Введение

Любой материал имеет свойство изнашиваться и впоследствии разрушаться. В результате на его поверхности появляются дефекты в виде трещин. Это может создавать разного рода проблемы и даже чрезвычайно опасные ситуации, скорость и методы решения которых в первую очередь зависят от сферы деятельности. Среди таких сфер можно выделить следующие.

1. Реконструкция фасадов зданий.

Для выявления причины появления трещин используют так называемые «маяки» (щелемеры), которые крепятся непосредственно в области разрушения и помогают отслеживать динамику его развития. Однако такие приспособления имеют ряд недостатков. «Простые» версии ограничены в своей информативности, а современные электронные аналоги часто подвержены краже и вандализму.

1. Краш-тесты.

Различные материалы подвергают критическим нагрузкам, вследствие чего также появляются трещины.

1. Отбраковка продукции на производстве.

При производстве газобетонных блоков, древесины и т. п. очень важно вовремя исключать из конвейера изделия с дефектами

Методы машинного обучения позволяют улучшить эффективность проведения описанных выше мероприятий используя системы фото и видеонаблюдение. Анализируя поток изображений с камеры, можно определять наличие дефектов и выделять сегменты трещин. Такой информации достаточно для расчетов всех необходимых метрических характеристик трещин.

В компьютерном зрении задача сегментации трещин на поверхности относится к семейству задач «Semantic Image Segmentation» [1]. Цель такого рода задач заключается в попиксельной локализации целевого объекта на изображении. Для каждой отдельно взятой картинки результатом локализации будет выступать так называемая «бинарная маска» — одноканальное бинарное изображение (Рисунок 1). Пиксели такой маски обычно обозначают 1, если на соответствующем пикселе входного изображения присутствует искомый целевой объект, и обозначают 0, если на соответствующем пикселе объект отсутствует.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1. Пример сегментации трещин по картинке

Однако, для получения однозначного ответа на вопрос «Есть ли трещина?», необходимо реализовать дополнительный алгоритм, анализирующий полученную бинарную маску. Такой подход требует дополнительных издержек как вычислительных, так и временных, которые являются излишеством в следующих ситуациях.

1. Получение ответа на вопрос «Есть ли трещина?» является более приоритетным, чем получение бинарной маски. Возможно, в отрицательном случае такая маска вовсе не нужна.
2. Анализируемый поток информации по большей части не содержит «трещин», и дополнительного анализа таких данных, в том числе и построение маски, вовсе не требуется.

Дальше будет рассмотрен подход, позволяющий определить наличие трещины на картинке, не требующий построения бинарной маски, делая это самое построение вариативным.

Постановка задачи

Целью данной работы является реализация обучаемой модели для определения и сегментации трещин на поверхности по изображению.

**Сегментация.** Для решения задачи сегментации в глубоком обучении используют различные архитектуры генеративных нейронных сетей, относящихся семейству «Variance Autoencoders (VAEs)» [2]. Такие архитектуры состоят из трёх основных компонент, которые взаимодействуют между собой по следующему принципу (Рисунок 2.).

1. Энкодер «Encoder». Исходный набор картинок подается в энкодер. Задача энкодера определенным образом «сжать» информацию в так называемое «сжатое представление». Такое сжатие происходит путем многократного пропуска информации через последовательность предобученных блоков. Такие блоки состоят из слоев двухмерных свёрток «Convolution» с обучаемым ядром «kernel» и слоев пулинга «pooling».
2. Ботлнек «Bottleneck». Слой сжатого представления. Здесь сжатая информация дополнительно проходит через предобученные блоки. Однако на этом уровне никакого сжатия не происходит. Цель этого блока – обработать сжатое представление и подготовить его к обратной реконструкции, которой занимается декодер.
3. Декодер «Decoder». Завершающий слой, реконструирующий сжатую информацию к необходимому виду. Реконструкция происходит за счет заполнения сжатых образов пустотами, переводя представление в более разреженный вид. После чего разреженные сегменты заполняются путем прохода через последовательность предобученных блоков. Такая операция называется обратной или «транспонированной» свёрткой «Transposed Convolution» с обучаемым ядром.

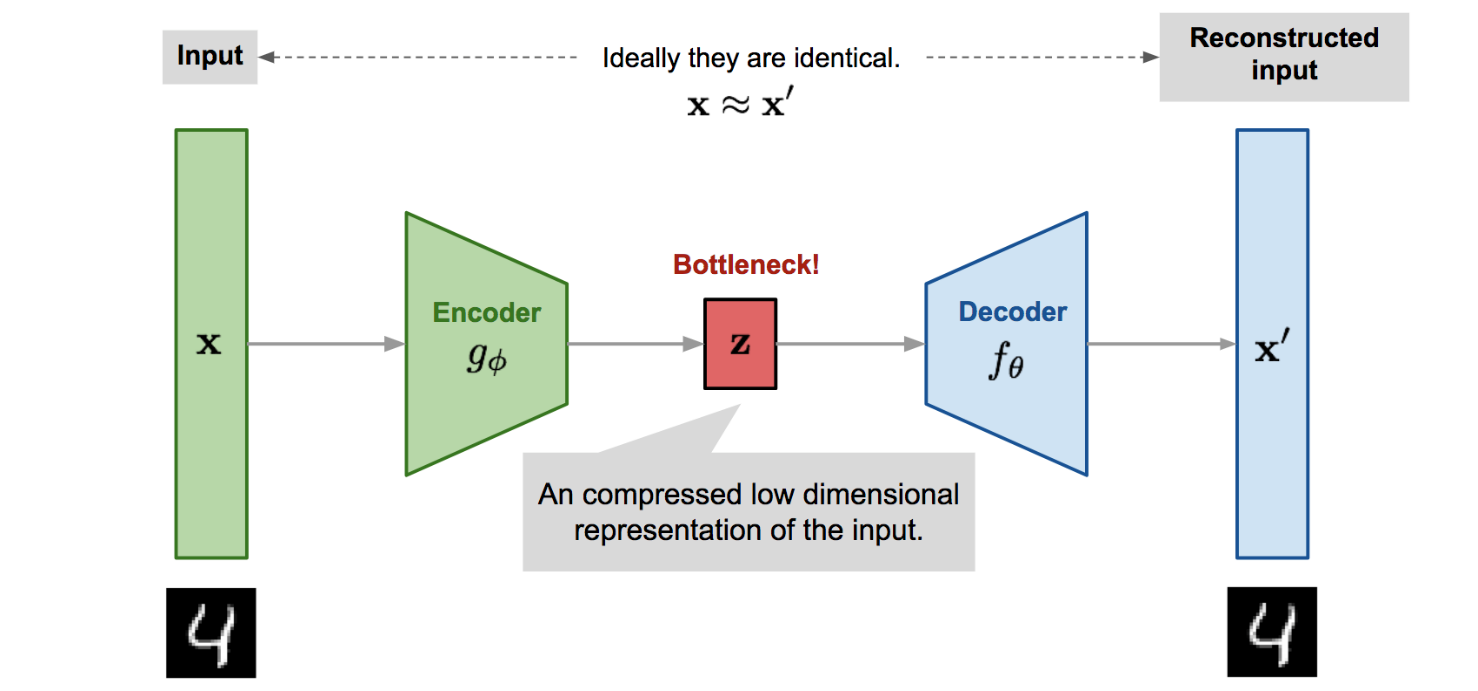


Рисунок 2. Принцип работы архитектур семейства «VAEs».

Таким образом, имея входные данные на выходе необходимо получить представление вида следующим образом.

Где – слой энкодера, – сжатое представление, – слой декодера.

**Классификация.** Задачу определения наличия трещины можно сформулировать как задачу классификации. С этой точки зрения класс 1 будет обозначать присутствие трещины на картинке, а класс 0 её отсутствие. Тогда рассматривая все те же входные данные необходимо получить бинарное представление .

Где – вероятность того, что на картинке присутствует трещина, – некоторый фиксированный порог «строгости классификации».

Для реализации описанных моделей необходимо определиться с набором инструментов. В первую очередь нужно выбрать такой язык программирования, на котором вся разработка сведется к сбору, систематизации и анализу данных. После чего на основе полученной информации можно будет легко создавать и настраивать алгоритмы для глубокого обучения. Среди популярных языков для анализа данных можно выделить MATLAB [3], R [4] и Python [5]. Такой язык также должен обладать библиотеками и фреймворками, которые позволяют производить математические расчеты на видеокартах, активно развиваются и поддерживаются, а также имеют крупное сообществом разработчиков. Всем этим критериям явно подходит язык Python. Из подходящих фреймворков можно выделить два наиболее крупных и известных: PyTorch [6] и TensorFlow [7]. TensorFlow, в отличии от PyTorch, имеет более простые инструменты для создания и обучения модели. Однако такая простота усложняет работу в случаях, когда нужно совершить более низкоуровневые настройки моделей. PyTorch предоставляет более низкоуровневый интерфейс для настройки, что повышает порог входа для изучение фреймворка в начале и облегчает работу в дальнейшем. Оба фреймворка предоставляют интерфейс для вычислений на видеокартах. Такие вычисления легко распараллеливаются благодаря так называемого «графа вычислений», который лежит в основе арифметики обоих фреймворков. Однако, в TensorFlow работа с таким графом производится в явном виде. В PyTorch же граф вычислений реализован в неявном виде, и все вычисления автоматически передаются в него.

Оба рассмотренных фреймворка дают возможность создавать модели глубоких нейронных сетей и производить все вычисления на видеокартах. Но они не имеют интерфейса для обучения самих моделей. Процесс обучения можно определить как отдельную задачу, т. к. он имеет свои параметры и проблемы, которые также решаются различными техниками. Поэтому для каждого из перечисленных фреймворков были разработаны дополнительные фреймворки, которые содержат весь необходимый набор инструментов для обучения. Для TensorFlow есть официальное решение под названием Keras [8], в то время как для PyTorch существует несколько неофициальных решений, разработанные сообществом фреймворка. Самым крупным, с точки зрения предоставляемых возможностей, является фреймворк под названием Catalyst [9], разработанный отечественными энтузиастами. Помимо полного набора инструментов и удобного интерфейса для обучения, данный фреймворк содержит в себе все необходимое метрики и функции потерь, которые будут необходимы на этапе обучения модели.

Глава 1. Подготовка данных

1.1 Описание набора данных

Для обучения и тестирования модели был взят набор данных под названием «Crack Segmentation» [\*]. Это большая база, состоящая из 11,298 пар изображений размера 448x448 пикселей. В каждой такой паре первая картинка представляет из себя цветную фотографию некоторой поверхности, а вторая картинка это бинарная маска первой, в которой все пиксели, содержащие трещины, окрашены белым, а все остальные черным.

Рисунок 3. Представление данных.

Техника «обучение с учителем» – когда модели обучаются на заранее размеченном наборе данных, предполагает разделение всего набора данных на 2 части – тренировочную (train) и проверочную (valid). На тренировочной подвыборке модель изменяет своё состояние, то есть обучается, а на проверочной нет. Такой подход гарантирует честность и объективность подсчета метрик в процессе обучения. Описанный выше набор данных изначально разбит на 2 соответствующих каталога, из которых 9,603 экземпляров относятся к тренировочной подвыборке и 1,695 к проверочной.

1.2 Аугментация данных

Увеличение исходного набора данных оказывает положительное влияние на обучение модели. Это не только помогает избежать проблемы «переобучения», но и даёт возможность обучаемой сети познакомиться с большим набором уникальных ситуаций и выявить больше закономерностей. Процесс искусственного увеличения набора данных называется «аугментацией». Такой эффект достигается путем пропуска данных через различные, в зачастую случайные, операции обработки изображений. Рассмотрим примеры таких операций.

1. Горизонтальное отображение (Рисунок 4). Позволяет из 1 уникального экземпляра изображения сделать 2 уникальных экземпляра. С точки зрения модели нейронной сети это будут две разные сцены.

Изображение выглядит как земля, внешний, строительный материал, камень

Автоматически созданное описание Изображение выглядит как земля, внешний, строительный материал, камень

Автоматически созданное описание

Рисунок 4. Горизонтальное отображение.

1. Вертикальное отображение (Рисунок 5). Аналогично горизонтальному. Используя оба вида отображения, можно из 1 уникального экземпляра сделать 4 уникальных.

Изображение выглядит как земля, внешний, строительный материал, камень

Автоматически созданное описание Изображение выглядит как земля, внешний, строительный материал, камень

Автоматически созданное описание

Рисунок 5. Вертикальное отображение.

1. Поворот на 0, 90, 180 и 270 градусов (Рисунок 6). Каждые 90 градусов дает новые уникальные экземпляры. В совокупности со всеми видами отображение можно из 1 уникального экземпляра сделать 16 уникальных.

Изображение выглядит как земля, внешний, строительный материал, камень

Автоматически созданное описание Изображение выглядит как земля, внешний, строительный материал, камень

Автоматически созданное описание Изображение выглядит как земля, внешний, строительный материал, камень

Автоматически созданное описание Изображение выглядит как земля, внешний, строительный материал, камень

Автоматически созданное описание

Рисунок 6. Поворот на 0, 90, 180 и 270 градусов.

Чередование случайным образом описанных операций позволяет увеличить исходный набор данных в 16 раз.

В рамках исследования размер картинок 448x448 пикселей является избыточным. Чем больше входная информация, тем мощнее должна быть модель, которая эту информацию будет обрабатывать. Мощность модели повышается за счет увеличения количества обучаемых параметров. Это значительно увеличивает требования к минимальному объему видеопамяти. Поэтому в процессе аугментации данных, каждая «аугментированная» картинка также будет разделена на 5 равных частей следующим образом (Рисунок 7).



Рисунок 7. Принцип разделение на 5 частей.

Так как на части картинок трещины занимает меньше половины площади, операция разделения создаст изображения с отсутствием трещин, что даст необходимый набор данных для последующего решения задачи классификации.

Глава 2. Применение глубоких нейронных сетей

* 1. Свёрточная нейронная сеть

Свёрточная нейронная сеть – архитектура искусственных нейронных сетей, которую впервые предложил Ян Лекун в 1998 году [11]. В общем случае такая архитектура устроена аналогичным образом, что и архитектура простой полносвязной нейронной сети: Слой извлечения признаков Слой нормализации Слой активации. Полносвязный слой извлечения признаков, он же перцептрон, определяется следующим образом.

Где – входной вектор данных размерности , – выходной вектор размерности , – матрица весов размерности , – вектор смещения размерности . Обработка двумерных данных обычными полносвязными (линейными) нейронными сетями имеет два критических недостатка. Рассмотрим их на примере задачи классификации на наборе данных MNIST (Рисунок 8.).



Рисунок 8. Набор данных MNIST [10].

Представленная база состоит из 10 наборов (классов) изображений рукописных цифр. Картинки представлены в одноканальном виде и имеют разрешение 28x28 пикселей. Перед подачей такого изображения на вход полносвязной нейронной сети его необходимо представить в одномерном виде, то есть растянуть в вектор длинной 284. При обработке такого вектора возникают следующие недостатки.

1. Нарушаются пространственные отношения на картинке. Полносвязная нейронная сеть лишена способности выявлять двумерные паттерны. Это приводит к дополнительным сложностям распознавания, когда объект, не изменив своих внешних характеристик, смещается в другие области изображения.
2. Полносвязная нейронная сеть требует слишком большую входную размерность. Например, для обработки изображения размера 224х224 пикселей, необходимо 50,176 входных нейронов.

Архитектура свёрточной нейронной сети решает описанные выше проблемы путем использования свёрточных слоев вместо полносвязных для извлечения признаков. Слои нормализации и активации также адаптируются под двумерное представление данных. Кроме того, используются дополнительные слои субдискретизации.

Последовательность таких слоев формируют блоки архитектуры свёрточной нейронной сети. А последовательность таких блоков формируют непосредственно саму архитектуру. Рассмотрим принципы работы описанных слоев и основанные на них архитектуры для решения различных задач.

2.1.1 Свёрточный слой

Свёрточный слой определяется следующим образом.

Где – тензор весов, – вектор смещений, и – входной и выходной тензоры, и – размеры ядра (фильтра) свёртки, и – высота и ширина входного изображения, а и – шаг по вертикали и горизонтали, а и – количество каналов входного и выходного изображения соответственно.

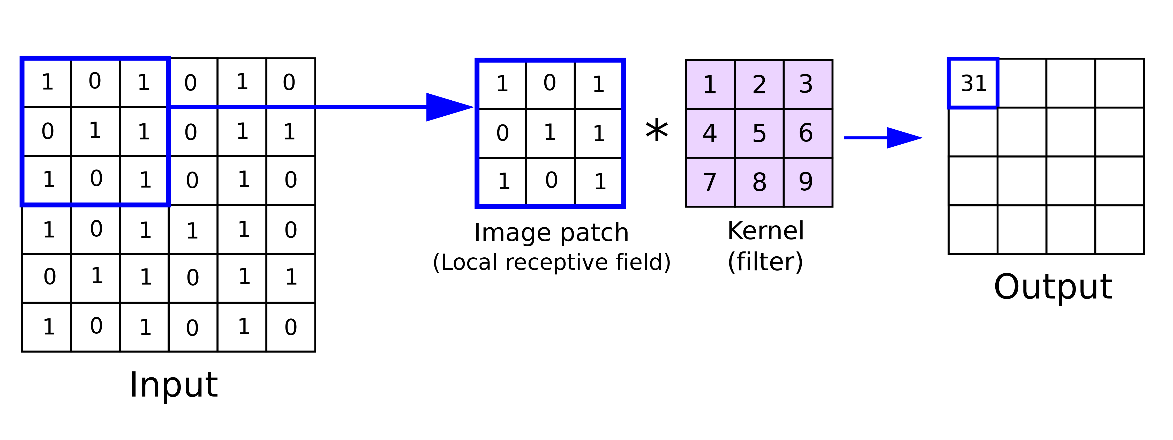


Рисунок 9. Демонстрация работы свёртки.

Веса фильтров свёртки общие – это позволяет использовать относительно малое количество параметров, при этом выявлять паттерны, не зависящие от топологии изображения. В общем случае, вход и выход свёрточного слоя являются трёхмерными тензорами, где первые две размерности – это высота и ширина изображения, а третья размерность – это количество каналов, они же карты признаков «feature map». Такие карты позволяют использовать ядра свёртки для поиска различных паттернов. Количество каналов свёрточного слоя – это параметр аналогичный количеству нейронов полносвязного слоя.

2.1.2 Слой нормализации

При обучении глубоких нейронных сетей возникает проблема, когда распределение входных признаков каждого слоя меняется в связи с изменениями параметров предыдущего слоя во время обучения. В результате сам процесс обучения замедляется. Для решения данной проблемы используется дополнительный слой нормализации признаков [12] с обучаемыми параметрами, как часть архитектуры нейронной сети.

В процессе обучения практически всегда возникает проблема, когда невозможно загрузить все данные на видеокарту в силу ограничений видеопамяти. Тогда применяется подход, при котором данные разбиваются на несколько равных по объему групп, называемых батчами. Слой нормализации применяется непосредственно к самим батчам данных, накапливая необходимую статистику для обобщения по всей выборке. Таким образом, слой батч нормализации связывает входные и выходные группы тензоров и следующим образом.

Где индекс – обозначает -ый батч, и – накапливаемые статистики выборочного среднего и стандартного отклонения, посчитанные по всей подвыборке , и — обучаемые вектора параметров, используемые для масштабирования и смещения признаков к требуемому распределению, а — очень малое число-константа, необходимое для предотвращения деления на ноль.

2.1.3 Слой активации

В качестве функции активации в архитектурах свёрточных нейронных сетей в подавляющем большинстве случаев используют функцию ректификации под названием «ReLU» [13], график которой изображен на рисунке 10.

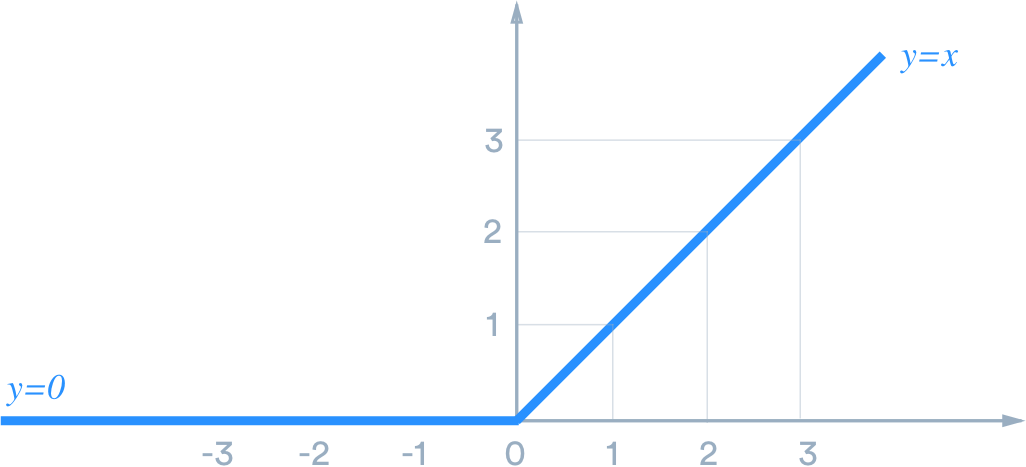


Рисунок 10. График функции активации ReLU.

С точки зрения обратного распространения ошибки [14], производная данной функции активации имеет следующий физический смысл: ошибка распространяется неизменной по тем путям, где сигнал был положителен и не распространяется вовсе, где сигнал был отрицателен.

2.1.4 Слой субдискретизации

Слой субдискретизации – это нелинейное преобразование, использующее функции максимума (MAX), минимального (MIN), или среднего (AVG), действующие на группы компонентов тензора. Такой слой ещё называют слой пулинга, и используют его для нелинейного уплотнения карты признаков. Слой пулинга с функцией максимума определяется следующим образом.

Где – входной тензор размера , – выходной тензор, – окно пулинга, а и – величины шага по вертикали и горизонтали соответственно. Такой слой субдискретизации уменьшает размерность выходных признаков, оставляя лишь наиболее «важные» в смысле максимума (Рисунок 11.).



Рисунок 11. Демонстрация работы макс пулинга.

2.2 Свёрточная нейронная сеть – классификатор

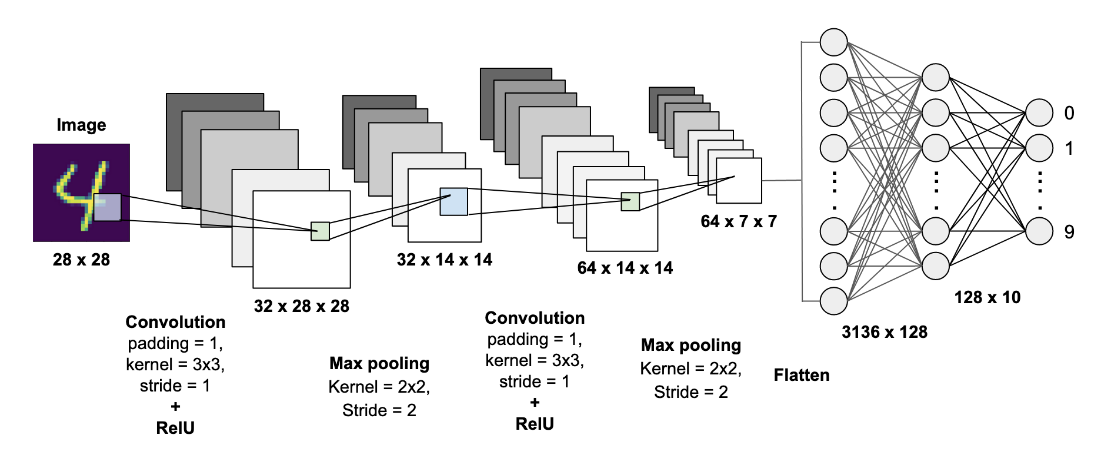


Рисунок 9. Пример архитектуры свёрточной нейронной сети.

Глава 3. Эксперименты и результаты

…

Выводы

…

Заключение

…

Список литературы

[1] <https://arxiv.org/abs/2001.05566>

[2] <https://arxiv.org/abs/1906.02691>

[3] <https://www.mathworks.com/help/matlab/language-fundamentals.html>

[4] <https://www.r-project.org/>

[5] <https://www.python.org/>

[6] <https://pytorch.org/>

[7] <https://www.tensorflow.org/>

[8] <https://keras.io/>

[9] <https://catalyst-team.github.io/catalyst/>

[10] <http://yann.lecun.com/exdb/mnist/>

[11] <http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf>

[12] <https://arxiv.org/abs/1502.03167>

[13] <https://en.wikipedia.org/wiki/Rectifier_(neural_networks)>

[14] <https://arxiv.org/ftp/arxiv/papers/1708/1708.06243.pdf>