**Введение**

Любой твёрдый материал со временем изнашивается и начинает разрушаться. В результате на его поверхности появляются дефекты в виде трещин. Особенно это актуально для фасадов зданий. Например, если на несущей стене появилась трещина, то в первую очередь необходимо выявить причину её появления. Для этого 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. К такой системе мониторинга в зачастую привлекают специалистов по реконструкции, которые принимают решения на основе собранной информации о дефектах.

Для улучшения эффективности проведения описанных выше мероприятий можно использовать системы фото и видеонаблюдение в совокупности с системами глубокого обучения. Такое программное обеспечение могло бы с помощью анализа потока изображений с камеры самостоятельно определять наличие дефектов и строить бинарную карту трещин. Данной информации было бы достаточно, чтобы рассчитать некоторые метрические характеристики трещин.

В качестве примера похожего, с точки зрения концепции, программного обеспечения можно привести специальный тепловизор [\*], позволяющий анализировать температуру поверхностей через смартфон.

Изображение выглядит как текст, монитор, внутренний, электроника

Автоматически созданное описание

(Описание)

**Постановка задачи**

**(Техническое задание)**

Целью данной работы является описание и реализация модели глубокого обучения для определения и сегментации трещин на поверхности по изображению. Задача сводится к построению модели , такой что:

где – входное изображение, – бинарный показатель, говорящий о присутствие или отсутствие трещин на изображении, и – бинарная маска для , отображающая карту трещин. Задача «определения» в данном случае эквивалентна задаче «классификации». Таким образом, реализуемая модель должна единовременно решать сразу две задачи: классификацию и сегментацию.

**Классификация**. Рассмотрим общую постановку задачи бинарной классификации. Получив на вход некоторое изображение , от модели ожидается на выходе пара чисел – вектор :

где – это некоторый количественный показатель того, что на изображении преобладает -й класс. Для интерпретации полученных показателей в бинарном виде нужно сначала представить их в вероятностном виде. Расчет вероятностного показателя выполняется с помощью функции «Softmax» [\*]:

Таким образом, искомый бинарный показатель будет получен следующим образом:

Однако, здесь может возникнуть неопределенность при . Для этого вводится некоторый порог , такой что:

Сама задача бинарная классификация является частным случаем общей задачи классификации. Это позволяет рассмотреть постановку задачу в ином виде: принадлежность к тому или иному классу можно интерпретировать одним вероятностным показателем . Следовательно, выход из модели будет представлять из себя одно число. Тогда находится следующим образом:

Таким образом, если вероятность превышает некоторый порог , значит на изображении представлен класс 1, иначе представлен класс 0.

Выбор того или иного подхода происходит на начальном этапе, при построении выходов в архитектуре модели.

**Сегментация**. Получив на вход изображение размера пикселей по ширине и пикселей по высоте, на выходе от модели ожидается одноканальное изображение того же размера, которое можно представить в виде матрице :

Такая матрица будет иметь соответственно строк и столбцов. Получение бинарной маски из матрицы достигается аналогичным бинарной классификации образом. Каждый элемент матрицы представляется в вероятностном виде, после чего проходит пороговую функцию:

Пиксель бинарной маски интерпретируются как 1, если на нем присутствует трещина, и 0 если отсутствует.

**(Средства)**

Для реализации описанной модели необходимо определиться с набором инструментов. В первую очередь 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 [\*], R [\*] и Python [\*]. Обязательно, чтобы язык обладал такими 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 [\*] и TensorFlow [\*]. 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. PyTorch и TensorFlow предоставляют интерфейс для вычислений на видеокартах. Такие вычисления легко распараллеливаются благодаря так называемого «графам вычислений», который лежит в основе арифметики обоих фреймворков. Однако, в TensorFlow такой граф необходимо явно инициализировать и передавать ему переменные и операции. В PyTorch 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 [\*], 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 [\*], 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.

**(Описание данных)**

Для обучения и тестирования модели был взят набор данных «Crack Segmentation» [\*]. Это большая база, состоящая из 11,298 пар изображений размера 448x448 пикселей. В каждой такой паре первая картинка является цветной фотографией поверхности с трещинами, а вторая бинарной маской первой, где пиксели, содержащие трещины, окрашены белым, а все остальные черным.



(Описание)

В рамках исследования размер картинок 448x448 пикселей является избыточным, т. к. значительно увеличит требования к объему видеопамяти, за счет увеличения объема самой модели. Поэтому в процессе обучения, из каждой такой картинки будет случайным образом вырезаться изображение размером 112x112. Так как на части картинок трещины занимает меньшую половину площади, операция вырезания создаст изображения с отсутствием трещин, что даст необходимый набор данных для решения задачи классификации. Сама себе классификация нужна здесь по двум причинам. Первая причина — это последующее использование модели в жизни. Если получать от модели только бинарную маску, для того чтобы получить минимальный ответ – есть трещина или нет, достаточно будет обратиться к вероятностному показателю, вместо проверки каждого пикселя бинарной маски. Вторая причина – бинарный показатель играет роль в непосредственном обучении модели. Подробнее этот момент будет рассмотрен в дальнейшем описании функции потерь.

Также к вырезанному экземпляру картинки в процессе обучения будут случайным образом применяться дополнительные операции обработки изображений. Данный процесс называется аугментацией данных, и нужен чтобы разнообразить поток данных, предоставляемый нейронной сети на этапе обучения.