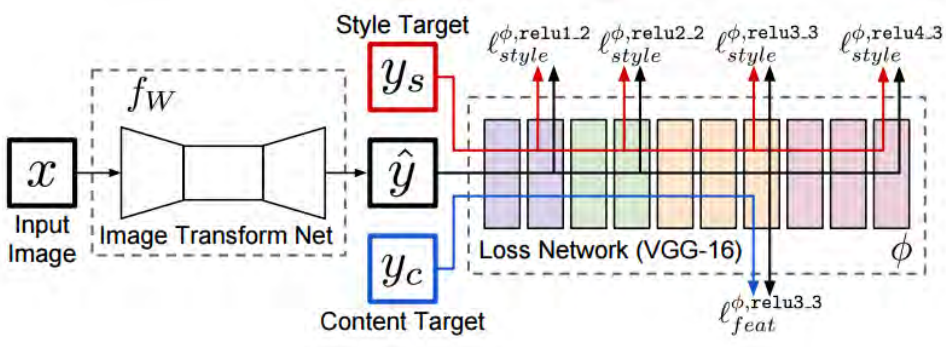


Рис. 3: Схема работы метода [6] с использованием генеративной нейронной сети

Впоследствии стилизация любого изображения осуществляется простым применением генеративной сети к нему. Это позволяет существенно уменьшить время переноса стиля, сохранив качество оптимизационного подхода (1). Однако в данном методе теряется гибкость, связанная с возможностью переносить стиль любого изображения, так как требуется наличие предобученной сети для каждого изображения стиля.

3.4 Многостилевые генеративные сети

Одним из недостатков использования генеративной сети для переноса стиля изображений является необходимость обучения отдельной сети для каждого стиля. Это влечет за собой необходимость хранения большого количества сетей, что может быть невозможным при малом объеме памяти, например, на мобильных устройствах. Кроме того, это ограничивает выбор пользователя заданным набором стилевых изображений. После подтверждения гипотезы о схожей функциональной зависимости процессов переноса одного стиля на разные изображения возник вопрос о возможности описания переноса разных стилей единой функциональной зависимостью от входных изображений. Возникает необходимость нахождения архитектуры нейронной сети, позволяющей использовать одну генеративную сеть для переноса различных стилей.

Instance Normalization В работе [9] был проведен анализ влияния батч-нормализации [10] (batch normalization), метода регуляризации нейронных сетей, на обучение генеративной сети для переноса стиля изображений. Батч-нормализация состоит в нормализации промежуточных активаций сети с использованием матожидания и дисперсии признаков по батчу. Пусть $x \in \mathbb{R}^{N \times C \times H \times W}$ — карта признаков для набора из N изображений. Тогда нормализация происходит следующим образом:

$$
\begin{aligned}
y_{nkij} &= \frac{x_{nkij} - \mu_k}{\sqrt{\sigma_k^2 + \epsilon}} \\
\mu_k &= \frac{1}{NHW} \sum_{n=1}^{N} \sum_{i=1}^{H} \sum_{j=1}^{W} x_{nkij} \\
\sigma_k^2 &= \frac{1}{NHW} \sum_{n=1}^{N} \sum_{i=1}^{H} \sum_{j=1}^{W} (x_{nkij} - \mu_k)^2
\end{aligned}
\tag{10}
$$

Авторы [9] пришли к выводу, что батч-нормализация оказывает негативное влияние как на скорость обучения, так и на результат работы генеративной сети. Они обосновывают это тем, что среднее и дисперсия для каждого канала карты признаков содержат информацию о стиле изображения, а в батч-нормализации эти статистики вычисляются по нескольким изображениям, из-за чего информация о стиле теряется. Взамен для регуляризации генеративной сети было предложено использовать индивидуальную нормализацию (instance normalization), так как в ней статистики вычисляются отдельно для каждого объекта в батче следующим образом:

$$\begin{aligned} y_{nkij} &= \frac{x_{nkij} - \mu_{nk}}{\sqrt{\sigma_{nk}^2 + \epsilon}} \\ \mu_{nk} &= \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} x_{nkij} \\ \sigma_{nk}^2 &= \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} (x_{nkij} - \mu_{nk})^2 \end{aligned} \tag{11}$$

Данный способ регуляризации дает существенное улучшение скорости обучения и качества работы алгоритма, так как удаляет особенности общей яркости и контрастности стилизуемого изображения.

Conditional Instance Normalization На основе предыдущей работы в [11] авторы предлагают использовать разные обучаемые параметры в индивидуальной нормализации в зависимости от изображения стиля. Данный подход был назван обусловленной индивидуальной нормализацией (conditional instance normalization). Его суть состоит в использовании одной генеративной сети для переноса различных стилей, при этом в зависимости от переносимого стиля используются разные параметры индивидуальной нормализации. В этом случае для каждого слоя l с количеством каналов C_l есть две матрицы γ и β размера $N \times C_l$, где N — количество моделируемых стилей. Обуславливание признаков x на стиль s выглядит следующим образом:

$$z_{nkij} = \gamma_{sk} \frac{x_{nkij} - \mu_{nk}}{\sqrt{\sigma_{nk}^2 + \epsilon}} + \beta_{sk}, \tag{12}$$

где n — номер объекта в батче, k — номер канала, μ_{nk} и σ_{nk} вычисляются по формулам (11), γ_s и β_s получены из матриц γ и β выбором соответствующих стилю

строк. Таким образом, использование обуславливания выходов слоев позволяет использовать одну сеть для переноса различных стилей. Количество параметров для обуславливания гораздо меньше общего числа параметров сети, что уменьшает затраты на хранение модели. Однако все еще остается необходимость обучения параметров в обусловленной индивидуальной нормализации для новых стилей.

Adaptive Instance Normalization В работе [12] было продолжено исследование различных нормализаций. В результате был предложена адаптивная индивидуальная нормализация (adaptive instance normalization), в которой вместо параметров γ_s и β_s из обусловленной нормализации используются стандартное отклонение и среднее карты признаков для изображения стиля:

$$z_{nkij} = \sigma_k^s \frac{x_{nkij} - \mu_{nk}}{\sqrt{\sigma_{nk}^2 + \epsilon}} + \mu_k^s, \tag{13}$$

где μ_k^s и σ_k^s соответствуют среднему и дисперсии канала k карты признаков для изображения стиля s.

Данное решение показывает сравнимое качество с предыдущим подходом. Преимуществом метода является то, что с его использованием одна генеративная сеть способна переносить любые стили, в том числе отсутствовавших при обучении.

MSG-Net Предыдущие работы были связаны с преобразованиями среднего и дисперсии по отдельным каналам карты признаков. Однако в работе [3] было показано, что немало информации о стиле содержится в матрице ковариаций каналов карты признаков. Поэтому в работе [13] предлагается использовать матрицу Грама изображения стиля как один из параметров для переноса стиля. Для этого авторы предлагают использовать новый слой под названием CoMatchLayer:

$$CoMatchLayer\left(F_l(x_{content}), F_l(x_{style})\right) = F_l(x_{content}) \times W \times G(F_l(x_{style})), \tag{14}$$

где $F_l(x) \in \mathbb{R}^{H_l W_l \times C_l}$ — выход слоя l генеративной сети для изображения x,
$G(F_l(x)) \in \mathbb{R}^{C_l \times C_l}$ — матрица Грама для карты признаков $F_l(x)$, $W \in \mathbb{R}^{C_l \times C_l}$ — обучаемые параметры.

В данном методе в одну и ту же генеративную сеть подают на вход изображения $x_{content}$ и x_{style}. Первую половину сети обычно называют кодировщиком (encoder) и

используют для получения высокоуровневых признаков для изображений. Затем с помощью CoMatchLayer признаки $F(x_{content})$ преобразуют по формуле (14), внося в них информацию о переносимом стиле. После этого вторая половина сети, называемая декодировщиком (decoder), возвращает результат переноса стиля. К преимуществу данного метода можно отнести возможность использования одной сети для переноса любых стилей. На Рис. 4 приведена схема работы генеративной сети, которую авторы назвали MSG-Net.

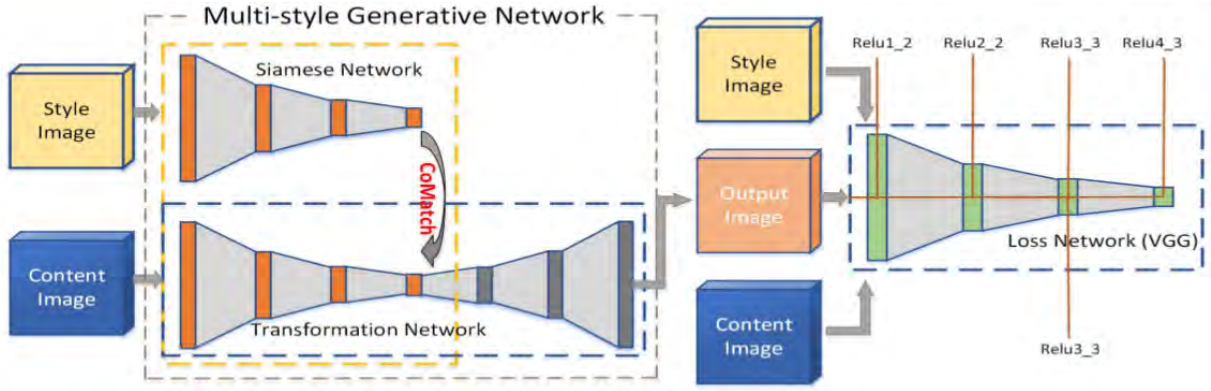


Рис. 4: Схема работы сети MSG-Net

3.5 Контроль размера художественной кисти

Задачу переноса стиля можно рассматривать как задачу переноса текстур с одного изображения на другое. С этой точки зрения существенной характеристикой является размер переносимой текстуры. Поэтому одной из подзадач переноса стиля является контроль размера переносимой текстуры, чаще называемой задачей контроля размера художественной кисти.

В работе [14] обращают внимание на то, что размер изображения стиля оказывает существенное влияние на масштаб переносимого стиля. Это связано с тем, что сверточная нейронная сеть, используемая для вычисления функции потерь, имеет фиксированную область восприятия. Если взять одно и то же изображение в разных разрешениях, то признаки, полученные с помощью сверточной нейронной сети для изображения большего размера, будут описывать текстуры меньшего размера. Авторы работы [14] предлагают использовать это наблюдение для контроля размера переносимой текстуры. Для этого во время обучения при подсчете функции потерь предлагается использовать изображения стиля меньшего размера, если необходимо

получить больший размер текстур. Однако данный подход не позволяет менять размер кисти в режиме реального времени, так как генеративная сеть обучается переносить только один размер текстур, присутствовавший при обучении. Поэтому приходится обучать свою генеративную сеть для каждого размера кисти, что увеличивает затраты по памяти и времени обучения.

В работе [13], посвященной архитектуре MSG-Net, предлагают обучать единую сеть для разных масштабов переносимого стиля. Для этого во время обучения предлагается использовать разные размеры изображения стиля для подачи на вход как генеративной сети, так и используемой при подсчете функции потерь предобученной сверточной нейронной сети. Авторы метода полагают, что слоя CoMatchLayer достаточно для контроля размера кисти. Тогда после обучения пользователь может оказывать влияние на масштаб переносимого стиля, изменяя размер подаваемого в генеративную сеть изображения стиля.

4 Предлагаемый метод

4.1 Анализ влияющих на размер художественной кисти факторов

В обзоре литературы было упомянуто, что размер изображения стиля влияет на размер переносимой текстуры. Уменьшение размера изображения позволяет увеличить область восприятия признаков, полученных с помощью сверточной нейронной сети. Это наводит на мысль, что можно контролировать размер переносимых при стилизации текстур, варьируя область восприятия генеративной сети. Это можно проинтерпретировать тем, что обучение генеративной сети можно представить как обучение ядер сверток воспроизведению текстур фиксированного размера. В разделе 3.1 показывалось, что количество слоев в сверточной нейронной сети оказывает влияние на размер области восприятия. Поэтому варьирование количества слоев можно использовать в качестве дополнительного способа контроля области восприятия генеративной сети.

4.2 Описание метода

Основной идеей предлагаемого подхода является то, область восприятия генеративной сети должна быть параметром, определяемым пользователем в зависимости от желаемого размера кисти. Для этого в рамках единой генеративной сети должна быть возможность применения разного количества слоев. Достичь этого можно с помощью блока ветвления, в котором для разных размеров кисти применяются разные ветви. Идея ветвления взята из статьи [15], однако в предлагаемом подходе помимо выбора размера кисти предлагается возможность выбора переносимого стиля в режиме реального времени. При этом каждая ветвь имеет бóльшую область восприятия по сравнению с предыдущей. Для избежания резкого роста числа весов генеративной сети применяется разделение параметров между различными ветвями, при котором каждая следующая ветвь строится на основе предыдущей добавлением дополнительных слоев. Такой модуль позволяет легко регулировать область восприятия генеративной сети с помощью выбора ветви с необходимым количеством слоев.

Сеть состоит из четырех последовательно соединенных блоков. В начале расположен кодировщик, задачей которого является получение признакового описания для изображений $x_{content}$ и x_{style}. Затем расположен CoMatchLayer, описанный в разделе 3.4, который добавляет в признаки изображения $x_{content}$ информацию о стиле изображения x_{style} по формуле (14). После CoMatchLayer расположен блок ветвления, в котором происходит моделирование текстур разного размера. Количество ветвей в блоке зависит от того, сколько размеров кисти мы собираемся моделировать и насколько плавно мы хотим осуществлять переход от меньшего масштаба стиля к большему. Выход блока ветвления подается на вход декодировщику, преобразующему высокоуровневые признаки в итоговое стилизованное изображение. Схема генеративной сети изображена на Рис. 5. Содержимое кодировщика, декодировщика и одной ветви блока ветвления изображено на Рис. 6.

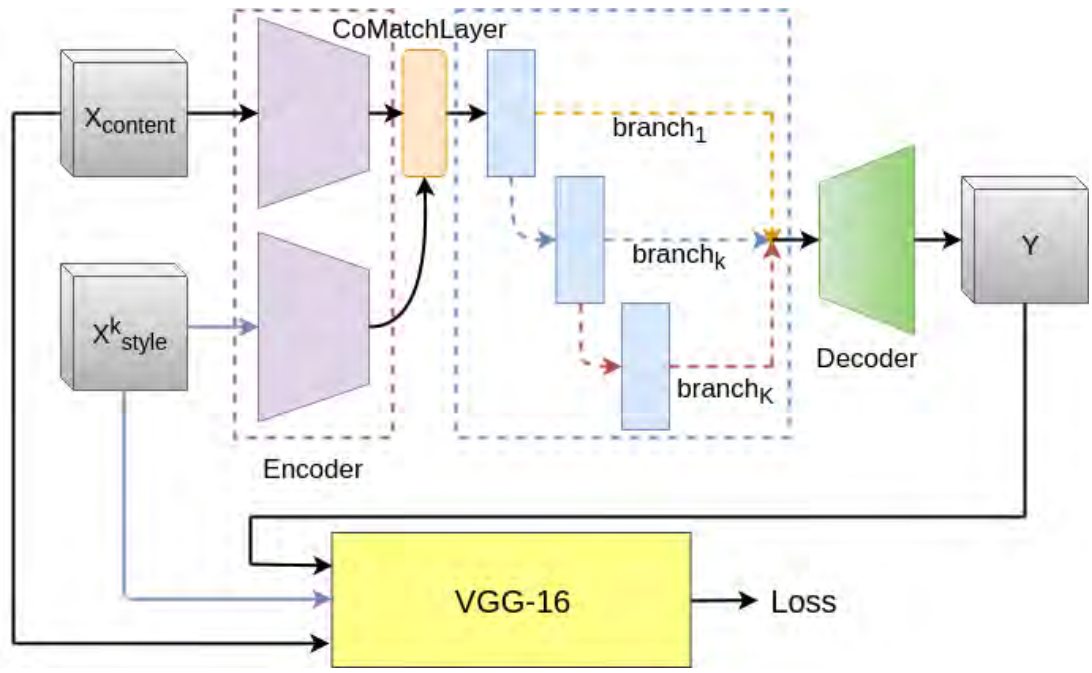
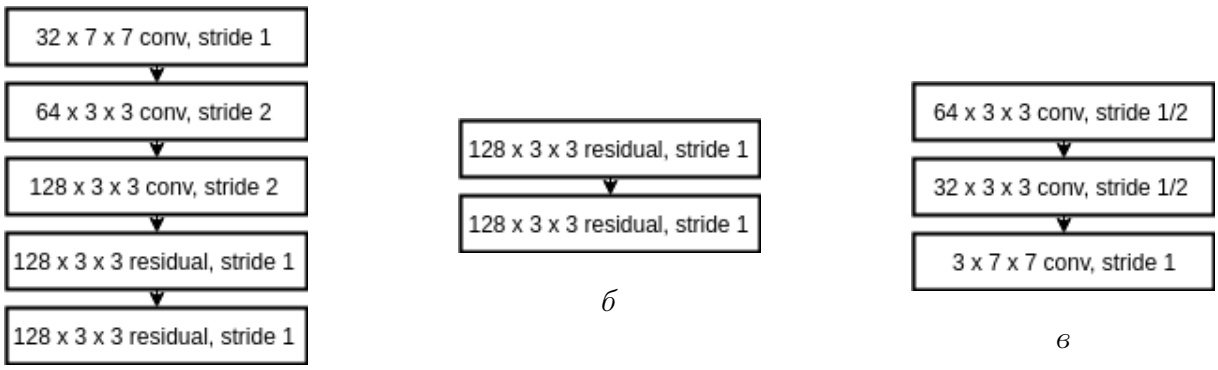


Рис. 5: Схема генеративной сети с блоком ветвления. Во время одной итерации размеру кисти k соответствуют масштабированное изображение стиля x^k_{style} и ветвь $branch_k$.



а *б* *в*

Рис. 6: Содержимое компонент генеративной сети с блоком ветвления. Схема *а* соответствует кодировщику, *б* — одной ветви блока ветвления, *в* — декодировщику. Запись вида $C \times H \times W$ означает, что слой состоит из C фильтров с размером ядра свертки $H \times W$. Для регуляризации использовалась индивидуальная нормализация [9], в качестве нелинейной функции активации был выбран ReLU. Шаг сдвига 2 использовался для уменьшения высоты и ширины изображения, шаг $\frac{1}{2}$ — для увеличения изображения способом, описанным в работе [13]. Описание residual слоя содержится в работе [16].

Для того, чтобы ветви обучались воспроизводить разные размеры текстур, необходимо использовать различные функции потерь. Достичь этого можно с помощью следующей процедуры обучения. При обучении генеративной сети во время каждой итерации выбирается текущий обучаемый размер кисти $k \in \mathbb{K}$. Размеру кисти k в блоке ветвления соответствует ветвь $branch_k$, которая будет использоваться при стилизации на текущей итерации обучения. Кроме того, изображение стиля x_{style} масштабируется к размеру $H_k \times W_k$. Числа H_k и W_k выбираются по стратегии, описанной в разделе 3.5, согласно которой большему размеру текстур соответствует изображение стиля меньшего размера. Обозначим результат масштабирования как x^k_{style}. Изображения x^k_{style} и $x_{content}$ подаются на вход как генеративной сети, так и сети VGG-16 [4], используемой при подсчете функции потерь. Таким образом, функция потерь, используемая на текущей итерации, зависит от размера кисти k и вычисляется по следующей формуле:

$$\mathcal{L}^{total}_k\left(x_{content}, x^k_{style}, f_W(x_{content}, x^k_{style})\right) = \mathcal{L}^{content}\left(x_{content}, f_W(x_{content}, x^k_{style})\right) + \\ +\alpha\mathcal{L}^{style}\left(x^k_{style}, f_W(x_{content}, x^k_{style})\right) + \gamma\mathcal{L}^{TV}\left(f_W(x_{content}, x^k_{style})\right) \quad (15)$$

Algorithm 1: Процедура обучения генеративной сети с блоком ветвления

$\mathbb{X}_c = \{x_{content,i}\}_{i=1}^{N}$ — обучающая выборка

$\mathbb{X}_s = \{x_{style,i}\}_{i=1}^{N_{styles}}$ — множество моделируемых стилей

$\mathbb{K}$ — множество моделируемых масштабов переносимого стиля

W — все веса генеративной сети

W^k — веса генеративной сети, задействованные при использовании ветви $branch_k$

for $i = 1, \ldots, n_{iter}$ **do**

    Выбираем случайно $x_{content} \in \mathbb{X}_c$, $x_{style} \in \mathbb{X}_s$, $k \in \mathbb{K}$

    Масштабируем изображение стиля к размеру k: $x_{style} \to x^k_{style}$

    Проводим стилизацию с выбором ветви $branch_k$: $y = f_{W^k}(x_{content}, x^k_{style})$

    Обновляем веса W^k, используя градиент функции потерь $\mathcal{L}^{total}_k\left(x_{content}, x^k_{style}, y\right)$, вычисленной по формуле (15)

end for

Благодаря такой процедуре обучения ветвь $branch_k$ генеративной сети можно использовать для моделирования размера кисти k. Для этого во время применения

генеративной сети достаточно выбрать соответствующую ветвь в блоке ветвления. Кроме того, с помощью генеративной сети можно получать стилизованные изображения с не присутствовавшим при обучении размером кисти. Этого можно достичь с помощью интерполирования выходов различных ветвей сети. Пусть ветви $branch_i$ и $branch_j$ моделируют размеры кисти i и j соответственно, причем $i < j$. Тогда для моделирования размера кисти $l \in [i, j]$ можно попробовать подобрать коэффициент $\alpha \in [0, 1]$ такой, что линейная комбинация выходов ветвей $branch_i$ и $branch_j$ с весами α и $1 - \alpha$ будет моделировать размер кисти l.

5 Эксперименты

5.1 Детали реализации

Для подсчета функции потерь использовалась предобученная на задаче классификации сеть VGG-16. Качество сохранения содержания вычислялось с использованием слоя ReLU3_3, качество переноса стиля оценивалось с помощью слоев ReLU1_2, ReLU2_2, ReLU3_3 и ReLU4_3. Для обучения генеративной сети в качестве набора изображений $\{x_{content,i}\}_{i=1}^{N}$ использовалась выборка Microsoft COCO [17], в которой содержится около 80000 изображений. Важность компонент в функции потерь (15) была задана следующим образом: $\alpha = 5$, $\gamma = 1 \times 10^{-6}$. В качестве алгоритма для стохастической оптимизации использовался Adam [18] со скоростью обучения 1×10^{-3}. Архитектура генеративной сети была реализована на языке Python с использованием библиотеки Pytorch.

5.2 Сравнение качества контроля размера художественной кисти

Для сравнения качества работы были выбраны следующие методы контроля размера кисти в многостилевых генеративных сетях:

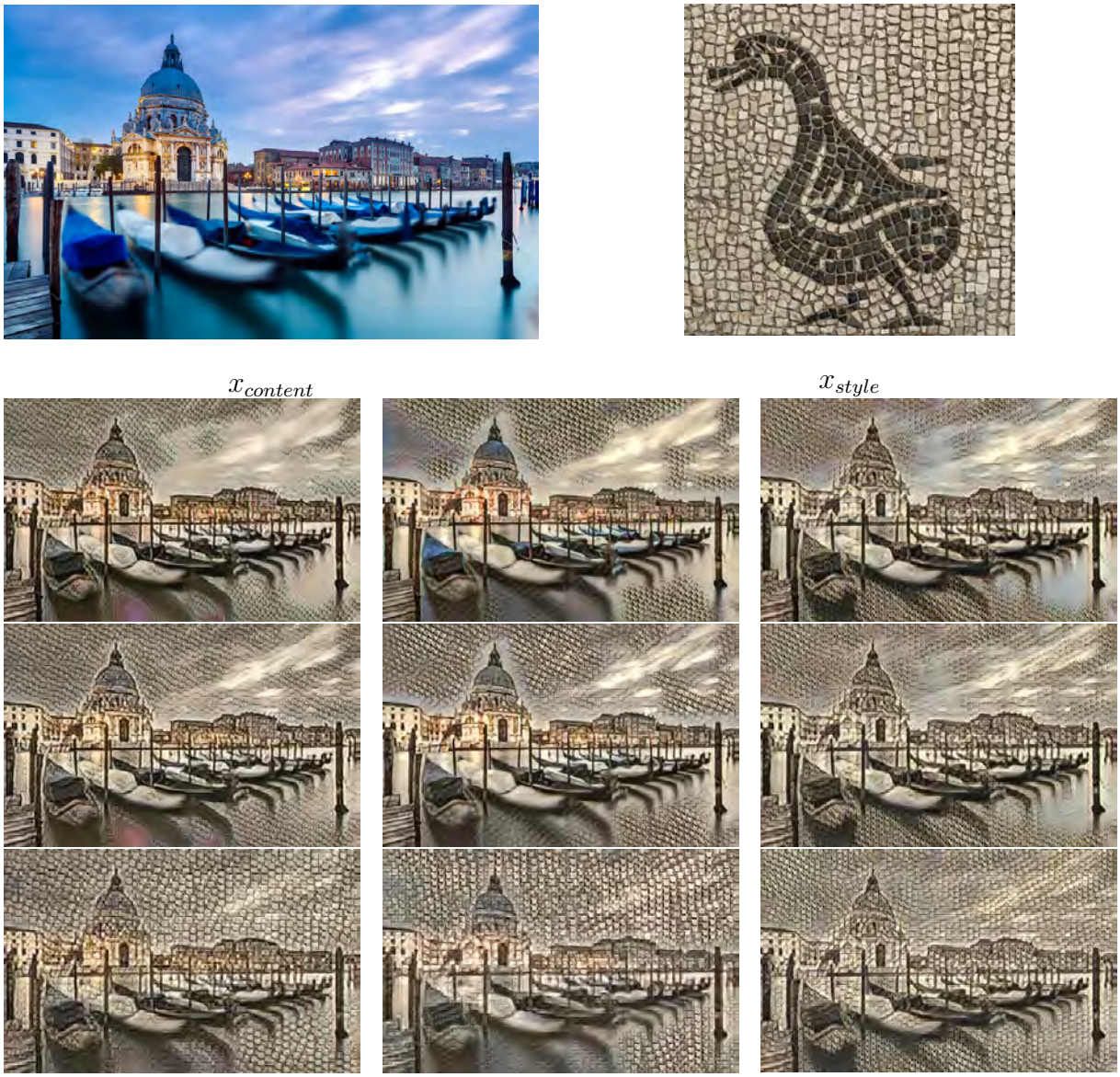
- Предложенный метод с использованием блока ветвления в генеративной сети для моделирования разных размеров кисти;

- Обучение отдельной генеративной сети для каждого требуемого размера кисти [14];

- Использование одной генеративной сети, в которой масштаб переносимого стиля задается только размером изображения x_{style} [13].

Пример контроля размера кисти этими методами можно увидеть на рисунках 7 и 8. Из них можно увидеть, что первые два алгоритма справляются с задачей контроля размера переносимой текстуры. Например, на Рис. 7 в соответствующих этим алгоритмам столбцах *а* и *б* при увеличении размера кисти на стилизованных изображениях увеличивается размер „кирпичей“, из которых состоят изображения. Предложенный алгоритм с использованием блока ветвления достигает сравнимых результатов с работой [14] в плане качества контроля масштаба переносимого стиля.

Третий метод [13], в котором масштаб переносимого стиля задается только размером изображения стиля, показывает меньшую вариативность размера кисти. Это может быть связано с тем, что в такой архитектуре есть только один зависящий от размера изображения стиля слой CoMatchLayer, который не способен так же качественно адаптироваться к разным масштабам переносимого стиля, как специально для этого обученные ветви в блоке ветвления. Это показывает, что предложенная архитектура генеративной сети с блоком ветвления лучше моделирует разные масштабы переносимого стиля, чем метод [13].



а *б* *в*

Рис. 7: Результат стилизации при различных размерах кисти. В первом ряду по горизонтали расположены содержательное и стилевое изображения. В последующих рядах расположены результаты стилизации разными методами по мере увеличения размера кисти. Столбец *а* получен с помощью предложенной генеративной сети с блоком ветвления, *б* — тремя генеративными сетями, обученными по методу [14], *в* — генеративной сетью, обученной по методу [13].

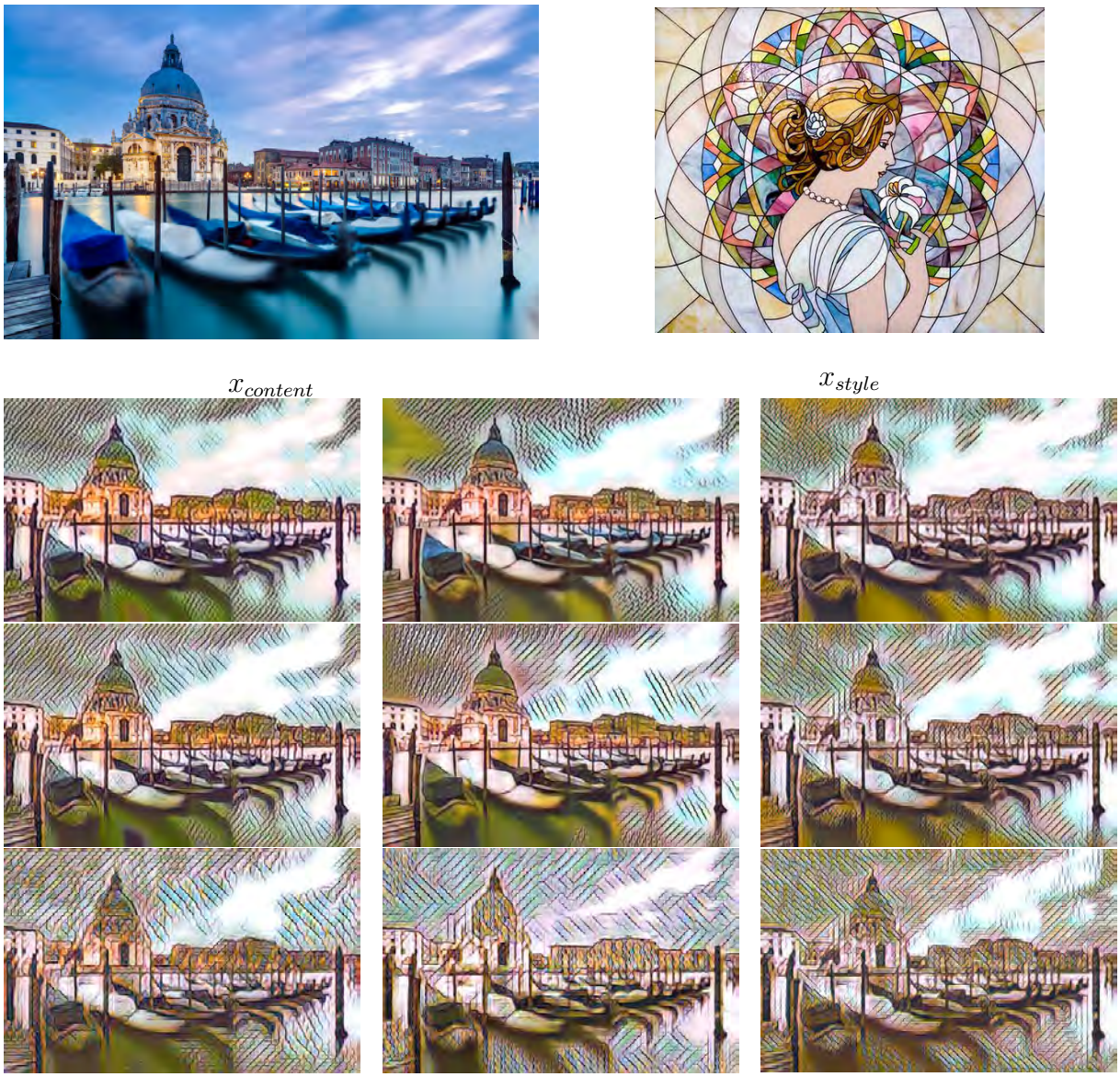


Рис. 8: Результат стилизации при различных размерах кисти. В первом ряду по горизонтали расположены содержательное и стилевое изображения. В последующих рядах расположены результаты стилизации разными методами по мере увеличения размера кисти. Столбец *а* получен с помощью предложенной генеративной сети с блоком ветвления, *б* — тремя генеративными сетями, обученными по методу [14], *в* — генеративной сетью, обученной по методу [13].

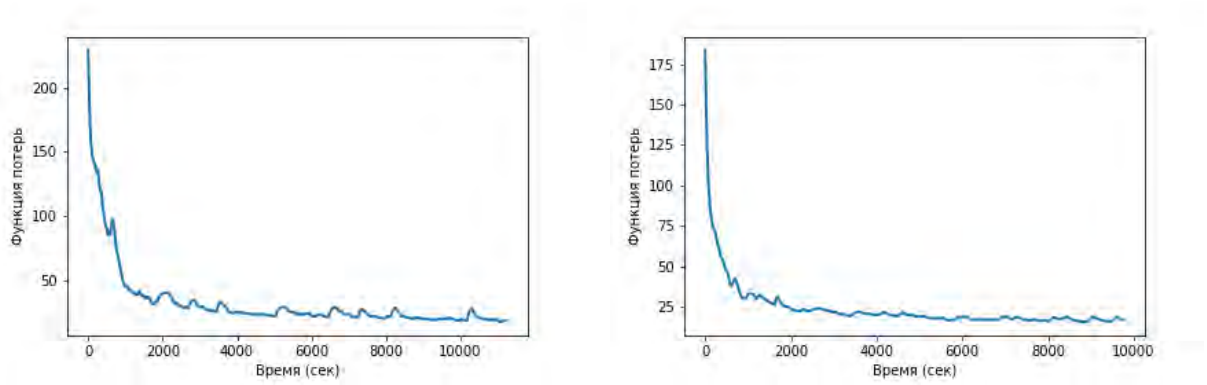
5.3 Сравнение количества параметров и времени обучения

Особенностью предложенного блока ветвления в генеративной сети является построение следующей ветви на основе предыдущей с помощью добавления дополнительных слоев. Это позволяет использовать меньшее количество параметров по сравнению с методом, в котором для каждого размера кисти требуется своя генеративная сеть. В таблице 1 показано количество используемых параметров в этих методах. Предложенный метод имеет более чем в два раза меньшее число параметров, причем разница при увеличении числа моделируемых размеров кисти будет только нарастать.

Также представляют интерес время обучения моделей. На Рис. 9 приведены графики зависимости функции потерь от времени при обучении генеративной сети как с блоком ветвления, так и без него. Результаты показывают, что скорость обучения обеих генеративных сетей имеет схожий порядок. Однако генеративная сеть с блоком ветвления за это время обучается моделировать три размера кисти, а обычная сеть — один размер кисти. Таким образом, предложенный метод требует меньше времени на обучение, если необходимо моделировать больше одного размера переносимой текстуры.

Метод	Количество параметров для моделирования трех размеров кисти (млн)
Генеративная сеть с тремя ветвями в блоке ветвления	1.37
Своя генеративная сеть для каждого моделируемого размера кисти	3.24

Таблица 1: Сравнение общего количества параметров в разных архитектурах.



а *б*

Рис. 9: Графики зависимости функции потерь от времени при обучении генеративной сети. График *а* соответствует обучению генеративной сети с блоком ветвления, моделирующей три размера кисти. График *б* соответствует обучению обычной генеративной сети, моделирующей один размер кисти.

5.4 Выводы

Результаты экспериментов показали, что предложенный метод достигает схожего качества работы с существующими методами контроля размера художественной кисти для многостилевых генеративных сетей. При этом он обладает следующими преимуществами:

- Требуется меньше памяти для хранения модели. Это преимущество может стать ключевым при использовании нейросетевого переноса стиля на мобильных устройствах;
- Обучение моделирующей K размеров кисти генеративной сети с блоком ветвления требует меньше времени, чем обучение K генеративных сетей, каждый из которых моделирует свой размер переносимой текстуры.

6 Заключение

В данной работе были получены следующие результаты:

- Рассмотрены основные архитектуры многостилевых генеративных сетей;
- Проведен анализ существующих методов контроля размера переносимой текстуры, указаны их преимущества и недостатки;
- Предложена архитектура сети, объединяющая как возможность стилизации в реальном времени с использованием многих стилей, так и возможность контролировать масштаб применяемого стиля в непрерывной шкале. Было проведено сравнение предложенного метода с существующими алгоритмами контроля размера кисти;
- Предложенная архитектура генеративной сети была реализована на языке Python с использованием библиотеки Pytorch.

Список литературы

[1] *Shin H. C. et al.* Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning //IEEE transactions on medical imaging. – 2016. – Т. 35. – №. 5. – С. 1285-1298.

[2] *Pan S. J., Yang Q.* A survey on transfer learning //IEEE Transactions on knowledge and data engineering. – 2010. – Т. 22. – №. 10. – С. 1345-1359.

[3] *Gatys L. A., Ecker A. S., Bethge M.* Image style transfer using convolutional neural networks //Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on. – IEEE, 2016. – С. 2414-2423.

[4] *Simonyan K., Zisserman A.* Very deep convolutional networks for large-scale image recognition //arXiv preprint arXiv:1409.1556. – 2014.

[5] *Li Y. et al.* Demystifying neural style transfer //arXiv preprint arXiv:1701.01036. – 2017.

[6] *Johnson J., Alahi A., Fei-Fei L.* Perceptual losses for real-time style transfer and super-resolution //European Conference on Computer Vision. – Springer, Cham, 2016. – С. 694-711.

[7] *Hecht-Nielsen R.* Theory of the backpropagation neural network //Neural networks for perception. – 1992. – С. 65-93.

[8] *Ulyanov D. et al.* Texture Networks: Feed-forward Synthesis of Textures and Stylized Images //ICML. – 2016. – С. 1349-1357.

[9] *Ulyanov D., Vedaldi A., Lempitsky V.* Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis //Proc. CVPR. – 2017.

[10] *Ioffe S., Szegedy C.* Batch normalization: Accelerating deep network training by reducing internal covariate shift //arXiv preprint arXiv:1502.03167. – 2015.

[11] *Dumoulin V., Shlens J., Kudlur M.* A learned representation for artistic style //CoRR, abs/1610.07629. – 2016. – Т. 2. – №. 4. – С. 5.

[12] *Huang X., Belongie S.* Arbitrary style transfer in real-time with adaptive instance normalization //CoRR, abs/1703.06868. – 2017.

[13] *Zhang H., Dana K.* Multi-style generative network for real-time transfer //arXiv preprint arXiv:1703.06953. – 2017.

[14] *Gatys L. A. et al.* Controlling perceptual factors in neural style transfer //IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017.

[15] *Jing Y. et al.* Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields //arXiv preprint arXiv:1802.07101. – 2018.

[16] *He K. et al.* Deep residual learning for image recognition //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – C. 770-778.

[17] *Lin T. Y. et al.* Microsoft coco: Common objects in context //European conference on computer vision. – Springer, Cham, 2014. – C. 740-755.

[18] *Kingma D. P., Ba J.* Adam: A method for stochastic optimization //arXiv preprint arXiv:1412.6980. – 2014.