Санкт–Петербургский государственный университет

Кузнецов Дмитрий Алексеевич

Выпускная квалификационная работа

Применение искусственных нейронных сетей в задаче восстановления изображений

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2016 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Математическое и программное обеспечение вычислительных машин»

Научный руководитель:
доцент кафедры ТСУЭФА, к.ф. - м.н.
Козынченко Владимир Александрович

Рецензент:
профессор кафедры теории управления, д.ф. - м.н.
Котина Елена Дмитриевна

Санкт-Петербург
2020 г.

Содержание

Введение

Восстановление изображения — это процесс восстановления недостающей области изображения. Такое изображение представляется в виде одной (если изображение черно-белое) или нескольких (если изображение цветное) матриц, в которых пикселям, значения которых нужно восстановить, соответсвует значение 0 в матрице, называемой маской изображения, а пикселям, не требующим восстановления, соответсвует единица. Выпускная квалификационная работа посвящена решению задачи восстановления изображений, в которых присутсвуют области с дефектами, например посторонние предметы, или повреждения произвольных форм и размеров, шумы, трещины и т.п.

Первоначально для решения задачи использовались итеративные алгоритмы, которые ищут похожие совпадения в доступной области изображения и/или усредняют значения соседних пикселей, двигаясь от границы недостающей области до ее внутренней части. Использование искусственных нейронных сетей, а конкретннее сверточных, позволило улучшить качество и точность обработки, придя на смену классическим итеративным алгоритмам. В последнее время особую популярность приобрели методы, основанные на генеративно-состязательных нейронных сетях, состоящих из двух сетей: первая генерирует образцы, а вторая пытается отличить, реальное изображение ей пришло на вход или сгенерированное.

В Главе 1 представлена общая теория нейронных сетей, процессы их работы и обучения, дается характеристика U-net архитектуры. В Главе 2 рассматриваются три группы методов восстановления изображений — итеративные алгоритмы, методы, основанные на сверточных нейронных сетях, и методы, основанные на генеративно-состязательных нейронных сетях, их преимущества и недостатки. Метод, с помощью которого реализуется поставленная задача, использует сверточную нейронную сеть. Описание алгоритма, а также параметров сети приведено в Главе 3. Глава 4 посвящена особенностям реализации выбранного алгоритма и описанию проведенных экспериментов.

Постановка задачи

Цель работы — решить задачу восстановления изображений с применением искусственных нейронных сетей. Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить наиболее популярные подходы, основанные как на искусственных нейронных сетях, так и на классических итеративных алгоритмах, рассмотреть их преимущества и недостатки;

2. Выбрать и реализовать один из методов, основанный на глубоком обучении;

3. Провести эксперименты, для чего нужно:

    - Подготовить датасет — изображения для обучения сети и проверки точности работы алгоритма;
    - Сгенерировать нерегулярные области отсутсвущих пикселей (маски), которые представляют из себя матрицы размерности равной размерности изображения, состоящие из нулей и единиц, и которые обозначают, значения какой группы пикселей нужно восстановить;
    - Используя подготовленный датасет, обучить модель;

4. Проверить работу сети.

Обзор литературы

Исследованием нейронных сетей занимаются со второй половины прошлого века. За это время написано множество работ, среди которых можно выделить вводные в теорию искусственных нейронных сетей (далее — ИНС): Уоссермен Ф. [2], Хайкин С. [3], Круглов В. В., Борисов В. В. [4], в которых рассматриваются и общедоступно излагаются основные парадигмы ИНС, их строгое математическое обоснование, роль и применение, а повествование сопровождается примерами, практическими задачами и экспериментами.

С необходимостью распознавать изображения возникла и новая проблема — каким образом подавать изображения в ИНС? Можно, конечно, поставить каждому значению пикселя в соответсвие один входной нейрон, но такое представление неустойчиво к изменению масштаба и поворотам. Для решения этой проблемы группа ученых во главе с Яном ЛеКуном представила сверточную нейронную сеть, которая позволяла решить эти проблемы. Информацию об этой архитектуре можно найти в [5].

На сегодняшний день системы глубокого обучения стали действительно точными в выявлении закономерностей в изображениях, видео и тексте. Но они с трудом создают изображения, предложения, абзацы или переводы, которые воспринимаются человеком как натуральные. Генеративные состязательные сети (GAN) предлагают многообещающее решение этих проблем, соединяя две конкурирующие нейронные сети: одну, которая генерирует контент, и другую, которая отбрасывает образцы с низким качеством. Такие сети также используют в задаче восстановления изображений — image inpainting. Более подробно про них изложено в [6, 7]

Еще одной важной задачей является задача сегментации объектов на изображении, в которой нужно классифицировать не изображение, а каждый пиксель этого изображения. Информацию о них можно найти в [8, 9, 10]

Глава 1. Нейронные сети

1.1 Сети прямого распространения

В 1943 году У. Маккалок и У. Питтс впервые ввели понятие искусственного нейрона и искусственной нейронной сети в работе «Логическое исчисление идей, относящихся к нервной активности» [1]. Искусственный нейрон — математическая функция, задуманная как модель биологических нейронов. Он является элементарной единицей в ИНС и состоит из следующих частей:

1. Набор *связей*, характеризующихся весом (weight) w_{ij}. Входной сигнал x_j на входе синапса j, связанного с нейроном k, умножается на вес w_{kj} (Рис. 1).

2. *Сумматор* складывает входные сигналы, взвешенные относительно соответствующих связей нейрона.

3. *Функция активации* осуществляет нелинейное преобразование выходного сигнала. Обычно значения функции лежат в диапазоне $[0, 1]$ или $[-1, 1]$

   Математически функционирование можно нейрона представить:

$$y_k = \varphi(\sum_{j=1}^{m} w_{kj}x_j + b_k)$$

Простейшая сеть состоит из нейронов, объеденненных в слои, которые связаны между собой и в которых выходной сигнал каждого нейрона слоя q является входным сигналом для нейронов слоя $q + 1$.

Результатом работы сети является вектор выходных сигналов $\{y_j^{(Q)}\}$ последнего слоя Q. Обучение обычно происходит с учителем. Это означает наличие обучающего набора, который содержит пары входных и выходных значений. Для обучения сети используется функция потерь $E = E(\{w_{ij}\}) = E(Y^{(Q)}, D) \in \mathbb{R}$. Она принимает в качестве аргументов выходное значение модели $Y^{(Q)} = \{y_j^{(Q)}\} \in \mathbb{R}^l$ и ожидаемое истинное значение

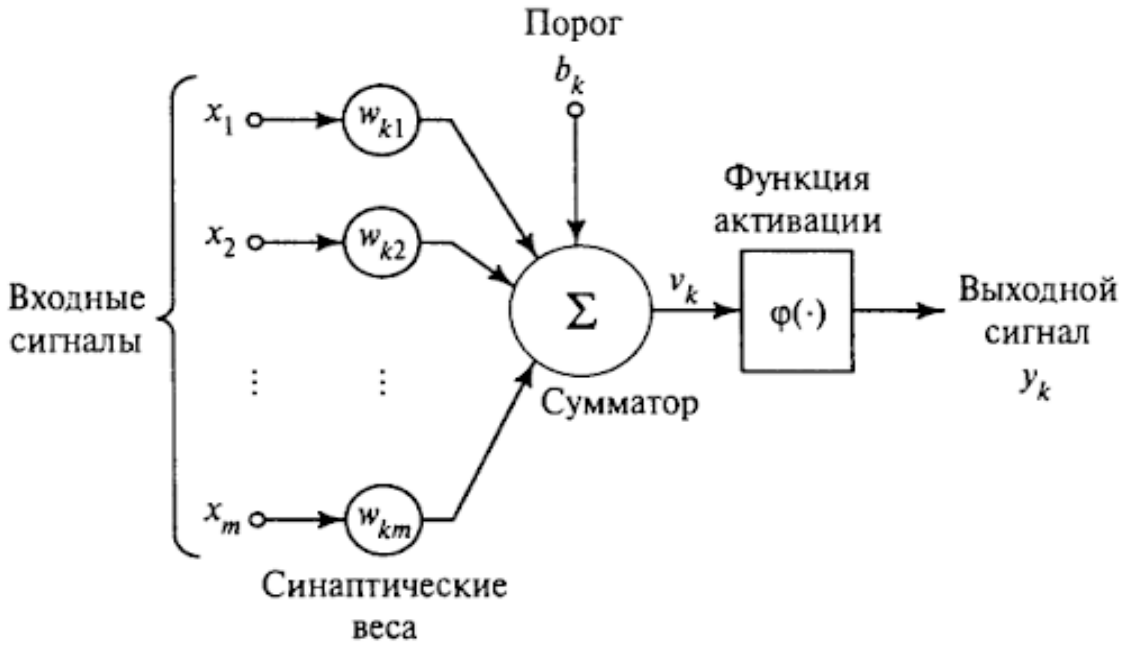

Рис. 1: Модель нейрона МакКалока-Питса.

$D \in \mathbb{R}^l$. Часто в роли этой функции выступает среднеквадратичное отклонение:

$$E(\{w_{ij}\}) = \frac{1}{2}\sum_j (y_j^{(Q)} - d_j)^2,$$

где $y_j^{(Q)}$ — реальное выходное состояние нейрона j выходного слоя Q, d_j — требуемое выходное состояние этого нейрона, а суммирование ведется по всем нейронам выходного слоя. Задача обучения состоит в том, чтобы минимизировать эту функцию, т.е. найти $\{w_{ij}\} = \arg\min(E(\{w_{ij}\}))$. Для этого используются методы градиентного спуска, вычисляющие антиградиент методом обратного распространения ошибки: последовательно от последнего слоя к первому вычисляется градиент функции потерь E по параметрам w_{ij} текущего слоя. Часто ИНС обучаются стохастически, то есть на каждом шаге используются различные части данных. Это избавляет от необходиости хранить в памяти большое количество обучающих примеров и позволяет избежать застреваний в локальных минимумах. Стандартным методом обучения является метод стохастического градиентного спуска,

который обновляет каждый параметр следующим образом:

$$\Delta w_{ij}^{(q)} = -\eta \frac{\partial E}{\partial w_{ij}},$$

где w_{ij} — весовой коэффициент синаптической связи, соединяющей $i-$й нейрон слоя $q-1$ с $j-$м нейроном слоя q, η — коэффициент скорости обучения, $0 < \eta < 1$.

Однако неправильный выбор шага может привести к расхождению метода или к его медленной сходимости, поэтому существуют его модификации, которые изменяют шаг в процессе обучения.

1.2 Сверточные нейронные сети

В 1989 году Ян ЛеКун совместно с группой ученых опубликовал статью о новой нейросетевой архитектуре — сверточной нейронной сети (далее СНС) [11]. В 2012 году на соревновании по компьютерному зрению, которое проводилось на базе изображений ImageNet, Алекс Крижевский с командой исследователей разработал СНС, которая смогла с ошибкой в 15.8% классифицировать по тысячам категорий миллионы изображений [13]. В отличие от многослойного персептрона, СНС частично устойчива к искажениям изображений: изменениям масштаба, поворотам, смещениям и др. Секрет успеха СНС в использовании свертки — особого вида математической операции. Пусть $I_{M\times N} \in \mathbb{R}^2$ — исходная матрица изображения, $K_{P\times Q} \in \mathbb{R}^2$ — ядро свертки. Тогда *линейной сверткой* матрицы I с ядром K будем называть матрицу $S_{(M-P+1)\times(N-Q+1)} = \{S(i,j)\} \in \mathbb{R}^2$, элементы которой определены как:

$$S(i,j) = \sum_{p=1}^{P} \sum_{q=1}^{Q} I(p+i, q+j) K(p,q). \tag{1}$$

То есть мы проходим по матрице изображения окном $X_{P\times Q}$, на каждом шаге вычисляем сумму элементов матрицы $X \odot K$ ($\odot$ — поэлементное умножение) и записываем полученное значение в новую матрицу (рис. 2).

Размерность выходного изображения зависит от размерности ядра,

от шага, с которым движется окно, и от способа обработки границ — как будут учитываться пиксели, выходящие за пределы изображения, — их можно отсечь, а можно заполнить нулями или другими значениями.

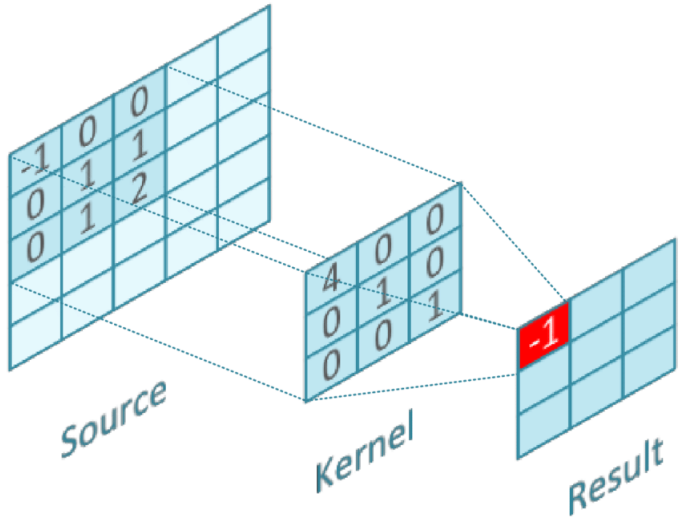

Рис. 2: Визуальное представление операции свертки.

СНС состоит из последовательно чередующихся сверточных слоев и слоев субдескритизации (или пулинга). Сверточный слой представляет из себя фильтр — набор ядер (или одно ядро), которые используются для свертки. По сути, ядро — это матрица чисел — весов — которые настраиваются в процессе обучения с целью поиска определенных признаков на изображении. Ядро перемещается по изображению и определяет, есть ли данный признак в конкретной части изображения. Для этого используется операция свертки. Полученная в результате свертки матрица называется картой признаков.

Слой пулинга представляет собой сжатие карты признаков. Его суть можно передать так: если на предыдущем слое были выявлены определенные признаки, то для дальнейшей работы достаточно менее подробного изображения, и поэтому карта сжимается. Также сжатие помогает сети не переобучиться, так как ненужные детали удаляются. Поэтому для преобразований берутся непересекающиеся прямогульники или квадраты (обычно размера 2×2) пикселей, из каждого такого квадрата по определенному правилу выбирается один пиксель. Чаще всего в роли правила выступает

функция максимума (max pooling). Слой пулинга позволяет существенно снизить размерность изображния. Он вставляется после слоя свертки перед слоем следующей свертки. Пример работы max pooling представлен на рис. 3.

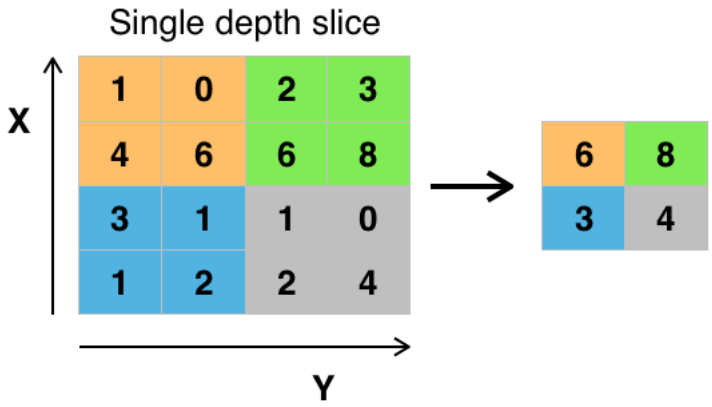

Рис. 3: Визуальное представление операции max pooling.

После прохождения нескольких слоев свертки и пулинга система перестраивается из определенной сетки пикселей с высоким разрешением в более абстрактные карты признаков, количество которых, как правило, увеличивается на каждом последующем слое, что происходит благодаря увеличению количества ядер в фильтрах свертки. Размер же каждой карты уменьшается. В конце концов, остается большой набор карт, которые хранят небольшое количество значений (и даже одно), которые интерпретируются как наиболее абстрактные характеристики, полученные из исходного изображения. Эти данные объединяются и передаются в обычную полносвязную нейронную сеть, которая также может состоять из нескольких слоев. В то же время, полносвязные слои уже теряют пространственную структуру пикселей и имеют сравнительно небольшой размер (относительно количества пикселей в исходном изображении).

1.3 Архитектура U-net

U-net — это сверточная нейронная сеть, изначально разработанная для сегментации биомедицинских изображений на факультете компьютерных наук Университета Фрайбурга в Германии [21]. Архитектура сети пред-

ставляет собой полностью сверточную нейронную сеть [22], т.е. не имеющую полносвязных слоев.

Сеть состоит из двух путей, которые придают ей U-образную архитектуру (рис. 4): сжимающего (слева) и расширяющего (справа). Сжимающий путь — типичная архитетектура сверточной сети. Каждый шаг состоит из двух сверточных слоев, в которых мнократного примененяется свертка ядрами размера 3×3, с функцией активации ReLU: $f(x) = \max(0, x)$. Далее следует слой max pooling с шагом 2 для понижения разрешения. На каждом шаге max pooling число карт признаков удваивается.

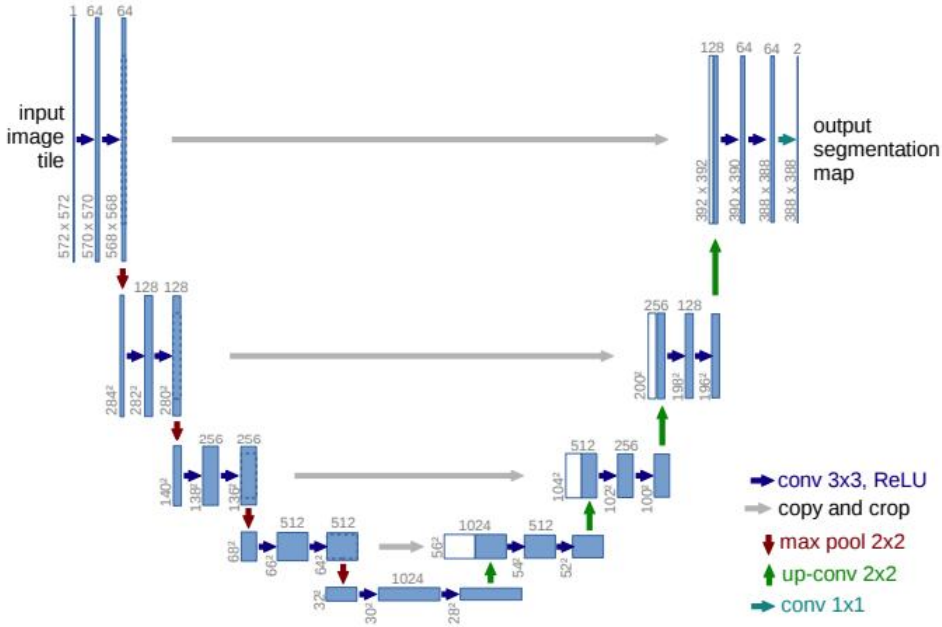

Рис. 4: Архитектура U-net сети.

Каждый шаг расширяющего пути состоит из слоя, повышающего разрешение изображения (up-conv) в два раза, после которого идет сверточный слой с ядром 2×2, который вдвое сокращает число карт признаков. После идет конкатенация с соответсвующей обрезанной картой признаков, полученной на сжимающем пути и два сверточных слоя с ядром 3×3 и функцией активации ReLU. Обрезание необходимо из-за потери граничных пикселей на каждом этапе свертки. На последнем слое используется свертка ядром 1×1 для отображения каждого 64-элементного вектора признаков на требуемое количество классов.

Глава 2. Восстановление изображений

Восстановление изображения — это процесс востановления недостающей области изображения или удаления некоторых объектов, добавленных к нему с сохранением семантической согласованности изображения без видимых изменений. Любые области пикселей, значения которых нужно восстановить, будем называть искажениями. Их примеры представлены на рис. 5.

Операция восстановления зависит от формы искажения и семантики изображения. Было предложено много подходов, которые, согласно [14], можно разделить на три категории: итеративные алгоритмы, подходы на основе сверточных нейронных сетей (CNN) и подходы на основе генеративно-состязательных нейронных сетей (GAN).

Блоки Тексты Шумы Объекты Маски Трещины

Рис. 5: Прмеры искажений [14]

2.1 Итеративные алгоритмы

Подходы, основанные на итеративных алгоритмах, можно разделить на две категории: основанные на патчах (patch) — небольших квадратных областях — и на диффузии.

Методы, основанные на патчах, заполняют область отсутствующих значений патч за патчем путем итеративного поиска наиболее подходящих патчей (т.е. патчей-кандидатов) в неповрежденной части изображения и копирования их в соответствующие места. Так, например, в алгоритме PatchMatch [15] вводится поле ближайших соседей (NNF) как функция $f : \mathbb{R}^2 \to \mathbb{R}^2$ смещений, определяемая на всех возможных координатах исходного патча. То есть для данной координаты патча a и соотвествующей координаты любого другого патча $b : f(a) = b - a$. Таким образом, выбирается патч, для которого значение f — минимально.

Методы, основанные на диффузии, заполняют недостающую область путем плавного распространения содержимого изображения от границы недостающей области до ее внутренней части, при этом также используя доступную информацию изображения.

К преимуществам этого подхода можно отнести отсутствие видимых наблюдаемых изменений на границах области отсутствующих пикселей. К недостаткам — невозможность семантически корректно восстановить изображение при отсутствии в данном изображении подходящего совпадения.

2.2 Методы на основе сверточных нейронных сетей

В последнее время сильный потенциал глубоких сверточных сетей (CNN) проявляется во всех задачах компьютерного зрения, особенно при восстановлении изображений. В основном, к результату восстановления предъявляются два требования: глобальная семантическая согласованность и высокая детализация текстур. Классические методы рисования, описанные в предыдщуем разделе, не в состоянии определить глобальную семантику изображния, что остается все еще сложной задачей. Для ее решения было предложено немало методов, основанных на сверточных нейронных сетях, их эффективность также обусловлена способностью сети к обучению

на большом количестве данных. Например, Shift-net [28] использует специальный слой shift-connection, чтобы соединить признаки известной области с неизвестной. В том же контексте Weerasekera с соавт. [23] используют карту глубины изображения в качестве входных данных.

2.3 Методы на основе генеративно-состязательных нейронных сетей

Генеративно-состязательные сети (GAN) были впервые рассмотрены Яном Гудфеллоу в 2014 году в [16]. GAN представляет структуру, которая содержит две сети прямого распространения, генератор G, «разворачивающий» изображение на базе скрытых параметров, и дискриминатор D, реализованный как сверточная нейронная сеть. Обе сети настроены на работу друг против друга: генеративная сеть G получает случайный шум z в качестве входного сигнала и генерирует некоторые ложные изображения, похожие на реальные, тогда как дискриминационная сеть D (дискриминатор) должна научиться определять, реальными или поддельными являются образцы. Эту связь можно рассматривать как игру двух игроков, в которой соревнуются G и D: G пытается минимизировать функцию потерь, то есть состязательную потерю, а D — максимизировать:

$$\mathcal{L}_D = \sum_{i=1}^{m} \left(\log \left(D(x^{(i)}) \right) + \log \left(1 - D(G(z^{(i)})) \right) \right),$$

$$\mathcal{L}_G = \sum_{i=1}^{m} \log \left(D(G(z^{(i)})) \right),$$

где $z^{(i)}$ — случайный вектор шума, $x^{(i)}$ — реальное изображение, полученное в результате работы сети, m — число сгенерированных примеров. Недавно GAN был применен к нескольким методам семантического рисования, чтобы естественным образом заполнить области отсутствия пикселей.

Суть метода [17] состоит в локальном и глобальном согласовании результата. Для обучения сети используются глобальные и локальные контекстые дискриминаторы, которые обучены отличать реальные изображения от полученных в результате восстановления. Глобальный дискрими-

натор просматривает все изображение, чтобы оценить, является ли оно в целом согласованным, в то время как локальный дискриминатор смотрит только на небольшую область с центром в восстановленной области. Затем сеть обучается обманывать обе сети распознавания контекста, что требует от нее генерирования изображений, которые неотличимы от реальных с точки зрения общей согласованности, а также в деталях.

А подход в [18] основывается на модели, которая может не только синтезировать новые структуры изображений, но и явно использовать особенности доступных пикселей изображений в качестве ссылок во время обучения сети, чтобы делать более точные прогнозы. Модель состоит из двух частей. Первая часть – сверточная сеть, обученная с функцией ошибки — reconstruction loss. Контекстное внимание интегрируется на втором этапе. Его основная идея заключается в использовании известных патчей в качестве сверточных ядер для обработки уже созданных патчей.

2.4 Выбор алгоритма

При таком подходе к восстановлению изображений, как PatchMatch [15], не используется глубокое обучение, а производится поиск соответствий в оставшемся изображении, что приводит к ограниченности использования. Так, например, на рис. 6 (b) этот алгоритм смог плавно заполнить недостающие фрагменты картины, используя патчи стены с тенью, но при семантическом подходе следовало бы использовать патчи картины.

Методы, основанные глубоком обучении, изучают семантику изображения, а сети используют сверточные ядра на изображениях для замены удаленного контента фиксированным значением. В результате эти подходы сильно зависимы от начальных значений отсутсвующих пикселей, что часто проявляется в отсутствии текстуры в областях отсутствия пикселей или явных цветовых контрастах. Примеры использования архитектуры U-net с обычными сверточными слоями можно увидеть на рис. 6(e) и 6(f).

Хотя использование генеративных методов, таких как GAN, для задачи покраски изображений является привлекательной идеей, их использование также имеет недостатки. Во-первых, GAN должны быть обучены на

(a) Изображение с маской

(b) Алгоритм PatchMatch

(c) Iizuka с соавт. [17]

(d) Yu с соавт. [18]

(e) Hole=127.5

(f) Hole=Mean

(g) Partial Conv [19]

(h) Исходное изображение

Рис. 6: [19] 6(a): Изображение с маской. 6(b): результат восстановления алгоритмом PatchMatch, 6(c): Результат работы алгоритма Iizuka с соавт. [17]. 5(d): Результат работы алгоритма Yu с соавт. [18]. 6(e) и 6(f) используют U-net подобную архитектуру [24] с обычными сверточными слоями, в 6(e) вместо неизвестных значений записаны 127.5, а в 6(f) — средние по ImageNet датасету. 6(g): та же U-net архитектура с использованием частичных сверточных слоев вместо обычных сверточных.

большом наборе данных для получения хороших результатов. Во-вторых, перерисовка с помощью GAN довольно медленная, а обусловливание вывода значений пикселей в отсутствующей области в конечном итоге приводит к различным типам визуальных неточностей, которые требуют дорогостоящей последующей обработки. Например, Iizuka с соавт. [17] используют пуассоновское смешивание изображений [20], а Yu с соавт. [18] используют уточняющую сеть для корректировки необработанных предсказаний. Однако эти уточнения не могут устранить такие искажения, как 6(c) и 6(d). В связи с этим был выбран алгоритм, цель которого — добиться семантически значимого прогнозирования отсутствующих значений независимо от инициализации отсутствующих значений и без какой-либо дополнительной постобработки.

Глава 3. Алгоритм на основе частичного сверточного слоя.

Алгоритм, выбранный для реализации, основан на новом подходе, использующем вместо сверточных слоев частичные сверточные слои. Сверточный слой представляет из себя операцию частичной свертки и функцию обновления маски — бинарной матрицы размером равным размеру изображения, состоящей из нулей и единиц и обозначающей, значения каких пикселей неизвестны (закрашены).

3.1 Частичная свертка

Пусть $W-$ ядро свертки, $X'-$ значения пикселей для текущего окна свертки, а $M'-$ соответствующая бинарная маска. Тогда частичная свертка в каждом месте определяется как:

$$x' = \begin{cases} W * (X' \odot M') \frac{sum(1)}{sum(M')}, & \text{if } sum(M') > 0 \\ 0, & \text{otherwise} \end{cases},$$

где $\odot-$ поэлементное умножение, $A * B = sum(A \odot B)-$ операция свертки, аналогичная (1), $\frac{sum(1)}{sum(M')}$ – масштабирование. После этого происходит обновление маски по следующему правилу:

$$m' = \begin{cases} 1, & \text{if } sum(M') > 0 \\ 0, & \text{otherwise} \end{cases}. \tag{2}$$

3.2 Архитектура сети

Сеть представляет собоой U-net подобную архитектуру [21], аналогичную той, что использовалась в [24]. Но все сверточные слои заменены частичными сверточными слоями, а также на этапе декодирования используется nearest neighbour up-sampling.

Вместо типичного заполнения пространства за пределами изображе-

ния (padding) в качестве значений используются результаты частичной свертки, а само пространство интерпретируется как еще одна область отсутствующих пикселей. Это гарантирует, что на неокрашенное содержимое на границе изображения не будут влиять недопустимые значения вне изображения.

3.3 Функция потерь

Функция потерь нацелена на точность восстановления как отдельных пикселей, так и композиции в целом, то есть на то, как плавно согласуются предсказанные значения в отсутствующей области с окружающим их контекстом. Поэтому ее можно рассматривать как составную:

$$\begin{gathered}\mathcal{L}_{total} = \alpha_1 \mathcal{L}_{valid} + \alpha_2 \mathcal{L}_{hole} + \alpha_3 \mathcal{L}_{perceptual}, \\ \alpha_i \in \mathbb{R}, \quad \forall i = \overline{1,3}.\end{gathered} \tag{3}$$

Величины коэффициентов были использованы, согласно [19]: $\alpha_1 = 1, \alpha_2 = 6, \alpha_3 = 0.05$.

Для начала определим функцию потерь для попиксельного восстановления. Пусть $\mathbf{I}_{in}$ — входное изображение, $\mathbf{M}$ — заданная бинарная маска (отсутствующим значениям соотвествует 0, остальным — 1), $\mathbf{I}_{out}$ — восстановленное изображение и $\mathbf{I}_{gt}$ — реальное изображение. Тогда $\mathcal{L}_{hole} = \|(1 - \mathbf{M}) \odot (\mathbf{I}_{out} - \mathbf{I}_{gt})\|_1$ и $\mathcal{L}_{valid} = \|\mathbf{M} \odot (\mathbf{I}_{out} - \mathbf{I}_{gt})\|_1$. Это L^1 ошибка для областей отсутствующих значений и не требующих восстановления соответственно. Перцепционная потеря, т.е. отражающая восприятие, может быть выражена, согласно [25]:

$$\mathcal{L}_{perceptual} = \sum_{n=0}^{N-1} \|\Psi_n(\mathbf{I}_{out}) - \Psi_n(\mathbf{I}_{gt})\|_1 + \sum_{n=0}^{N-1} \|\Psi_n(\mathbf{I}_{comp}) - \Psi_n(\mathbf{I}_{gt})\|_1, \tag{4}$$

где $\mathbf{I}_{comp}$ — необработанное изображение $\mathbf{I}_{out}$. Перцепционная потеря вычисляет L^1 расстояние между $\mathbf{I}_{out}$ и $\mathbf{I}_{comp}$ и оригинальным изображением, но после проецирования этих изображений в пространственные объекты более высокого уровня с использованием сети VGG-16 Ψ_n, предварительно обученной на ImageNet датасете.

3.4 Алгоритм

Случайным образом инициалищируются параметры (веса) сети. Сеть обучается с помощью итеративного алгоритма. На каждом шаге два этапа:

1. Прямое распространение;
2. Обратное распространение.

При прямом распространении генерируется обучающий пример, состоящий из реального изображения $\mathbf{I}_{in}$ (batch size равен 4) и сгенерированной случайным образом маски $\mathbf{M}$. Маска накладывается на изображение, как показано на рис.7 и полученное изображение $\mathbf{I}_{masked}$ подается на вход U-net подобной сети, принцип работы которой описан в 4. Далее сеть с весами, полученными на предыдущем шаге, предсказывает некоторый результат $\mathbf{I}_{out}$. После чего рассчитывает функцию потерь $E = E(I_{in}, I_{out})$, определенную в (3). При обратном распространении рассчитывается градиент функции потерь E, которая после этого минимизируется методом Adam, а веса соответствующим образом корректируются.

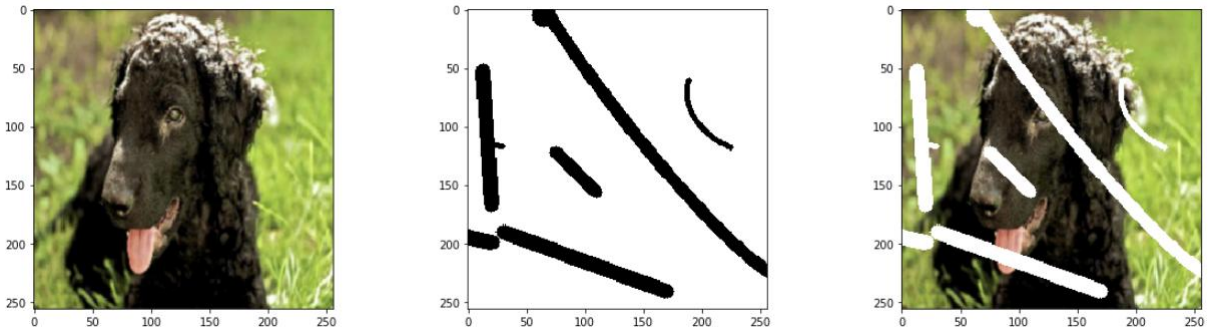

Рис. 7: Пример изображения с наложенной маской.

Глава 4. Реализация и эксперименты

Данный алгоритм был реализован на языке Python 3 с использованием надстройки над библиотекой для машинного обучения TensorFlow – библиотеки Keras в несколько этапов:

1. Генерация множества случайных нерегулярных масок;
2. Реализация слоя частичной свертки PConv2D;
3. Реализация U-net подобной архитектуры с PConv2D слоями;
4. Обучение сети на выбранном датасете;
5. Попытка предсказать недостающий фрагмент изображения.

Генерация множества масок. Для генерации нерегулярных масок — имеющих произвольную форму и размеры — была использована библиотека компьютерного зрения OpenCV. С ее помощью случайным образом сгенерированы и соединены непрерывные линии и эллипсы. Интерфейс представлен классом MaskGenerator. Примеры масок на рис. 8.

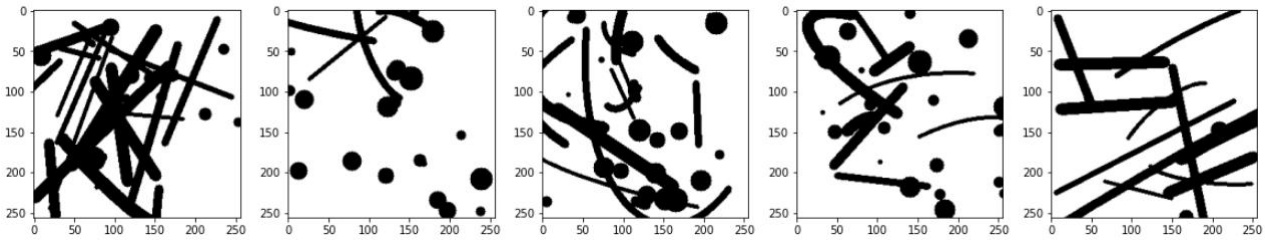

Рис. 8: Примеры полученных масок.

Частичный сверточный слой. Представляет собой класс PConv2D, который расширяет класс Conv2D в библиотеке Keras, реализуя частичную свертку и функцию обновления маски, описанные в 3.1 по следующему алгоритму:

Пусть:

- $X \in \mathbb{R}^{H \times W}$ — матрица входного изображения

- $M \in \mathbb{R}^{H \times W}$ — матрица входной маски
- $K \in \mathbb{R}^{H_k \times W_k}$ — ядро свертки
- $U \in \mathbb{R}^{H_k \times W_k}$ — матрица, состоящая из единиц
- $sum(A) \in \mathbb{R}$ — сумма элементов матрицы A
- $A \odot B$ — поэлементное умножение двух матриц одинаковой размерности
- $A * B = sum(A \odot B) \in \mathbb{R}$ — линейная операция свертки матрицы A ядром B, элемент $S(i,j)$ матрицы S, согласно формуле (1)
- $conv(A, B)$ — операция сверточного слоя, матрица $S = \{S(i,j)\}$, описанная в 1.2.
- $M'(p,q)$ — подматрица матрицы M, образованная пересечением H_k строк и W_k столбцов матрицы M, взятых последовательно, $M'(p,q) = \{m_{p+i,q+j}\}_{i=0,j=0}^{H_k-1,W_k-1}$, $M = \{m_{ij}\}$

Тогда результат частичного сверточного слоя может быть вычислен в несколько шагов:

1. Вычисляем матрицу элементов $M_{sum} = \{sum(M')\} = conv(M, U)$;
2. Вычисляем матрицу $M_{ratio} = \{m_{ratio}\}$, где $m_{ratio} = \dfrac{(H_k \cdot W_k)}{(m_{sum} + 10^{-8})}$, 10^{-8} необходимо для того, чтобы не происходило деление на 0: полученные большие значения, как будет показано далее, учитываться не будут;
3. Элементы матрицы $sum(M)$ — целые числа, если значение равно 0, значит для M' окна $sum(M') = 0$ и значения пикселей на этом этапе не предсказываются, поэтому заменим все ненулевые элементы матрицы единицей: $m_{sum} \to \{0, 1\}$ согласно формуле (2);
4. Далее отсеем большие значения, полученные на шаге 2: $M_{ratio} = M_{ratio} \odot M_{sum}$;

5. Применим операцию свертки для изображения с наложенной маской, т.е $Y = \{y\} = conv((X \odot M), K)$, где $y = (X' \odot M') * K$;

6. Получим результат частичного сверточного слоя — новую матрицу изображения $X_{new} = \{x_{new}\} = Y \odot M_{ratio}$, т.е $x_{new} = \dfrac{sum(U)}{sum(M')}$ $X_{new} = pconv((X, M), K)$.

Архитектура сети. Используется U-net подобная архитектура, как в [24], в которой все сверточные слои заменены слоями частичными сверточными слоями PConv2D. Размеры входного и выходного слоев были изменены на 256×256 для увеличения пропускной способности сети. Также используется предобученная сеть VGG-16 [26] для вычисления пиксельных признаков на результатах работы сети на каждом шаге обучения.

Датасет и обучение. В качестве датасета используется набор изображений ImageNet 256×256 объемом 25 000 изображений для обучающего множества, 2 500 для валидационного и 5 000 для проверочного. Модель обучалась по эпохам. В каждой эпохе модели предоставлялось по 10 000 обучающих примеров и 1 000 валидационных. Размер батча равен 4, коэффициент обучения 0.0005. Для обучения сети были использованы облачные вычисления, предоставляемые компанией Google. Время обучения составило чуть более 25 часов. После каждой эпохи состояние модели сохранялось, если значение функции потерь (ошибка) на валидационном множестве было меньше по сравнению с ошибкой на предыдущей эпохе. Конечное число эпох равно 24, после чего изменения значения функции потерь на валидационном множестве после каждой эпохи оказались менее 0.5%. График изменения ошибки представлен на рис. 9.

Эксперименты. После каждой эпохи состояние модели сохранялось, если значение функции потерь (ошибка) на валидационном множестве было меньше по сравнению с ошибкой на предыдущей эпохе. График ошибки по эпохам представлен на рис. 9.

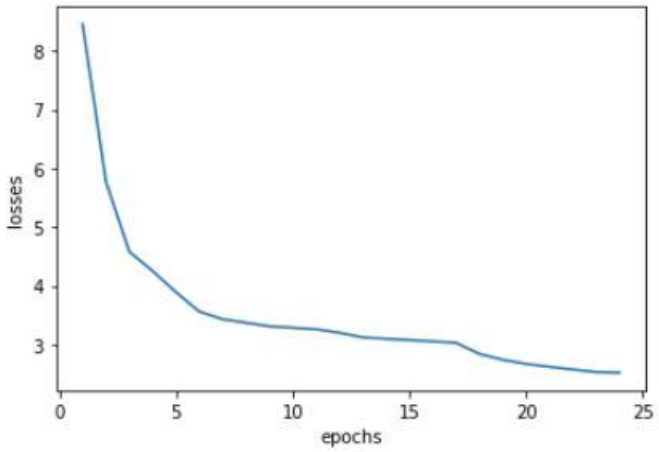

Рис. 9: График значений функции потерь в зависимости от эпохи обучения.

Некоторые примеры работы алгоритма:

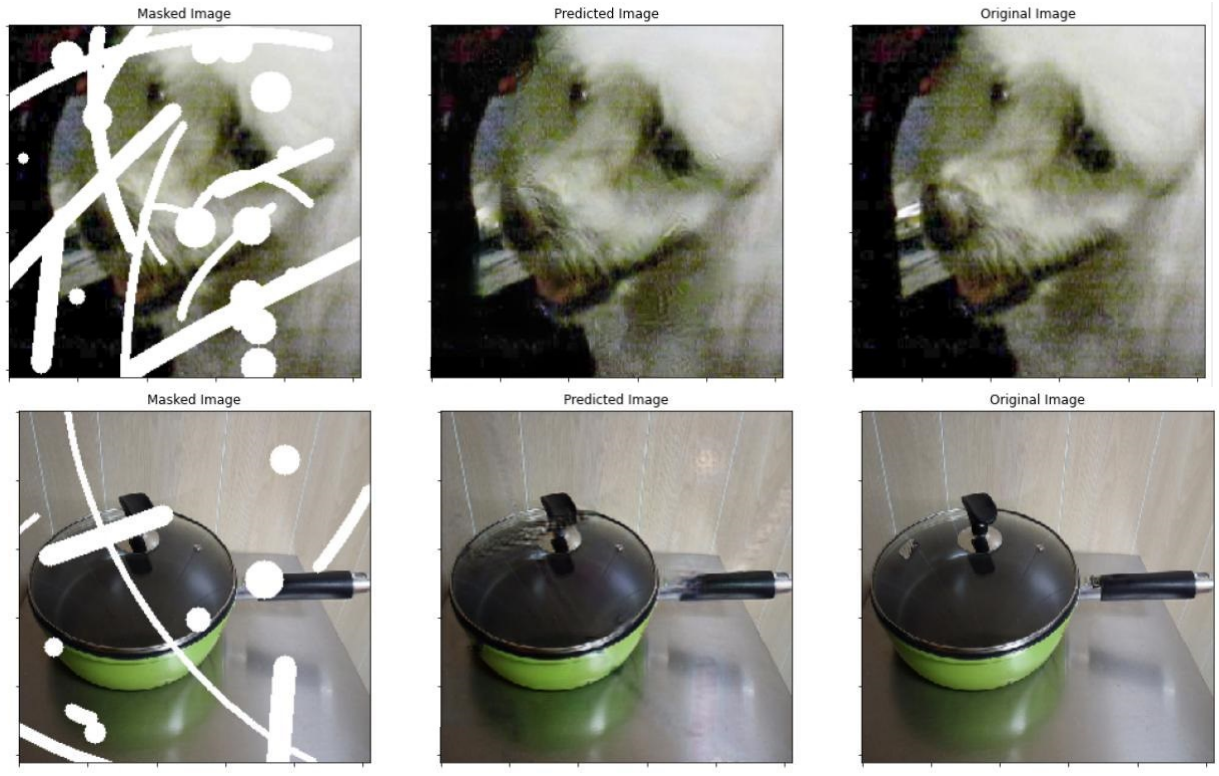

Рис. 10: Примеры работы алгоритма.

Были и неудачные примеры, например, сеть показывает плохие результаты с лицами людей (рис. 12) или с текстом (рис. 13). Это связано с тем, что сеть обучалась на общих фотографиях, в которых было мало примеров, содержащих лица или текст. Для решения этой проблемы мо-

жет быть применено обучение на датасетах, содержащих лица (например Celeba) или текст. Также сеть плохо себя показывает, если отсутствующая область слишком большая (рис. 14).

Рис. 11: Примеры работы алгоритма.

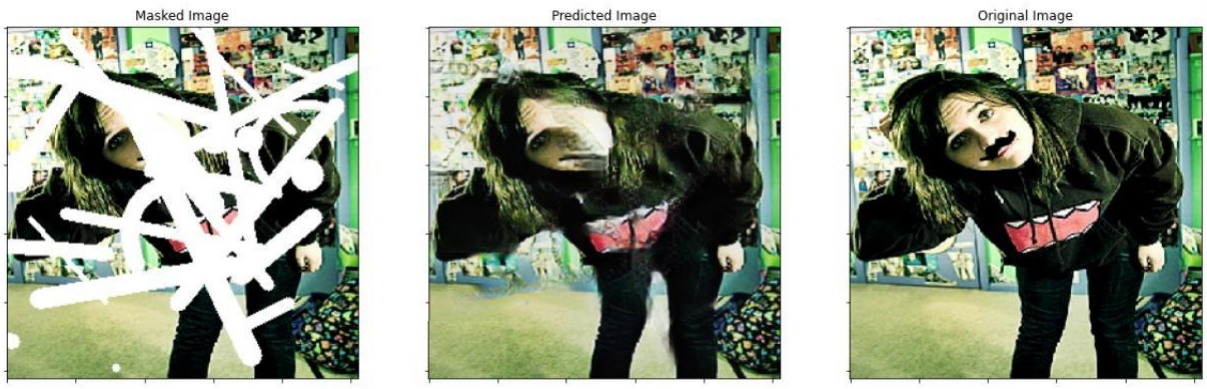

Рис. 12: Пример неудачной работы алгоритма на изображениии, содержащим лицо.

Рис. 13: Пример неудачной работы алгоритма на изображении, содержащим текст.

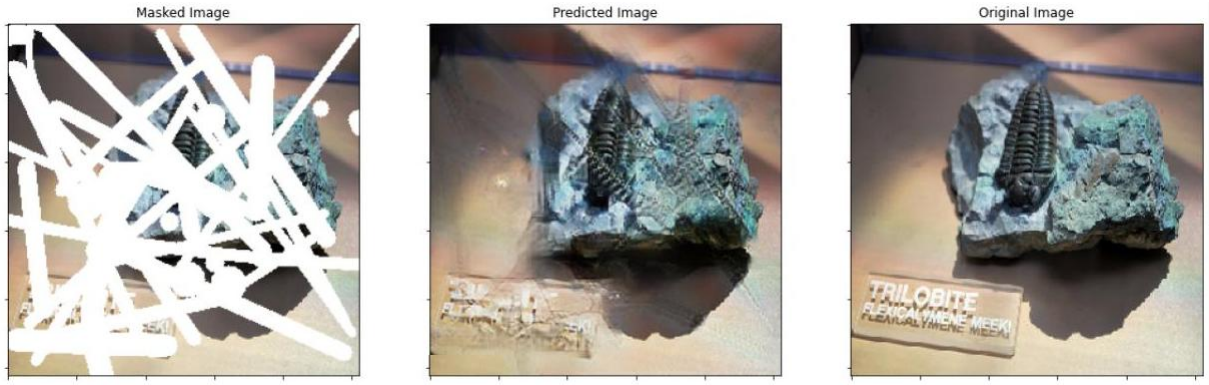

Рис. 14: Пример неудачной работы алгоритма на изображении с большой областью отсутсвующих значений.

Так же на рис.15 представлен пример того, как сеть прогрессировала в процессе обучения.

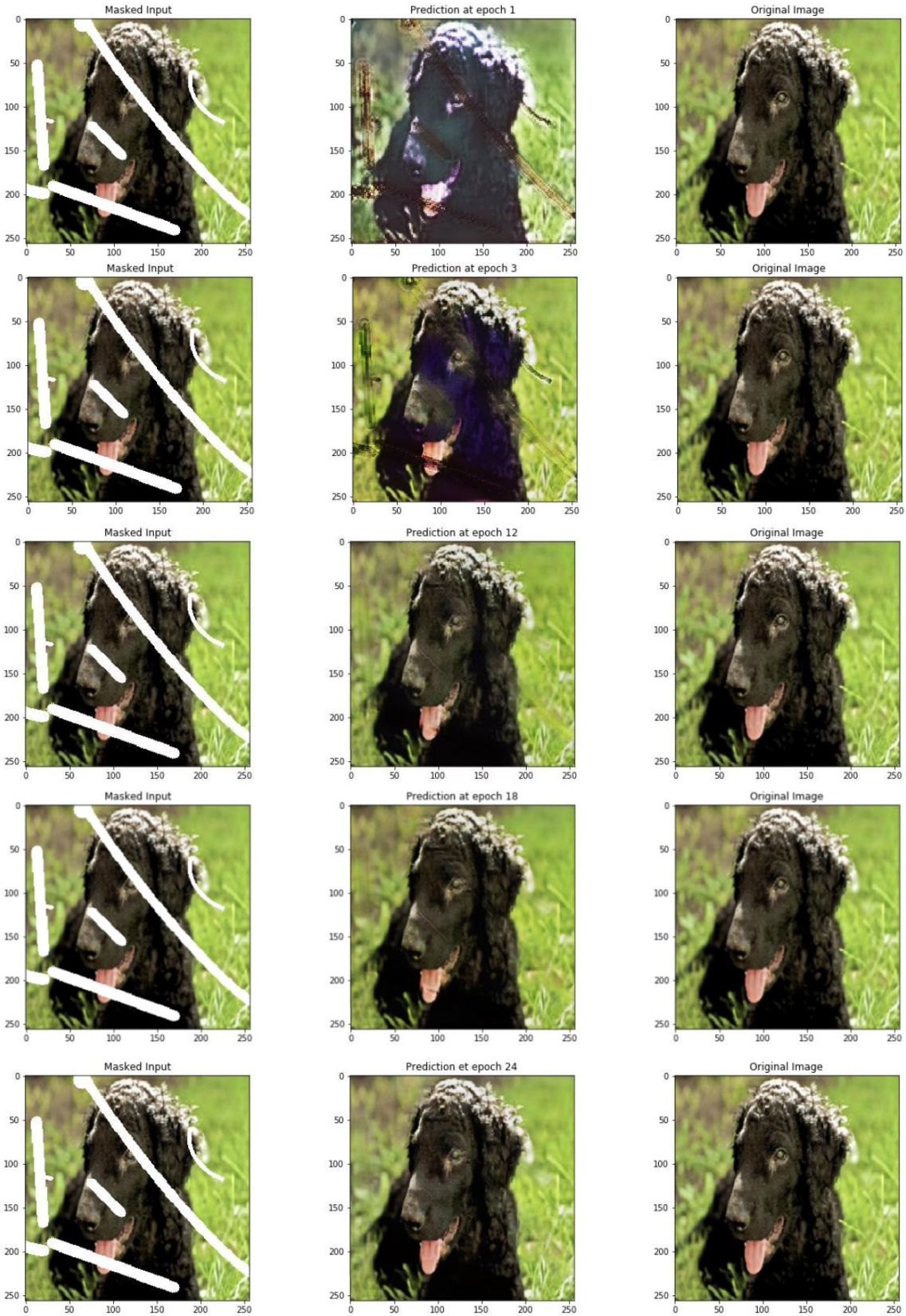

Рис. 15: Пример работы сети во время обучения по эпохам: 1, 3, 12, 18, 24

Заключение

В выпускной квалификационной работе была решена задача восстановления изображений с использованием искусственных нейронных сетей. В ходе работы были решены следующие задачи:

- Проведен обзор методов восстановления, разделенных на три группы: итеративные алгоритмы, методы, основанные на сверточных нейронных сетях, и методы, основанные на генеративно-состязательных нейронных сетях.
- Был выбран и реализован метод, использующий частичную сверточную сеть, который хорошо себя показал в востановлении изображений с нерегулярными областями отсутствующих значений. Для реализации было подготовлено множество картинок для обучения (25 000 изображений) и проверки качества работы сети, а также множество масок.
- Представлены результаты работы сети.

Список литературы

[1] McCalloch W.S., Pitts W. A logical calculus of the ideas immanent in nervous activity, 1943.

[2] Уоссермен Ф. Нейрокомпьютерная техника. М.: Мир, 1992, 184 с.

[3] Хайкин С. Нейронные сети: полный курс. 2-е изд. М.: Вильямс, 2008, 1104 с.

[4] Круглов В. В., Борисов В. В. Искусственные нейронные сети. М.: Горячая линия — Телеком, 2002, 382 с.

[5] Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение / пер. с анг. А. А. Слинкина. – 2-е изд., испр. – М.: ДМК Пресс, 2018. – 652 с.

[6] GANs in Action: Deep learning with Generative Adversarial Networks 2019, 276 с.

[7] Foster D. Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play, 2019, 330 с.

[8] Ronneberger O., Fischer Ph., and Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation

[9] Jegou S., Drozdzal M., Vazquez1 D., Romero A., Bengio Y. The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation

[10] Zheng S., Jayasumana S., Romera-Paredes b., Vineett V., Su Z., Du D., Huang C., and Torr Ph.: Conditional Random Fields as Recurrent Neural Networks

[11] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.

[12] Lecun, Y., et al.: "Gradient-Based Learning Applied to Document Recognition." Proceedings of the IEEE, vol. 86, no. 11, 1998, pp. 2278–2324., doi:10.1109/5.726791.

[13] Krizhevsky, Alex, et al.: "ImageNet Classification with Deep Convolutional Neural Networks." Communications of the ACM, vol. 60, no. 6, 2017, pp. 84–90., doi:10.1145/3065386.

[14] O. Elharroussa, N. Almaadeeda, S. Al-Maadeeda6 Y. Akbaria: Image inpainting: A review, Department of Computer Science and Engineering, Qatar university, Doha, Qatar, 2019

[15] C. Barnes, E. Shechtman, A. Finkelstein, D. B Goldman PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing, ACM Transactions on Graphics (Proc. SIGGRAPH), 2009

[16] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y.: (2014) Generative adversarial nets, In: Advances in neural information processing systems, pp 2672–2680

[17] S. Iizuka, E. Simo-Serra, H. Ishikawa: Globally and Locally Consistent Image Completion, ACM Transactions on Graphics, 2017

[18] J. Yu, Zh. Lin, J. Yang, X. Shen, X. Lu, Th. Huang: Generative Image Inpainting with Contextual Attention, IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018

[19] G. Liu, F. A. Reda, K. J. Shih, Ting-Chun Wang, A. Tao B. Catanzaro: Image Inpainting for Irregular Holes Using Partial Convolutions, ECCV, 2018

[20] P´erez, P., Gangnet, M., Blake, A.: Poisson image editing. ACM Transactions on graphics (TOG) 22(3), 313–318, 2003

[21] O. Ronneberger, Ph. Fischer, Th. Brox: U-Net: Convolutional Networks for Biomedical Image Segmentation, Computer Science Department and BIOSS

Centre for Biological Signalling Studies, University of Freiburg, Germany, 2015

[22] Long, J.; Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation, 2014

[23] Weerasekera C.S., Dharmasiri T., Garg R., Drummond T., Reid I.: Just-in-time reconstruction: inpainting sparse maps using single view depth predictors as priors. In: 2018 IEEE international conference on robotics and automation (ICRA), IEEE, 2018 c. 1-9

[24] Isola P., Zhu J.Y., Zhou T., Efros A.A.: Image-to-image translation with conditional adversarial networks. arXiv preprint, 2017

[25] Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style, 2015

[26] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition, 2014

[27] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. 2014

[28] Zhaoyi Yan, Xiaoming Li, Mu Li, Wangmeng Zuo, Shiguang Shan: Shift-Net: Image Inpainting via Deep Feature Rearrangement, 2018