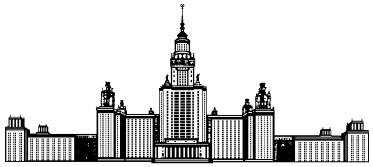
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

ДИПЛОМНАЯ РАБОТА СТУДЕНТА 417 ГРУППЫ

«Оптимизация учета глубины изображения в задаче автоматического переноса стиля»

Выполнил:

студент 4 курса 417 группы

Козловцев Константин Анатольевич

Научный руководитель:

к.ф.-м.н.

Китов Виктор Владимирович

Москва, 2018

Содержание

1 Введение

В последнее время стала очень популярной модификация фотографий с целью их улучшения. И если обычные операции вроде размытия, выделения деталей, настройки контраста, или наложения фильтров достаточно просто решаются попиксельными и морфологическими операциями, то задача наложения стиля, позволяющая сделать из фотографии картины, и которая была предложена и решена L.Gatys в 2016 году [2], требует использования более сложного метода, которым в итоге стали сверточные нейросети. Сверточные сети способны находить на изображениях отдельные объекты или значимые части объектов, которые также называют высокоуровневыми признаками (тогда как пиксели изображения можно считать низкоуровневыми признаками). Эта их способность позвляет эффективно различать объекты на изображении, восстанавливать некоторые величины, например расстояние до объектов, и модернизировать изображения с сохранением высокоуровневых признаков. На последнем принципе и основан нейронный перенос стиля.

В случае переноса стиля Gatys наложение стиля зачастую делает фотографию плоской. Для сохранения глубины изображения используется сверточная нейросеть, которая может эту глубину оценить, и перенести на результатирующее изображение вместе со стилем.

Таким образом можно получить картину из любого изображения без использования сложных графических редакторов и без умения рисовать (достаточно иметь картину-образец). В случае использования рассмотренных и предложенного ниже метода, на полученной картине будет сохранена перспектива, и положение объектов в глубину, друг относительно друга.

2 Постановка задачи

Задача переноса стиля изображений заключается следующем: даны два изображения — стиль и контент, требуется сгенерировать изображение, похожее на контент по изображенным на нем объектам и с сохранением общей композиции контента, но результатирующее изображение должно быть в одном стиле с изображением-стилем рис. 2.

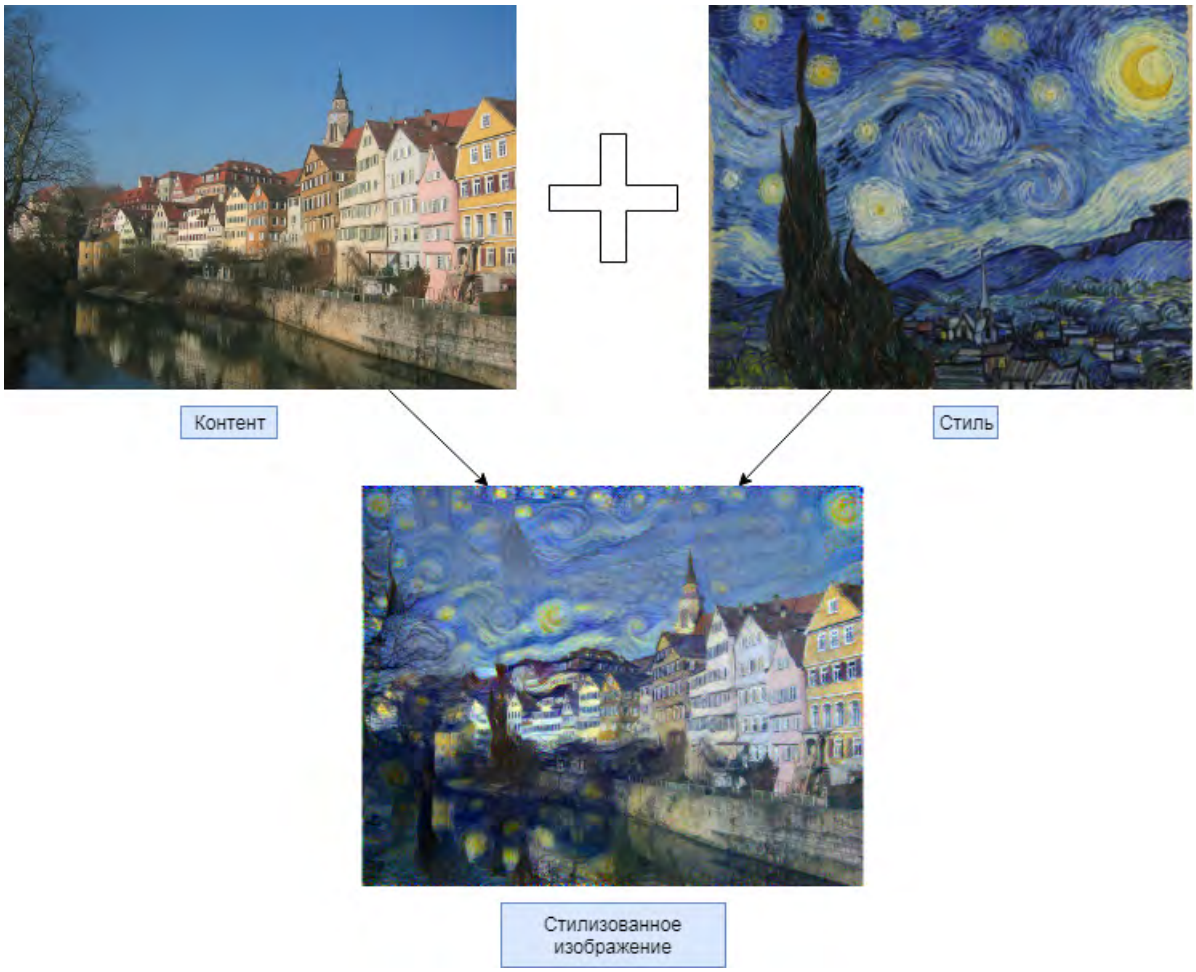


Рис. 1: Пример переноса стиля

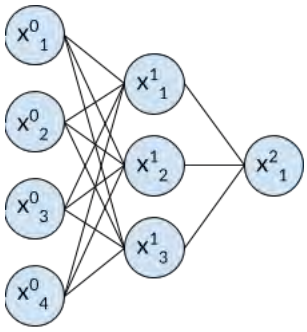
Практическое применение такой задачи — создание «картин» из фотографий, в некотором художественном стиле или похожими на некоторую определенную картину.

Проблема классического подхода к стилизации изображений заключается в том, что изображения с сильно наложенным стилем начинают выглядить плоскими. Решением этой проблемы призваны быть методы учета глубины изображения. В данной работе реализуется и рассматривается существующий метод учета глубины, и предлагается один новый метод.

3 Обзор литературы и существующих методов

3.1 Нейронные сети

Нейронные сети — достаточно популярный в последнее время метод машинного обучения позволяющий относительно просто находить сложные зависимости. В основе нейронной сети лежит понятие нейрона — математической модели биологических нейронов. Нейрон представляет собой совокупность из: значения полученного на вход (аналогия — пришедший электрический импульс), нелинейного преобразования над этим значением, также называемым функцией активации (по аналогии — нейрон может возбудиться от поданного сигнала, или остаться в спокойном состоянии), и передачи преобразованного значения другим нейронам с некоторым весом (аналог — передача импульсов по аксонам нейрона). Передача сигнала ведется послойно от одного слоя нейронов к следующему.



Математически такая модель записывается как $x_j^{l+1} = \sum_{j=1}^{k} f(x_i^l) w_{ij}^l$, где x_i^l — значение в нейроне передающем сигнал, x_j^{l+1} — значение в нейроне, который принимает сигналы от предыдущего слоя нейронов, и w_{ij}^l - вес с которым сигнал передается от i-го нейрона к j-му между слоями l и $l+1$. То же самое в матричной форме:

$$x^{l+1} = W^l f(x^l),\ l = 0, \ldots, L-1$$

На вход такой сети подается представление объекта в виде вектора $x^0 = x$, значение на выходе $\hat{y} = x^L$ оценивается некоторой дифференцируемой функцией потерь $L(\hat{y}, y)$, здесь y — известный ответ, функция потерь штрафует разницу между выходом сети и этим ответом. Так как функция потерь дифференцируема, при выборе дифференцируемых функций активации можем найти градиент $\frac{\partial L}{\partial W}$, а значит и вы-

полнить оптимизацию $W^* = \underset{W}{argmin}\, L(\hat{y}(W), y)$ при помощи градиентного спуска и другими связанным с ним методами, а значит и обучить нейросеть.

3.2 Сверточные нейронные сети

Сверточные нейронные сети — это модификация обычной нейронной сети с полносвязными слоями, которая в основном ориентирована на работу с изображениями. Обычная нейронная сеть при работе с изображением никак не учитывает близость отдельных пикселей, на вход ей можно подать только изображение в вытянутом в вектор виде. Также проблемой является большое количество нейронов, и соответственно большое количество весов для обучения.

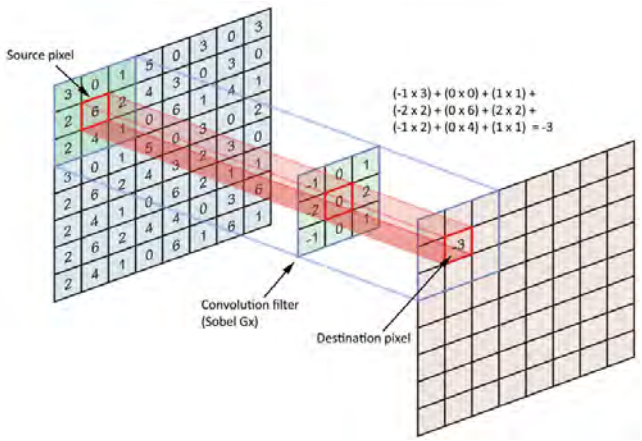


Рис. 2: Принцип операции свертки

Для того чтобы учесть взаимное расположение отдельно взятых пикселей на изображении, предлагается использовать сверку: $X_i^{l+1} = \sum_j^{C^{l+1}} f(X_j^l) * W_i^l, i = 1, \ldots, C^l$, где $*$ — операция двумерной (в случае работы с изображением) свертки, $(A * B)_{xy} = \sum_{ij} A_{x-i,y-j} B_{ij}$, здесь A и B двумерные массивы, результат свертки — также двумерный массив, индексы i, j принимают все возможные значения, при которых определены индексируемые ячейки. W_i^l называют ядром свертки или фильтром сверточной сети, их может быть несколько, каждый фильтр выдает на выходе один канал изображения. C^l и C^{l+1} — количество каналов в слое l и $l+1$ соответственно.

При выборе относитльно небольшого размера ядра свертки (обычно это квадрат со стороной 3, 5 или 7 пикселей), в вычислении одного пикселя в канале слоя $l+1$ участвуют только те пиксели из слоя l, которые находятся рядом не далее чем на

расстоянии размера ядра (область влияния одних нейронов на другие называется рецептивным полем). При увеличении количества сверточных слоев рецептивное поле увеличивается, захватывая большее число пикселей, и нейросеть находит все более высокоуровневые признаки на изображении.

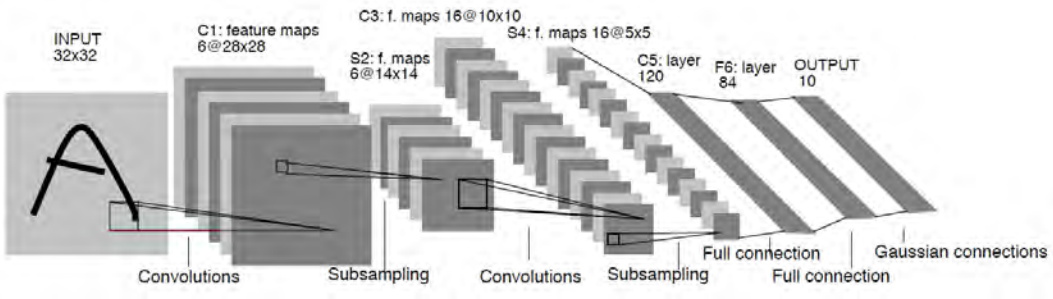


Рис. 3: Пример сверточной нейронной сети для распознавания рукописных символов Lenet-5.

3.3 Нейронный перенос стиля

Перенос стиля это метод генерации изображений по двум другим — стилю (Y_{style}) и контенту ($Y_{content}$), таким образом, что на сгенерированном изображении присутствуют те же объекты что и на изображении-контенте, но при этом изображение похоже мелкими деталями на стилевое изображение. Обычно в качестве контента берут фотографию, в качестве стиля — картину или рисунок в определенном жанре, на выходе данного метода будет эта же фотография, но как будто нарисованная в том же стиле что и картина.

Решение этой задачи впервые представлено в работе [2]. Задача решается при помощи нейронной сети VGG — предобученная на большом наборе изображений imagenet нейросеть, которая решает задачу поиска объектов. Метод устроен следующим образом: исходые изображения пропускаются через предобученную нейросеть, после чего для некоторого изображения X рассчитываются следующие две величины:

- Функция потерь контента,

$$L^l_{content}(X, Y_{content}) = \frac{1}{HWC} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} (X^l_{ijc} - Y^l_{ijc})^2,$$

где X^l, Y^l — карты признаков, полученные после применения нейросети к изображениям X и Y, в l-м слое нейросети, H, W, C — их размеры и число каналов соответственно. Такая функция показывает насколько похожи между собой изображения X и Y, но не сравнивает их попиксельно, а находит похожие высокоуровневые признаки и сравнивает их.

- Функция потерь стиля,

$$L^l_{style}(X, Y_{style}) = \sum_i^C \sum_j^C (G^{X^l}_{ij} - G^{Y^l_{style}}_{ij})^2,$$

где G^A — матрица Грамма для карты признаков A: $G^A_{ij} = \frac{1}{HWC} \sum_{x=1}^{W} \sum_{y=1}^{H} a^i_{xy} a^j_{xy}$, $i, j = 1, \ldots, C$, здесь a^c_{xy} — нейрон с координатами xy на канале c карты признаков A. Матрица Грамма показывает зависимость между нейронами на разных каналах, функция потерь стиля штрафует разницу в соответствующих матрицах и показывает насколько изображения похожи по стилю.

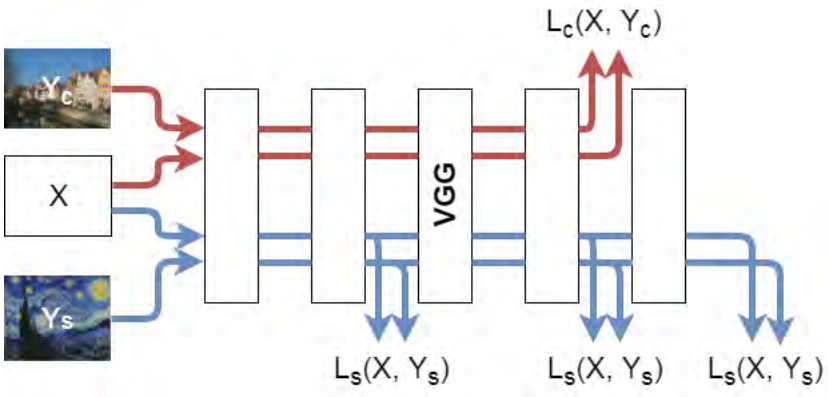


Рис. 4: Принцип вычисления функций потерь контента и стиля

Далее выбирается некоторое изображение X (как правило X инициализируется изображением-контентом), и решается задача оптимизации:

$$X = \underset{X}{argmin} \left[\alpha \sum_{l \in Q_{content}} L^l_{content}(X, Y_{content}) + \beta \sum_{l \in Q_{style}} L^l_{style}(X, Y_{style}) \right],$$

Здесь $Q_{content}$ и Q_{style} — наборы номеров слоев внутри которых необходимо вычислить функции потерь, α, β — веса контента и стиля соответственно (чем больше соотношение $\frac{\alpha}{\beta}$, тем менее стилизованным будет изображение, и наоборот).

3.4 Восстановление глубины по одному изображению

Задача восстановления глубины по изображению описана подробно в работе [4]. В ней предлагается использовать сверточную нейросеть которая принимает на вход трехканальное изображение (в RGB формате) а на выходе выдает одноканальное изображение такго же размера (глубину). Полная топология сети представлена на рис 3.4. Нейросеть состоит из inception блоков, общий вид которых представлен на рис 6

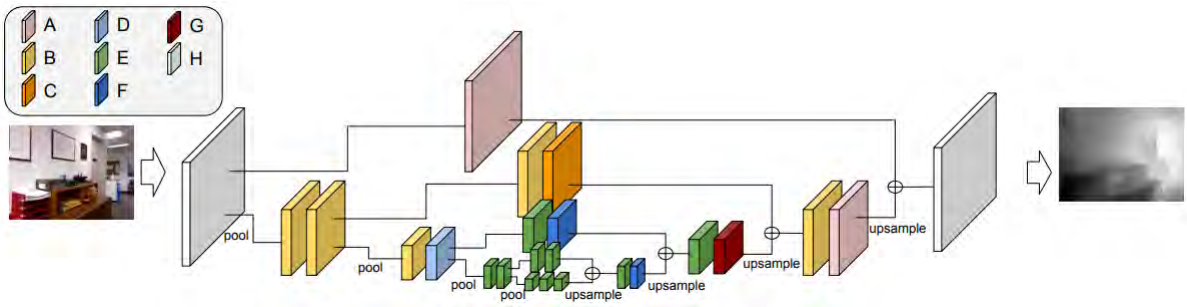


Рис. 5: Топология сети hourglass

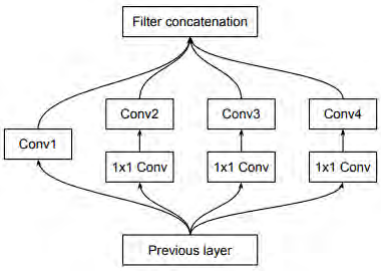


Рис. 6: Общий вид inception блока

блок	A	B	C	D	E	F	G
каналы на входе	128	128	128	128	256	256	128
каналы на выходе	64	128	128	256	256	256	128
каналы на выходе свертки 1×1 Conv	64	128	128	256	256	256	128
размер ядра Conv1	1	1	1	1	1	1	1
размер ядра Conv2	3	3	3	3	3	3	3
размер ядра Conv3	7	5	7	5	5	7	5
размер ядра Conv4	11	7	11	7	7	11	7

Каждый из представленных на рис. 3.4 цветных блоков представляет собой inception блок с некоторым набором параметров. Белые (входной и выходной) слои — это сверточные слои с размерами ядра 7×7 и 3×3 соответственно. Параметры inception блоков представлены в таблице. Все сверточные слои не уменьшают размер карты признаков за счет того что свертка по ядру с размером k производится с отступами на $(k-1)/2$. Все сверточные слои, кроме слоя на выход нейросети используют активацию ReLU.

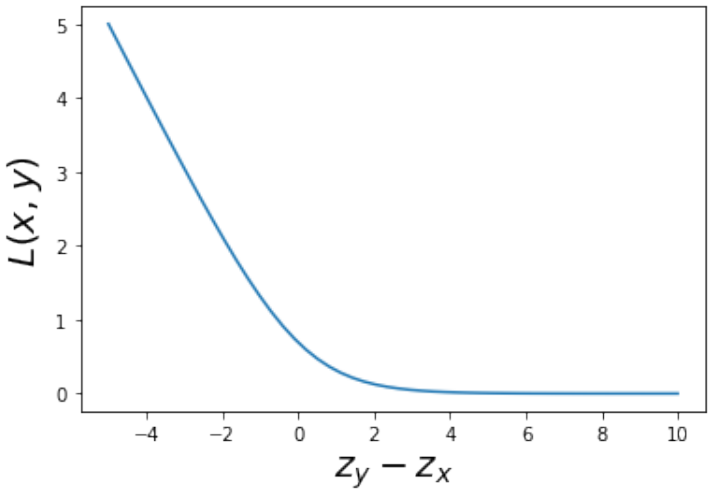
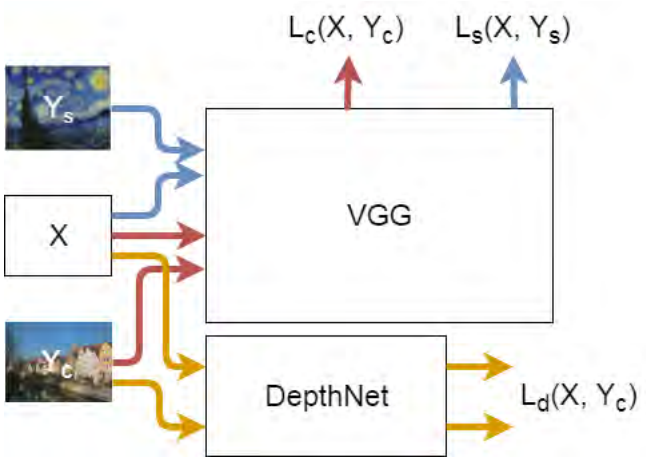


Рис. 7: Функция потерь для нейросети оценивающей глубину

Для обучения такой нейросети используются наборы данных, размеченные при помощи специальных средств (таких как Microsoft Kinect) и вручную размеченных данных. Для каждого изображения вводится набор пар точек x, y для которых известно, что глубина в точке x меньше чем в точке y: $z_x < z_y$. Пусть на выходе нейросети — карта глубин $\hat{Z}$, тогда функция потерь для пары точек выглядит следующим образом:

$$L(\hat{Z}, \{x, y\}) = log(1 + exp(-(\hat{z}_y - \hat{z}_x))).$$

Как видно на рис. 7, такая функция штрафует пропорционально разнице глубин в случае нарушения неравенства $z_x < z_y$, и поощаряет большую разность при ее соблюдении (но в целом потери около 0).



3.5 Учет глубины при переносе стиля

Для того чтобы при переносе стиля изображение не становилось плоским, к обычному методу переноса стиля добавляется третья функция потерь — потери глубины [1]. Она определяется следующим образом: $L_{depth}(X, Y_{content}) = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}(X_{depth} - Y_{depth})^2$, где X_{depth}, Y_{depth} — выход нейросети оценивающей глубину изображения [4]. Для построения конечного изображения необходимо решить задачу вида:

$$X = \underset{X}{argmin}\left[\alpha \sum_{l \in Q_{content}} L^l_{content}(X, Y_{content}) + \beta \sum_{l \in Q_{style}} L^l_{style}(X, Y_{style}) + \gamma L_{depth}(X, Y_{content})\right].$$

4 Предлагаемый метод

Алгоритм описанный в [1] хорошо сохраняет глубину изображения и границы между объектами, и одновременно достаточно интенсивно стилизует его. Тем не менее в результате работы алгоритма, стиль наносится одинаково интенсивно как на объекты, находящиеся в непосредственной близости от камеры, так и на далекие объекты. Обычно более близкие объекты больше привлекают внимание, и сцена выстраивается относительно объектов на переднем плане, тогда как более далекие объекты создают фон, и меньше концентрируют на себе внимание. Поэтому было бы логичнее сохранять ближайшие объекты более четкими, и больше размывать фон, то есть накладывать стиль на фон с большей интенсивностью.

В данной работе предлагается метод, который позволяет использовать карту глубины изображения для раздельного взвешивания нейронов в разных частях карт признаков для функции потерь контента.

4.1 Описание метода

Допустим мы хотим взвешивать потери контента пропорционально коэффициенту α_{ij} для пикселя ij в исходном изображении. Однако, как правило, функция потерь контента считается по некоторому скрытому слою в сети VGG, поэтому напрямую взвесить попиксельную разность нельзя. Ведем для каждого слоя α^l_{ij}, $\alpha^0_{ij} = \alpha_{ij}$, где $alpha_{ij}$ — вес контента для пикселя ij, пропорциональный глубине этого пикселя. Также поставим соответствие между α^l_{ij} и α^{l+1}_{ij} при прохождении через сверточный слой с размером ядра свертки $h \times w$:

$$\alpha^{l+1}_{ij} = \frac{\sum_{x=max(1,i-h)}^{min(H,i+h)} \sum_{y=max(1,j-w)}^{min(W,j+w)} \alpha^l_{xy}}{\sum_{x=max(1,i-h)}^{min(H,i+h)} \sum_{y=max(1,j-w)}^{min(W,j+w)} 1},$$

по сути эта та же свертка, но с ограничением на границах (при неполном наложении ядра свертки на карту признаков), такой выбор связан с тем, что в VGG используются несжимающие сверточные слои, т.е. слои с отступами на $\frac{k-1}{2}$, где k — размер ядра, и заполнением нулями, а при подсчете потерь контента в достаточно глубоком слое, нули могут накопить ошибку, и весь стиль будет прорисован у границы изображения. При прохождении через слой подвыборки (pooling) с размером ядра k производится подвыборка по средним значениям (average pooling) с аналогичным ядром. Таким образом α^l_{ij} соответствует взвешенному набору значений α_{xy} которые попадают в рецептивное поле нейрона ij на карте признаков из слоя l.

Таким образом можем переопределить функцию потерь для контента, вместо $\alpha L_{content}(X, Y_{content})$ будем использовать

$$L_{content}(X, Y, \alpha) = \frac{1}{HWC} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \alpha^l_{ij} (X^l_{ijc} - Y^l_{ijc})^2$$

. Решая описанную выше задачу оптимизации, получим изображение, с неравномерным наложением стиля, при больших α стиль будет подавляться контентом в данной точке. Подставляя в качестве карты весов α нормализованную карту глубин, получим изображение с акцентированным передним планом и размытым задним.

5 Эксперименты

Для экспериментов была использована нейросеть VGG19, для вычисления функций потерь контента и стиля, для контента был использован слой $conv\_2\_2$, для стиля — слои $conv\_1\_2, conv\_2\_2, conv\_3\_3, conv\_4\_3$. Для потерь глубины была использована нейросеть hourglass из работы [4], которая была портирована с torch7 под pytorch и дообучена на датасете KITTI (подробнее ниже). Стилизация изображений происходила для размеров 512×640 для того чтобы изображения помещались в память, перед нейросетями добавлялся слой сжатия изображений в 2 раза (average pooling).

5.1 Дообучение нейросети

В качестве первого эксперимента нейросеть hourglass была дообучена на датасете KITTI[5], который представляет собой съемку городских улиц с панорамной камеры автомобиля и данные с лидаров (датчик определения расстояния до точки, с помощью света). Данные неполные, и на карте глубин отмечены значения лишь некоторых точек.

Рис. 8: Пример объекта из набора данных KITTI. a) Исходное изображение, b) данные с лидара

Для обучения на данном датасете, для каждое изображение обрезалось и сжималось до размера 224×288 После чего на карте глубин для известных значений находились пары пикселей, расстояние между которыми не превышало 40 пискелей, перепад глубины был не ниже 1% от перепада между максимальной и минимальной высотой (чтобы ограничить шум от неточности лидара). Далее для каждой пары ставилась метка какой из 2х пикселей ближе, и производилось обучение.

Результаты обучения измерялись на валидационных выборках датасетов NYU, DIW и KITTI, первые два датасета взяты из оригинальной статьи, NYU — это фотографии интерьеров комнат с оценкой глубины при помощи системы Microsoft kinect. Датасет DIW — это фотографии с открытого сервиса хранения фотографий Flickr, размеченные вручную, на одной фотографии рассматривается только одна пара точек. Про KITTI было описано выше.

В качестве метрики для сравнения были выбраны функция потерь, на которой обучалась нейросеть, описанная выше в пункте 3.4, а также общая точность которая ищется как доля пар точек $\{x, y\}$ для которых выполнено неравенство: $z_x < z_y$. Результаты дообучения представлены в таблице 1.

		NYU	DIW	KITTI
Оригинальная сеть [4]	relative loss	0.613	**0.250**	0.420
	accuracy	0.659	**0.9**	0.863
Дообученная нейросеть	relative loss	**0.464**	1.108	**0.196**
	accuracy	**0.782**	0.8	**0.926**

Таблица 1: Результаты дообучения на новом датасете.

Видно, что при дообучении точность повышается на 2х наборах данных из 3х, как по точности, так и по функции потерь, которая поощаряет больший разброс между точками при правильном ответе, таким образом можно утверждать, что дообученная сеть в качестве результата дает больший разброс между ближайшей и дальнейшей точками, особенно хорошо это видно на изображениях с сильно выраженной перспективой (рис. 9).

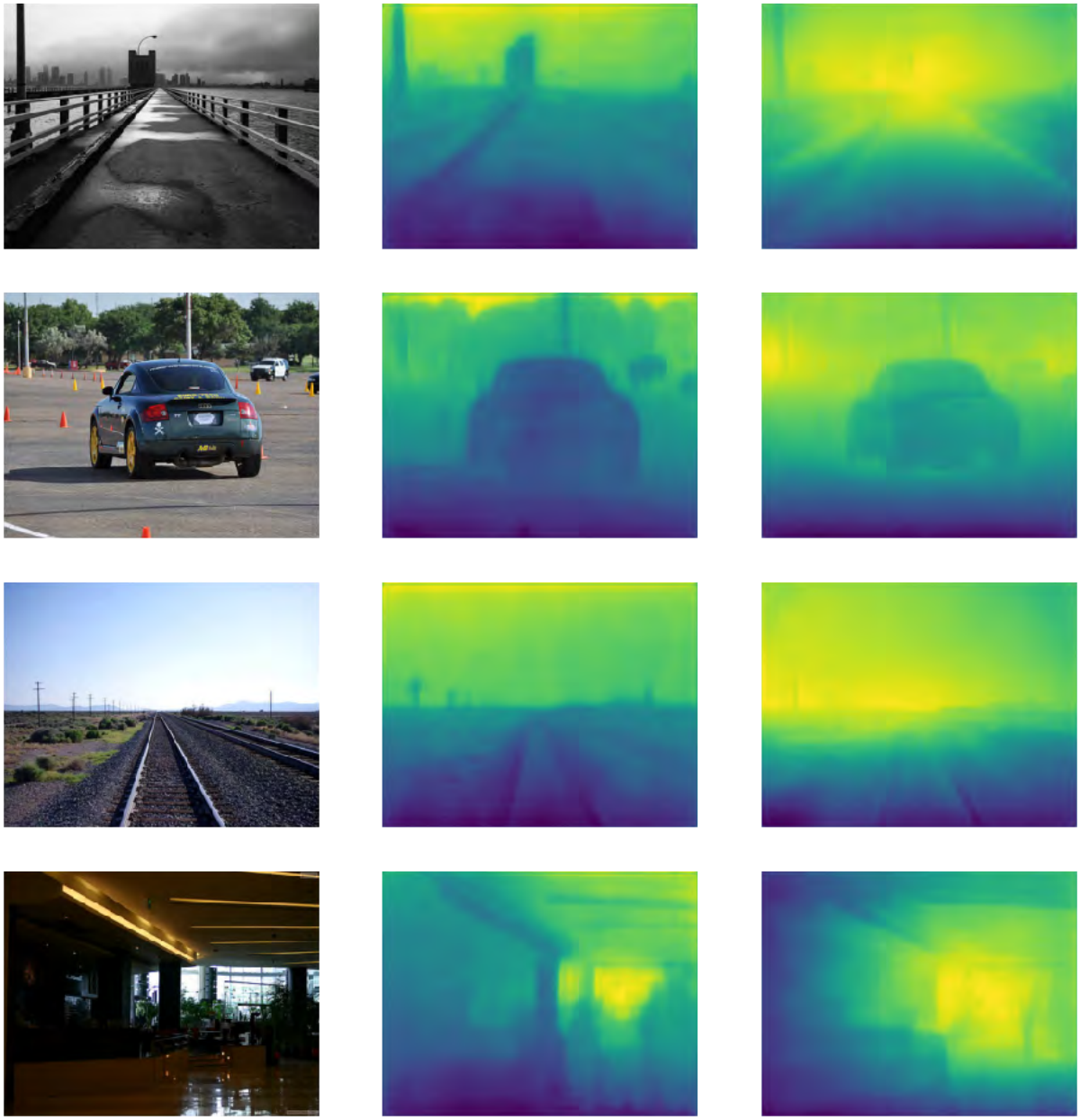


Рис. 9: Результаты на изображениях с выраженной перспективой. Слева направо: оригинальное изображение, глубина полученная при помощи нейросети до обучения, и после дообучения

5.2 Функция потерь контента с попиксельным взвешиванием

В качестве карты весов использовалась карта глубин (кроме примера для сгенерированного градиента на рис 10). Для более равномерного наложения стиля, веса изменяются от меньшего к большему по логарифмической шкале. В модельном примере (рис. 10) вес меняется с 1 до 10 слева направо. Видно что правая половина

изображения почти не затронута стилем, тогда как слева стиль гораздо более интенсивен, при этом изменение стиля происходит плавно.

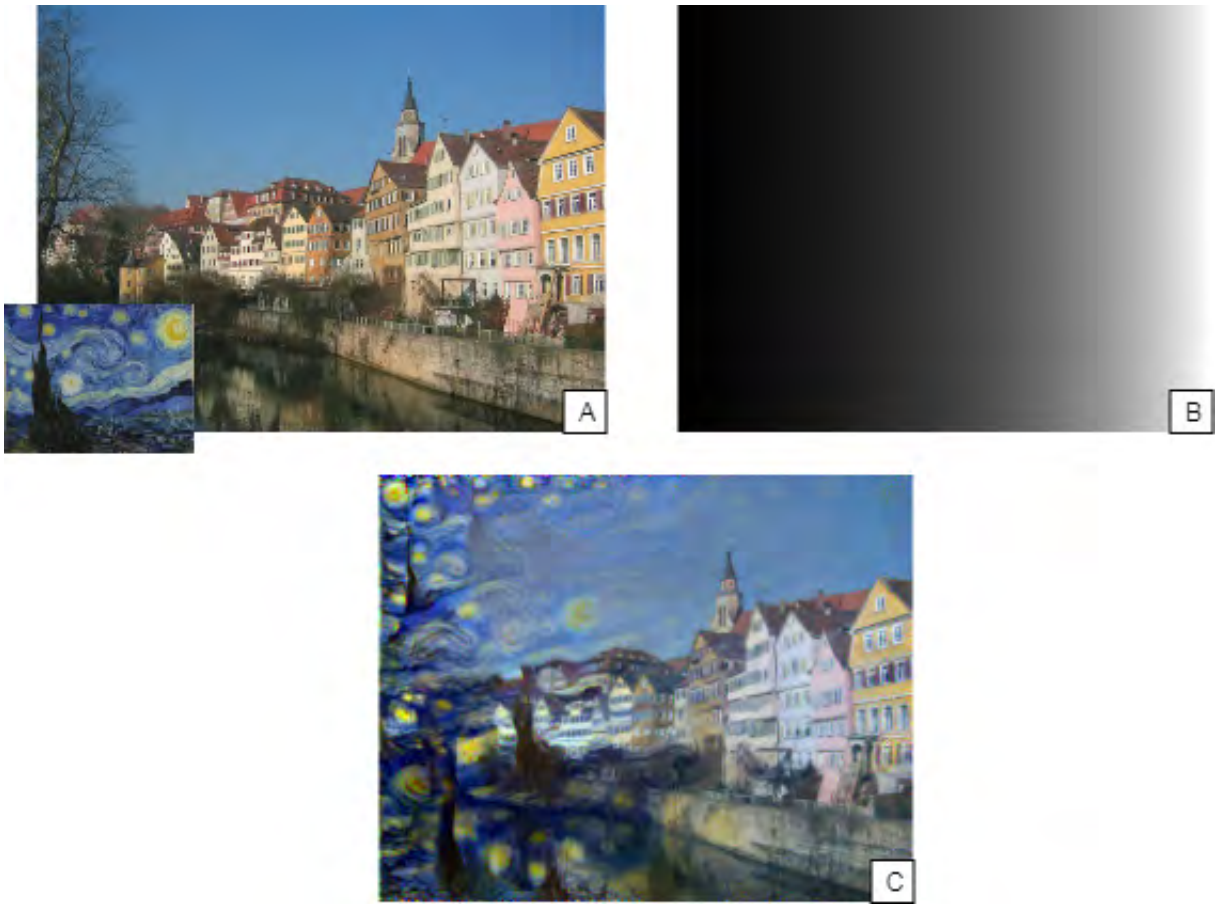


Рис. 10: Пример взвешенного переноса стиля на модельных данных. a) контент и стиль b) карта весов c) стилизованное изображение

5.3 Сравнение методов

Ниже сравниваются методы: обычный перенос стиля [2], перенос стиля с картой весов, перенос стиля с функцией потерь глубины [1], и комбинация этих методов. Попиксельное взвешивание функции потерь контента производилось следующим образом: для изображения вычисляется глубина при помощи дообученной нейросети, после чего ближайшему пикселю присваевается значение $\alpha_{ij} = \alpha_{max}$, самому дальнему пикселю значение $\alpha_{ij} = \alpha_{min}$, остальные значения задаются пропорционально, по логарифмической шкале между двумя этими значениями, таким образом получаем что на удаленных частях изображения контент минимален, и больше применяется стиль, и наоборот, ближние объекты более хорошо прорисованы, и лучше сохранен контент.

Под комбинацией методов подразумевается одновременное использование попиксельного взвешивания на штраф контента и введения штрафа на глубину. Для всех экспериментов были использованы значения $\alpha_{min} = 1, \alpha_{max} = 5, \beta = 1000, \gamma = 20$, где β — вес штрафа стиля, и γ — вес штрафа глубины. Для обычного переноса стиля вес контента выбирался равным α_{min}.

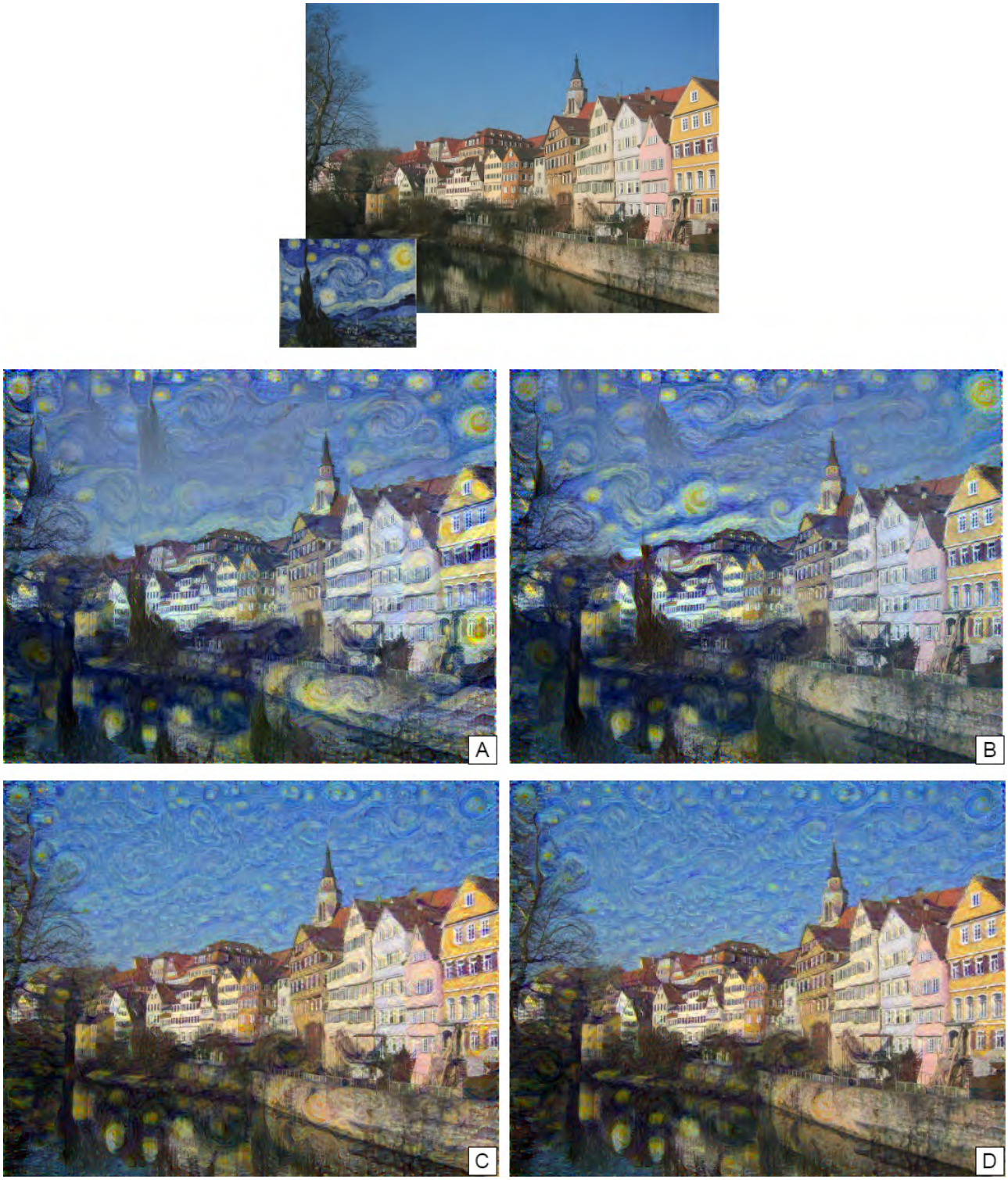


Рис. 11: a) Обычная стилизация [2], b) предложенный метод, c) стилизация с учетом глубины [1], c) комбинация методов.

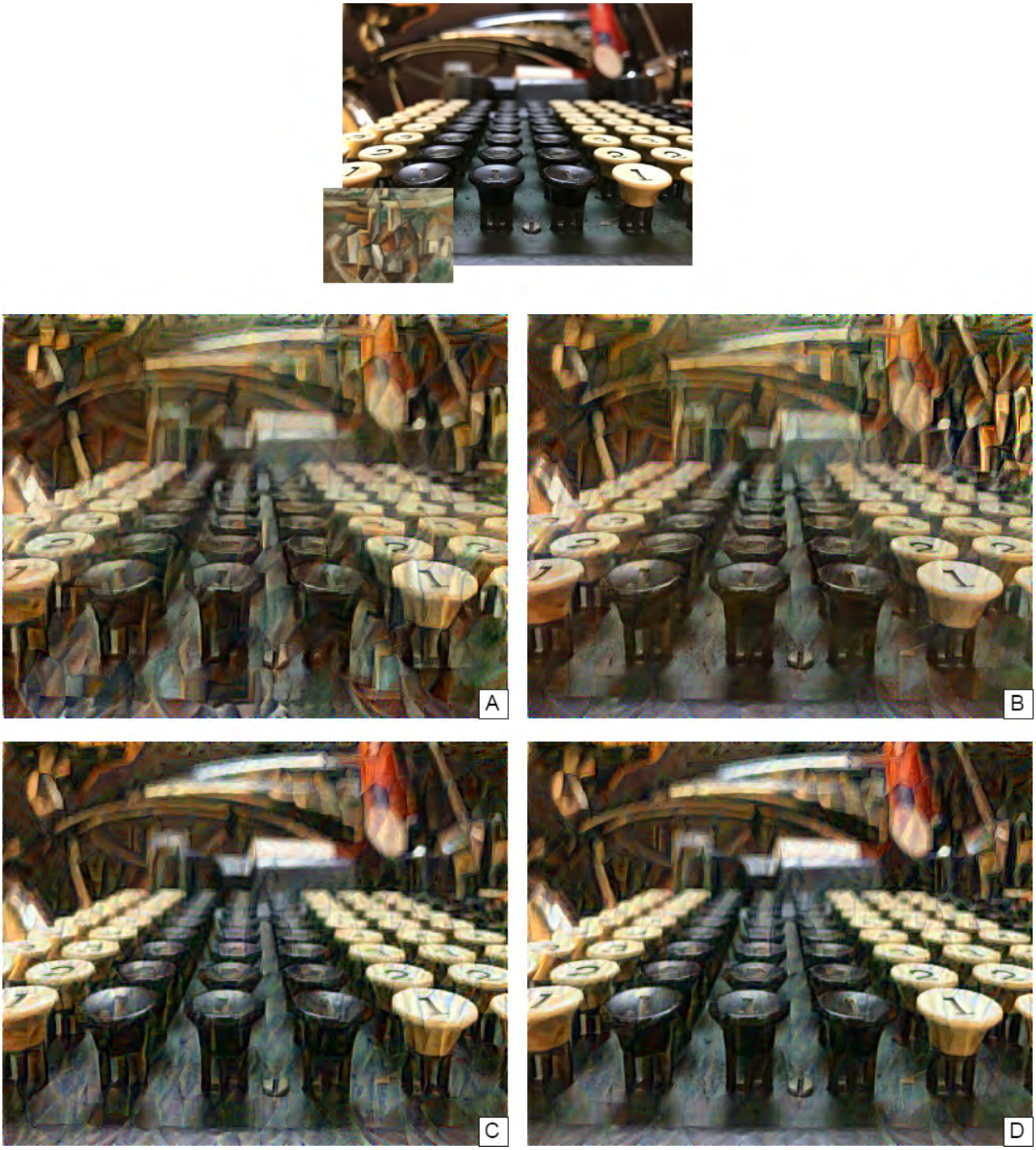


Рис. 12: a) Обычная стилизация [2], b) предложенный метод, c) стилизация с учетом глубины [1], c) комбинация методов.

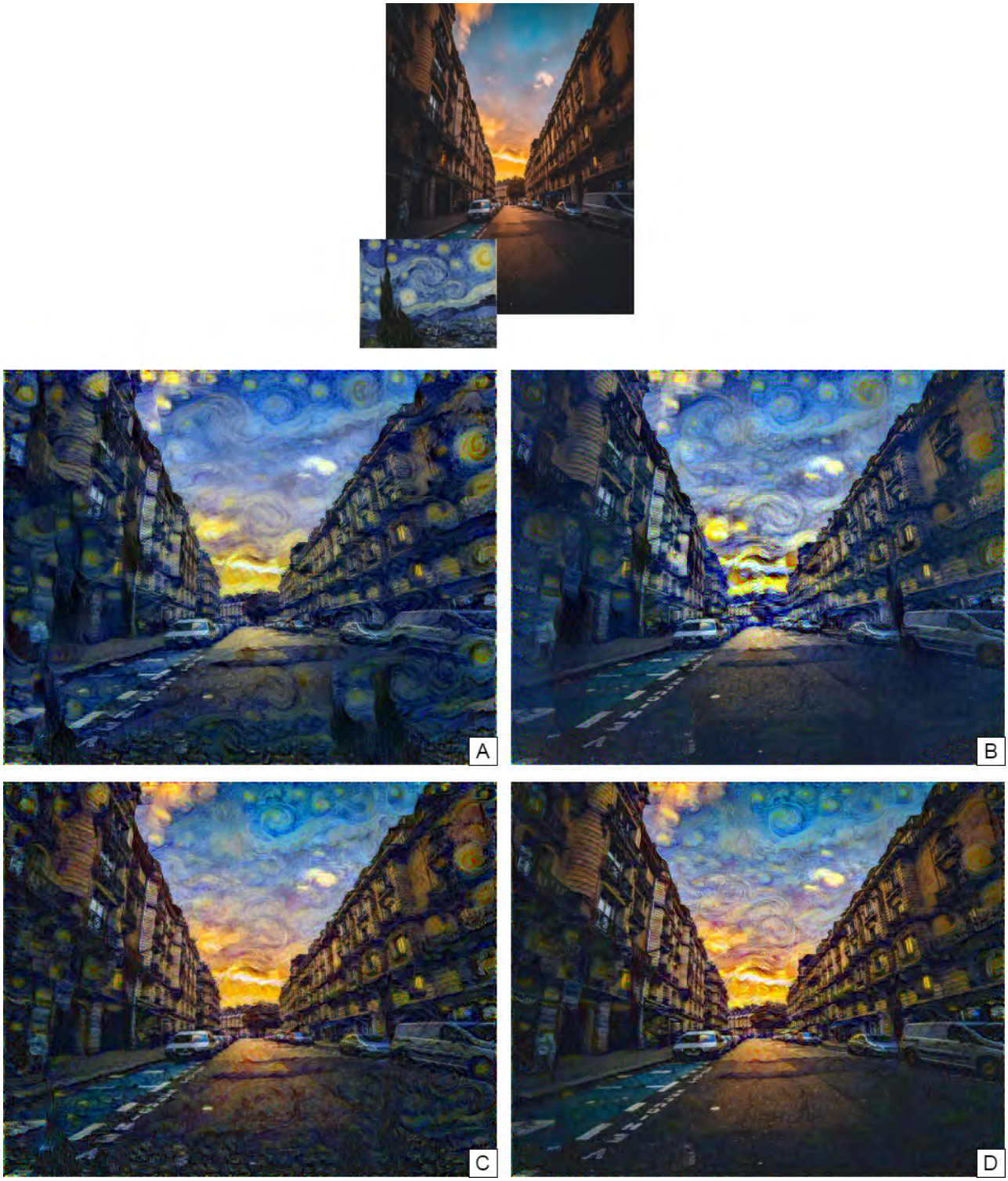


Рис. 13: a) Обычная стилизация [2], b) предложенный метод, c) стилизация с учетом глубины [1], c) комбинация методов.

Видно что за счет штрафа за сильную стилизацию ближайших объектов, изображения выглядят лучше чем совсем плоские фотографии, тем не менее такой метод

учета глубины явно проигрывает методу с функцией потерь для глубины. Тем не менее на фотографиях, где методы скомбинированны можно видеть что передние объекты становятся чуть четче, что визуально улучшает изображение. Особенно хорошо этот эффект виден на последней фотографии, где более гладкая дорога на переднем плане сильно контрастирует с небом и уходящей вдаль части улицы, на которые стиль сильнее действует.

6 Заключение

Результатами данной работы являются:

- Реализация нейросети оценивающей глубину изображения на python под библиотеку pytorch.
- Дообучение нейросети на наборе данных KITTI. Результатом дообучения стала нейросеть, которая более плавно распознает глубину, показывает лучшие результаты на открытом пространстве типа дорог, полей и т.д.
- Реализация метода попиксельного взвешивания контента, и метода учета глубины с его помощью.
- Сравнение методов: обычного переноса стиля, переноса стиля с учетом глубины, при помощи функции потерь глубины, переноса стиля методом попиксельного взвешивания контента, и комбинации двух последних методов.

  Возможное направление дальнейшего исследования: в работе везде для экспериментов был использован классический перенос стиля [2] (в связи с ограничением на доступную вычислительную мощность). Предлагатеся дополнительно исследовать работу методов в случае быстрого переноса стиля [3], в котором необходимо для каждого стиля обучить сеть преобразующую случайные изображения-контенты, в стилизованное изображение (соответственно все модификации легко переносятся на случай быстрого переноса стиля).

Список литературы

[1] Depth-aware neural style transfer / X.-C. Liu, M.-M. Cheng, Y.-K. Lai, P. L. Rosin // Proceedings of the Symposium on Non-Photorealistic Animation and Rendering / ACM. — 2017. — P. 4.

[2] *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. — Pp. 2414–2423.

[3] *Johnson J., Alahi A., Fei-Fei L.* Perceptual losses for real-time style transfer and super-resolution // European Conference on Computer Vision. — 2016.

[4] Single-image depth perception in the wild / W. Chen, Z. Fu, D. Yang, J. Deng // Advances in Neural Information Processing Systems. — 2016. — Pp. 730–738.

[5] Sparsity invariant cnns / J. Uhrig, N. Schneider, L. Schneider et al. // International Conference on 3D Vision (3DV). — 2017.