Промежуточный отчёт по воспроисведению результатов статьи WESPE

Алексей Калинов
Факультет компьютерных наук
Высшая школа экономики
aakalinov@edu.hse.ru

Денис Беляков
Факультет компьютерных наук
Высшая школа экономики
dobelyakov@edu.hse.ru

Abstract

В данном отчёте описывается процесс воспроизведения статьи WESPE Ignatov et al. [2018]. Сначала приводится краткое описание задачи и методов её решения, предлагаемых авторами. Затем излагаются эксперементы, подлежащие воспроизведению, и отличия при их проведении от оригинальной статьи.

1 Постановка задачи

В работе рассматривается задача автоматического улучшения качества изображений, снятых на камеры современных мобильных телефонов, до качества, получаемого с помощью цифрового зеркального фотоаппарата. В качестве входных данных выступают два набора изображений. Первый получен с помощью бюждетной камеры, встроенной в современный сматрфон, второй с помощью более дорогой и громоздкой цифровой камеры. Стоит заметить, что наборы могут быть непарными, то есть совсем не требуется, чтобы для каждой фотографии в первом наборе был такой же снимок во втором, то есть задача формулируется в weakly supervised манере.

Имея вышеописанные данные, необходимо предоставить алгоритм, способный переводить изображения из первого набора данных в изображения, соответствующие по качеству второму датасету. Так как понятие качества изображенийя сложно и математически строгие критерии его оценки зачастую не могут охватить все его аспекты, применяется adversarial подход, при котором алгоритм оценки обучается совместно с алгоритмом преобразования.

2 Набор данных DPED

В оригинальной статье авторы используют собранный лично ими набор данных DPED: DSLR Photo Enhancement Dataset Ignatov et al. [2017], включающий в себя изображения с трех сматрфонов: sony, iphone и blackberry, а также изображения с цифровой камеры. Большинство из них запечатлевают городские пейзажи, в том числе, дома, машины и прохожих. Для получения изображений была использована специальная установка, на который были закреплены все устройства. Захват изображений управлялся с помощью внешнего микроконтроллера и выполнялся синхронно. Таким образом, выборка состоит из попарно соответсвующих изображений. Тем не менее, при проведении экспериментов эта связь специально не использовалась.

Preprint. Work in progress.

3 Предложеный алгоритм и его реализация

В своей статье авторы предлагают воспользоваться генеративными конкурентными сетями для решения поставленной задачи. В частности, архитектура состоит из полносвёрточного генератора и нескольких дискриминаторов. Генератор содержит в себе 4 resnet блока со свёртками $3 \times 3 \times 64$ и layer norm слоями, и завершается ещё тремя полносвёрточными слоями. В качестве активаций везде, кроме последнего слоя, используются ReLU.

Дискриминатор также является полносвёрточной сетью с большими свёртками $11 \times 11, 5 \times 5$ в начале и обычными свёрточными слоями 3×3 далее. После каждого слоя, начиная со второго используется layer norm. В качестве активаций выбраны LeakyReLU. Заканчивается дискриминатор полносвязным слоем с сигмоидальной функцией активации для классификации изображения.

Интересные особенности:

1. Выход последнего слоя генератора интересен тем, что во всех реализациях используют отмасштабированный гиперболический тангенс, а именно $y = 0.58 \tanh(x) + 0.5$, где y — выход последнего слоя, а x — результат после применения свёрток на последнем слое. Кажется, что правильно масштабировать, уменьшая выход тангенса ровно вдвое, но в дальнейших экспериментах использовалась именно такая функция.
2. В реализации на github выход последнего слоя дискриминатора предсказывает вероятности сразу двух классов $fake, 1 - fake$. При воспроизведении результатов также произоводилось предсказание единственного класса $fake$, так как вероятность реального класса вычисляется из вероятности $fake$ напрямую и не требует отдельного предсказания моделью. В проведенных экспериментах различий между предсказанием одного класса и предсказанием двух классов не наблюдалось.

Функция потерь для обучения генератора состоит их четырех частей: color loss, texture loss, reconstruction loss и tv loss. Первые две из них вычисляются отдельным дискриминатором. Color loss вычисляется по сглаженным гауссовским фильтром цветным изображениям-выходам генератора и отвечает, по сути, за содержание изображения, а texture loss вычисляется по чёрно-белым изображениям и отвечает за оценку текстур на изображении. Perception loss вычисляются с помощью ещё одного генератора такой же структуры. Штраф накладывается, если полученное изображение не совпадает с исходным в пространстве признаков сети VGG19. Total variation loss является регуляризатором и вычисляется как сумма норм градиентов сгенерированного изображения. Это заставляет соседние пиксели быть схожими и всю картинку гладкой.

Все модели оптимизируются с помощью Adam. В ходе экспериментов размер шага варьировался от 0.0001 до 0.0005. Количество итераций составляло 20 тысяч. Размер батча менялся от 2 до 44. Соотношение реальных изображений из цифровой камеры и улучшенных изображений с помощью генератора в каждом батче составляет $1 : 1$.

Заметим, что и в оригинальной статье, и при её воспроизведении используются не целые изображения, а вырезанные из них куски размером 100×100. Они предоставляются авторами статьи, так что для более точного соответствия используются именно они. На рисунке 1 показаны примеры изображений. Видно, что изображения со смартфона более зернистые и не такие чёткие.

Модель была реализована с использованием фреймворка PyTorch.

4 Эксперименты

Основным результатом, который в первую очередь хотелось воспроизвести, является получение реалистичных изображений. К сожалению, получить такие же отличные результаты, как и в оригинале, не удалось, но мы опишем, какие приёмы позволили получить хоть какую-то генерацию по сравнению с промежуточным отчётом.

Рис. 1: Пример изображений в наборе данных DPED. Верхний ряд: изображения, полученные с помощью камеры телефона sony. Нижний ряд: изображений, снятые с использованием цифровой камеры.

4.1 Промежуточные эксперименты

Данные эксперименты выполнялись только с двумя функциями потерь: color loss и texture loss и соответствуют промежуточному отчёту. Они приведены здесь для полноты картины.

После не очень большого количества итераций генератор схлопывается и начинает генерировать изображения постоянного цвета. При этом, его функция потерь продолжает убывать. Также после некоторого количества итераций обучения лосс дискриминатора резко возрастает и впоследствии не восстанавливается.

На рисунках 2 – 4 приведены графики функции потерь генератора и дискриминатора от количества итераций обучения, сопровождающиеся примерами работы моделей. На каждом рисунке верхний ряд изображений получен цифровой камерой, а нижний ряд содержит сгенерированные изображения.

Рисунок 2 показывает одну полную эпоху обучения при размере обучающей выборки в 30000 пар. На нём видны резкие скачки color loss и texture loss дискриминатора и схлопывание генератора к генерации однотонных серых изображений.

На рисунке 3 можно наблюдать осмысленные результаты работы модели при монотонном уменьшении лосса генератора и колебаний color loss и texture loss дискриминатора. Данный рисунок получен в самом начале обучения, то есть изначально модель пытается сгенерировать похожие изображения, а уже потом схлопывается в постоянный цвет.

Рисунок 4 показывает полный процесс обучения, состоящий из четырех эпох с увеличенным размером обучающей выборки до 150000 пар. Наблюдаем резкий скачок color loss и texture loss дискриминатора и схлопывание генератора к генерации однотонных красных изображений

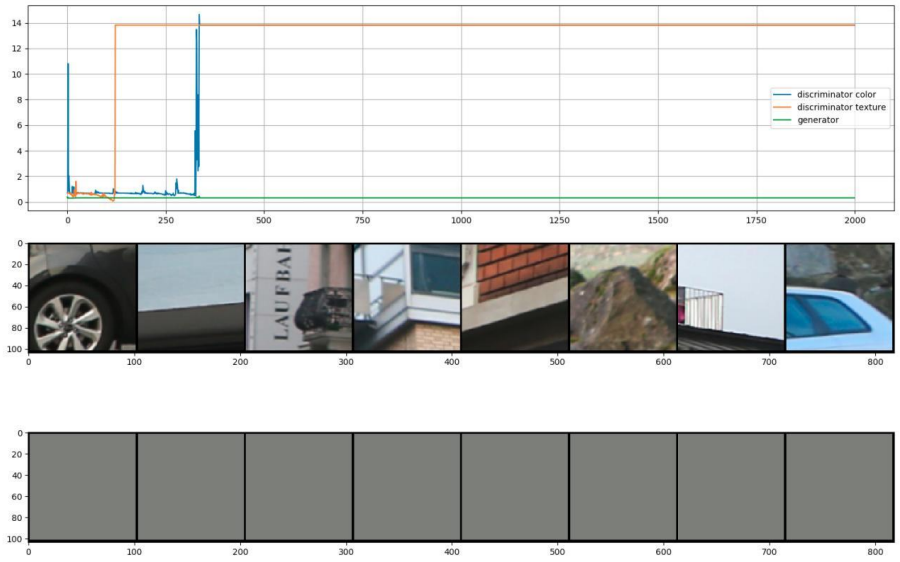

Рис. 2: Обучение с двумя компонентами функции потерь. График функций потерь, целевые и сгенерированые изображения. Итераций попеременного обучения: 2000. Размер батча: 30.

Эксперименты с изменением соотношения между количеством шагов оптимизации генератора и дискриминатора, величины шага Adam и количеством изображений в батче данную ситуацию не исправляли.

4.2 Финальные эксперименты

Ясно, что основной целью на данный момент являлось получение хоть каких-то изображений, соответствующих заявленным в статье.

Первым шагом для достижения цели являлось добавление total variation loss и content loss для обучения генератора. Данный шаг практически не изменил ситуацию, генерация оставалась всё такой же плохой.

После нескольких экспериментов, мы заметили, что дискриминатор почти всегда слишком плохо обучается, и в какой-то момент не может догнать генератор. Оперевшись на tensorflow реализацию этой статьи, мы изменили batch norm слой на instance norm, а также изменили инициализацию весов на kaiming uniform из статьи He et al. [2015].

После таких изменений дискриминатор стал обучаться действительно лучше, даже чересчур хорошо, так как его лосс стремился к сходимости около нуля, а генератор начинал генерировать однотонные изображения, то есть теперь генератор страдал от недообучения. Изменения learning rate не исправили ситуацию. Пример обучения и результатов представлены на графике 5.

Имея такие результаты на руках, было решено изменить соотношение итераций обучения в пользу генератора. На маленьком количестве итераций это, казалось, исправило ситуацию, но при обучении на сервере с большим батчом (здесь и далее: 24-30) оказалось, что уже генератор обучается слишком хорошо, а дискриминатор застревал на случайном предсказании (значение функции потерь около $2 \log 2$). Данная ситуация изображена на рисунке 6.

Соответственно, теперь необходимо было решить противоположную задачу и усложнить дискриминатор. Для этого было произведено увеличение количества каналов в свертках. Но тогда при балансе обучения 1:1 и 2:1 в пользу генератора проблема с коллапсом

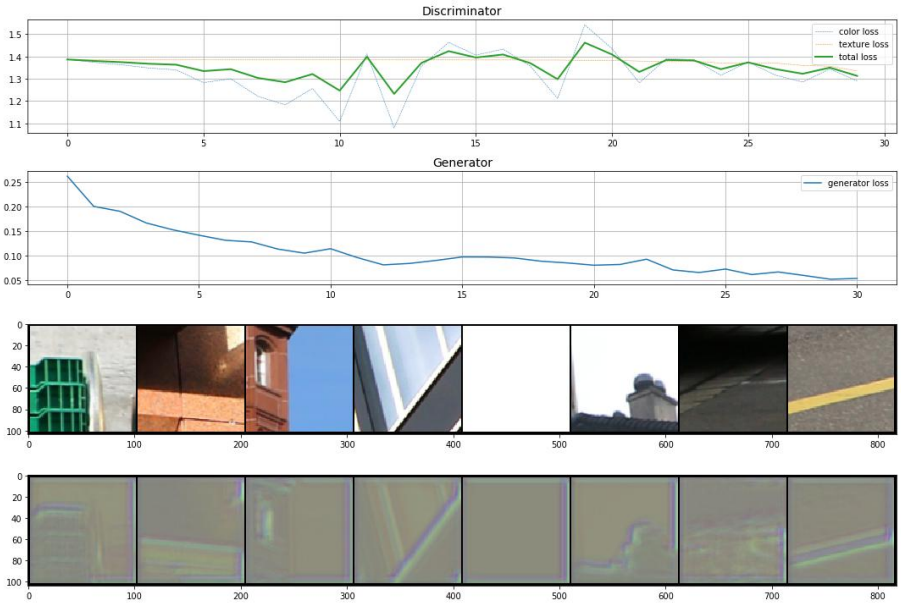

Рис. 3: Обучение с двумя компонентами функции потерь. График функций потерь, целевые и сгенерированые изображения. Итераций попеременного обучения: 30. Размер батча: 30.

генератора вернулась. Пример обучения и картинок в данной постановке указан на рисунке 7

Воодушевившись найденными в сети трюками с GANами, появилась идея разморозить градиенты дискриминатора и делать 1 шаг генератора на несколько шагов дискриминатора. На маленькой выборке результаты были многообещающими, но, в итоге, обучение тоже свелось к однотонному предсказанию еще где-то на 1000-й итерации и больше не улучшалось.

Не оставляя надежду получить хоть какой-то вменяемый результат, была добавлена динамическая схема подбора баланса обучения между генератором и дискриминаторами. Два пороговых параметра настраивались, исходя из среднего по последним значениям функции потерь. Логика первого параметра заключается в том, чтобы не позволять дискриминатору достигать совсем малых значений функции потерь, после которых эмпирически генератор всегда сходится к однотонным предсказаниям. Если средний лосс дискриминаторов этот параметр превосходит, устанавливается соотношение шагов обучения 3 : 1 в пользу генератора. Второй параметр вводит ограничение сверху на значение функции потерь генератора. Если среднее по пяти последним лоссам генератора превосходит его пороговый параметр (в данном случае выбран порог 0.015), обучение дискриминатора останавливается, и обучается только генератор (до стабильного снижения лосса). Данное значение выбрано таким, так как в среднем ранее при значениях около 0.02 и неизменном расписании обучения генератор сходился к генерированию однотонных изображений и больше не восстанавливался.

Итоговые результаты изображены на рисунках 8, 9. Несмотря на то, что они далеки от идеальных, тем не менее, на них можно различить черты исходных изображений.

Список литературы

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision, pages 1026–1034, 2015.

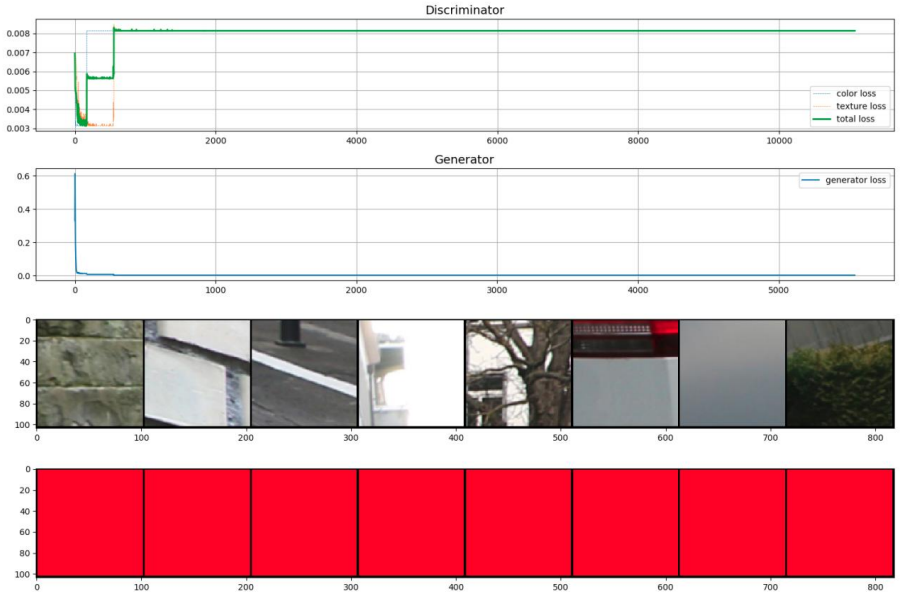

Рис. 4: Обучение с двумя компонентами функции потерь. График функций потерь, целевые и сгенерированые изображения. Итераций попеременного обучения: 12000. Размер батча: 44.

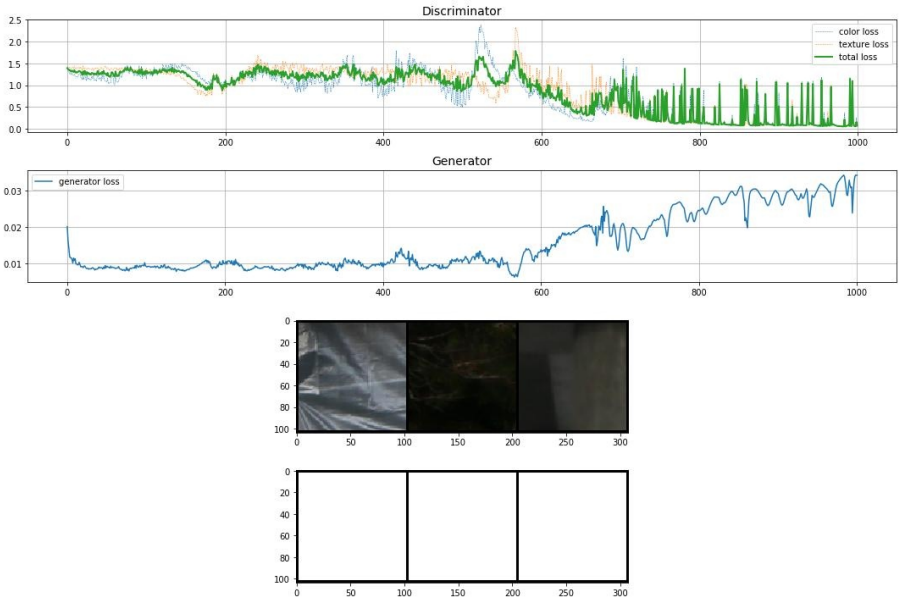

Рис. 5: Полное обучение. Пример недообучения генератора. Появляется генерация однотонных изображений и функция потерь на постоянном уровне.

A. Ignatov, N. Kobyshev, R. Timofte, K. Vanhoey, and L. Van Gool. Dslr-quality photos on mobile devices with deep convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, pages 3277–3285, 2017.

A. Ignatov, N. Kobyshev, R. Timofte, K. Vanhoey, and L. Van Gool. Wespe: weakly supervised photo enhancer for digital cameras. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 691–700, 2018.

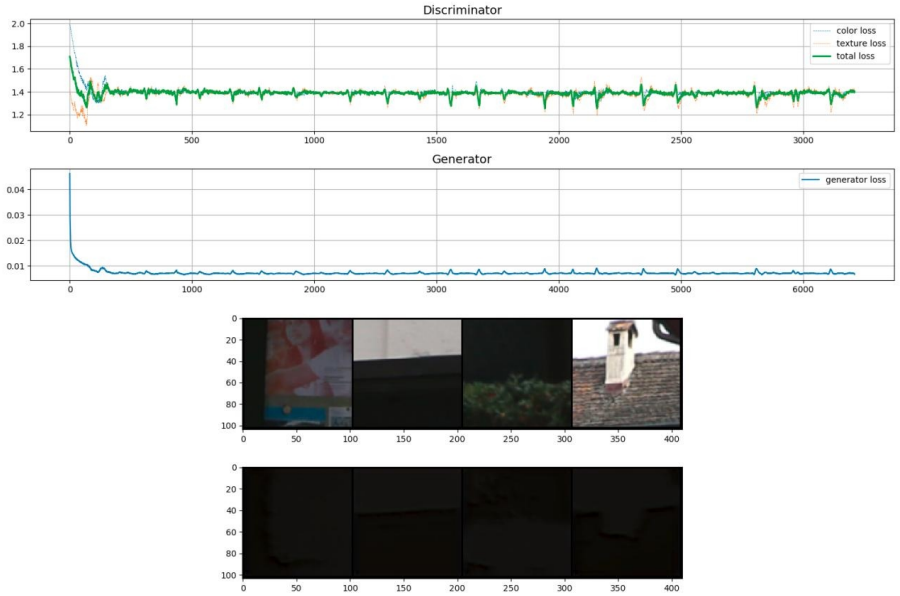

Рис. 6: Полное обучение. Пример слабого дискриминатора. Функция потерь генератора почти нулевая, в то время как функция потерь дискриминатора принимает характерное значение $2 \log 2$.

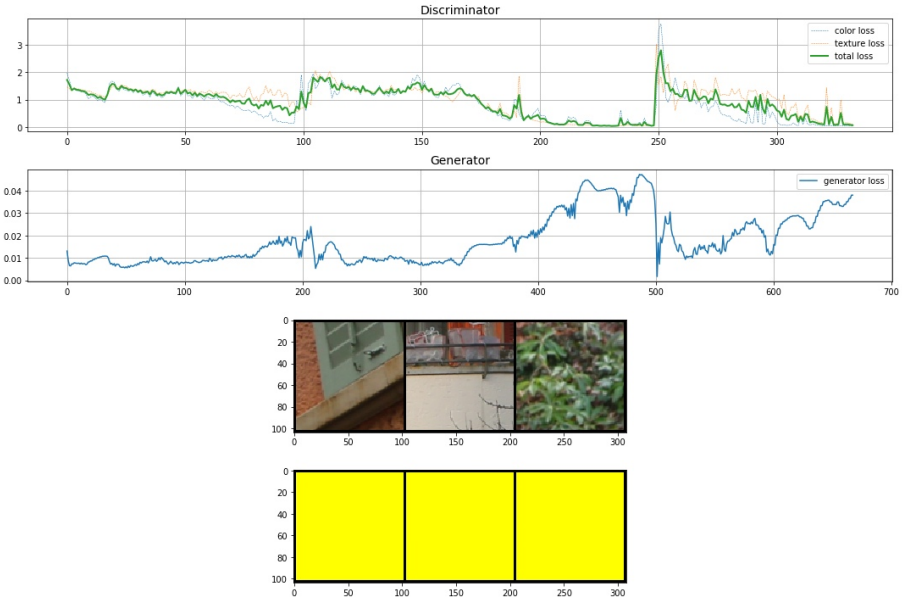

Рис. 7: Полное обучение. Усложненный генератор и дискриминатор. Снова видна проблема недообучения генератора.

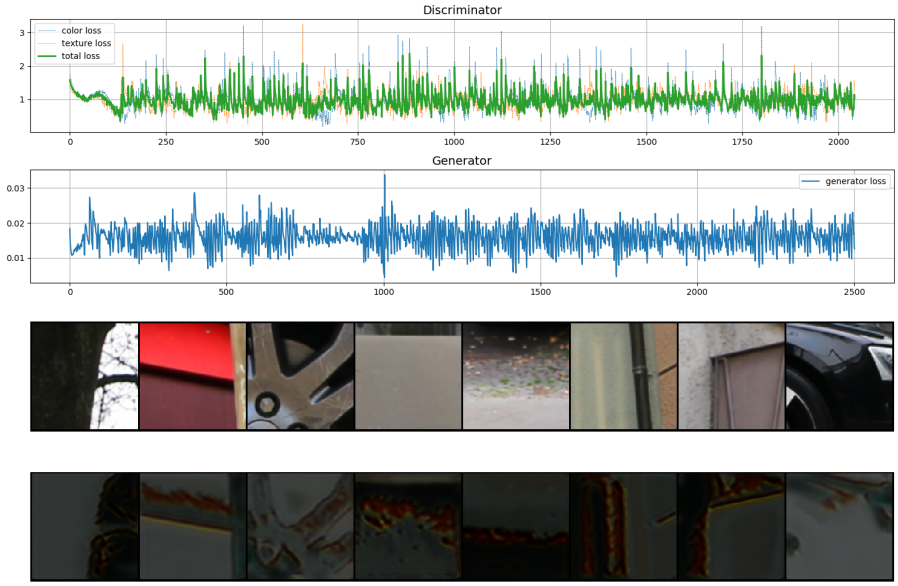

Рис. 8: Финальный результат. Усложненный генератор и дискриминатор с динамическим подбором количества итераций. Полученные изображения не являются идеальными, но похожи на исходные изображения.

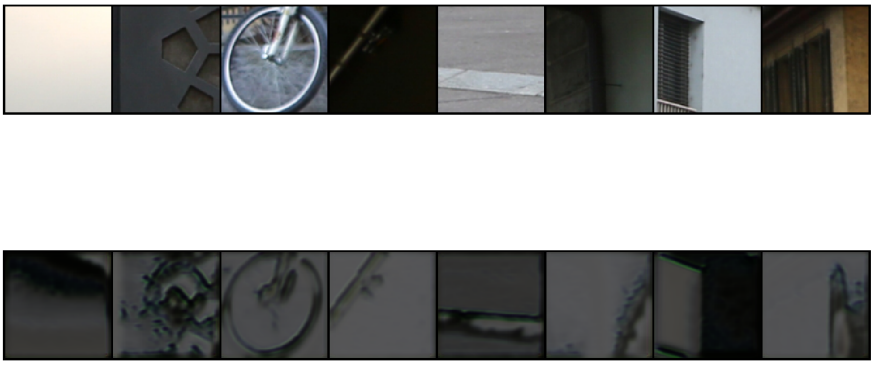

Рис. 9: Финальный результат. Полученные изображения не являются идеальными, но похожи на исходные изображения.