МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский нижегородский государственный университет им. Н.И. Лобачевского
Институт информационных технологий, математики и механики
Кафедра математического обеспечения и суперкомпьютерных технологий

Отчёт по практике

Тема :

Сети Generative adversarial networks для синтеза изображений по текстовому описанию

Выполнил:
студент гр. 381706-1
Митягина Дарья Сергеевна
Научный руководитель:
Профессор, доктор технических наук
Турлапов Вадим Евгеньевич

Нижний Новгород 2019

Содержание

1 Введение

Одной из самых сложных проблем в мире Computer Vision является синтез высококачественных изображений из текстовых описаний. Без сомнения, это интересно и полезно, но современные системы искусственного интеллекта далеки от этой цели.

Генерация изображений имеет множество возможных применений в будущем, когда технологии будут готовы для коммерческого применения. Люди смогут создавать схему расположения мебели для своего дома, просто описывая ее на компьютере, а не тратя много часов на поиск нужного дизайна.
Создатели контента смогут творить в более тесном сотрудничестве с машиной, используя естественный язык. Кроме того, данная тема может стать более интересной, если ее развить. А именно, перевести задачу из генерации 2d изображения в построение 3d сцены и даже, возможно, формирование видео материалов с использование полученных сцен.

В данном отчете будут
- рассмотрены уже существующие решения задачи генерации изображений по текстовому описанию
- разобраны основные составляющие части программной реализации этих решений
- описаны основные положения из теории, лежащие в основе выбранных методов
- предъявлены результаты некоторых проведенных экспериментов

Постановка задачи :

1. Исследовать различные публикации на поставленную тему.
2. Изучить средства и методы решения задачи синтеза изображений из их словесного описания.
3. Начать проведение практических экспериментов.

2 Необходимый теоретический минимум

1. Граф сцены [1] [2]
Граф сцены представляет структуру, которая содержит логическое и зачастую (но не обязательно) пространственное представление графической сцены. Определение графа сцены нечёткое, поскольку программисты, осуществляющие его реализацию в приложениях, — и, в частности, в индустрии разработки игр — берут базовые принципы и адаптируют их для применения в конкретных приложениях. Это означает, что нет договорённости о том, каким должен быть граф сцены.

Граф сцены представляет собой набор узлов такой структуры, как граф или дерево. Узел дерева может иметь множество потомков, но зачастую только одного предка, причём действие предка распространяется на все его дочерние узлы; эффект действия, выполненного над группой, автоматически распространяется на все её элементы. Во многих программах ассоциирование матрицы преобразования на уровне любой группы и умножение таких матриц представляет собой эффективный и естественный способ обработки таких действий. Общей особенностью, к примеру, является способность группировать связанные формы/объекты в составной объект, который можно перемещать, трансформировать, выбирать и т. д. так же просто, как и одиночный объект.

Простейшая форма графа сцены использует массив либо структуру данных связный список, а отображение его форм есть лишь последовательная итерация узлов один за другим. Другие общие операции, такие как определение, какая форма пересекается с курсором мыши (например, в приложениях, основанных на графическом интерфейсе пользователя) также осуществляются путём линейного поиска. Для небольших графов сцены этого обычно достаточно.

Более масштабные графы сцены приводят к заметному замедлению линейных операций, так что используются более сложные структуры хранения основных данных, наиболее популярная и общая форма — это дерево. В этих графах сцены составной шаблон проектирования зачастую призван создать иерархическое представление узлов группировки и листовых узлов. Сгруппированные узлы могут иметь любое количество прикреплённых дочерних узлов. Сгруппированные узлы включают узлы трансформаций и коммутационные узлы. Листовые узлы — это узлы, которые в действительности подлежат визуализации, либо узлы, на которых виден результат какого-либо действия. Они включают объекты, спрайты, звуки, источники освещения и всё, что может рассматриваться как подлежащее «рендерингу» в некотором абстрактном смысле.

2. Свёрточная нейронная сеть [3]
Идея свёрточных нейронных сетей заключается в чередовании свёрточных слоёв (англ. convolution layers) и субдискретизирующих слоёв. Структура сети — однонаправленная (без обратных связей), принципиально многослойная. Для обучения используются стандартные методы, чаще всего метод обратного распространения ошибки. Функция активации нейронов (передаточная функция) — любая, по выбору исследователя.

Архитектура и принцип работы :

В обычном перцептроне, который представляет собой полносвязную нейронную сеть, каждый нейрон связан со всеми нейронами предыдущего слоя, причём каждая связь имеет свой персональный весовой коэффициент. В свёрточной нейронной сети в операции свёртки используется лишь ограниченная матрица весов небольшого размера, которую «двигают» по всему обрабатываемому слою (в самом начале — непосредственно по входному изображению), формируя после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией. То есть для различных нейронов выходного слоя используются одна и та же матрица весов, которую также называют ядром свёртки. Её интерпретируют как графическое кодирование какого-либо признака, например, наличие наклонной линии под определённым углом. Тогда следующий слой, получившийся в результате операции свёртки такой матрицей весов, показывает наличие данного признака в обрабатываемом слое и её

координаты, формируя так называемую карту признаков (англ. feature map). Естественно, в свёрточной нейронной сети набор весов не один, а целая гамма, кодирующая элементы изображения (например линии и дуги под разными углами). При этом такие ядра свёртки не закладываются исследователем заранее, а формируются самостоятельно путём обучения сети классическим методом обратного распространения ошибки. Проход каждым набором весов формирует свой собственный экземпляр карты признаков, делая нейронную сеть многоканальной (много независимых карт признаков на одном слое). Также следует отметить, что при переборе слоя матрицей весов её передвигают обычно не на полный шаг (размер этой матрицы), а на небольшое расстояние. Так, например, при размерности матрицы весов 55 её сдвигают на один или два нейрона (пикселя) вместо пяти, чтобы не «перешагнуть» искомый признак.

Операция субдискретизации (англ. subsampling, англ. pooling, также переводимая как «операция подвыборки» или операция объединения), выполняет уменьшение размерности сформированных карт признаков. В данной архитектуре сети считается, что информация о факте наличия искомого признака важнее точного знания его координат, поэтому из нескольких соседних нейронов карты признаков выбирается максимальный и принимается за один нейрон уплотнённой карты признаков меньшей размерности. За счёт данной операции, помимо ускорения дальнейших вычислений, сеть становится более инвариантной к масштабу входного изображения.

Рассмотрим типовую структуру свёрточной нейронной сети более подробно. Сеть состоит из большого количества слоёв. После начального слоя (входного изображения) сигнал проходит серию свёрточных слоёв, в которых чередуется собственно свёртка и субдискретизация (пулинг). Чередование слоёв позволяет составлять «карты признаков» из карт признаков, на каждом следующем слое карта уменьшается в размере, но увеличивается количество каналов. На практике это означает способность распознавания сложных иерархий признаков. Обычно после прохождения нескольких слоёв карта признаков вырождается в вектор или даже скаляр, но таких карт признаков становятся сотни. На выходе свёрточных слоёв сети дополнительно устанавливают несколько слоёв полносвязной нейронной сети (перцептрон), на вход которому подаются оконечные карты признаков.

Слой свёртки. Нейроны слоя свёртки, преобразуемые по нескольким выходным каналам Слой свёртки (англ. convolutional layer) — это основной блок свёрточной нейронной сети. Слой свёртки включает в себя для каждого канала свой фильтр, ядро свёртки которого обрабатывает предыдущий слой по фрагментам (суммируя результаты поэлементного произведения для каждого фрагмента). Весовые коэффициенты ядра свёртки (небольшой матрицы) неизвестны и устанавливаются в процессе обучения.

Особенностью свёрточного слоя является сравнительно небольшое количество параметров, устанавливаемое при обучении. Так например, если исходное изображение имеет размерность 100100 пикселей по трём каналам (это значит 30000 входных нейронов), а свёрточный слой использует фильтры с ядром 3x3 пикселя с выходом на 6 каналов, тогда в процессе обучения определяется только 9 весов ядра, однако по всем сочетаниям каналов, то есть 936 =162, в таком случае данный слой требует нахождения только 162 параметров, что существенно меньше количества искомых параметров полносвязной нейронной сети.

Слой активации. Скалярный результат каждой свёртки попадает на функцию активации, которая представляет собой некую нелинейную функцию. Слой активации обычно логически объединяют со слоем свёртки (считают, что функция активации встроена в слой свёртки). Функция нелинейности может быть любой по выбору исследователя, традиционно для этого использовали функции типа гиперболического тангенса
$f(x) = \tanh x$
$f(x) = |\tanh x|$
или сигмоиды
$f(x) = (1 + e^{-x})^{-1}$
$f(x) = (1 + e^{-x})^{-1})$.

Однако в 2000х годах была предложена и исследована новая функция активации — ReLU (сокращение от англ. rectified linear unit), которая позволила существенно ускорить процесс обучения и одновременно упростить вычисления (за счёт простоты самой функции), что означает блок линейной ректификации, вычисляющий функцию
$f(x) = \max(0, x)$
$f(x) = \max(0, x)$.
То есть по сути это операция отсечения отрицательной части скалярной величины. По состоянию на 2017 год эта функция и её модификации (Noisy ReLU, Leaky ReLU и другие) являются наиболее часто используемыми функциями активации в глубоких нейросетях, в частности, в свёрточных. Существует методика определения оптимального числа блоков линейной ректификации.

Пулинг или слой субдискретизации. Пулинг с функцией максимума и фильтром 22 с шагом 2 Слой пулинга (иначе подвыборки, субдискретизации) представляет собой нелинейное уплотнение карты признаков, при этом группа пикселей (обычно размера 22) уплотняется до одного пикселя, проходя нелинейное преобразование. Наиболее употребительна при этом функция максимума. Преобразования затрагивают непересекающиеся прямоугольники или квадраты, каждый из которых ужимается в один пиксель, при этом выбирается пиксель, имеющий максимальное значение. Операция пулинга позволяет существенно уменьшить пространственный объём изображения. Пулинг интерпретируется так: если на предыдущей операции свёртки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного. К тому же фильтрация уже ненужных деталей помогает не переобучаться. Слой пулинга, как правило, вставляется после слоя свёртки перед слоем следующей свёртки.

Кроме пулинга с функцией максимума можно использовать и другие функции — например, среднего значения или L2-нормирования. Однако практика показала преимущества именно пулинга с функцией максимума, который включается в типовые системы.

В целях более агрессивного уменьшения размера получаемых представлений, всё чаще находят распространение идеи использования меньших фильтров[8] или полный отказ от слоёв пулинга.

3. Генеративно-состязательная сеть (GAN) [4] [5]
Алгоритм машинного обучения без учителя, построенный на комбинации из двух нейронных сетей, одна из которых (сеть G) генерирует образцыа другая (сеть D) старается отличить правильные («подлинные») образцы от неправильных. Так как сети G и D имеют противоположные цели — создать образцы и отбраковать образцы — между ними возникает Антагонистическая игра. Генеративно-состязательную сеть описал Ян Гудфеллоу из компании Google в 2014 году.

Использование этой техники позволяет в частности генерировать фотографии, которые человеческим глазом воспринимаются как натуральные изображения.

Дискриминатор. Дискриминационные алгоритмы пытаются классифицировать входные данные. Учитывая особенности полученных данных, они стараются определить категорию, к которой они относятся.

К примеру, пробегая все слова в письме дискриминационный алгоритм может предсказать, является сообщение спамом или не спамом. Спам — это категория, а пакет слов, собранный из электронной почты — образы, которые составляют входные данные. Математически категории обозначают y, а образы обозначают x. Запись p(y|x) используется для обозначения «вероятности y при заданном x», которая обозначает «вероятность того, что электронное письмо является спамом при имеющемся наборе слов».

Итак, дискриминационные функции сопоставляют образы с категорией. Они заняты только этой корреляцией.

Генератор. Генеративные алгоритмы заняты обратным. Вместо того, чтобы предсказы-

вать категорию по имеющимся образам, они пытаются подобрать образы к данной категории.

В то время как дискриминационные алгоритмы волнует взаимосвязь между y и x, генеративные алгоритмы волнует "откуда берутся x". Они позволяют находить p(x|y), вероятность x при данном y или вероятность образов при данном классе (генеративные алгоритмы также могут использоваться в качестве классификаторов. Они могут делать больше, чем классифицировать входные данные.)

Еще одно представление о работе генеративных алгоритмов можно получить, разделяя дискриминационные модели от генеративных таким образом:

1. Дискриминационные модели изучают границу между классами;
2. Генеративные модели моделируют распределение отдельных классов.

Как работают GAN

Одна нейронная сеть, называемая генератором, генерирует новые экземпляры данных, а другая — дискриминатор, оценивает их на подлинность; т.е. дискриминатор решает, относится ли каждый экземпляр данных, который он рассматривает, к набору тренировочных данных или нет.

Предположим, мы пытаемся сгенерировать рукописные цифры, подобные тем, что имеются в наборе данных MNIST. Цель дискриминатора — распознать подлинные экземпляры из набора.

Между тем, генератор создает новые изображения, которые он передает дискриминатору. Он делает это в надежде, что они будут приняты подлинными, хотя являются поддельными. Цель генератора состоит в том, чтобы генерировать рукописные цифры, которые будут пропущены дискриминатором. Цель дискриминатора — определить, является ли изображение подлинным.

Шаги, которые проходит GAN :

1. Генератор получает рандомное число и возвращает изображение.
2. Это сгенерированное изображение подается в дискриминатор наряду с потоком изображений, взятых из фактического набора данных.
3. Дискриминатор принимает как реальные, так и поддельные изображения и возвращает вероятности, числа от 0 до 1, причем 1 представляет собой подлинное изображение и 0 представляет фальшивое.

Таким образом, у есть двойной цикл обратной связи:

1. Дискриминатор находится в цикле с достоверными изображениями.
2. Генератор находится в цикле вместе с дискриминатором.

Вы можете представить GAN как фальшивомонетчика и полицейского, играющих в кошки мышки, где фальсификатор учится изготавливать ложные купюры, а полицейский учится их обнаруживать. Оба динамичны; т. е. полицейский тоже тренируется (возможно, центральный банк отмечает пропущенные купюры), и каждая сторона приходит к изучению методов другого в постоянной эскалации.

Сеть дискриминаторов представляет собой стандартную сверточную сеть, которая может классифицировать изображения, подаваемые на нее с помощью биномиального классификатора, распознающего изображения как реальные или как поддельные. Генератор в некотором смысле представляет собой обратную сверточную сеть: хотя стандартный сверточный классификатор принимает изображение и уменьшает его разрешение, чтобы получить вероятность, генератор принимает вектор случайного шума и преобразует его в изображение. Первый отсеивает данные с помощью методов понижения дискретизации, таких как maxpooling, а второй генерирует новые данные.

Обе сети пытаются оптимизировать целевую функцию или функцию потерь в игре zero-zum. Это, по сути, модель актера-критика (actor-critic). Когда дискриминатор меняет свое поведение, то и генератор меняет, и наоборот.

Автокодеры [6] [7] и VAE [8]. Полезно сравнить генеративные состязательные сети с другими нейронными сетями, такими как автокодеры (автоэнкодеры) и вариационные автокодеры.

Автокодеры кодируют входные данные в векторы. Они создают скрытое или сжатое представление необработанных данных. Они полезны при уменьшении размерности: вектор, служащий в качестве скрытого представления, сжимает необработанные данные в меньшее количество. Автокодеры могут быть сопряжены с так называемым декодером, который позволяет восстанавливать входные данные на основе их скрытого представления, как и в случае с машиной Больцмана.

Вариационные автокодеры являются генеративным алгоритмом, который добавляет дополнительное ограничение для кодирования входных данных, а именно то, что скрытые представления нормализуются. Вариационные автокодеры способны сжимать данные как автокодеры и синтезировать данные подобно GAN. Однако, в то время как GAN генерируют данные детализовано, изображения, созданные VAE, бывают более размытыми. Примеры Deeplearning4j включают в себя как автокодеры, так и вариационные автокодеры. Вы можете разделить генеративные алгоритмы на три типа, которые имея:

1. категорию, предсказывают связанные функции (Naive Bayes);
2. скрытое представление, предсказывают связанные функции (VAE, GAN);
3. некоторые образы, предсказывают остальное (inpainting, imputation);

4. Архитектура нейронной сети [9] [10]

1. Входные узлы (входной слой): вычислений в этих слоях нет, они просто передают информацию следующему слою.
2. Скрытые узлы (скрытый слой): в скрытых слоях выполняется промежуточная обработка или вычисления, после чего происходит перенос весов с входного слоя на следующий слой.
3. Выходные узлы (выходной слой): здесь мы наконец используем функцию активации.
4. Соединения и веса: сеть состоит из соединений, каждое соединение передает выход нейрона i на вход нейрона j. В этом смысле i является предшественником j, а j является преемником i. Каждому соединению присваивается вес $W_{i,\,j}$.
5. Функция активации: функция активации узла определяет выходные данные этого узла с учетом входных данных или набора входных данных.
6. Правило обучения. Правило обучения - это правило или алгоритм, который изменяет параметры нейронной сети для того, чтобы данный вход в сеть создавал предпочтительный результат. Этот процесс обучения обычно сводится к изменению весов и порогов.

3 Обзор источников

На данную тему существует сравнительно немного литературы, что дает понять ее новизну и необходимость дальнейшего ее изучения.
Несомненно, одним из наиболее важных аспектов работы являются данные, использованные для обучения моделей.
Каждая статья отличается своим подходом к решению данной проблемы, но в большинстве своем они содержат в качестве составной части генеративно-состязательную сеть и различные ее модификации(ObjGAN, StackGAN, StackGAN++, AttnGAN и так далее).
Каждый из существующих методов дает отличные от других результаты. Если попытаться обобщить, то можно сказать, что есть алгоритмы, прекрасно выполняющие свою задачу при несложной структуре сцены и маленьком количестве объектов, но показывающие менее впечатляющие результаты при усложнении одного из критериев.
Примеры статей, представляющих подобные алгоритмы :

1. Генерация изображений по текстовому описанию с помощью генеративной сети внимания (Attentional Generative Adversarial Networks)
   Статья AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks, Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, Xiaodong He [11].
   Будет рассмотрена подробна в части 4 (Разбор статей).

2. Синтез изображений по текстовому описанию с оптимизацией информации
   Статья "Text to Image Synthesis with Mutual Information Optimization Lihang Liu, Mei Wang [12].
   Целью является генерирование изображения из описаний, т.е. использование неструктурированные текстовые описания в качестве входных данных и создание соответствующих изображений. Эта задача сочетает в себе две сложные компоненты : языковое моделирование и генерация изображений. В этой статье авторы предложили метод, который объединяет оба пайплайна [пайплайн(pipeline) - это процесс разработки (подготовки, производства), программный конвейер] синтез текста в изображение и подписи к ним чтобы помочь максимизировать поток информации через эти конвейеры и улучшить качество генерируемых изображений. В экспериментах на CUB и наборах данных Oxford-102, авторы демонстрируют эффективность метода по генерации изображений определенных категорий из неструктурированных текстовые описания.

Есть авторы, попытавшиеся расширить границы возможностей GAN в данной области.
Например :

1. Генерация мелкозернистого изображения с помощью асимметричной тренировки.
   Статья "CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, Gang Hua [13].

   Авторы представляют вариационные генеративные состязательные сети, общая структура обучения которых сочетает в себе auto-encoder с генеративной состязательной сетью для синтеза изображения в мелкозернистых категориях, таких как лица конкретного человека или объекты опредеенной категории.

2. Синтез фотореалистичных изображений по текстовому описанию, используя стек генеративно-состязательных сетей.
   Статья "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks , Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas [14].

   В этой статье предложен стек генеративныо-состязательных сетей (StackGAN) для генерации 256 256 фотореалистичных изображений с учетом текстовых описаний. Авторы

разлагают сложную проблему на более управляемые подзадачи в процессе уточнения эскиза. Stage-I GAN рисует примитивную форму и цвета объекта на основе данного текстового описания, получая изображения низкого разрешения Stage-I. Этап II ГАН принимает Stage-I результаты и текстовые описания в качестве входных данных, а также генерирует изображения с высоким разрешением с фотореалистичными деталями. Это способно исправлять дефекты в результатах Стадии I и добавлять неотразимые детали с процессом уточнения.

В данном отчете будут рассмотрены примеры из обеих выделенных мною категорий.
А теперь уделим внимание данным, на которых обучались сети. Как говорил профессор Эндрю Ын в своих курсах: «В деле машинного обучения достигает успеха не тот, у кого есть наилучший алгоритм, а тот, у кого есть наилучшие данные». Чаще всего используются стандартные наборы данных(Data set) с фотографиями и текстовым описанием интересующих объектов(в большей части статьей такими объектами являются птицы, цветы, иногда и лица людей и так далее). Примеры используемых dataset'ов : COCO [15], LSUN[16], CUB, MS COCO и т.п.

Далее приведен более подробный анализ двух конкретных статей :
- Image Generation from Scene Graphs, Justin Johnson, Agrim Gupta, Li Fei-Fei;
- AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks, Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, Xiaodong He

4 Разбор статей

4.1 Статья 1 : Генерация изображений по графу сцены

В данной части будет рассмотрена статья "Image Generation from Scene Graphs"Justin Johnson, Agrim Gupta, Li Fei-Fei [17].
Большинство существующих методов дают потрясающие результаты на ограниченных доменах, таких как описания птиц или цветов, но не могут точно воспроизвести сложные предложения со многими объектами и отношениями.
Для преодоления этого ограничения в данной статье авторы предлагают метод генерирования изображений из графов сцен, позволяющих явно рассуждать об объектах и их отношениях. Представленная в публикации модель использует свертку графа для обработки входных графов, вычисляет макет сцены, прогнозируя границы Bounding Box (BBox, ограничивающий параллелепипед) и маски сегментации для объектов, и преобразует макет в изображение с каскадным уточнением сети.

Основы метода :
Предложение представляет собой линейную структуру, в которой одно слово следует за другим; однако, информация, передаваемая сложным предложением, часто может быть более явно представлена в виде графа сцены объектов и их отношений.

- Для обработки входных данных графа сцены используется сеть свертки графа.
- Для создания изображения, которое соответствует макету применяется cascaded refinement network (CRN), которая обрабатывает макет.
- Чтобы убедиться, что сгенерированные изображения реалистичны и содержат необходимые объекты разумно применить генеративно-состязательные сети, работающие на патчах изображений и сгенерированных объектах.
- Сквозной процесс обучения можно разделить на два основных компонента :
Учебный компонент - первый этап, на котором машина записывает все параметры, выполняемые оператором (через Сверточные нейронные сети (CNN)).
Компонент логического вывода возможен тогда, когда машина действует на основе ранее полученного опыта от компонента обучения сквозного процесса обучения.

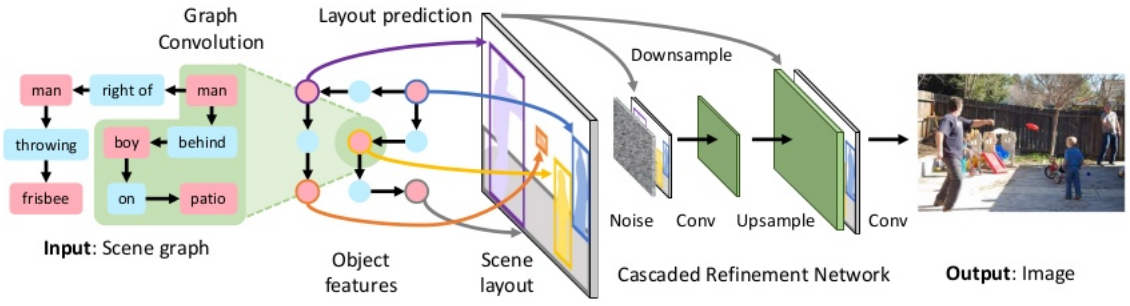

*Рис. 1 - Схема, описывающая метод

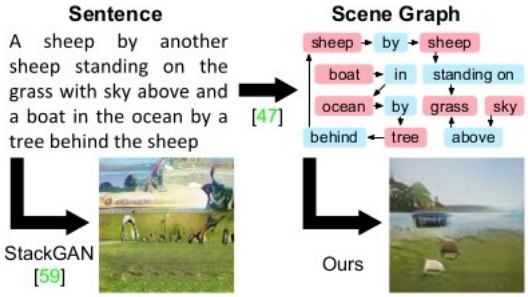

*Рис. 2 - Генерация графа сцены по предложению

Входом для описанной модели является граф сцены, описывающий объекты и связи между ними. Дан набор категорий объектов C и набор категории R, граф сцены - это граф (O, E), где O = o_1,. , , , o_n - множество объектов с каждым o $i \in C$, и $E \subseteq O \times R \times O$- множество направленных ребер вида (o_i, r, o_j) где o_i, $o_j \in O$и$r \in R$.

Конкретно, заданные входные векторы v_i, v_r $\in \mathrm{R}^D$ для всех объектов $o_i \in O$ и ребра (o_i, r, o_j) $\in E$, вычисляем вывод векторы для v_i', v_r' $\in \mathrm{R}^D_{out}$ для всех узлов и ребер с использованием три функции g s, g p и g o, которые принимают в качестве входных данных тройку векторов (v_i, v_r, v_j) для ребра и вывод новых векторов для субъекта o_i, предиката r и объекта o_j соответственно. Чтобы вычислить выходные векторы v_r' для ребер, мы просто устанавливаем $v_r = g_p\ (v_i, v_r, v_j)$.
Аналогично использовалось g_o для вычисления набора кандидатов V_o^i для всех ребер, оканчивающихся на o_i. В частности,
$V_s^i = g_s\ (v_i, v_r, v_j)$: $(o_i, \mathrm{r}, o_j) \in E$.
$V_o^i = g_o\ (v_j, v_r, v_i)$: $(o_j, \mathrm{r}, o_i \in E$.
Выходной вектор для объекта o i затем вычисляется как $v_i' = \mathrm{h}\ (V_s^i \bigcup V_o^i)$, где h - симметричная функция, которая объединяет входной набор векторов в один выходной вектор.

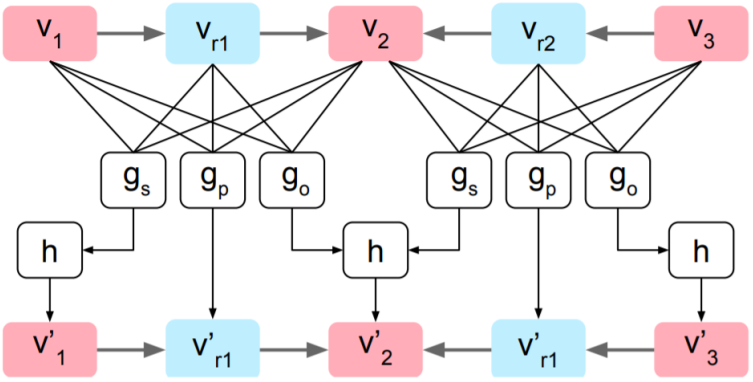

*Рис. 3 - Вычислительный граф

Вычислительный граф, иллюстрирующий один граф сверточного слоя. Граф состоит из трех объектов o_1, o_2 и o_3 и два ребра (o_1, r_1, o_2) и (o_3, r_2, o_2). Вдоль каждого края три входные вектора передаются в функции g_s, g_p и g_o; g_p напрямую вычисляет выходной вектор для ребра, а g_s и g_o вычисляют векторы-кандидаты, которые подаются к симметричной функции пула h для вычисления выходных векторов для объектов.

Авторы генерируют реалистичные выходные изображения, обучая сеть генерации изображений e состязательно против пары дискриминаторных сетей D_{img} и D_{obj}. Дискриминатор D пытается классифицировать входные данные x как реальные или поддельные путем максимизации $L_{GAN} = \underset{x \sim p_{real}}{E} logD(x) + \underset{x \sim p_{fake}}{E} log(1 - D(x))$
гдех$\sim$ p_{fake} - выход из сети генерации f. В то же время, f пытается генерировать выходные данные, которые будут обмануть дискриминатор путем минимизации L_{GAN}. D_{img} дискриминатор изображения на основе патча обеспечивает общий вид сгенерированных изображений реалистичен; он классифицирует регулярно расположенный, перекрывающийся набор изображений патчи как реальные или фальшивые. Дискриминатор объекта D_{obj} гарантирует, что каждый объект на изображении выглядит реалистичным. В дополнение к классификации каждого объекта как реальный или фальшивый, D_{obj} также гарантирует, что каждый объект распознаваем, используя вспомогательный классификатор, который предсказывает категория объекта; и D_{obj}, и F пытаются максимизировать вероятность того, что D_{obj} правильно классифицирует объекты.

4.2 Статья 2 : Генерация изображений по текстовому описанию с помощью генеративной сети внимания (Attentional Generative Adversarial Networks)

Рассмотрим статью "AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks"Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, Xiaodong He. [11]
В этой статье предложена Attentional Generative Adversarial Network (AttnGAN), которая обеспечивает сосредоточенное, многоэтапный синтез изображения из текста. Благодаря новой генеративной сети внимания AttnGAN можно синтезировать мелкозернистые детали в разных областях изображения, обращая внимание на соответствующие слова в описании на естественном языке. Предложенная Генеративная Состязательная Сеть (AttnGAN) имеет два новых компонента: генеративная сеть с вниманием (attentional generative network) и глубокая мультимодальная модель подобия внимания(the deep attentional multimodal similarity model).

Описание метода :
Современные GAN-модели для генерации текста в изображения обычно кодируют целое предложение в текстовое описание в одном векторе как условие для генерации изображения, но не хватает детальной информации на уровне слов. В этом разделе авторы статьи предлагают новую модель внимания, которая позволяет генеративной сети рисовать различные субрегионы изображения, основанные на отношении к этим субрегионам. Предложенная генеративная сеть внимания имеет m генераторов (G_0, G_1, ..., G_{m-1}), которые берут скрытые состояния (h_0, h_1, ..., h_{m-1}) в качестве входных данных и генерируют изображения малых и больших масштабов ($\hat{x^0}$, $\hat{x^1}$, ..., $\hat{x^{m-1}}$).
В частности, $h_0 = F_0$ (z, F^{ca} ($\overline{e}$));
$h_i = F_i$ (h_{i-1}, F_i^{attn}(e, h_{i-1})) для i = 1, 2, ..., m - 1;
$\hat{x^i} = G_i$ (h_i).
Здесь z - вектор шума, обычно выбираемый из стандартного нормального распределения, $\overline{e}$ является глобальным вектором предложения, а e является матрица векторов слов. F^{ca} представляет собой кондиционирование дополнение, которое преобразует вектор предложения e в вектор кондиционирования. F^{attn} - предлагаемая модель внимания на i-м этапе AttnGAN. F^{ca}, F_i^{attn}, F_i и G_i моделируются как нейронные сети.
Для создания реалистичных изображений с несколькими уровнями (т.е. уровень предложения и уровень слова) условий, определяется конечная целевая функция генеративной сети внимания:
$L = L_G + \lambda L_{DAMSM}$, where $L_G = \sum_{i=0}^{i<m1} L_{G_i}$

Модель глубокого внимательного мультимодального сходства DAMSM изучает две нейронные сети, которые отображают субрегионы изображения и слова предложения в общее семантическое пространство, таким образом, измеряет сходство изображения с текстом на уровне слова, чтобы вычислить мелкозернистые потери при генерации изображения.

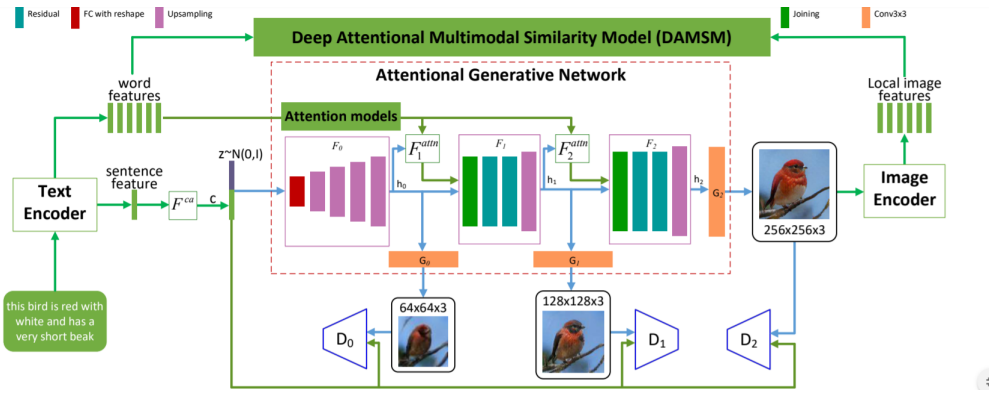

*Рис. 4 - Схема, описывающая метод

5 Результаты практики

Была предпринята попытка расширить теоретические знания в области нейронных сетей (в том числе применительно к задаче генерации изображений по текстовому описанию сцены).

В результате работы был реализован простейший автокодировщик, являющийся составной частью одного из описанных методов.
Так же была реализована простая версия одной из частей конволюционной(сверточной) нейронной сети. Выбрана именно конволюционная сеть, т.к. она(ее модификация) применяется в рассматриваемой в курсовой работе статье.
Использовался набор данных Fashion MNIST с помощью Google Colab Laboratory.

Необходимо сравнивать точность Training (обучение) и Validation (тест), чтобы определить переобучение. Точность обучения, которая субъективно намного выше, чем точность теста, указывает на него.
Полученная модель дала следующие результаты :

```
In [15]: epoch_list = list(range(1, len(hist.history['acc']) + 1))
         plt.plot(epoch_list, hist.history['acc'], epoch_list, hist.history['val_acc'])
         plt.legend(("Training Accuracy", "Validation Accuracy"))
         plt.show()
```

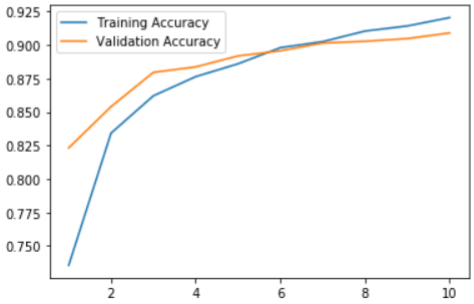

*Рис. 5 - Точность участка Training против точности Validation. Простейший пример.

6 Заключение

В ходе проделанной работы были достигнуты поставленные цели.

1. Исследованы различные публикации на поставленную тему (2).

   Какие же выводы можно сделать по рассмотренным двум статьям? Они дают различные результаты : Статья 1 (генерация изображений по графу сцены) дает более хорошие результаты при наличии сложных отношений между объектами, присутствующими в текстовом описании сцены в то время, как Статья 2 (генерация изображений по текстовому описанию с помощьюгенеративной сети внимания) при данных условиях уступает.

   Методы, описанные в каждой из представленных публикаций, абсолютно разные.

   Составные компоненты первого метода :

   (a) Сеть свертки графа.
   (b) Cascaded refinementnetwork (CRN).
   (c) Сверточные нейронные сети (CNN).

   Составные компоненты второго метода :

   (a) Генеративная сеть с вниманием (attentional generative network).
   (b) Глубокая мультимодальная модель подобия внимания(the deep attentional multimodal similarity model)

   Однако, если обобщить, то каждое решение представляет собой особую модификацию GAN.

2. Изучены некоторые средства и методы решения задачи синтеза изображений из их словесного описания.

3. Проведены практические эксперименты.

В дальнейшем планирую рассмотреть следующую литературу:

1. Generative Adversarial Networks Cookbook by Josh Kalin, December 2018
2. Photographic Text-to-Image Synthesis with a Hierarchically-nested Adversarial Network, Zizhao Zhang, Yuanpu Xie, Lin Yang

7 Ссылки

1. D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei. Scene graph generation by iterative message passing. In CVPR, 2017.
2. J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In CVPR, 2015
3. Convolutional neural networks, Jianxin Wu, LAMDA Group, National Key Lab for Novel Software Technology, Nanjing University, China
4. Generative Adversarial Nets, Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair , Aaron Courville, Yoshua Bengio
5. A Beginner's Guide to Generative Adversarial Networks (GANs), https://pathmind.com/wiki/generat adversarial-network-gan
6. Autoencoders, Unsupervised Learning, and Deep Architectures, Pierre Baldi
7. A Tutorial on Deep Learning Part 2: Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks, Quoc V. Le
8. VAE (Variational autoencoders), https://seas.ucla.edu/ kao/nndl/lectures/vae.pdf
9. Как работает нейронная сеть: алгоритмы, обучение, функции активации и потери, https://neurohive.io/ru/osnovy-data-science/osnovy-nejronnyh-setej-algoritmy-obuchenie-funkcii-aktivacii-i-poteri/
10. Нейронные сети. Полный курс. Саймон Хайкин. Университет McMaster. Онтарио, Канада
11. AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks, Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, Xiaodong He
12. Text to Image Synthesis with Mutual Information Optimization, Lihang Liu, Mei Wang
13. CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training, Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, Gang Hua
14. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris Metaxas
15. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollar, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: Computer Vision - ECCV 2014 - 13th European Conference. pp. 740–755 (2014)
16. Yu, F., Zhang, Y., Song, S., Seff, A., Xiao, J.: LSUN: Construction of a Largescale Image Dataset using Deep Learning with Humans in the Loop. CoRRabs/1506.0 (2015)
17. Image Generation from Scene Graphs, Justin Johnson, Agrim Gupta, Li Fei-Fei
18. Text-to-Image Synthesis Based on Machine Generated Captions, Marco Menardi, Alex Falcon, Saida S.Mohamed, Lorenzo Seidenari, Giuseppe Serra, Alberto Del Bimbo and Carlo Tasso
19. MirrorGAN: Learning Text-to-image Generation by Redescription Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao
20. Object-driven Text-to-Image Synthesis via Adversarial Training, Wenbo Li, Pengchuan Zhang, Lei Zhang, Qiuyuan Huang, Xiaodong He, Siwei Lyu, Jianfeng Gao