

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Кафедра технологий моделирования сложных систем

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
«Программная инженерия»,
профессор департамента
программной инженерии, канд. техн. наук

_____ В.В. Шилов
« ____ » _____ 2018 г.

Выпускная квалификационная работа

на тему “Программа обнаружения изменений на аэрофотоснимках с помощью методов
глубинного обучения”

по направлению подготовки 09.03.04 «Программная инженерия»

ПРИЛОЖЕНИЯ

<p style="text-align: center;">Научный руководитель к.ф.-м.н., доцент кафедры технологий моделирования сложных систем</p> <p style="text-align: center;">Е. В. Бурнаев</p> <p style="text-align: center;">_____ Подпись, Дата</p>	<p style="text-align: center;">Выполнила студентка группы БПИ141 4 курса бакалавриата образовательной программы «Программная инженерия»</p> <p style="text-align: center;">М. В. Колос</p> <p style="text-align: center;">_____ Подпись, Дата</p>
---	---

Москва 2018

Реферат

Данная работа посвящена реализации различных алгоритмов обнаружения изменений на аэрофотоснимках с использованием сверточных нейросетей.

Приведен сравнительный анализ современных подходов к решению поставленной задачи – с использованием классических методов компьютерного зрения, статистического анализа и глубинного обучения. Обоснованы преимущества решений с применением сверточных нейронных сетей.

Работа содержит описание теоретической части – алгоритмы, использованные метрики, методы валидации – и детали практической реализации сиамских модификаций state-of-the-art CNN архитектур, генерации синтетических данных, которые потребовались в процессе работы наряду с описанием процесса обучения нейронных сетей и обработки данных. Запуск модели представлен в веб-приложении.

Разработанное решение может найти применение в областях мониторинга охранных зон, кадастровой аналитики и управления чрезвычайными ситуациями.

Работа содержит 39 страниц, 3 главы, 26 рисунков, 5 таблиц, 22 источника, 4 приложения

Ключевые слова: обнаружение изменений, глубокое обучение, генерация синтетических данных, сиамские нейронные сети.

Abstract

This work is dedicated to aerial imagery change detection using deep convolutional neural networks.

The work contains an overview of existing approaches for the given problem including both – common computer vision and statistical analysis methods and ones using deep learning. Convolutional neural networks' superiority over other approaches explained.

The work contains both theoretical basis – the hypothesis suggested, metrics used and validation techniques along with implementation details – siamese modifications of state-of-the-art CNN architectures, generation of synthetic data followed by neural networks training and data processing. Model inference was wrapped into a web application.

The proposed methodology will find possible applications in regional resource monitoring, disaster management, land development, and environmental planning

This paper contains 39 pages, 3 chapters, 26 illustrations, 5 tables, 22 bibliography items, 4 appendices.

Key words: change detection, deep learning, synthetic data generation, siamese neural networks

Основные определения, термины и сокращения

CNN – Convolutional Neural Network, сверточная нейронная сеть.

Датасет - размеченный набор данных.

ImageNet – самая массивная базы данных размеченных изображений (около 15 млн), представляющих более 20 000 классов.

State-of-the-art – понятие превосходства модели в рамках конкретной задачи (например, семантическая сегментация, классификация) на конкретных эталонных данных (benchmark datasets), например – ImageNet, CIFAR.

Тензор – упрощенно – многомерный массив.

GPU – Graphical Processing Unit, графический процессор.

Learning rate – параметр градиентных методов оптимизации, позволяющий управлять величиной коррекции весов на каждой итерации.

Supervised – методы обучения с учителем (включающие полноценную разметку). Также существуют методы обучения без учителя (unsupervised) и подходы semi-supervised (разметка неполная).

Аугментация - целенаправленное искажение исходного материала для увеличения датасета.

Пиксель - наименьший логический элемент двумерного цифрового изображения в растровой графике.

Ground truth – истина в контексте разметки датасета.

Энкодер – блок нейросети для повышения размерности и извлечения признаков.

Декодер – блок нейросети для снижения размерности.

Генератор – итератор подвыборок из исходных данных для обучения нейросети. Принимает на вход полный набор данных и для каждой итерации выдает подвыборку $\{[X]_k, [y]_k\}$, где k – размер подвыборки. Используется для работы с большими объемами данных.

Картографическая проекция - математически определенный способ отображения поверхности Земли.

TP — истинно-положительное решение.

TN — истинно-отрицательное решение.

FP — ложно-положительное решение.

FN — ложно-отрицательное решение.

Footprint – отпечаток здания

WMTS – (Web Map Tile Service) - стандарт веб публикации географических карт с использованием кэшированных пирамидальных фрагментов изображений (tile).

Содержание

Реферат	2
Abstract.....	3
Основные определения, термины и сокращения.....	4
Введение	7
Глава 1. Обзор и анализ источников, выбор методов	9
1.1. Алгебра изображений.....	9
1.2. Классификационные подходы.....	9
1.3. Методы, основанные на извлечении и трансформации признаков	9
1.4. Условные случайные поля.....	10
Выводы по главе	10
Глава 2. Алгоритмы, модели и данные.....	10
2.1. Сверточные нейронные сети	10
2.1.1. Топология	11
2.1.2. Слои.....	11
2.1.2.1. Сверточные	11
2.1.2.2. BatchNormalization	13
2.1.2.3. Активации	14
2.1.2.4. Pooling	15
2.1.2.5. Полносвязные	15
2.2. Метрики и функции потерь	16
2.3. Данные.....	17
2.3.2. Калифорнийские пожары.....	17
2.3.3. Синтетические данные	17
2.3.4. Inria Aerial Image Label Dataset.....	20
2.3.5. Functional Map of the World.....	20
2.3.6. Север Москвы.....	20
2.3.7. Цунами в Японии.....	20
2.4. Алгоритмы.....	21
2.4.2. Используемые аугментации.....	21
2.4.1. Сегментационный подход.....	21
2.4.1.1. Параллельная сегментация.....	21
2.4.1.1.1. Обучение сегментационной модели.....	21
2.4.1.1.2. Анализ разности масок сегментации.....	23

2.4.1.2.	Реализация алгебры изображений с помощью сверточных нейросетей	23
2.4.1.2.1.	Сиамский U-net.....	23
2.4.1.2.1.1.	Независимые подсети VGG16	26
2.4.1.2.1.2.	Разделенные веса Inception Resnet v2	26
2.4.2.	Классификационный подход	27
2.4.3.	Результаты экспериментов.....	29
2.4.3.1.	Сегментация.....	29
2.4.3.2.	Классификация	29
	Выводы по главе, выбор алгоритма.....	30
	Глава 3. Программная реализация	30
2.5.	Инструменты разработки.....	30
2.4.4.	Обучение и backend	30
2.4.5.	Frontend	31
2.4.6.	Развертывание	31
2.4.7.	Генерация искусственного набора данных	32
2.6.	Веб-приложение	32
1.1.1.	Tileservers	34
1.1.2.	Frontend	35
1.1.3.	Backend	36
1.1.3.1.	API	36
1.1.3.1.1.	Реализация распределенной очереди запросов	36
	Заключение.....	37
	Список литературы.....	38

Введение

Обнаружение изменений на последовательности изображений является одной из фундаментальных задач машинного зрения.

На сегодняшний день особенно актуальны методы автоматизированного мониторинга земной поверхности. Примерами прикладных задач являются разметка территорий, пострадавших в результате ЧС, мониторинг охранных зон, кадастровая аналитика.

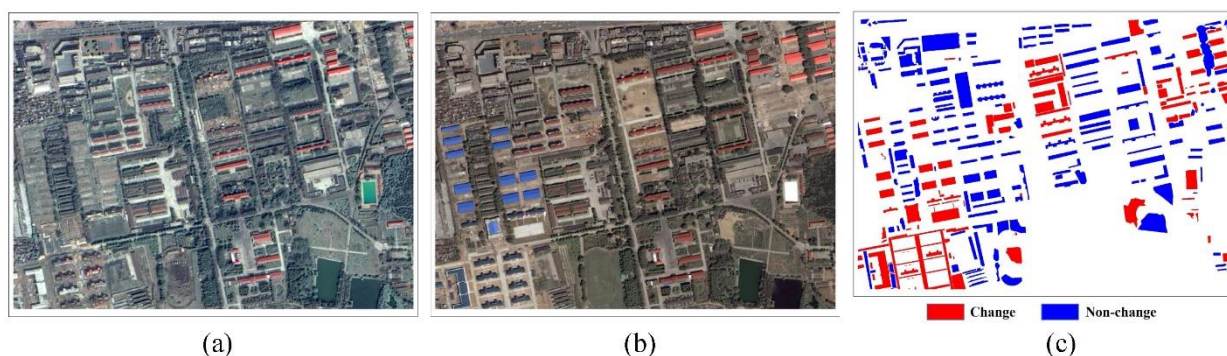


Рисунок 1. Пример решения задачи обнаружения изменений

В случае стихийных бедствий, таких как землетрясение, лесные пожары или цунами, необходимо оперативно разметить пострадавшие районы для дальнейшей организации спасательных операций, оценки ущерба и информирования населения.

В настоящее время большая часть экстренной разметки выполняется вручную – пары изображений до и после происшествия сравниваются, в ходе чего отмечаются пострадавшие районы. Этот крайне кропотливый, длительный процесс, в то время как скорость является одним из главных приоритетов наряду с точностью.

Очевидная тривиальность обнаружения изменений в случае с одинаковыми условиями съемки и освещения (тогда решением задачи будет попиксельная разница с предварительной классификацией) является обманчивой. Входные данные задачи представляют собой пару аэрофотоснимков одной и той же территории. Однако, если данные представлены одним источником (аэрофотоаппаратом) варьируются только угол съемки и освещение. В противном случае, когда источники разные, нет гарантии совпадения FOV и проекционных параметров.

Наиболее эффективные методы решения подобных задач используют сверточные нейронные сети. Они позволяют извлекать устойчивые признаки из изображений, за счет чего обладают большой обобщающей способностью.

Целью работы является разработка сервиса для обнаружения изменений на данных аэрофотосъемки.

Для достижения цели необходимо решить следующие задачи:

1. Изучить существующие алгоритмы и подходы к обнаружению изменений на аэрофотоснимках;
2. Предложить и реализовать модификации текущих методов для улучшения качества работы моделей.
3. Разработать веб-приложение для демонстрации возможностей нейросетевых алгоритмов обнаружения изменений на аэрофотоснимках.

Работа структурирована следующим образом: первая глава посвящена обзору существующих подходов к обнаружению изменений на аэрофотоснимках. Вторая глава описывает метод, использующий сверточные нейронные сети и искусственные данные (с описанием процесса их генерации). Третья глава содержит технические детали реализации проекта и результаты работы алгоритма на тестовых данных, после чего следует заключение и список литературы.

Глава 1. Обзор и анализ источников, выбор методов

1.1. Алгебра изображений

Наиболее прямолинейный подход к обнаружению изменений подразумевает анализ разности двух сегментационных масок или каких-либо других статистик. Среди подобных методов разность является наиболее точным методом, однако чувствительным к смещениям и изменению контраста.

Классический подход к анализу попиксельной разности использует попарные разности каналов мультиспектральных изображений (multivariate alteration detection) [1]. Анализ векторов разности - Change Vector Analysis (CVA) [2] представляет собой концептуальное расширение анализа попиксельной разности изображений с интегрированной теоретической основой. Данный метод описывает не только амплитуду изменений, но и их направление за счет использования мультиспектральных данных. Анализ векторов разности менее чувствителен к смещениям, но недостаточно точен, так как опирается только на статистические признаки. Кроме того, использование мультиспектральных данных высокого разрешения не всегда возможно в силу их высокой стоимости.

1.2. Классификационные подходы

Классификационные подходы опираются на маску объектов исходного изображения. По ней определяются фрагменты во втором изображении – кандидаты на классификацию.

В данном подклассе методов предполагается, что маска объектов известна изначально. Предложено большое число классификационных алгоритмов [3,4], нацеленных на повышение точности обнаружения изменений при известной карте объектов. В частности, метод обнаружения изменений с использованием подхода transfer learning [5] был предложен для обновления карт местности с помощью классификации временных снимков.

Данный подход является очень перспективным – state-of-the-art архитектуры сверточных нейросетей показывают качество классификации изображений, превосходящее человеческие способности [17].

1.3. Методы, основанные на извлечении и трансформации признаков

Одним из способов решения задачи обнаружения изменений является ручное извлечение признаков. Как правило, используются физические признаки – такие как NDVI, различные вегетационные и водные индексы [6, 7], что широко применяется для распознавания растительности и мониторинга охранных зон. Помимо физических признаков также применяются различные статистики, например в одной из последних работ по change detection в качестве признаков были использованы дивергенция Кульбака-Лейблера, взаимная информация (mutual information) и различные текстурные метрики. Полученные признаки были поданы в SVM.

Подобные методы привлекательны своей интерпретируемостью. Однако даже не смотря на такой существенный недостаток как полуавтоматика (что снижает скорость и повышает стоимость), подобные алгоритмы всегда имеют свой «потолок» качества – количество ручных признаков. В какой-то момент расширение представления объекта становится

невозможным, так как все статистики уже собраны. Сверточные нейронные сети здесь обладают значительным преимуществом – они позволяют извлекать признаки различной иерархии (от низкоуровневых до высокоуровневых) и обобщают объект через скрытое представление. Модификации архитектур и использование различных функций потерь позволяют экспериментировать с представлением объекта, что открывает новые перспективы качества нейросетевых моделей.

1.4. Условные случайные поля

Условные случайные поля (Conditional Random fields) – это дискриминативная вероятностная модель. Она применяется для моделирования постериорной вероятности разметки при условии наблюдаемых данных. Модель CRF была предложена Лафферти в 2001 году [8] для одномерной морфологической сегментации. Затем одномерная структура CRF была расширена до двумерной сетки, после чего метод получил широкое распространение в решении классификационных задач. Наиболее распространенной моделью CRF в обработке спутниковых снимков является парная CRF-модель, которая рассматривает спектральную информацию одного пикселя и контекстуальную информацию о соседних пикселях [7]. Различные вариации на тему использования CRF были предложены, в частности ансамблевая CRF-модель для достижения консенсуса нескольких исходных результатов изменений по различным признакам в вероятностной структуре и модель со связными регионами [10].

В рамках данной задачи условные случайные поля в особенности в рамках концептуальной комбинации с рекуррентными сетями [20] могут выступать в качестве метамоделей на более поздних стадиях разработки алгоритма обнаружения изменений и более сложных наборах данных (не парах изображений, а серий). Они способны учитывать симметрию изменений, что будет необходимо по мере расширения классов изменений и объемов данных.

Выводы по главе

Наиболее перспективным представляется развитие классификационных подходов. Направления для улучшений подразумевают реализацию сиамских архитектур state-of-the-art моделей, исследование подхода semi-supervised, а также использование сегментационных моделей.

Глава 2. Алгоритмы, модели и данные

2.1. Сверточные нейронные сети

Сверточная нейронная сеть – это архитектура нейронной сети, основанная на использовании сверток. Наиболее широко применяется для решения задач компьютерного зрения. Архитектура была впервые описана Яном ЛеКуном и др. в 1995 г. Идея чередования сверточных слоев подразумевает имитацию работы зрительной коры головного мозга: низкоуровневые признаки (контуры, штрихи) иерархически агрегируются в цельное представление изображения. В процессе обучения сеть настраивается на извлечение наиболее значимых признаков в рамках конкретной задачи.

2.1.1. Топология

Глубокие нейронные сети состоят из входного слоя, блока скрытых слоев и выходного слоя. Скрытые слои состоят из набора нейронов. В полносвязных сетях каждый нейрон связан со всеми нейронами предыдущего слоя и где нейроны в одном слое функционируют полностью независимо.

Полносвязные сети не масштабируются для работы с изображениями. Например, для трехканального изображения размера 200×200 пикселей единственный полносвязный нейрон имел бы $200 \times 200 \times 3 = 120\,000$ параметров. Такая сеть с несколькими слоями будет иметь огромное количество параметров, что не только вычислительно сложно, но и приводит к переобучению.

Архитектура сверточных нейронных сетей адаптирована для работы с изображениями. Слои сверточной сети расположены в трех измерениях: в ширину, высоту и глубину (рис. 11). Нейроны в каждом слое связаны только с небольшой областью впереди стоящего слоя, а не со всеми нейронами слоя, как в случае с полносвязной архитектурой.

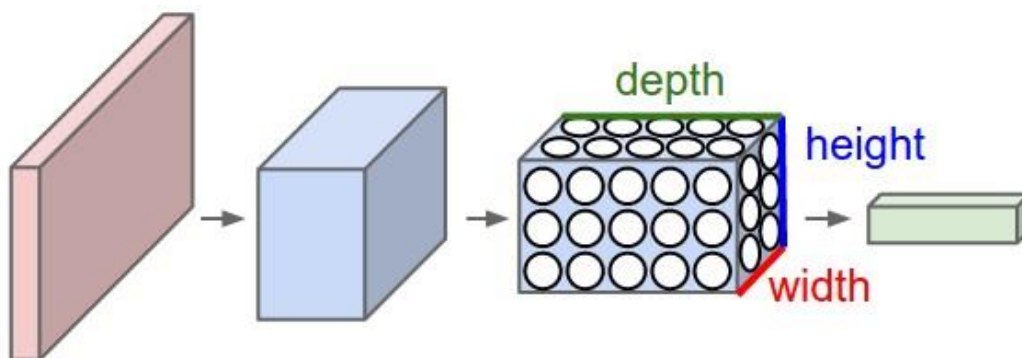


Рисунок 2. Визуализации топологии сверточного слоя

2.1.2. Слои

Архитектуры CNN состоят из сверточных слоев, нормализационных, слоев pooling и полносвязных. За счет использования сверточных слоев и пулинга сверточные нейронные сети являются инвариантными к сдвигам.

2.1.2.1. Сверточные

Сверточные слои используют операторы свертки для извлечения значимых признаков из изображения.

Операторы свертки заданной размерности являются параметрами нейросети и обучаются с помощью алгоритма обратного распространения ошибки.

Слой свертки включает в себя четыре основных параметра:

- 1) *количество фильтров*; они определяют глубину выходного тензора, каждый фильтр независимо обрабатывает сигналы предыдущего слоя;

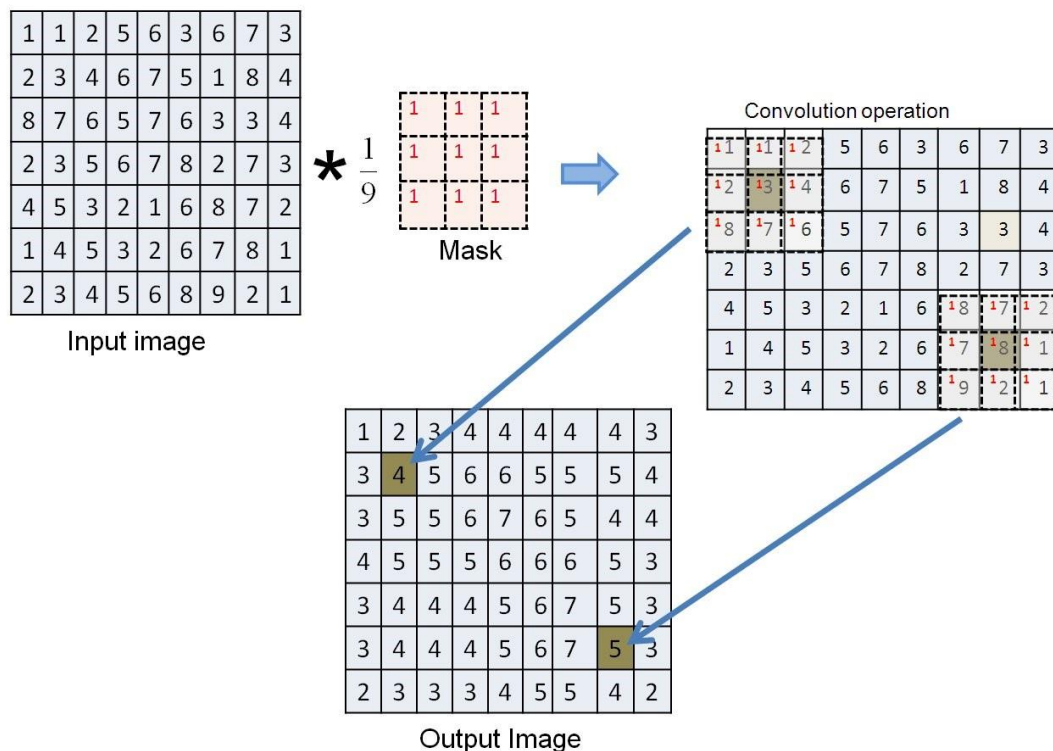


Рисунок 3. Свертка матрицы пикселей оператором размера 3x3

- 2) *размер матрицы оператора*;
- 3) *параметры отступа (padding)* - определяют стратегию сохранения размерности, в рамках которой используется заполнение границ матрицы нулями. Ширина отступа и является параметром;
- 4) *шаг свертки*, с которым фильтр проходит входную матрицу. Определяя частоту перемещения шаг свертки влияет на размер выходного тензора.

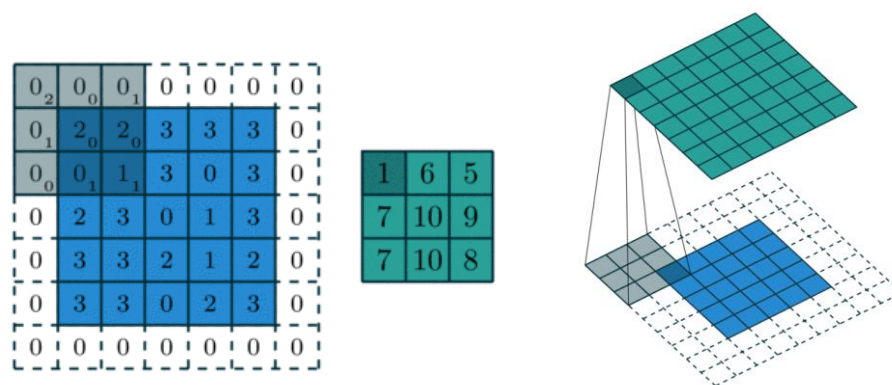


Рисунок 4. Иллюстрации padding с параметром 1 (слева) и 3 (справа)

Выходом сверточного слоя является трехмерный тензор с картами признаков (feature maps).

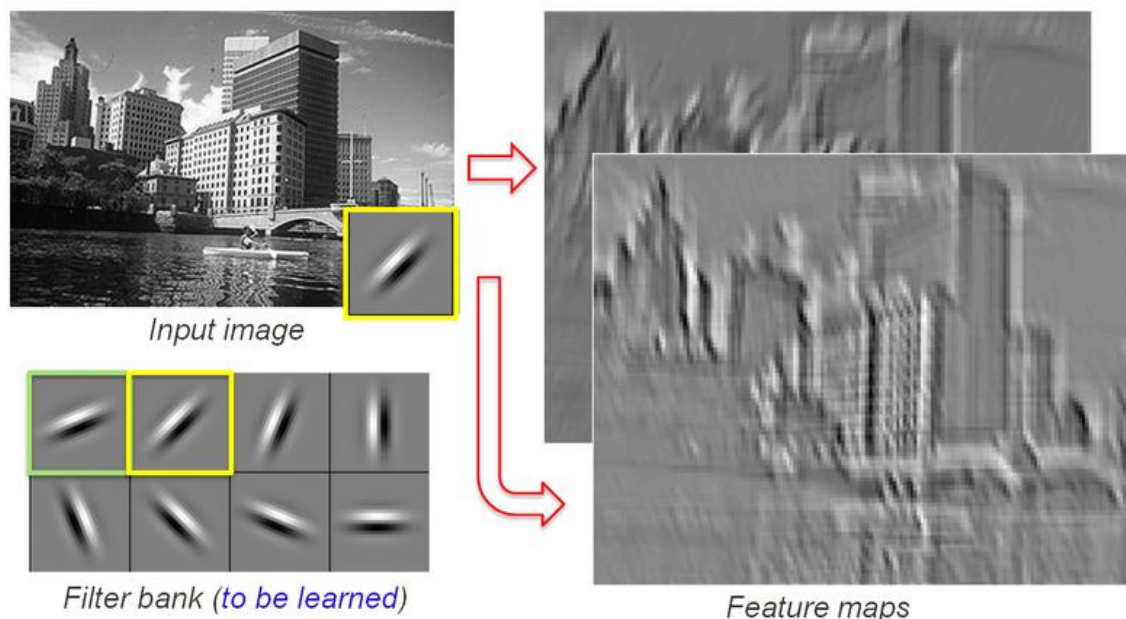


Рисунок 5. Иллюстрация создания независимых карт признаков каждым из фильтров

2.1.2.2. BatchNormalization

При работе с большим объемом изображений, на один проход нейросети данные подаются небольшими подвыборками фиксированного размера (batch).

В 2015 Google предложили следующий способ ускорения обучения нейронных сетей [20].

Стандартный способ нормировки подвыборки подразумевает вычитание среднего и деление на отклонение выборки. Результат - распределение с центром в 0 и дисперсией 1. Такое распределение ускорит сходимость сети, т.к. все значения лежат в одном диапазоне.

Концептуальным улучшением данной идеи является введение двух переменных для среднего и отклонения.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots x_m\}$;	
Parameters to be learned: γ, β	
Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$	
$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i$	// mini-batch mean
$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2$	// mini-batch variance
$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$	// normalize
$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i)$	// scale and shift

Рисунок 6. Оригинальный алгоритм из статьи [20]

Эти параметры будут входить в алгоритм обратного распространения ошибки. Таким образом, слой batch normalization содержит $2 \cdot k$ параметрами, где k – размер подвыборки. Как правило, он добавляется после сверточного слоя.

2.1.2.3. Активации

После нормализации выходов сверточного слоя применяется одна из стандартных функций активации.

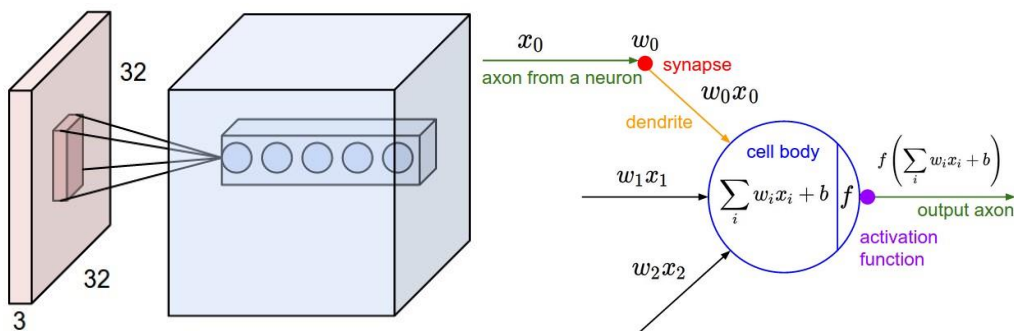


Рисунок 7. Иллюстрация последовательности применения активации

ReLU

$$\text{ReLU}(x) = \max(0, x)$$

В сверточных сетях наиболее широко используется функция активации ReLU (rectified linear unit, по аналогии с однополупериодным выпрямителем в электротехнике). Нейроны с данной функцией активации реализует простой пороговый переход в нуле: $f(x) = \max(0, x)$.

Ниже приведена иллюстрация применения функции ReLU к изображению.

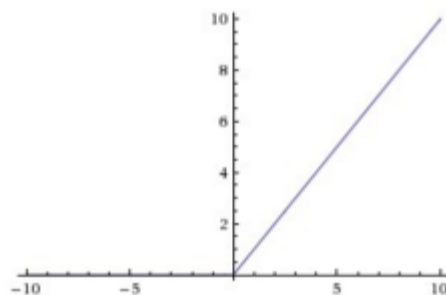


Рисунок 8. График функции ReLU

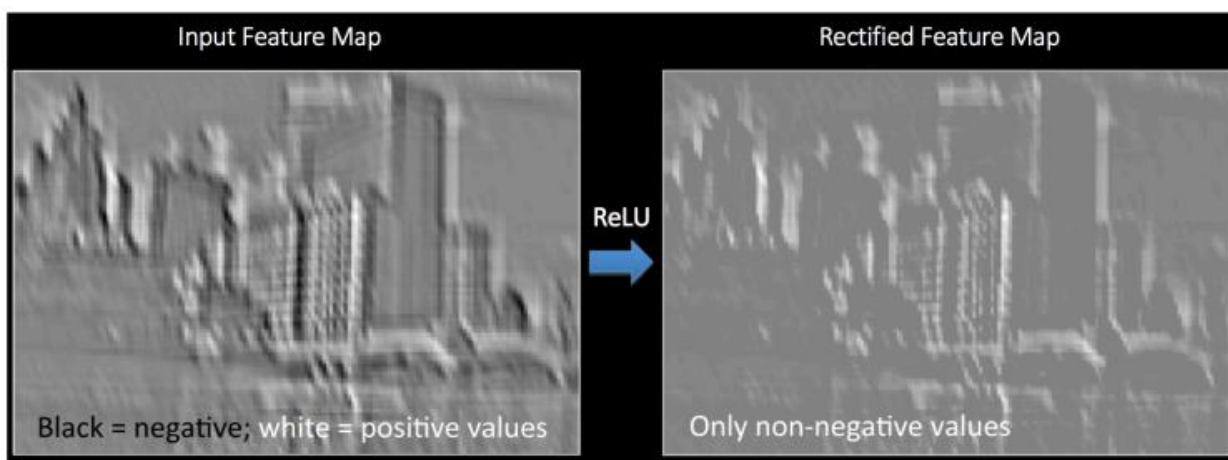


Рисунок 9. Иллюстрация применения функции ReLU

Softmax

$$\text{Softmax}(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$

Функция Softmax является обобщением логистической функции для многомерного случая и представляет собой взвешенную и нормированную на единицу сумму экспонент.

Функция преобразует вектор, сохраняя входную размерность так что, каждая координата полученного вектора представлена вещественным числом в интервале $[0,1]$ и сумма координат равна 1, что используется для вероятностной интерпретации.

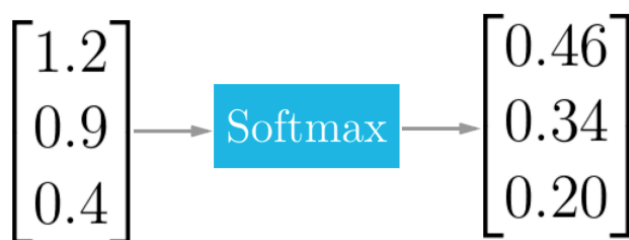


Рисунок 10. Иллюстрация применения функции Softmax

2.1.2.4. Pooling

Пулинг понижает размерность входной карты признаков, сохраняя наиболее значимую информацию. Пулинг основан на функции, применяемой к подматрице карты. Размер подматрицы и шаг – параметры слоя. Возможные функции – максимум (Max - наиболее распространенный), среднее (Average), сумма (Sum) и так далее.

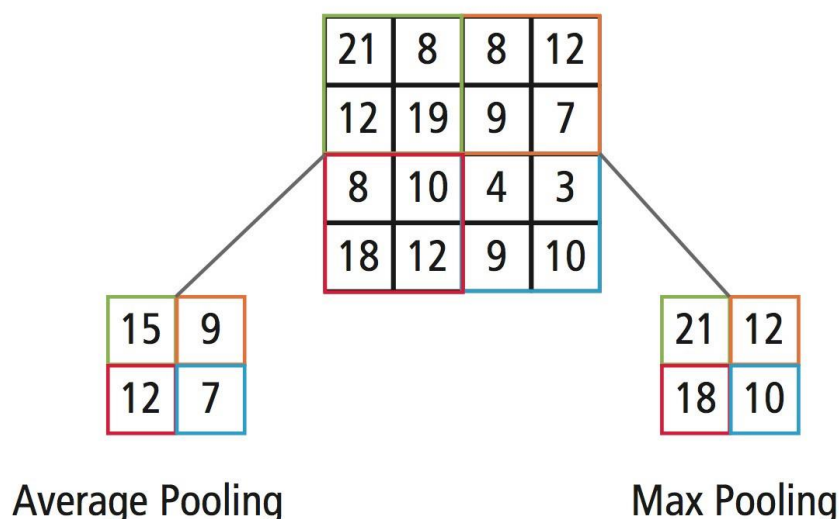


Рисунок 11. Иллюстрация пулинга на основе функций максимума и среднего

2.1.2.5. Полносвязные

Полносвязный слой представляет собой традиционный многослойный персептрон, который использует функцию активации Softmax на выходном уровне (могут быть использованы другие классификаторы, например SVM).

Выходные данные из сверточных и пулинговых слоев представляют собой высокоуровневые признаки входного изображения (на практике, однако, часто используют прием с residual connections, когда выходы более ранних слоев (более низкоуровневые признаки) также подаются на полносвязный слой). Цель полносвязного слоя - использовать эти функции для классификации входного изображения в различные классы на основе набора тренировочных данных.

2.2. Метрики и функции потерь

Мера Жаккара или intersection over union (IoU) – классическая метрика в задачах сегментации. Она описывает то, насколько точно предсказываемая область совпадает с истинной.

$$IoU = \frac{n(A \cap B)}{n(A) + n(B) - n(A \cap B)} = \frac{n(A \cap B)}{n(A \cup B)}$$

Бинарная кросс-энтропия (BCE) - функция потерь, которая позволяет приблизить распределение предсказания модели к реальному распределению, посредством максимизации вероятности наблюдаемых данных.

$$BCE = -\frac{1}{N} \sum_{i=1}^N (y_i \cdot \log(h_{\theta}(x_i)) + (1 - y_i) \cdot \log(1 - h_{\theta}(x_i)))$$

В ходе обучения функционал BCE минимизируется относительно параметров модели θ .

Точность (Precision) – это доля объектов, действительно принадлежащих данному классу относительно всех объектов которые система отнесла к этому классу.

$$Precision = \frac{TP}{TP + FP}$$

Полнота (recall) – это доля найденных классификатором объектов, принадлежащих классу относительно всех объектов этого класса в тестовой выборке.

$$Recall = \frac{TP}{P} = \frac{TP}{FN + TP}$$

F-мера является одной из самых важных метрик в классификации.

Очевидно, чем выше точность и полнота, тем лучше. Но в решении реальных задач максимальная точность и полнота не гарантированы одновременно, что вынуждает искать некий баланс. F-мера объединяет себе информацию о точности и полноте модели. Это очень ценная информация в рамках принятия решения о том какую модель запускать в конечном продукте, в частности, когда в предметной области (разметка территорий пострадавших в результате ЧС) цена ложно-положительных и ложно-отрицательных результатов очень высока.

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

F-мера представляет собой гармоническое среднее между точностью и полнотой. Она стремится к нулю, если точность или полнота стремится к нулю.

2.3. Данные

Наборы данных представлены в порядке появления и применения при разработке.

2.3.2. Калифорнийские пожары

В декабре 2017 года в Калифорнии произошли лесные пожары, после чего DigitalGlobe [11] оперативно опубликовали спутниковые данные пострадавшей территории (провинции Вентура и Санта-Роза).

Данные были размечены вручную с помощью онлайн ГИС-сервиса. Отношение площади пикселей целевого класса к общей площади изображения – 2,2%.

Набор данных Калифорнии является основным в данной работе, так как представляет наиболее востребованную область применения (emergency mapping) разрабатываемого алгоритма.

2.3.3. Синтетические данные

В рамках экспериментальной части работы была выдвинута гипотеза о том, что искусственные данные повысят качество модели за счет увеличения объема тренировочных данных и 100% точной разметки. Данная гипотеза опирается на работы [12] и [13], в которых использование искусственных данных показало впечатляющие результаты.

Таким образом, целью текущего этапа было создание синтетического набора данных, который представляет собой набор пар изображений, которому соответствует бинарная маска ground truth: 1 при наличии структурных изменений для одного из целевых классов иначе - 0. Набор данных должен содержать 4 целевых класса (листва, вода, дороги и здания) и случайный набор нецелевых изменений: различные углы съемки, перемена освещения.

2.3.3.1. Реализация.

В ходе реализации поставленная задача была разбита на следующие подзадачи:

- Генерация геометрии, текстурирование моделей
- Генерация и текстурирование террейна
- Реализация алгоритма переключения объектов и создания случайных нецелевых изменений.

2.3.3.2. Генерация геометрии, текстурирование моделей

Для создания сцен больших масштабов (порядка 10 км²) был выбран алгоритм генерации геометрии, основанный на использовании данных картографической разметки.

Входные данные: файл xml, экспортированный с портала OpenStreetMap. Он содержит в себе информацию о размеченных объектах в ограниченной прямоугольной области в векторном формате, средняя высота домов, два набора текстур для фасада и крыш.

Алгоритм

1. Начинаем проход по каждому полигону, рассматривая пару вершин на очередном шаге внутреннего цикла. Для каждой пары последовательных точек строим стену высотой в заданном диапазоне вокруг средней высоты.

2. Дублируем полигон в качестве крыши.
3. Вычисляем размер тайла для наложения текстуры на основе данных о площади и высоте дома.
4. Выбираем случайную текстуру фасада и крыши. Накладываем текстуру с вычисленным размером тайла.

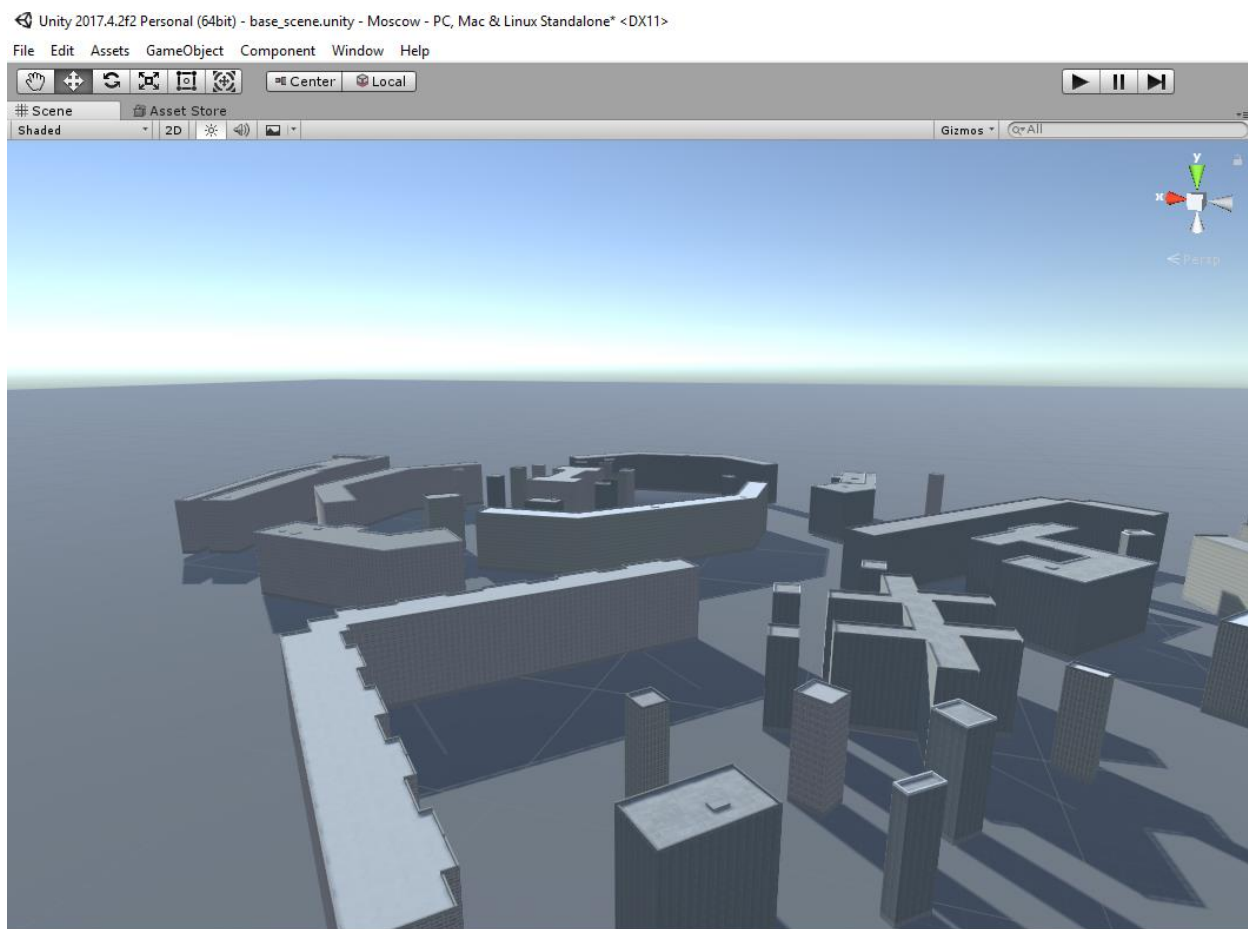


Рисунок 12. Здания, сгенерированные на основе xml-файла территории Мытищ

2.3.3.3. Генерация и текстурирование террейна

Входные данные: карта высот для заданной территории (SRTM) и слой OpenStreetMap, текстуры травы, земли, асфальта.

Алгоритм

1. Данные представлены в проекции WGS84 (EPSG4326 - широта, долгота), что означает, что визуально данные немного искажены, следовательно, требуют репроекции. Целевая система координат - EPSG:32653. На текущем шаге производим репроекцию данных с помощью `gdal_warp` [14]
2. Генерация геометрии на основе карты высот.
3. Создание шейдера на основе векторных масок со слоя OpenStreetMap. Текстурирование террейна.

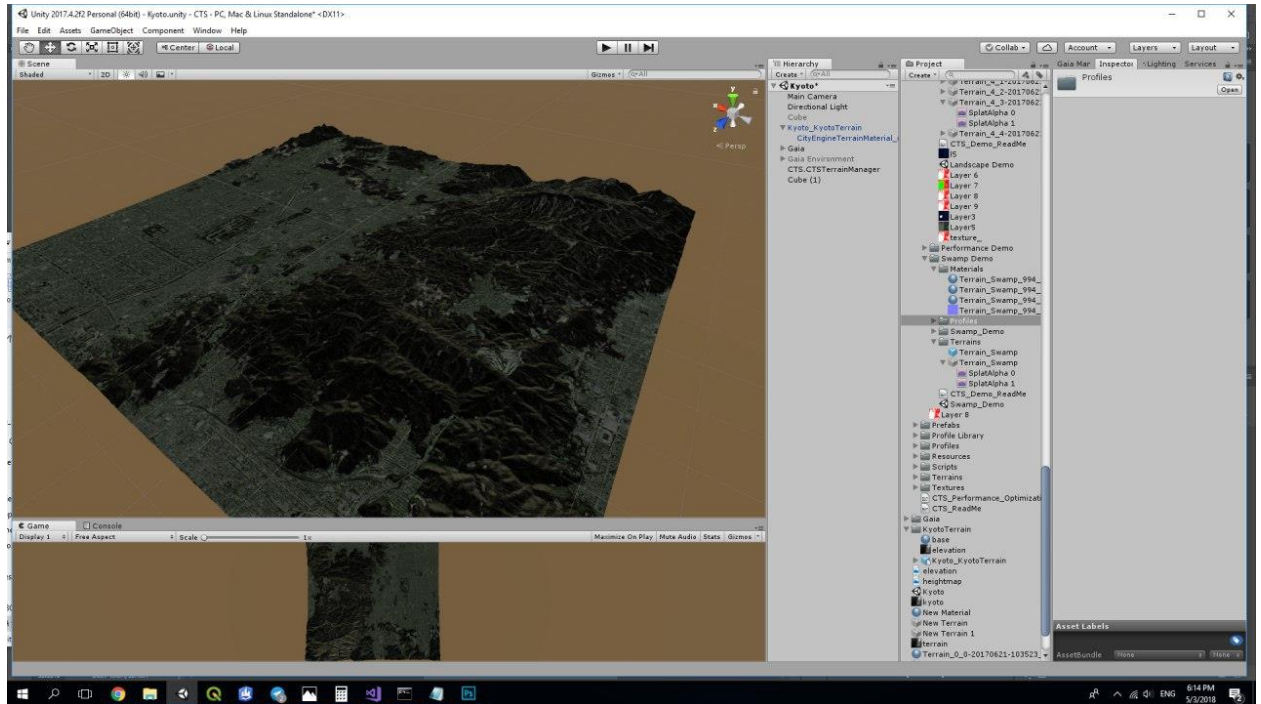


Рисунок 13. Текстурирование террейна, Unity

2.3.3.4. Реализация алгоритма переключения объектов и создания случайных нецелевых изменений

Предполагаемая высота симулируемой аэрофотосъемки – 1500 м. Координата высоты (Y) была зафиксирована при нижеописанном перемещении камеры.

Входные данные: сгенерированные модели в формате fbx, диапазон углов съемки для вариации, время суток.

Алгоритм

1. Сбор всех объектов из ресурсов
2. Для очередного кадра выбираем 30% объектов заданного класса и выключаем их.
3. Меняем угол обзора и время, снимаем ту же сцену.
4. Помещаем на сцену только те объекты, которые были отключены. Рендерим кадр. Получившееся изображение – маска ground truth.

Для создания набора данных на OpenStreetMap была выбрана территория Питсбурга.

Сгенерированный набор содержал 15 изображений 5000x5000, включающий 5% площади целевого класса.

К сожалению, использование искусственных данных не дало прироста в качестве, а в некоторых случаях и вовсе его ухудшало. Это объясняется тем, что изображения отрендеренные через игровой движок, во-первых, не являются достаточно реалистичными (визуально), а также содержат другие низкоуровневые признаки (например, шумы) нежели аэрофотоснимки. Подобные проблемы также возникают при попытке применить модель обученную на снимках с одного спутника на изображениях с другого аппарата. Для решения этой проблемы применяются различные техники domain adaptation с использованием генеративно-состязательных сетей.

От дальнейших экспериментов в этой области было принято решение отказаться (по причине большой трудоёмкости методов) в пользу исследования моделей, обученных наборах реальных данных с открытых соревнований.

2.3.4. Inria Aerial Image Label Dataset

Набор данных Inria Aerial Image Label был представлен на одноименном соревновании по объектной сегментации [15]

Датасет включает:

- Покрытие 810 км² (405 км² для обучения и 405 км² теста)
- Геоскорретированную трёхканальную аэрофотосъемку с пространственным разрешением 0.3 м
- Бинарные маски для двух классов: *здания* и *не здания* (только для тренировочного набора данных)

Набор данных содержит крайне разнообразные изображения городских поселений – от густонаселенных районов (например, Сан-Франциско) до альпийских городов (Лиенц в Австрии).

Данный набор данных позволил обучить сегментационную модель с большой обобщающей способностью, что очень сильно повысило качество экспериментальных алгоритмов.

2.3.5. Functional Map of the World

Данные с одноименного соревнования по объектной классификации спутниковых снимков.

Датасет представлен в двух версиях: с многоканальными спутниковыми данными и RGB.

Разметка представляет собой ограничивающие прямоугольные области (bounding boxes) для объектов из различных классов зданий: жилые, офисы, башни, заводы и т. д.

Классификационная модель, используемая в дальнейших экспериментах, была обучена на данном датасете.

2.3.6. Север Москвы

Разновременные данные с открытых WMTS-слоев Mapbox (до) и Yandex (после) были отобраны вручную – на карте были выделены интересующие регионы, каждый около 2км².

Полученные растры были размечены обученной сегментационной сетью. Разметка для обнаружения изменений была получена сравнением объектов по IoU.

Данный датасет был создан для проверки обобщающей способности модели на несколько других данных.

2.3.7. Цунами в Японии

ABCD (AIST Building Change Detection) – это размеченный набор данных для классификации разрушений. Датасет содержит пары изображений с изображением одной местности до и после цунами. Целевой класс представляет собой разрушенные дома

Данный набор данных был использован в качестве сравнения предложенных методов с результатами из одноименной статьи [16].

2.4. Алгоритмы

Сети в нижеописанных экспериментах были обучены с использованием фреймворка CuDNN для ускорения нейросетевых библиотек на видеокарте GeForce GTX 1080.

В качестве оптимизаторов были использован Adam с learning rate указанным для каждого эксперимента.

2.4.2. Используемые аугментации

Для повышения устойчивости моделей к различным модификациям входных данных при обучении были использованы аугментации из различных семейств: аффинные преобразования, цветокоррекция и зашумление.

2.4.1. Сегментационный подход

В рамках решения задачи сегментации функция потерь представляла собой взвешенную сумму IoU (0.6) и бинарной кросс-энтропии (0.4).

2.4.1.1. Параллельная сегментация

Данный подход подразумевал использование независимую сегментацию каждого из пары снимков и дальнейшую обработку двух полученных масок.

2.4.1.1.1. Обучение сегментационной модели

В качестве исходной архитектуры была взята state-of-the-art модель от Google из семейства ResNet.

Inception Resnet V2 Network

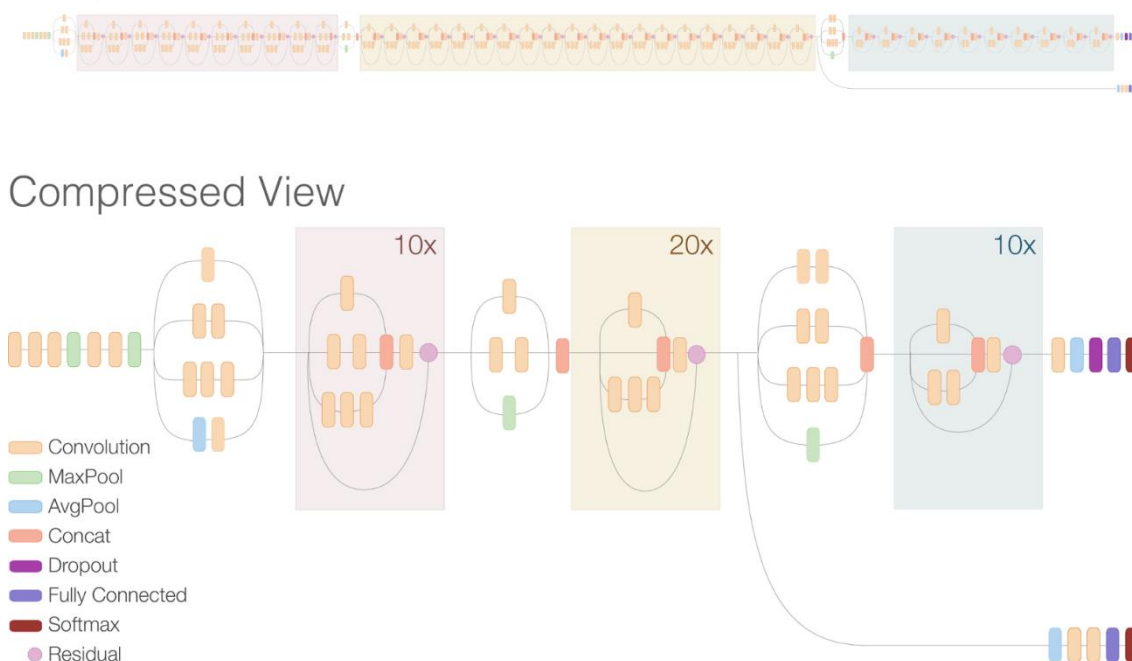


Рисунок 14. Полное (сверху) и сжатое представление архитектуры Inception Resnet V2

ResNet архитектуры были предложены для решения так называемой проблемы тренировки очень глубоких сетей [21]. Проблема заключалась в том, что увеличение количества слоев не давало прироста качества, и более того, приводило к переобучению.

Идея ResNet состоит в следующем:

- рассмотрим хорошую модель оптимальной глубины;
- очевидно, мы можем получить более глубокую модель с таким же качеством (в данном случае важно, что *не хуже*), добавив некоторое число «прозрачных» (identity) слоев. Такие слои просто будут пропускать сигнал, не влияя на качество модели.

Наблюдение, что всегда можно получить модель не хуже identity, и является основной концепцией ResNets.

Таким образом формулировка задачи строится следующим образом: более глубокие уровни должны предсказывать *разницу* между тем, что выдают предыдущие слои и канонической разметкой; следовательно, нерелевантные веса всегда могут быть обращены в 0, а сигнал быть пропущен, как через «прозрачный» слой. Отсюда и название Residual Networks (residual - остаточный, то есть предсказание отклонения от предшествующих слоев).

Реализация данной концепции представляет собой блочную архитектуру, где каждый блок сети представляет собой следующую конструкцию:

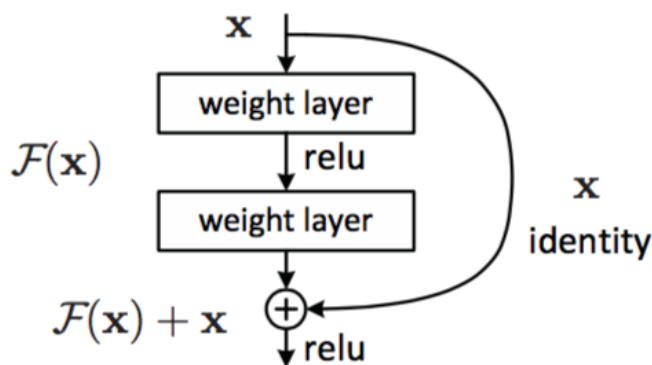


Рисунок 15. ResNet блок

В реализации Inception-ResNet-v2 были существенно упрощены блоки, что позволило сконструировать более глубокую модель с высочайшей точностью.

Архитектура Inception-ResNet-v2 может быть использована для решения задач классификации или сегментации, в зависимости от выходных слоев. Для использования модели для сегментации выход модели был пропущен через слой деконволюции с активацией ReLU. Результирующая модель содержала около 50 млн параметров.

Данная модель была обучена на наборе данных Inria Aerial Image Label с переменным learning rate (от 10^{-3} до 10^{-4}) за 200 эпох в течение двух дней.

2.4.1.1.2. Анализ разности масок сегментации

Обученная модель была использована для сегментации пары изображений. Полученные маски были обработаны с помощью следующей тривиальной процедуры:

- Маски векторизуются - связные области на бинарной маске ограничиваются полигонами с пиксельными координатами.
- В квадратичном цикле проверяется расстояние между центроидами (центр масс) полигонов из разных масок. Если они достаточно близко, то происходит проверка на пересечение. Если полигоны пересекаются, то значит объект на втором изображении не изменился.

2.4.1.2. Реализация алгебры изображений с помощью сверточных нейросетей

Идея прямого сравнения результатов сегментации довольно очевидна, но имеет два основных недостатка:

1. Полученная маска содержит большое количество объектов FP или FN из-за разницы в амплитудах соответствующих активаций.
2. Предлагаемый подход не является единым алгоритмом.

Поэтому предлагается альтернативный подход, основанный на разнице изображений: сиамская модификация современных CNN-архитектур.

2.4.1.2.1. Сиамский U-net

За основу была взята классическая архитектура U-Net — один из самых популярных инструментов для решения задач сегментации.

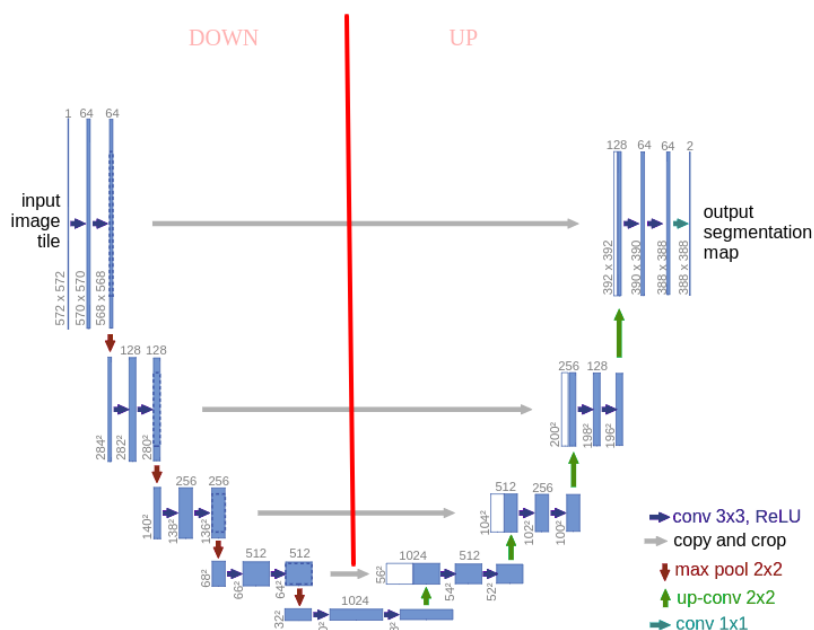


Рисунок 16. Классическая архитектура U-net

Формально, U-net – это две подсети – энкодер и декодер. Энкодер извлекает признаки и понижает размерность исходных данных, а декодер решает обратную задачу.

Модификация U-net для решения задачи обнаружения

Для наглядности последующей модификации на рисунке представлено упрощенное представление классической архитектуры U-Net

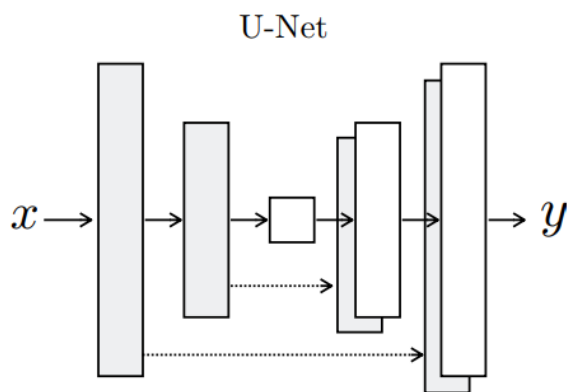


Рисунок 17. Упрощенное представление U-net

Модификация классической архитектуры U-Net для решения задачи обнаружения изменений подразумевает использование двух энкодеров, выходы которых конкатенируются перед переходом в декодер.

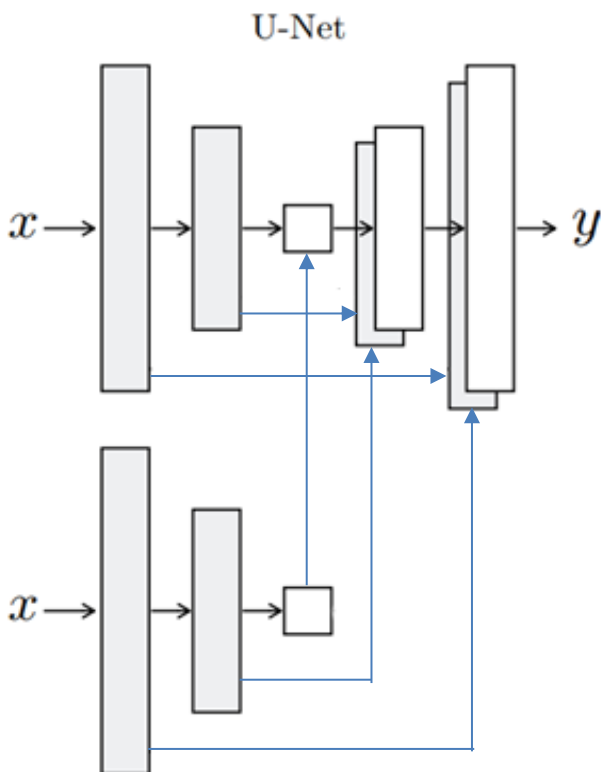


Рисунок 18. Сиамская модификация U-net

Такая архитектура подразумевает две возможных реализации – когда веса энкодеров одинаковые (weight sharing) и когда это две независимые подсети.

Последний подход гарантирует независимое извлечение признаков из каждого изображения. Отказ от разделения весов подходит в случае, когда обучающий набор данных достаточно большой, тогда данная реализация подразумевает так называемый domain adaptation: сеть настраивается извлекать нужные признаки из новых данных, которые могут принадлежать другому распределению.

Разделение весов целесообразно, когда в качестве энкодера взята обученная на большом наборе разнообразных данных сегментационная сеть. Данный подход относится к семейству методов transfer learning – переносу опыта. В таком случае, сеть уже настроена извлекать нужные признаки и не чувствительна к небольшим изменениям в гистограммах пары изображений.

2.4.1.2.1.1. Независимые подсети VGG16

В качестве подсетей были использованы энкодер и декодер на основе архитектуры VGG16 с инициализацией предобученными на ImageNet весами.

Сети основаны на сверточных блоках, каждый из которых включает в себя три слоя:

- Свертка
- BatchNormalization
- Активация

Параметры сверточного слоя определяют размерность данных на выходе. Каждый сверточный блок энкодера завершается слоем MaxPooling. Энкодер состоял из 5 таких блоков, декодер содержал 4 деконволюционных блока.

Итоговая архитектура содержала два независимых энкодера:

```
model_input = Input(shape=(None, None, n_channels), name='input')
# get two different encoders for each of two multi-channel image
left_encoder, left_activations =
_unet_vgg16_encoder(model_input=model_input, postfix='left',
weights_path=weights_path)
right_encoder, right_activations =
_unet_vgg16_encoder(model_input=model_input, postfix='right',
weights_path=weights_path)
```

Выходы и активации с промежуточных слоев каждого из энкодеров были скатенированы перед входом в единый декодер.

```
encoders_output = concatenate([left_encoder.output, right_encoder.output],
axis=-1)
# zip activations from encoder layers into pairs
activations = zip(left_activations, right_activations)
# build final model adding decoder
model = _unet_vgg16_decoder(n_classes, model_input, encoders_output,
activations)
```

Итоговая модель содержала около 48 млн параметров.

Так как данные на ImageNet представляют совершенно другую предметную область, ценность имели только самые первые слои, которые извлекают низкоуровневые признаки, например штрихи и контуры.

Модель сначала была обучена на 5 эпохах с $\text{learning rate} = 10^{-3}$ замороженными начальными слоями для корректной настройки остальных слоев, а потом дообучена еще 10 эпох с $\text{learning rate} = 10^{-4}$, будучи размороженной полностью.

2.4.1.2.1.2. Разделенные веса Inception Resnet v2

В данном эксперименте была использована предобученная на наборе данных Inria модель Inception-ResNet-v2.

Один экземпляр (instance) модели был использован для обоих изображений (таким образом keras позволяет переиспользовать веса слоев)

```
encoder = cd_inception_resnet_v2_encoder(n_filters=n_filters)
encoder_outputs_pred = encoder(img_pred)
encoder_outputs_post = encoder(img_post)
```

Предварительно, с каждой модели были собраны активации для дальнейшей передачи в декодер

```
# take intermediate layers of encoder for skip connctetions in decoder
skip_connections = list(reversed([9, 16, 260, 594]))
outputs = [encoder.output]

for i in skip_connections:
    output = encoder.layers[i].output
    outputs.append(output)
```

Полученные выходы были сконкатенированы и добавлены на различные слои декодера для деконволюции.

Так как данные Inria покрывают нашу предметную область (здания в Калифорнии), то начальные ветки были заморожены полностью. Таким образом, в результирующей модели с более чем 66 млн параметров обучаемых было только 14 млн.

Сеть обучалась в течение 5 эпох с переменным learning rate (от 10^{-3} до 10^{-4}).

2.4.2. Классификационный подход

Этот подход основан на идее пообъектной пост-классификации, где целевой класс представляет собой разрушенное/отсутствующее ¹ здание.

Подобная идея использовалась в статьях [16] и [5], однако в этих экспериментах исходная карта объектов для одного из изображений была изначально доступна.

Тем не менее, это не является необходимым условием, и подобный эксперимент может быть организован в рамках semi-supervised подхода: с помощью обученной сегментационной модели карта объектов может быть получена для любого из двух изображений из расчета симметрии изменений либо изменения «до»-«после», либо наоборот.

Предложенный метод состоит из следующих этапов:

1. Обучение сегментационной модели

Была использована модель Inception Resnet v2, обученная на Inria Aerial Image Label Dataset.

2. Получение карты объектов исходного изображения

Сегментационная модель выдает нормализованную растровую маску со значениями в диапазоне от 0 до 1, указывающими на степень уверенности попиксельной сегментации. Полученная маска далее векторизована с целью дальнейшего извлечения отдельных объектов.

¹ Можно классифицировать изображение «до», чтобы понять, где были построены новые здания на изображении «после»

Для повышения точности предсказаний сегментации можно использовать технику ТТА – test time augmentation. Исходное изображение изменяется в соответствии с определённым набором аугментация и для каждого модель предсказывает маску. Полученные маски усредняются (что было протестировано) или же обрабатываются каким-либо другим способом. Данный метод повышает точность предсказаний, что важно так как сегментационная модель часто пропускает маленькие здания.

3. Извлечение объектов из целевого изображения на основе карты объектов с предыдущего шага.

Минимальный ограничивающий квадрат вычисляется для каждого векторного объекта, и вырезается из изображения с небольшим отступом, чтобы сохранить некоторую информацию из фона. Такой выбор геометрии является наиболее удобным для представления данных для нейросети, так как данные в подвыборке генератора будут приведены к одной размерности, и использование квадратов сохранит геометрию исходных изображений.

4. Объектная классификация.

Полученные изображения подаются на вход генератору, где изменяются до целевого размера, тогда как важная информация об изначальном размере подается на конечный полносвязный слой.

В качестве оптимизируемого функционала используется бинарная кросс-энтропия.

В качестве базы классификационной модели была использована предобученная на данных соревнования Functional Map of the World (fMoW) [22] модель DenseNet.

Выбор модели обусловлен ее легкостью (всего 14 млн параметров) и превосходством над ResNet в решении задачи классификации на датасетах CIFAR/SVHN.

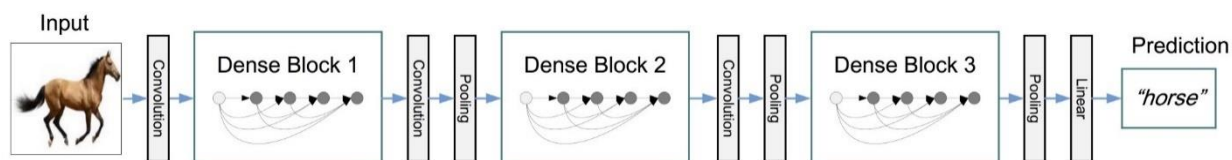


Рисунок 19. Схематическое представление архитектуры DenseNet

Каждый слой в Dense блоке пропускает выход предыдущего слоя наряду с входным сигналом, что гарантирует извлечение глубокого контекста.

В рамках экспериментов с классификацией было проведено три реализации:

1. Прямая классификация трехканальных фрагментов с второго изображения.
2. Классификация сиамской сетью по соответствующим фрагментам из обоих изображений.
3. Классификация второго изображения, умноженного на маску сегментации первого.

2.4.3. Результаты экспериментов

2.4.3.1. Сегментация

Калифорнийские пожары

	IoU	F-мера
Параллельная сегментация	0.78 +- 0.02	0.90+- 0.006
Сиамский U-net VGG16	0.75 +- 0.04	0.91+- 0.004
Сиамский U-net Inception Resnet v2	0.78 +- 0.02	0.91+- 0.006

Север Москвы

	IoU	F-мера
Параллельная сегментация	0.69 +- 0.02	0.78+- 0.007
Сиамский U-net VGG16	0.55+- 0.04	0.77+- 0.008
Сиамский U-net Inception Resnet v2	0.68 +- 0.02	0.78+- 0.007

2.4.3.2. Классификация

Калифорнийские пожары

	IoU	F-мера
Классификация второго изображения	0.94+- 0.02	0.98+-0.002
Классификация пары сиамской сетью	0.943+- 0.02	0.985+-0.002
Классификация второго, умноженного на маску сегментации первого	0.94+- 0.02	0.982+-0.002

Север Москвы

	IoU	F-мера
Классификация второго изображения	0.88+- 0.02	0.95+-0.002
Классификация пары сиамской сетью	0.91+- 0.02	0.96+-0.002
Классификация второго, умноженного на маску сегментации первого	0.90+- 0.02	0.955+-0.002

ABCD Tsunami

	F-мера
Классификация второго изображения	0.98+- 0.006
Классификация пары сиамской сетью	0.98+- 0.004
Классификация второго, умноженного на маску сегментации первого	0.98+- 0.006
Результаты из статьи	0.952 +-0.007

Выводы по главе, выбор алгоритма

Наилучшим образом себя показал классификационный подход. Это объясняется тем, что при классификации сеть настраивается на выявление тонких контекстных признаков, в то время как при сегментации модель тренируется на геометрию. В последнем случае, маленькие здания могут быть пропущены, что и показывает визуальный анализ.

Что касается данных, Калифорния представляет довольно простую и однообразную геометрию, и высокие результаты не являются удивительными. Для проверки обобщающих способностей алгоритмов, эксперименты были повторены на данных севера Москвы, которые представляют большое разнообразие геометрии – в плане и формы, и масштаба. Сложность данных объясняют скромные результаты сегментации, а высокие показатели на классификации подтверждают работоспособность алгоритма.

Предобученная классификационная модель была также протестирована на наборе данных с цунами. Результаты по IoU не приведены в силу того, что в набор данных ABCD не содержит сегментационной разметки.

Результаты превосходят описанные в статье в силу использования предобученной модели (в статье сеть обучали с нуля).

Для демонстрации в конечном приложении были выбраны два алгоритма:

- Сегментационный: параллельная сегментация
- Классификационный: классификация пары фрагментов сиамской сетью.

Глава 3. Программная реализация

2.5. Инструменты разработки

2.4.4. Обучение и backend

2.4.4.1. Языки

В качестве основного языка модуля используется Python.

Python – высокоуровневый интерпретируемый язык программирования, часто используемый для реализации научных алгоритмов. Для python реализовано множество библиотек для быстрых математических вычислений благодаря возможности интеграции Python с высокопроизводительным кодом на C/C++. Минималистичность языка позволяет сократить время разработки и повысить читаемость кода.

2.4.4.2. Инструменты

2.4.4.2.1. Keras

Keras – это открытая нейросетевая библиотека, написанная на языке Python. Она представляет собой надстройку над фреймворками DeepLearning4j, TensorFlow (в данной работе использован бэкенд Tensorflow) и Theano. Нацелена на оперативную работу с сетями глубинного обучения, при этом спроектирована компактной, модульной и расширяемой.

Библиотека содержит многочисленные реализации широко применяемых строительных блоков нейронных сетей, таких как слои, целевые и передаточные функции, оптимизаторы, и множество инструментов для упрощения работы с изображениями и текстом. Это позволяет быстрее реализовывать нейросетевые архитектуры, сохраняя код простым и читаемым.

2.4.4.2.2. Flask

Фреймворк для создания веб-приложений на языке программирования Python, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2.

2.4.4.2.3. Redis

Redis – инструмент для хранения и кэширования данных в оперативной памяти в распределённом состоянии, который также поддерживает функциональность брокера сообщений. Его внедрение позволяет распределять асинхронные ресурсоемкие задачи между сервисами-“рабочими” (workers).

2.4.4.3. Geoserver

Один из самых популярных сервисов для создания WMTS слоев из ортофотоснимков. Поддерживает REST API для загрузки (POST) векторных и растровых данных и получения (GET) ограниченной области. Скорее всего, вскоре будет заменен на другой tile-сервер с более высокой производительностью.

2.4.5. Frontend

2.4.5.1. Языки

В данном модуле используются языки JavaScript, HTML и CSS.

JavaScript – это мультипарадигменный язык программирования. Поддерживает объектно-ориентированную, императивную и функциональную парадигмы программирования

2.4.5.2. Инструменты

2.4.5.2.1. Vue

JavaScript - фреймворк для разработки пользовательских интерфейсов в парадигме реактивного программирования.

2.4.5.3. Bootstrap

JavaScript фреймворк для создания пользовательских интерфейсов, нацеленный на быстроту прототипирования. В качестве вспомогательных инструментов используется плагин Leaflet для отображения карт.

2.4.6. Развертывание

2.4.6.1. Docker

Инструмент для контейнеризации приложений для автоматизации развёртывания приложений в виртуальной среде на уровне операционной системы.

2.4.7. Генерация искусственного набора данных

Несмотря на неудачу экспериментов с искусственными данными, причина которой заключалась в том, что набор данных не был доведен до нужного уровня качества и реализма, практическая реализация этой идеи заняла значительную часть работы с использованием различных инструментов.

2.4.7.1. CityEngine

Инструмент 3D-моделирования для архитектурного планирования и создания городских ландшафтов. Поддерживает язык Python и собственный язык грамматик и правил для генерации геометрии из xml-данных.

2.4.7.2. QGis

QGis – это свободная кроссплатформенная геоинформационная система. Позволяет легко обрабатывать геоинформационные данные, осуществлять репроекции, работать с растровыми и векторными слоями.

2.4.7.3. Unity

Unity - межплатформенная среда разработки компьютерных игр. Поддерживает C# в качестве основного инструмента разработки и Cg (HLSL) – для написания шейдеров.

2.6. Веб-приложение

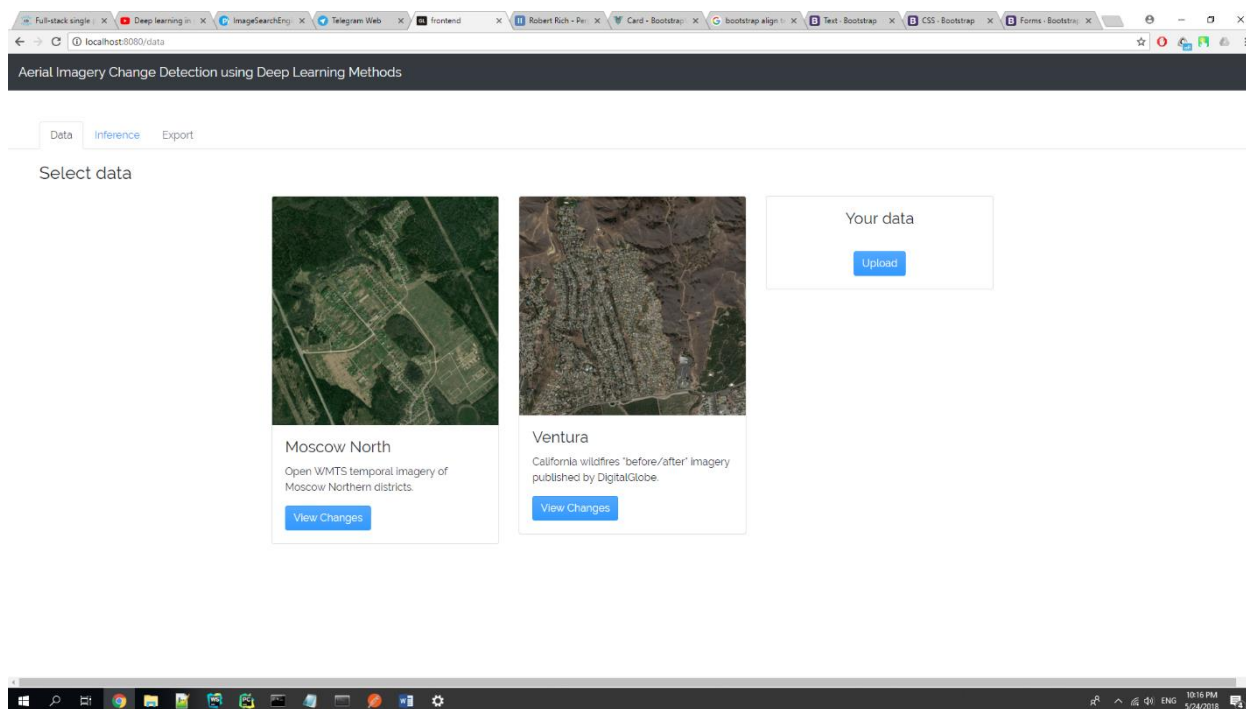


Рисунок 20. Веб-приложение. Начальный экран

Приложение представляет собой демонстрацию работы алгоритма: по паре ортофопланов одной местности составляется маска изменений.

Run inference



Рисунок 21. Полученная маска изменений

Сервис подразумевает работу с геопривязанными изображениями для удобства дальнейшей интеграции в картографические сервисы.

Пользователь может протестировать различные реализации алгоритма на встроенных наборах данных (парах изображений) или же загрузить свою пару геопривязанных снимков, удовлетворяющих определенным условиям.

Данная функциональность реализована в рамках многомодульной архитектуры

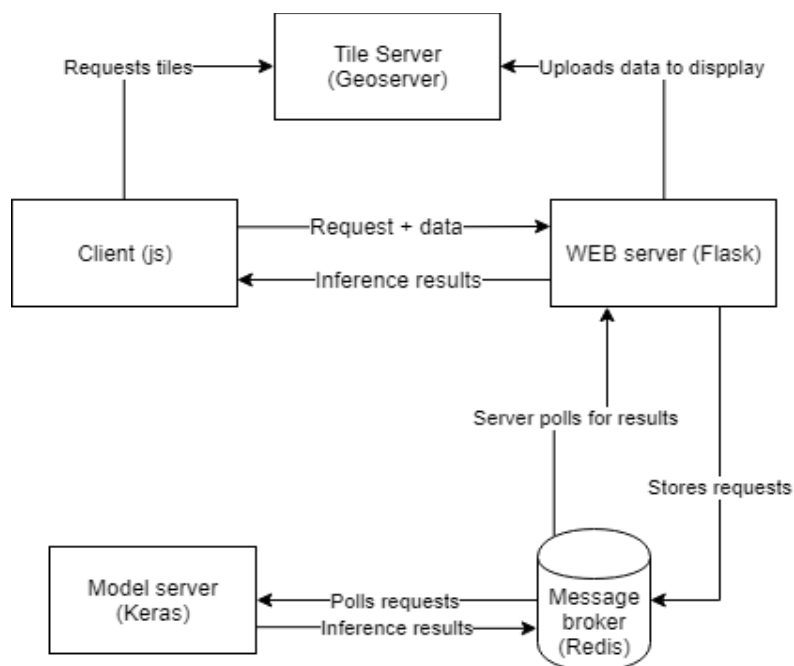


Рисунок 22. Архитектура приложения

Основные компоненты приложения (см. Рисунок 18) инкапсулированы в трех модулях.

Backend

Веб-сервер (Flask)

Содержит API для клиента для обработки запросов с помощью двух микросервисов.

- *Сервер модели (Keras)*
Осуществляет предсказание с помощью обученной модели
- *Брокер сообщений (Redis)*
Реализует распределённую очередь сообщений.

Frontend

Клиент (Vue.js)

TileServer

Geoserver – конвертирует исходные ортофотопланы в слои WMS и предоставляет посредством GET-запросов.

В качестве средства интеграции используется Docker. Для запуска модуля backend используется версия Docker с поддержкой NVIDIA для реализации вычислений на GPU. Более подробное описание каждого модуля приведено ниже.

1.1.1. Tileserver

Работа с геопривязанными изображениями (формата geotiff) подразумевает проекцию на карту. Для реализации отображения и рендеринга используется тайловый сервер (Geoserver). Он выполняет функцию единовременной пирамидальной (для каждого уровня зума) нарезки изображения на тайлы с последующей обработкой WMS GET запросов на определённую территорию. Исследуемые ортофотопланы проецируются на интерактивную карту.

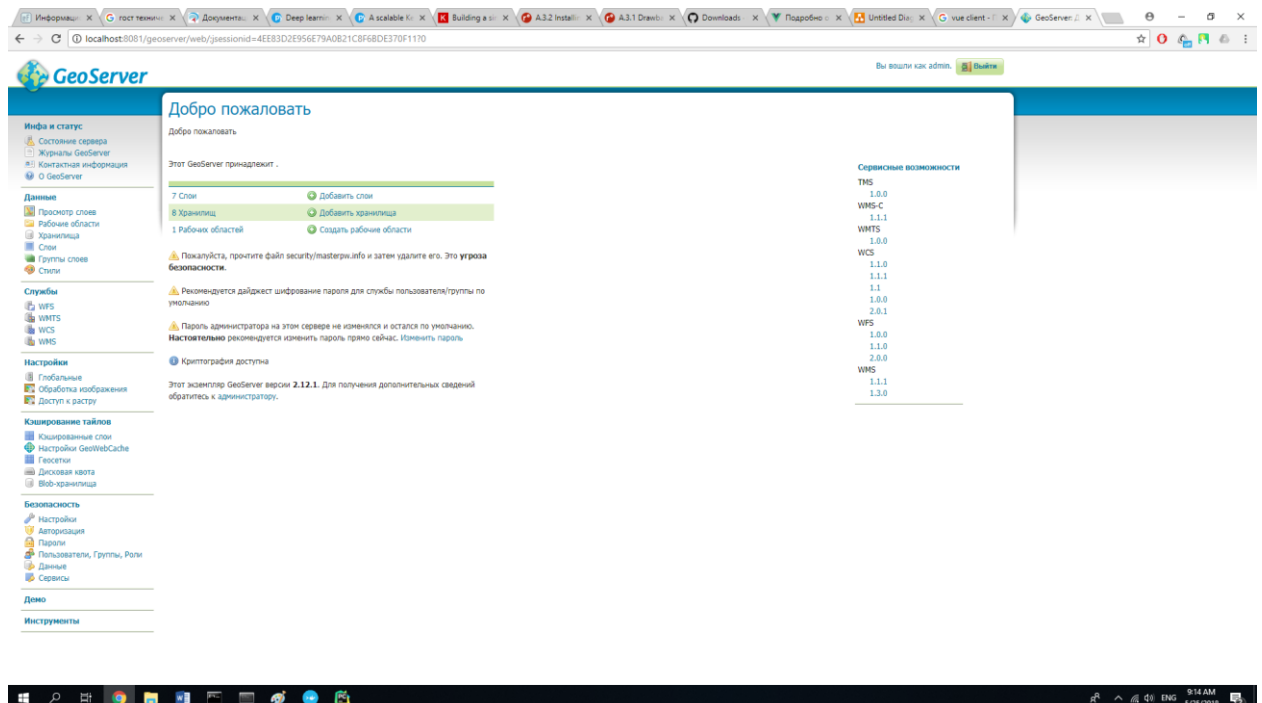


Рисунок 23. Интерфейс администратора GeoServer

1.1.2. Frontend

Для пользовательского интерфейса демонстрации возможностей алгоритма был выбран фреймворк Vue с использованием компонентов Bootstrap со стандартным стилистическим решением. Для отображения интерактивных карт используется плагин Leaflet.

Модуль обрабатывает действия пользователя, позволяя загружать данные, проводить inference на встроенных или загруженных данных, просматривать данные и спроецированные на карту.

Run inference



Рисунок 24. Интерактивная карта leaflet, крупный план

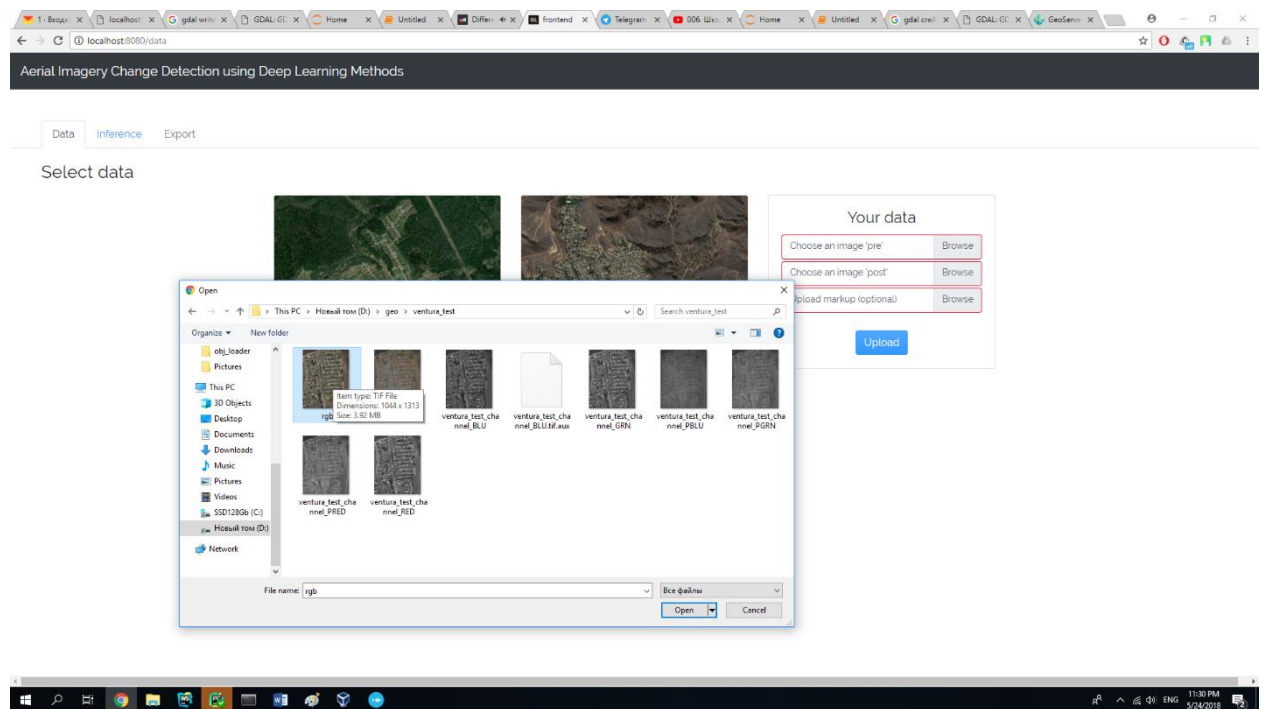


Рисунок 25. Загрузка данных

1.1.3. Backend

1.1.3.1. API

Backend сервис предоставляет API для взаимодействия с клиентским приложением.

API поддерживает обработку запросов на inference, добавляя очередной запрос в распределенную очередь сообщений (Redis). Inference может быть вызван либо для представленных пар изображений, либо для пары снимков, предоставленных пользователем.

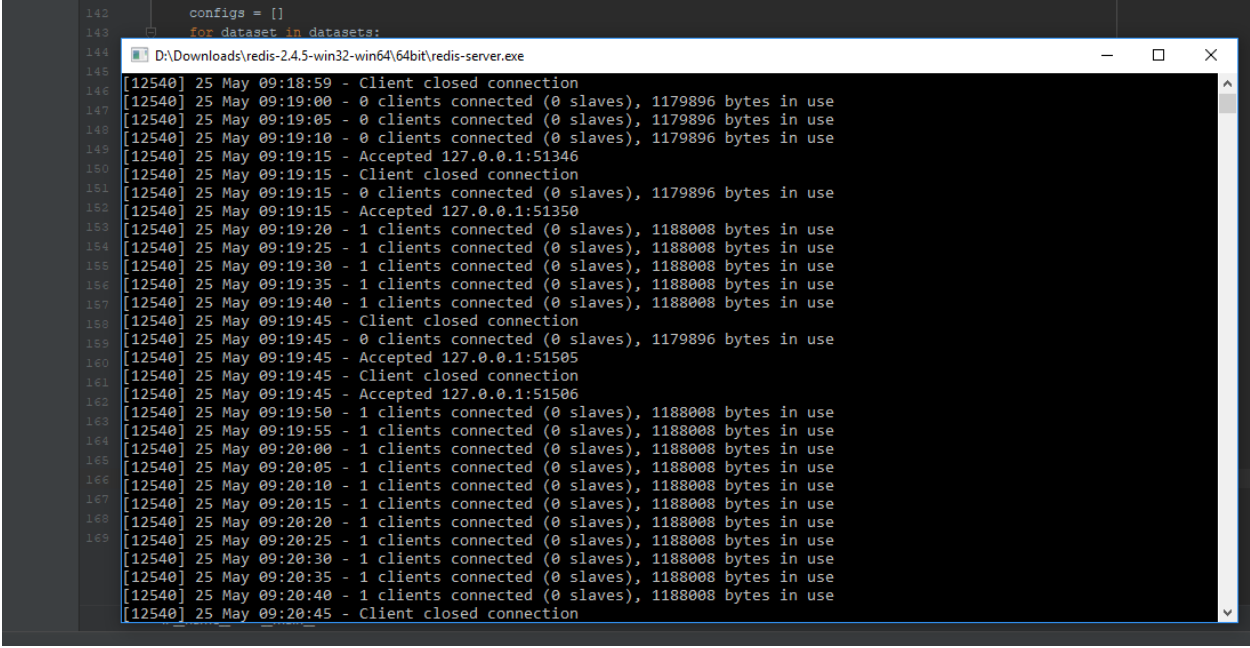
Модуль backend также содержит микроклиента для публикации новых данных на geoserver для дальнейшей отправки ссылок на соответствующие WMS слои на frontend.

1.1.3.1.1. Реализация распределенной очереди запросов

Модуль frontend общается с backend посредством POST и GET запросов. Отправленный пользователем запрос поступает на сервер (backend), где добавляется в очередь Redis на обработку и ожидает выполнения до тех пор, пока не освободится один из worker-сервисов.

Frontend запрашивает статус выполнения и отображает его пользователю. После освобождения worker-сервиса, исходные данные передается нейронной сети для inference. Полученные маски изменений отображаются в интерфейсе.

Мониторинг базы данных осуществляется посредством интерфейса командной строки.

A screenshot of a Redis console window. The window title is "D:\Downloads\redis-2.4.5-win32-win64\64bit\redis-server.exe". The console output shows a series of log messages from the Redis server. The messages include connection status updates, such as "Client closed connection", "0 clients connected (0 slaves), 1179896 bytes in use", and "Accepted 127.0.0.1:51346". The logs are timestamped with dates like "25 May 09:18:59" and "25 May 09:19:00". The console text is displayed on a dark background with light-colored text. On the left side of the image, there is a vertical list of line numbers from 142 to 169, corresponding to the lines of code or log output shown in the screenshot.

```
142 configs = []
143 for dataset in datasets:
144
145 [12540] 25 May 09:18:59 - Client closed connection
146 [12540] 25 May 09:19:00 - 0 clients connected (0 slaves), 1179896 bytes in use
147 [12540] 25 May 09:19:05 - 0 clients connected (0 slaves), 1179896 bytes in use
148 [12540] 25 May 09:19:10 - 0 clients connected (0 slaves), 1179896 bytes in use
149 [12540] 25 May 09:19:15 - Accepted 127.0.0.1:51346
150 [12540] 25 May 09:19:15 - Client closed connection
151 [12540] 25 May 09:19:15 - 0 clients connected (0 slaves), 1179896 bytes in use
152 [12540] 25 May 09:19:15 - Accepted 127.0.0.1:51350
153 [12540] 25 May 09:19:20 - 1 clients connected (0 slaves), 1188008 bytes in use
154 [12540] 25 May 09:19:25 - 1 clients connected (0 slaves), 1188008 bytes in use
155 [12540] 25 May 09:19:30 - 1 clients connected (0 slaves), 1188008 bytes in use
156 [12540] 25 May 09:19:35 - 1 clients connected (0 slaves), 1188008 bytes in use
157 [12540] 25 May 09:19:40 - 1 clients connected (0 slaves), 1188008 bytes in use
158 [12540] 25 May 09:19:45 - Client closed connection
159 [12540] 25 May 09:19:45 - 0 clients connected (0 slaves), 1179896 bytes in use
160 [12540] 25 May 09:19:45 - Accepted 127.0.0.1:51505
161 [12540] 25 May 09:19:45 - Client closed connection
162 [12540] 25 May 09:19:45 - Accepted 127.0.0.1:51506
163 [12540] 25 May 09:19:50 - 1 clients connected (0 slaves), 1188008 bytes in use
164 [12540] 25 May 09:19:55 - 1 clients connected (0 slaves), 1188008 bytes in use
165 [12540] 25 May 09:20:00 - 1 clients connected (0 slaves), 1188008 bytes in use
166 [12540] 25 May 09:20:05 - 1 clients connected (0 slaves), 1188008 bytes in use
167 [12540] 25 May 09:20:10 - 1 clients connected (0 slaves), 1188008 bytes in use
168 [12540] 25 May 09:20:15 - 1 clients connected (0 slaves), 1188008 bytes in use
169 [12540] 25 May 09:20:20 - 1 clients connected (0 slaves), 1188008 bytes in use
[12540] 25 May 09:20:25 - 1 clients connected (0 slaves), 1188008 bytes in use
[12540] 25 May 09:20:30 - 1 clients connected (0 slaves), 1188008 bytes in use
[12540] 25 May 09:20:35 - 1 clients connected (0 slaves), 1188008 bytes in use
[12540] 25 May 09:20:40 - 1 clients connected (0 slaves), 1188008 bytes in use
[12540] 25 May 09:20:45 - Client closed connection
```

Рисунок 26. Консоль Redis

Заключение

В работе были рассмотрены различные методы обнаружения изменений на трехканальных данных аэрофотосъемки с использованием глубоких сверточных сетей.

В работе приведен подробный обзор современных методов для решения данной задачи с обоснованием применимости различных алгоритмов. Были предложены экспериментальные методы на основе state-of-the-art классификационных и сегментационных архитектур для решения задачи обнаружения изменений. В работе приведены необходимая теоретическая база, инженерные детали реализации предложенных методов, а также результаты их работы на различных наборах данных.

В ходе выполнения ВКР были выполнены следующие задачи:

1. Проанализированы различные современные методы решения задачи обнаружения изменений;
2. Предложены и реализованы модификации различных архитектуры state-of-the-art, включая VGG16, Inception Resnet v2 и Densenet для улучшения существующих методов;
3. Приведены результаты экспериментов, в том числе превосшедшие описанные в одной из статей
4. Разработано веб-приложение на языке Python с веб-интерфейсом и реализацией распределенной очереди сообщений для демонстрации работы различных алгоритмов, а также наглядности возможности интеграции с картографическими сервисами.
5. Разработана техническая документация программного продукта, приведенная в приложениях А-Г.

Дальнейшие исследования включают в себя:

1. Создание end-to-end классификационного решения;
2. Исследование методов, использующие рекуррентные сети и условные случайные поля;
3. Обобщение приложенных методов на многоканальные данные спутниковой съемки;
4. Оптимизацию моделей.

Список литературы

1. Nielsen, A.A. The regularized iteratively reweighted mad method for change detection in multi- and hyperspectral data. *IEEE Trans. Image Process.* **2007**, 16, 463–478.
2. Bovolo, F.; Bruzzone, L. A theoretical framework for unsupervised change detection based on change vector analysis in the polar domain. *IEEE Trnas. Geosci. Remote Sens.* 2007, 45, 218–236.
3. Sinha, P.; Kumar, L.; Reid, N. Rank-Based Methods for Selection of Landscape Metrics for Land Cover Pattern Change Detection. *Remote Sens.* 2016, 8, 107, doi:10.3390/rs8020107.
4. Basnet, B.; Vodacek, A. Tracking Land Use/Land Cover Dynamics in Cloud Prone Areas Using Moderate Resolution Satellite Data: A Case Study in Central Africa. *Remote Sens.* 2015, 7, 6683–6709.
5. Demir, B.; Bovolo, F.; Bruzzone, L. Updating land-cover maps by classification of image time series: A novel change-detection-driven transfer learning approach. *IEEE Trans. Geosci. Remote Sens.* 2013, 51, 300–312.
6. Ganchev, T.D.; Jahn, O.; Marques, M.I.; Figueired, J.M.; Schuchmann, K.L. Automated acoustic detection of vanellus chilensis lampronotus. *Remote Sens. Environ.* 2015, 42, 6098–6111.
7. Morsier, F.; Tuia, D.; Borgeaud, M.; Gass, V.; Thiran, J.P. Semi-supervised novelty detection using svm entire solution path. *IEEE Trans. Geosci. Remote Sens.* 2013, 51, 1939–1950.
8. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. 18th International Conf. on Machine Learning*, 2001.
9. Learning deep structured network for weakly supervised change detection Salman Khan, , Xuming He , Fatih Porikli , Mohammed Bennamoun Ferdous Sohel and Roberto Togneri, 2017
10. Change Detection Based on Conditional Random Field With Region Connection Constraints in High-Resolution Remote Sensing Images, 2016
11. <https://www.digitalglobe.com/>
12. Synthetic Data Generation for Deep Learning in Counting Pedestrians Hadi Keivan Ekbatani, Oriol Pujol and Santi Segui, 2017
13. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, Stan Birchfield, 2018
14. <http://www.gdal.org/>
15. Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, Pierre Alliez, 2017
16. Aito Fujita, Ken Sakurada, Tomoyuki Imaizumi, Riho Ito, Shuhei Hikosaka and Ryosuke Nakamura, "Damage Detection from Aerial Images via Convolutional Neural Networks," *IAPR International Conference on Machine Vision Applications (MVA)*, 2017

17. <http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/>
18. Earthquake damage mapping: An overall assessment of ground surveys and VHR image change detection after L'Aquila 2009 earthquake Roberta Anniballea , Fabrizio Notob , Tanya Scaliaa , Christian Bignamic , Salvatore Stramondoc , Marco Chinid , Nazzareno Pierdicca, 2018
19. Conditional Random Fields as Recurrent Neural Networks Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr, 2015
20. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Sergey Ioffe, Christian Szegedy, 2015
21. Deep Residual Learning for Image Recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, 2015
22. Functional Map of the World, Gordon Christie, Neil Fendley, James Wilson, Ryan Mukherjee, 2017