

**ДИПЛОМНАЯ РАБОТА**

**на тему:**

**«Построение и обучение условной генеративной состязательной сети (cGAN)**

**для задач трансляции изображений»**

***Выполнил:***

Студент: Томилов Константин

***Проверил:***

Преподаватель: Баданина Наталья

г. Москва

2022

1. **Постановка задачи.**
   1. ***Описание исходной задачи***

Многие задачи в обработке изображений, компьютерной графике и компьютерном зрении связаны с преобразованием входного изображения в соответствующее выходное изображение.

По аналогии с тем, что один и тот же текст можно представить на английском или русском языке, так и изображение можно представить в RGB-цветах, в градиентах, в виде карты границ объектов, карты семантических меток и т.д. Традиционно каждая из этих задач выполняется отдельным специализированным приложением.

Вместе, с тем для задач трансляции изображения из одной области в другую, получая в итоге изображение со стилем (характеристиками) последней (рис.1), требуется иное решение.



Рис.1

Сформулируем определение задачи трансляции изображения (англ. Image-to-image translation) – это задача из области [компьютерного зрения](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D0%B7%D1%80%D0%B5%D0%BD%D0%B8%D0%B5), цель которой состоит в том, чтобы научиться строить соответствия между входным и выходным изображениями, используя тренировочные данные.

Для трансляции изображений подходят свёрточные нейросети (CNN), обученные минимизировать функцию потерь (мера расхождения между истинным значением оцениваемого параметра и оценкой параметра). Но, для нейросети необходимо показать, что конкретно нужно минимизировать, например, «минимизировать евклидово расстояние между предсказанными и настоящими пикселями». Так как, евклидово расстояние минимизируется путем усреднения всех правдоподобные результаты, то это ведёт к генерации смазанных изображений (рис. 2).

В этой связи, целесообразно и проще ставить нейросети высокоуровневые задачи (например, «сгенерировать изображение, неотличимое от реальности»), а затем автоматически обучить нейросеть для минимизации функции потерь, которая наилучшим образом выполняет поставленную задачу.

Для данных целей подходят генеративные состязательные сети (GAN). Впервые такие сети были представлены Иэном Гудфеллоу в 2014 году. Сеть GAN обучает функцию потерь, задачей которой является классифицировать изображение как «настоящее» или «поддельное», одновременно тренируя генеративную модель, чтобы минимизировать эту функцию. Здесь размытые изображения не пройдут проверку классификации как «настоящие». Генеративная модель G строит приближение распределения данных, а дискриминативная модель D оценивает вероятность, что образец пришел из тренировочных данных, а не сгенерирован моделью G (рис. 3). Обучение для модели G заключается в максимизации вероятности ошибки дискриминатора D.

Рис.2

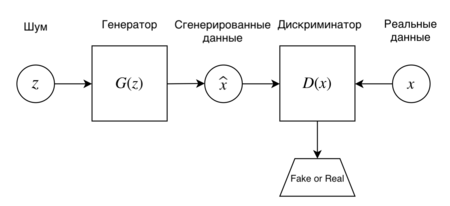


Рис. 3

В зависимости типа тренировочных данных и желаемого результата для генерации реалистичных изображений могут применяться различные варианты GAN.

* CycleGAN: меняет изображения c одного домена на другой, например, лошадей на зебр;
* SRGAN: создает изображения с высоким разрешением из более низкого разрешения;
* StackGAN: создает изображения по заданному тексту;
* MidiNet: генерирует последовательность нот, таким образом, создает мелодию.

***Задачей дипломной работы*** является построение и обучение условной генеративно-состязательной сети (cGAN – Conditional Genereative Adversarial Nets) для трансляции изображений с семантической окраской (Pix2Pix).

Основная цель данной дипломной работы - продемонстрировать, что такая глубокая нейронная сеть способна выполнять задачи построения реалистичных изображений из семантических изображений с приемлемым результатом.

Как и в других генеративных сетях, в cGAN нейросети воюют между собой. Одна из них (генератор) пытается создать фальшивое изображение, чтобы обмануть другую (дискриминатор). Со временем генератор обучается всё лучше обманывать дискриминатор, то есть генерировать более реалистичные изображения. В отличие от обычных GAN, в Pix2Pix одновременно и дискриминатор, и генератор имеют доступ к исходному изображению (рис. 4).

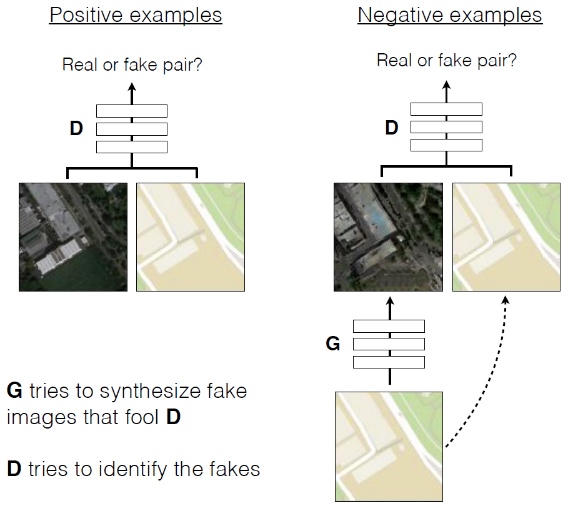


Рис. 4

* 1. ***Актуальность задачи, ее место в предметной области***

Преобразование изображения в изображение с помощью условных состязательных сетей можно применять для широкого круга задач, включая (рис. 5):

* синтез фотографий из карт меток;
* создание раскрашенных фотографий из черно-белых изображений;
* преобразование фотографий Google Maps в аэрофотоснимки;
* преобразование эскизов в фотографии.





Рис. 5

1. **Анализ**

***2.1. Исследование аналогичных решений***

Задачи перевода изображения в изображение часто формулируются как попиксельная классификация или регрессия (например, [7, 8, 9, 10, 11]). В этих формулировках выходное пространство рассматривается как «неструктурированное» в том смысле, что каждый выходной пиксель считается условно независимым от всех остальных пикселей входного изображения. Вместо этого условные GAN изучают структурированные потери. Структурированные потери наказывают совместную конфигурацию выхода.

В литературе описаны потери такого рода с помощью методов, включающих условные случайные поля [10], метрику SSIM [12], сопоставление признаков [15], непараметрические потери [14], сверточный псевдоаприорный анализ [16] и потери, основанные на сопоставлении ковариационной статистики [17].

В cGAN потери теоретически могут наказывать любую возможную структуру, которая имеет различия между выходом и целью. В описанных работах исследователи обусловливали GAN дискретными метками [13], текстом [18] и изображениями [5]. Модели с условными изображениями занимаются прогнозированием изображения на основе карты нормалей [19], прогнозированием будущего кадра [20], генерацией фотографий продукта [21] и генерацией изображений на основе разреженных аннотаций [22]. Несколько других статей также использовали GAN для сопоставления изображений с изображениями, но только безоговорочно применяли GAN, полагаясь на другие термины (такие как регрессия L2), чтобы заставить вывод быть обусловленным вводом. В этих работах были достигнуты впечатляющие результаты по рисованию [23], предсказанию будущего состояния [24], манипулированию изображениями в соответствии с пользовательскими ограничениями [25], переносу стиля [26] и сверхразрешению [27]. Каждый из методов был адаптирован для конкретного применения.

Применяемая в дипломной работе модель отличается от указанных работ вариантами архитектуры генератора и дискриминатора: для генератора используется архитектура на основе «U-Net» [28], а для дискриминатора используется сверточный классификатор «PatchGAN», который наказывает структуру только в масштабе фрагментов изображения. Аналогичная архитектура PatchGAN была ранее предложена в [26] для сбора статистики локального стиля.

В исследованиях по аналогичной структуре cGAN [5] показана эффективность модели, используемой в дипломной работе, в U-net в сравнении с энкодер-декодер, функции потерь L1+cGAN в сравнении L1, cGAN, свёртки 70х70 для PatchGAN, в сравнении со свёртками, 1х1, 286х286.

* 1. ***Предварительный анализ данных***

В дипломной работе используется набор данных «Cityscapes».

Набор создан исследовательской группой «Berkeley AI Research».

Данные «Cityscapes» содержат помеченные изображения, снятые из автомобилей в Германии.

Набор данных содержит 2975 файлов обучающих изображений (train) и 500 файлов проверочных изображений (val).

Каждый файл изображения имеет размер 256x512 пикселей, и представляет собой композицию, в которой исходная фотография находится в левой половине изображения, а изображение с меткой (результат семантической сегментации) - в правой половине (рис. 6).



Рис. 6

Ссылка на официальный сайт: <https://www.cityscapes-dataset.com/>

* 1. ***Алгоритмы и техники, применяемые для решения задачи***

В работе применяются библиотеки: Python 3.8, TensorFlow 2.4, Keras 2.4, Mathplotlib 3 и др.

Алгоритмы:

Свёрточная нейронная сеть – U-Net;

Свёрточный классификатор – PatchGAN;

Функции активации – Relu, LeaklyRelu, Tanh;

Оптимизация целевой функции – стохастический градиентный спуск Adam («метод адаптивной оценки моментов», англ. Adaptive Moment Estimation).

***Cсылка на код программы:*** [***https://github.com/constantintom/diplom/blob/main/City.ipynb***](https://github.com/constantintom/diplom/blob/main/City.ipynb)

1. **Методика решения**

***3.1. Преобразование данных***

*Подготовка и преобразование тренировочных данных:*

- Отделяем реальные изображения улиц от семантических изображений - все они будут иметь размер 256 x 256;

- Преобразуем файлы изображений в 2 тензора изображений;

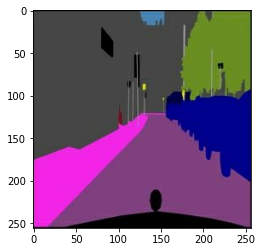
 

Рис. 7

- Применяем «случайное дрожание»: изменяем размер каждого изображения 256x256 на большую высоту и ширину - 286x286, затем произвольно обрезаем его обратно до 256x256.

Применяем «случайное зеркальное отображение»: произвольное отражение изображения по горизонтали, т.е. слева направо.

Нормализуем изображения в диапазоне [-1, 1].

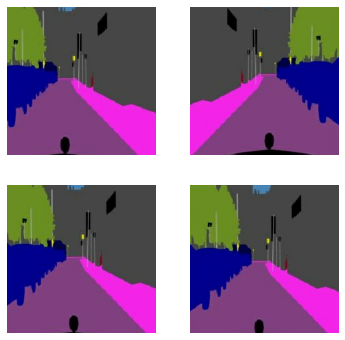


Рис. 8

*Подготовка и преобразование тестовых данных:*

Устанавливаем размер 256x256

Нормализуем изображения в диапазоне [-1, 1].

***3.2. Описание итоговой модели***

Условная генеративная состязательная сеть (сGAN) реализует архитектуру, где для генератора используется нейронная сеть U-Net, а для дискриминатора используется свёрточный классификатор PatchGAN, который штрафует алгоритм на уровне участков изображения.

Генератор сGAN работает следующим образом: на вход подается one-hot вектор класса x и в результате прохода через условный генератор выдается сгенерированное изображение этого класса. Таким образом, генератор можно представить, как следующую функцию: G:{x}→y.

Генератор обучается создавать максимально правдоподобные выходные изображения, дискриминатор же учится как можно лучше отличать фальшивые изображения от реальных.

Эта тренировочная процедура изображена на рисунке 9.

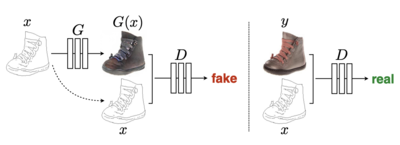
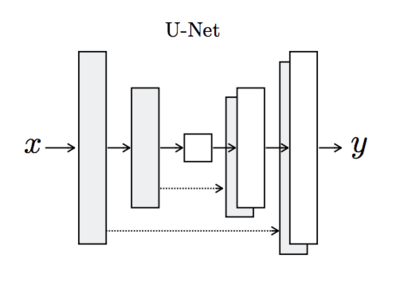


Рис. 9

***3.2.1. Архитектура Генератора***

Для генератора Pix2Pix используется U-net-генератор (рис. 10).

U-net-генератор - это модель encoder-decoder с добавлением пропускаемых соединений (англ. skip-connections) между зеркальными слоями в стеках кодировщика и декодера.

Пусть Ck обозначает слой Convolution-BatchNorm-LeakyReLU с k фильтрами.

BatchNorm не применяется к первому уровню C64 в кодировщике.

Все LeakyReLU в кодере имеют утечку с наклоном 0,2.

Рис.10

CDk обозначает слой Convolution-BatchNorm-Dropout-ReLU с коэффициентом отсева 50%.

Все свёртки представляют собой пространственные фильтры 4 × 4, применяемые с шагом 2. Свертки в кодере понижают дискретизацию в 2 раза, тогда как в декодере они передискретизируются в коэффициент 2.

Архитектура кодер-декодер состоит из:

кодер: C64-C128-C256-C512-C512-C512-C512-C512

декодер: CD512-CD512-CD512-C512-C256-C128-C64

После последнего слоя в декодере применяется свертка для сопоставления с количеством выходных каналов (3), за которым следует Tanh функция.

***Алгоритм работы генератора:***

- на вход подается изображение;

- далее последовательно применяются свертка, батч-нормализация (англ. Batch Norm layer), функция активации LeakyReLU и пулинг, что, тем самым, уменьшает количество признаков;

- при этом, следуя архитектуре U-net, добавляются пропускаемые соединения между каждым слоем i и слоем n−i, где n - общее количество слоев; каждое пропускаемое соединение просто объединяет все каналы на уровне i с другими на слое n−i; таким образом, информация, которая могла быть сильно сжата (потеряна), может доходить до некоторых более поздних слоев;

- после того, как получен слой минимального размера, начинается работа декодера, который делает то же, что и кодировщик, с отличием в слое, обратном пулингу, который увеличивает количество признаков;

- также в декодере добавляется dropout, чтобы достигнуть стохастичности на выходе генератора.

* + 1. ***Архитектура Дискриминатора***

Для дискриминатора используется свёрточный дискриминатор PatchGAN (рис. 11).

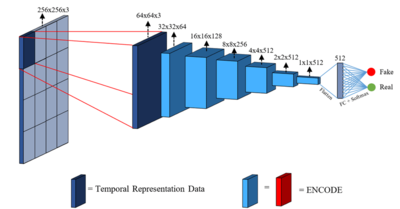


Рис. 11

PatchGAN дискриминатор - это тип дискриминатора для генеративных состязательных сетей, который штрафует алгоритм на уровне локальных фрагментов (патчей).

Архитектура дискриминатора 70 × 70:

C64-C128-C256-C512

После последнего слоя применяется свертка для сопоставления с одномерный выводом, за которым следует сигмовидная функция. BatchNorm не применяется к первому слою C64. Все LeakyReLU в кодере имеют утечку с наклоном 0,2.

Дискриминатор PatchGAN пытается определить, является ли каждый фрагмент размера N×N изображения настоящим или поддельным. Этот дискриминатор сверточно запускается по изображению, усредняя все ответы, чтобы посчитать окончательный результат D.

То есть, для каждого фрагмента определяется матрица классификаций, где все значения находятся в промежутке [0, 1], где 0 - подделка. Пройдя сверткой, в итоге получаем конечную матрицу классификаций. Таким образом, для поддельного изображения от генератора PatchGan должен попытаться вывести матрицу нулей.

N может быть намного меньше полного размера изображения и при этом давать результаты высокого качества. Это выгодно, потому что меньший PatchGAN имеет меньше параметров, работает быстрее и может применяться к изображениям большого размера.

Такой дискриминатор эффективно моделирует изображение как Марковское случайное поле, предполагая независимость между пикселями, разделенных диаметром более одного фрагмента.

* + 1. ***Функция потерь***

Целевая функция потерь сGAN может быть выражена как:

L*cGAN* (G, D) = Е*x,y*[log D(x, y)]+Е*x,z*[log(1-D(x, G(x, z))],

где G пытается минимизировать эту функцию потерь, а D пытается ее максимизировать, т.е.

G∗ = arg min*G* max*D* L*cGAN* (G, D).

В отдельных подходах [23], показано, что полезно смешивать потери (loss) GAN с более традиционными потерями, такими как расстояние L1 или L2 [23]. При этом L1 по сравнению с L2 способствует меньшему размытию. Работа дискриминатора остается неизменной, но перед генератором стоит задача не только обмануть дискриминатор, но и приблизиться к истине, поэтому его функция потерь выглядит следующим образом:

LL1(G) = Ex,y,z[||y − G(x, z)||1], где

||y − G(x, z)|| (|real\_output - generated\_output|) - попиксельная разница.

Итоговая функция потерь выглядит следующим образом:

LcGAN(G, D) + λ\*LL1(G).

* + 1. ***Метрики оценки качества решения***

Оценка качества синтезированных изображений является открытой трудной задачей [29]. Традиционные метрики, такие как среднеквадратическая ошибка на пиксель, не оценивают совместную статистику результат, и поэтому не измеряют саму структуру, которая структурированные потери направлены на захват.

В различных работах [29, 11] пытались использовать предварительно обученные семантические классификаторы измерить различимость генерируемых стимулов как псевдометрика. Если сгенерированные изображения реалистичны, классификаторы, обученные на реальных изображениях, смогут правильно классифицировать синтезированное изображение. Для этого обучали архитектуру FCN-8s. Затем оценивали синтезированные фотографии по точности классификации по меткам, из которых эти фотографии были синтезированы.

В данной работе оценка качества решения будет осуществляться:

- визуально;

- по значениям функции потерь (цель: убедиться, что ни модель генератора, ни модель дискриминатора не «выиграли»).

* gen\_gan\_loss или disc\_loss становятся очень низкими, это указывает на то, что одна модель доминирует над другой. Комбинированная модель обучается не успешно.
* Значение log(2) = 0.69 является хорошим показателем потерь, поскольку указывает, что дискриминатор в среднем одинаково неуверен в двух вариантах.
* disc\_loss < 0.69 означает, что дискриминатор работает лучше, чем случайный, в комбинированном наборе реальных и сгенерированных изображений.
* gen\_gan\_loss < 0.69 означает, что генератор лучше, чем случайный, обманывает дискриминатор.
* По мере обучения gen\_l1\_loss должен уменьшаться.
  + 1. ***Порядок обучения модели:***

- сегментированное изображение помещается в генератор U-Net, и он генерирует некоторый выход;

- дальше сгенерированное изображение соединяется с исходным входным сегментированным изображением, и это все идет в PatchGan дискриминатор, который выводит матрицу классификации, состоящую из значений между 0 и 1, которая показывает, насколько реальны или поддельны разные части этого изображения;

- затем для вычисления ошибки дискриминатора проводится 2 сравнения:

- сравнение:

матрицы классификации от {объединения сгенерированного изображения с исходным входным сегментированным изображением} с матрицей из всех 0;

матрицы классификация от {объединения реального изображения с исходным входным сегментированным изображением} с матрицей из всех 1;

- затем для вычисления ошибки генератора проводится сравнение матрицы классификации от {объединения сгенерированного изображения с исходным входным изображением} с матрицей из всех 1, которое считается с помощью BCE Loss, которое впоследствии суммируется с попиксельным сравнением реального изображения со сгенерированным, помноженным на λ;

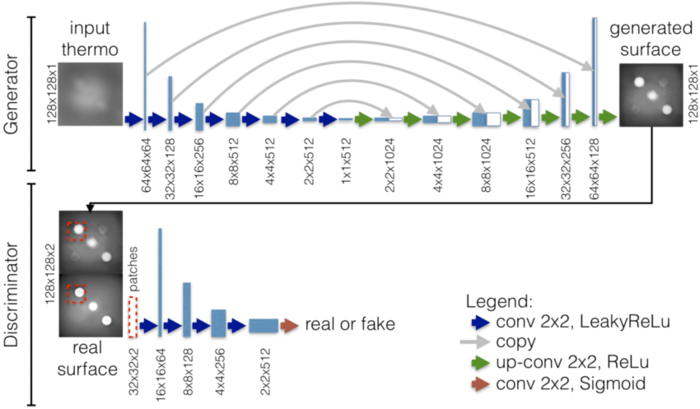
[](https://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:Pix2pix-UNet-128-GAN-network-architecture.png)

Рис. 12

Обучение сетей осуществляется с нуля.

Веса инициализируются из распределения Гаусса со средним значением 0 и стандартным отклонением 0,02.

В качестве оптимизатора используется Adam со скоростью обучения 0,0002, параметры импульса β1 = 0,5, β2 = 0,999.

Обучение на 200 эпохах (80\*103шагов), размер пакета - 1 (как показали эксперименты наиболее подходит для U-net).

Оптимизация сети проходит по стандартному подходу: вначале фиксируются веса G и обучается D максимизировать log D(x, G(x, z)), затем фиксируются веса D и обучается G минимизировать log(1 − D(x, G(x, z))). Как предложено в работе [1], вместо того, чтобы обучать G минимизировать log(1 − D(x, G(x, z))), обучаем максимизировать log D(x, G(x, z)). Кроме того, мы делим цель на 2 при оптимизации D, что замедляет скорость на который D узнает относительно G.

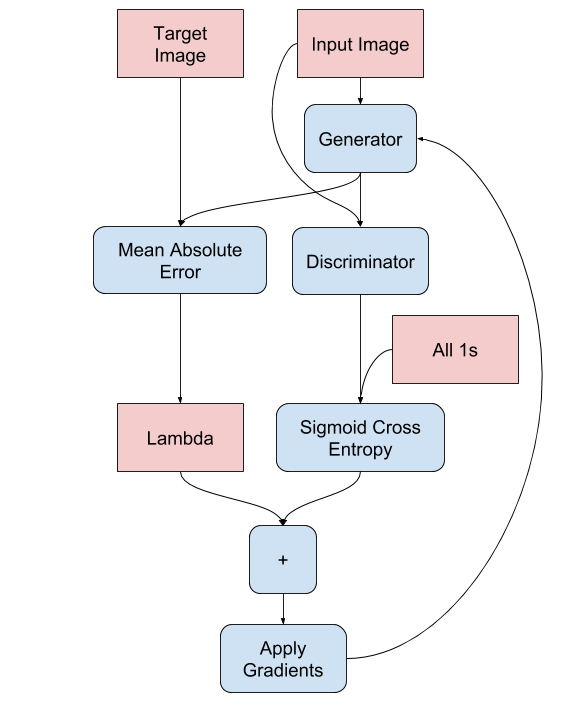
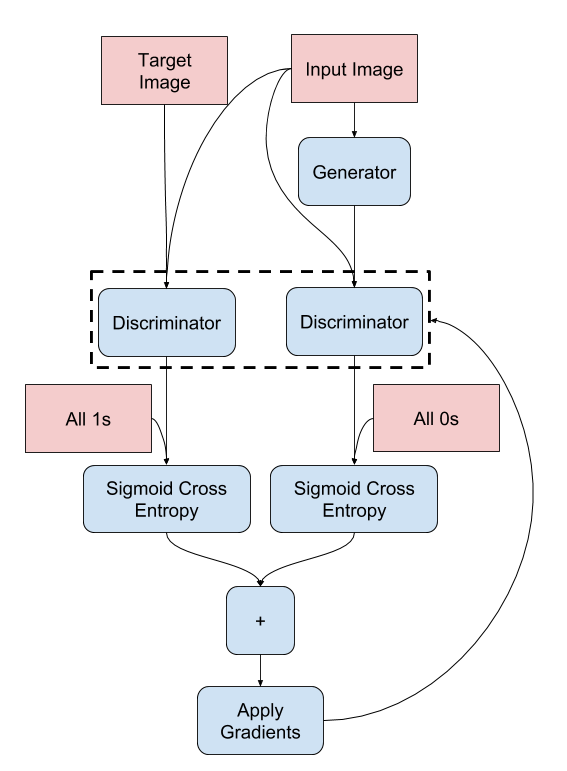


Рис. 13 Обучение Дискриминатора Рис. 14 Обучение Генератора

Во время вывода мы запускаем генераторную сеть точно так же, как и на этапе обучения. Это отличается от обычного протокола в том, что применяется отсев во время тестирования, и применяется пакетная нормализация, используя статистику тестовой партии, а не агрегированной статистики обучающей партии. Этот подход к нормализации партии, когда размер пакета установлен равным 1, он называется «нормализация экземпляра» и продемонстрировал свою эффективность в задачах генерации изображений [30].

Цикл обучения настроен на пошаговую работу, а не на эпохи.

Итерации проводятся по количеству шагов.

Каждые 1000 шагов: генерируется изображение, чтобы показать прогресс.

Каждые 5000 шагов: сохраняется контрольную точку.

Итоговый результат обучения модели (рис. 15):



Рис. 15

***4. Результаты***

***4.1. Значения целевых метрик итоговой модели***

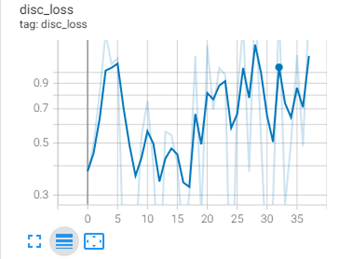
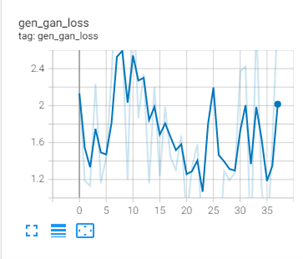


Рис. 16 Рис. 17

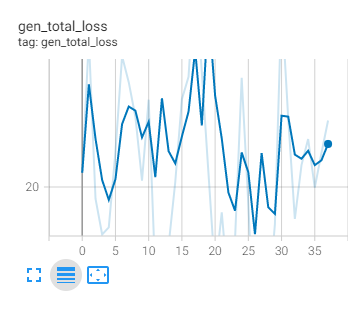
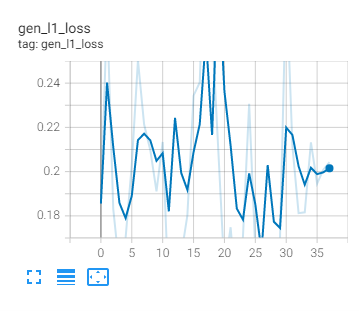


Рис. 18 Рис. 19

1. gen\_gan\_loss (рис. 16) или disc\_loss (рис. 17) не становятся слишком низкими, что указывает на то, что ни одна из моделей не доминирует над другой.

Комбинированная модель обучается успешно.

2. По мере обучения gen\_l1\_loss (рис. 18) и gen\_total\_loss (рис. 19) уменьшаются.

3. В среднем disc\_loss < 0.69 или близок к нему - дискриминатор работает лучше, чем случайный, в комбинированном наборе реальных и сгенерированных изображений.

4. gen\_gan\_loss > 0.69 означает, что генератор хуже, чем случайный, обманывает дискриминатор.

5. Необходимо дообучить модель генератора увеличив кол-во шагов или изменить размер Batch.

***Проведём визуальный анализ. Создадим несколько изображений используя тестовый набор:***

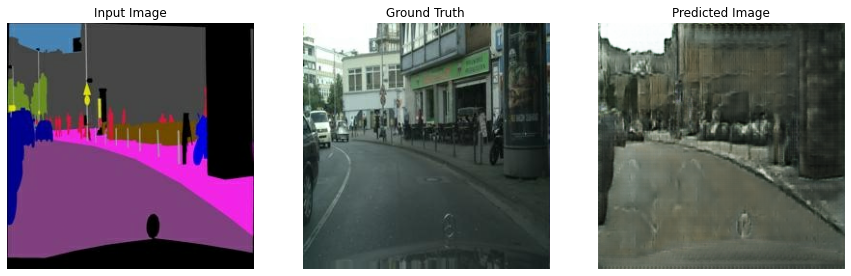


Рис. 20

***4.2. Область внедрения решения исходной задачи***

Описанная в дипломной работе сGAN успешно применяется специалистами по компьютерному зрению и графике, художниками-визуалистами к множеству задачи перевода изображения в изображение. Например:

удаление фона создание палитры эскиз-портрет



Скетч→Покемон передача позы «Делай, как я»



Фотогенератор



1. ***Выводы и заключение***

***В ходе выполнения дипломной работы поставленные цели достигнуты, задачи выполнены, а именно:***

- с использованием библиотек Python, TensorFlow, Keras построен код условной генеративной состязательной сети (cGAN) и проведено её обучение на тренировочном датасете «CityScapes» изображений с семантической окраской;

- полученные метрики качества (значения функций потерь) подтверждают выполнение задачи обучения сGAN;

- визуальный контроль полученных на тестовой выборке изображений подтверждает выполнение задачи получения приемлемого результата генерации реалистичных изображений из изображений с семантической окраской;

- условно-состязательные сети являются многообещающим подходом для многих задач перевода изображения в изображение, особенно тех, которые требуют высокой структурированный графический вывод, что делает их применимыми в самых разных условиях;

- дальнейшие пути развития и улучшения решения могут быть связаны с изменением параметров обучения (кол-во шагов обучения, гиперпараметры модели), с разработкой методик оценки качества модели, совершенствованием функции потерь, усложнением модели.

1. ***Список источников***
2. *Ян Дж. Гудфеллоу, Жан Пуже-Абади, Мехди Мирза, Бинг Сюй, Дэвид Уорд-Фарли, Шерджил Озаир, Аарон Курвиль и Йошуа Бенжио. Генеративно-состязательные сети. НИПС, 2014.*
3. *Алек Рэдфорд, Люк Мец, Сумит Чинтала. Неконтролируемое репрезентативное обучение с помощью глубоких сверточных генеративно-состязательных сетей . ICLR, 2016.*
4. *Джун-Ян Чжу, Филипп Крахенбуль, Эли Шехтман, Алексей А. Эфрос. Генеративное визуальное манипулирование естественным многообразием изображений . ECCV, 2016.*
5. *Jun-Yan Zhu\*, Taesung Park\*, Phillip Isola, Alexei A. Efros. Преобразование непарного изображения в изображение с использованием циклически согласованных состязательных сетей. ICCV, 2017.*
6. *Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros Image-to-Image Translation with Conditional Adversarial Networks, 2016.*
7. *L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In ICLR, 2015.*
8. *J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.*
9. *S. Xie and Z. Tu. Holistically-nested edge detection. In ICCV, 2015.*
10. *S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. ACM Transactions on Graphics (TOG), 2016.*
11. *G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. ECCV, 2016.*
12. *R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. ECCV, 2016.*
13. *Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4):600–612, 2004.*
14. *E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In NIPS, 2015.*
15. *C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. CVPR, 2016.*
16. *A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In NIPS, 2016.*
17. *S. Xie, X. Huang, and Z. Tu. Top-down learning for structured labeling with convolutional pseudoprior. In ECCV, 2015.*
18. *J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In ECCV, 2016.*
19. *S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In ICML, 2016.*
20. *X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. In ECCV, 2016.*
21. *M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. ICLR, 2016.*
22. *D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon. Pixellevel domain transfer. ECCV, 2016.*
23. *L. Karacan, Z. Akata, A. Erdem, and E. Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. arXiv preprint arXiv:1612.00215, 2016.*
24. *D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In CVPR, 2016.*
25. *Y. Zhou and T. L. Berg. Learning temporal transformations from time-lapse videos. In ECCV, 2016.*
26. *J.-Y. Zhu, P. Krahenb ¨ uhl, E. Shechtman, and A. A. Efros. ¨ Generative visual manipulation on the natural image manifold. In ECCV, 2016.*
27. *C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. ECCV, 2016.*
28. *C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In CVPR, 2017.*
29. *O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In MICCAI, 2015.*
30. *T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In NIPS, 2016.*
31. *D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022, 2016.*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*