САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Направление: 01.03.02 Прикладная математика и информатика

ООП: Прикладная математика, фундаментальная информатика и программирование

Корчевнюк Татьяна Сергеевна

Детектирование дефектов металлических пластин с помощью методов машинного обучения

Выпускная квалификационная работа

Научный руководитель:
к. т. н., доцент Гришкин В. М.
Рецензент:
к.ф.-м.н., доцент Козынченко В. А.

Санкт-Петербург
2022 год

SAINT PETERSBURG STATE UNIVERSITY

Specialty: 01.03.02 Applied mathematics and computer science

Main educational program: Applied mathematics, fundamental computer science and programming

Korchevniuk Tatiana Sergeevna

Bachelor Thesis

Detecting defects in metal plates using machine learning methods

Qualification level: **bachelor**

Scientific Supervisor:
Candidate of Technical Sciences,
Assoc. Prof. Grishkin V.M.
Reviewer:
Candidate of Physico-Mathematical Scienses,
Assoc. Prof. Kozynchenko V.A.

Saint Petersburg
2022

Содержание

Введение

В наше время, ввиду доступности инструментария для создания фотографий каждому человеку, ежедневно увеличивается количество изображений. Вследствие чего появляется необходимость как-то обрабатывать и анализировать все эти данные. Машинное обучение и компьютерное зрение являются важной составляющей этого анализа, поскольку отпадает необходимость просматривать каждое изображение вручную. Конечно же с автоматизацией обработки данных возникают и некоторые сложности. Если для человека представляется легкой задачей определить к какому из предложенных классов относится данное изображение или распознать какой-либо объект на фотографии, то для машины потребуется сложный алгоритм и большое количество размеченных данных на котором она будет обучаться. Кроме того для каждой конкретной задачи может понадобиться свой алгоритм и своя база данных. И если одна модель будет хорошо работать для одной задачи, то совсем не факт, что она будет иметь такой же успех для решения другой, хоть и похожей задачи.

Одно из направлений машинного обучения и компьютерного зрения – детектирование объектов. Оно включает в себя задачи локализации и классификации, а также находит применение в большом количестве областей таких как робототехника, социальные сети, поиск информации в интернете, медицине и прочих.

Отдельно хотелось бы отметить применение данного направления на производстве стали. Создание виртуального помощника, способного найти и классифицировать дефект является актуальной задачей. Сталь – это один из важнейших материалов, используемых для машиностроения и приборостроения. Изделия из нее устойчивы к естественному и искусственному износу, что обеспечило ее востребованность. Чтобы сделать производство более эффективным, необходимо качественное выявление дефектов.

Постановка задачи

Целью данной работы является решение задачи детектирования дефектов металлических пластин с помощью методов машинного обучения. В данной задаче необходимо локализовать дефекты и классифицировать их. Всего представлено четыре обезличенных класса, обозначенных соответствующими цифрами. Задача является multilabel, т.е. на одном изображении может быть один или несколько дефектов.

Для достижения поставленной цели необходимо решить следующие подзадачи:

- Произвести анализ данных снимков металлических пластин и их предварительную обработку;
- Проанализировать существующие подходы к решению задачи детектирования;
- Реализовать алгоритмы, позволяющие детектировать дефекты металлических пластин;
- Произвести тестирование реализованных методов;
- Сформировать и вывести результаты работы каждого алгоритма;
- Произвести сравнение результатов работ алгоритмов и сделать выводы.

Пример входных данных и ожидаемых результатов

На Рисунке 1 представлен пример **исходного снимка**, размера 256 × 1600.

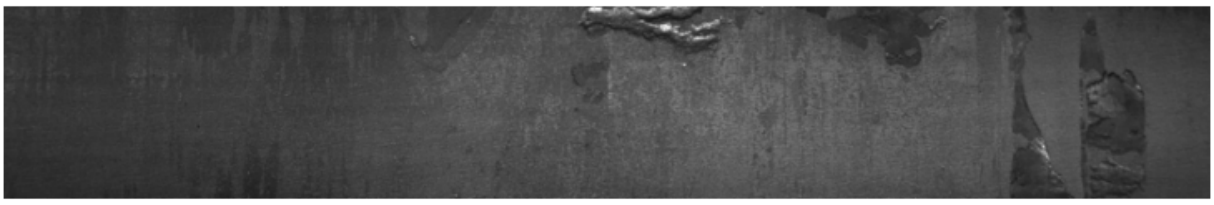

Рисунок 1 – исходный снимок

Ожидаемый результат: массив, содержащий номера классов, найденных на изображении и координаты дефектов. В данном примере найден один дефект: [4]. Для удобства визуализации представим их в виде сегментационной маски (Рисунок 2).

Рисунок 2 – семантическая маска

Используемые инструменты

Ввиду обработки большого объема данных и использования алгоритмов, требующих высоких вычислительных мощностей возникла необходимость в поиске среды разработки, предоставляющей необходимые возможности. А именно с достаточным количеством памяти и доступом к видеокарте, для ускорения работы моделей.

В качестве такой среды разработки была выбрана платформа Kaggle от компании Google. На ней предоставляется ОЗУ объемом 13 Гб, а также дается доступ к NVIDIA TESLA P100 GPU с памятью объемом 16 Гб, которая ускоряет обучение модели в 12.5 раз.

В данной работе задача решается на языке программирования Python с использованием библиотеки Pytorch.

Обзор литературы

MaskR-CNN

Mask R-CNN[1] – модель сверточной нейронной сети, предложенная, как модифицирование модели Faster R-CNN[2]. Она имеет два выхода для распознавания ограничивающей рамки и классификации объекта. В модели Mask R-CNN к ним параллельно добавляется третья ветвь для предсказания маски объекта, которая отличается от двух других. Схематическая архитектура модели представлена на Рисунке 3. Вместо преобразования в короткие выходные вектора полносвязными слоями, в этой ветви прогнозируется маска для каждой области, используя FCN[10]. Такой метод требует меньше параметров и является более точным.

Для того чтобы это реализовать применяется процедура RoIAlign. В отличии от процедуры RoIPool, которая была использована в модели Faster R-CNN, RoIAlign не имеет округления, а для вычисления точных значений входных признаков используется билинейная интерполяция[4] по четырем ближайшим целочисленным точкам в каждой области интереса. После чего результат агрегируется.

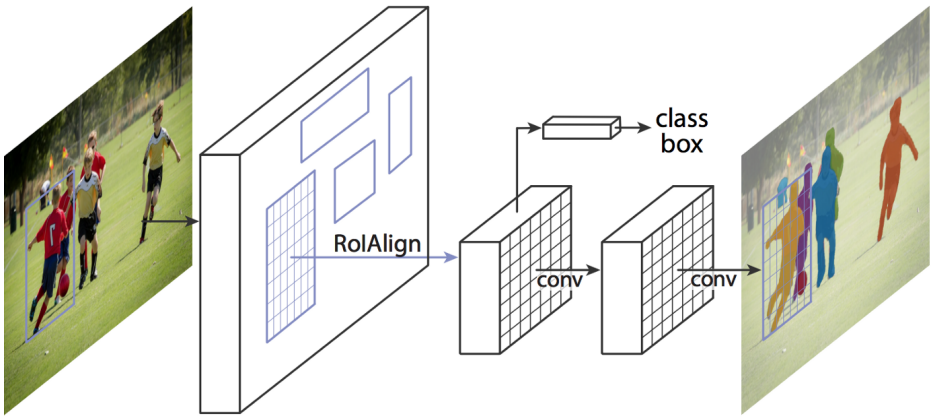

Рисунок 3 – Архитектура Mask R-CNN[1]

Данная модель получила широкое распространение в задачах распознавания опухолей молочной железы[5], меланомы[6], в задачах обнаружения судов[7], дефектов поверхности рельс[8] и других.

U-Net

U-Net[9] – нейронная сеть, основанная на архитектуре полностью сверточной сети (FCN[10]). Идея модели заключается в том, что в части повышающей дискретизации есть большое количество функциональных каналов, которые позволяют сети распространять контекстную информацию на уровни с более высоким разрешением. Как следствие, путь расширения более или менее симметричен пути сокращения и дает U-образную архитектуру (она представлена на Рисунке 4.). Часто левую часть сети называют кодером, а правую декодером.

Кроме того, у модели имеются пропускные соединения между слоями пути сокращения и расширения, что позволяет восстановить пространственную информацию, которая теряется во время максимального объединения.

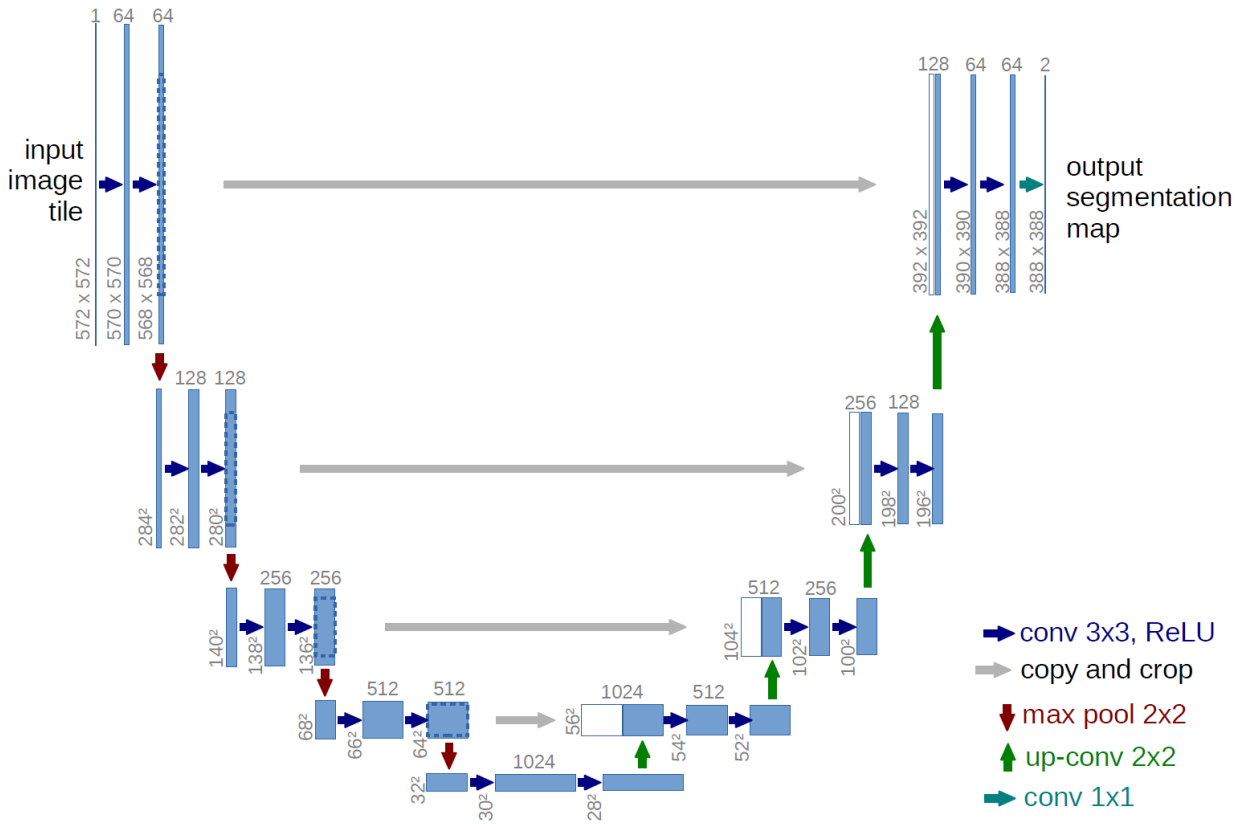

Рисунок 4 – Архитектура U-Net[9]

Одной из отличительных особенностей U-Net является то, что она хорошо решает задачи, даже при недостатке обучающих данных. Она была представлена как модель для сегментации медицинских снимков[11], [12], [13], однако хорошо себя показывает и в других областях. Например, в таких как

обнаружение трещин в бетоне[14], сегментация корней в почве[15] и других.

Идея данной модели получила большую популярность и появилось множество ее модифицированных версий. Одна из таких идей – использование в качестве кодера предобученной модели сверточной нейронной сети. Первыми из подобных моделей стали **TernausNet**[16], которая предсталяет собой U-Net подобную архитектру, где в качестве кодировщика используется предобученная нейронная сеть VGG16[17], и **LinkNet-34**[18] с кодировщиком ResNet34[19].

Отдельно хотелось бы остановиться на последней модели (ее архитектура представлена на Рисунке 5). Здесь в качестве кодировщика используется остаточная нейронная сеть, отличительной особенностью которой является наличие специальных «shortcut connections». Их идея состоит в том, что они пропускают один или несколько слоев и далее их выходы добавляются к выходам пропущенных слоев. С их помощью упрощается оптимизация модели, а также можно обучать более глубокие сети, не боясь проблемы затухания градиента.

Модель LinkNet-34[18] и ее аналоги с другим количеством слоев достаточно часто используют в задачах сегментации, поскольку она эффективна и проста в реализации. Так же как и U-Net получила распространение в медицине[20], [21], [22] кроме этого применялась к задачам распознавания облаков на спутниковых снимках[23], мониторинга заболеваний тепличных растений[24] и другим.

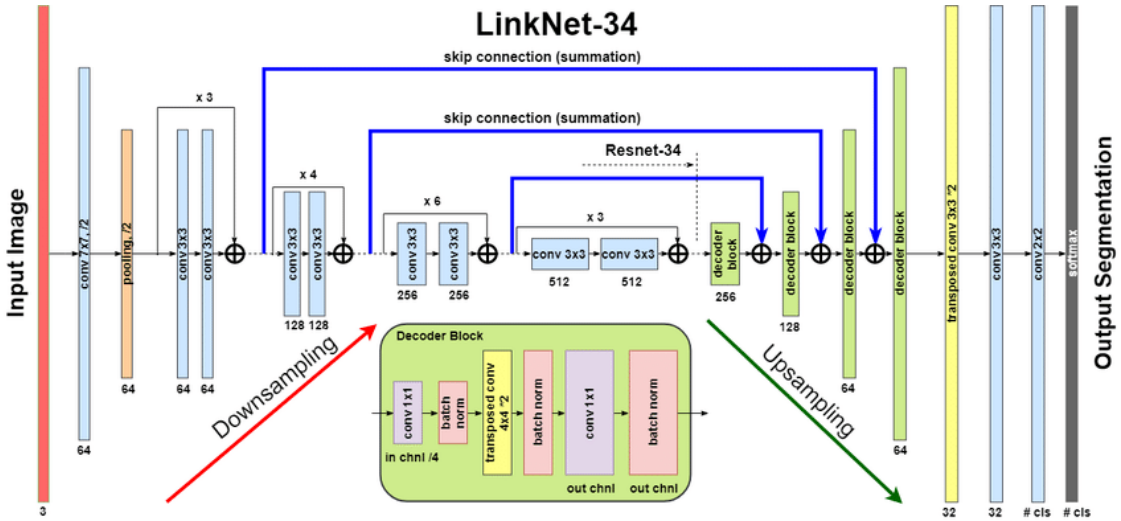

Рисунок 5 – Архитектура LinkNet-34[18]

U-Net++

Еще одна модификация модели U-Net направлена на то, чтобы преодолеть семантический разрыв между картами признаков кодировщика и декодера перед слиянием. В U-Net с помощью пропускных соединений обрезанные карты признаков кодировщика напрямую поступают в декодер, а в **U-Net++**[25] они перед этим подвергаются плотному блоку свертки, количество слоев свертки которого зависит от уровня сети.

Идея плотного блока свертки была предложена в рамках модели DenseNet[26]. Она состоит в том, что от каждого слоя блока проводится подключение к каждому последующему слою. Следовательно, каждый слой получает карты признаков всех предыдущих слоев.

Таким образом блок плотной свертки приближает семантический уровень карт признаков кодировщика к уровню карт признаков, ожидающих декодера. Использование этой идеи упростило обучение сети.

Unet++, также как и U-Net, была предложена для решения задачи сегментации медицинских изображений и хорошо показала себя в их решении: [27], [28], [29], [30]. Также используется для сегментации изображений и в других областях, например, для обнаружения ударных кратеров[31], сегментации дорожного мусора[32] и других.

Dice коэффициент

Dice коэффициент (индекс Серенсена-Дайса) представляет собой статистический инструмент, который измеряет сходство между двумя наборами данных. Данный коэффициент часто используется для оценки качества работы сегментационных моделей и вычисляется по следующей формуле:

$$D(X, Y) = \frac{2|X \cap Y|}{|X| + |Y|}, \tag{1}$$

где X - предсказанный набор пикселей, а Y - целевой.

Глава 1. Методы машинного обучения для детектирования дефектов

1.1. Анализ и предварительная обработка данных

Одним из важных этапов решения задачи является предварительная обработка данных. Правильный анализ позволит более грамотно выбрать метод решения наиболее подходящий для данной задачи.

1.1.1. Анализ данных

Первым этапом решения задачи является анализирование базы данных. Она представлена в виде csv-файла и набора изображений размера 256 × 1600. Всего 6666 изображений, из которых 4999 будут использоваться для обучения, а 1667 для теста. Csv-файл содержит в себе три колонки. В первой содержится перечень названий снимков, во второй класс дефекта и в третьей – его закодированная маска. Тип кодирования: RLE. Он заключается в том, что дается последовательность пар чисел. Первое из которых обозначает начальный пиксель, а второй их количество. Часть таблицы представлена на Рисунке 6.

	ImageId	ClassId	EncodedPixels
0	0002cc93b.jpg	1	29102 12 29346 24 29602 24 29858 24 30114 24 3...
1	0007a71bf.jpg	3	18661 28 18863 82 19091 110 19347 110 19603 11...
2	000a4bcdd.jpg	1	37607 3 37858 8 38108 14 38359 20 38610 25 388...
3	000f6bf48.jpg	4	131973 1 132228 4 132483 6 132738 8 132993 11 ...
4	0014fce06.jpg	3	229501 11 229741 33 229981 55 230221 77 230468...

Рисунок 6 – Csv-файл

Теперь визуализируем наши данные для каждого класса, выведем снимок с дефектом (дефекты разных классов обозначены разными цветами) и маску (на маске выводится только дефект, соответствующий номеру класса). Результат показан на Рисунке 7.

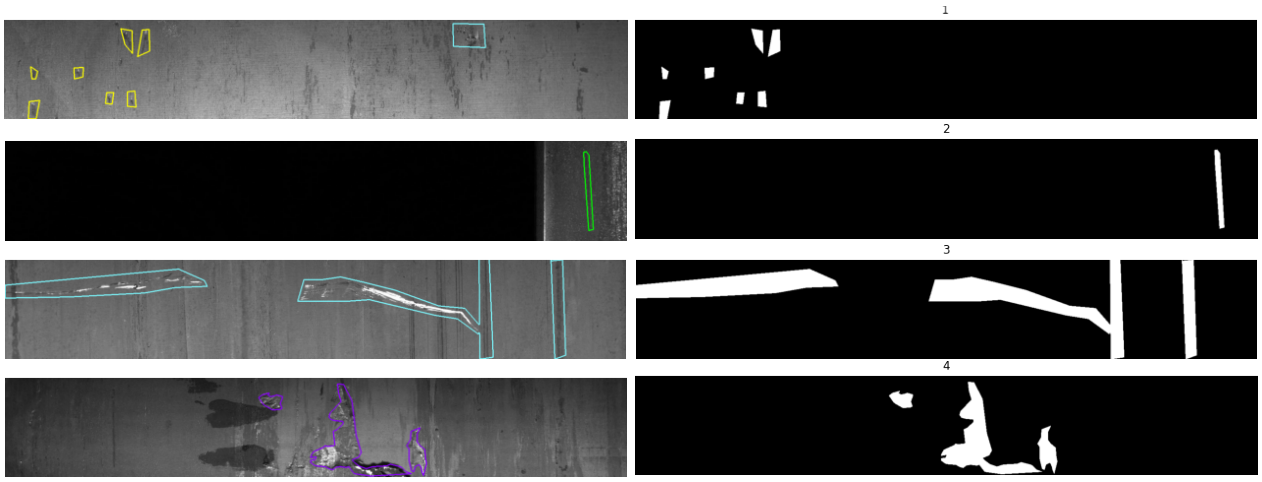

Рисунок 7 – Визуализация для каждого класса

После внимательного просмотра снимков и их сегментационных масок была замечена некоторая грубость в разметке данных. Часть размечена четко с учетом формы дефекта (Рисунок 8), а часть весьма простыми формами, не повторяющими очертания дефекта (Рисунок 9). Такое различие в разметке могло плохо сказаться не только на оценке качества работы модели, но и на ее обучении.

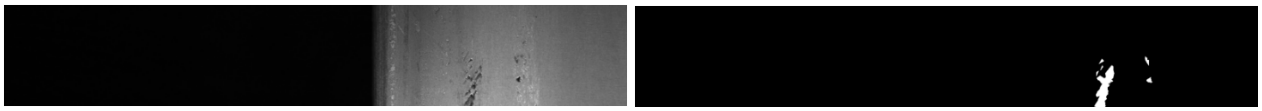

Рисунок 8 – пример разметки, учитывающей форму дефекта

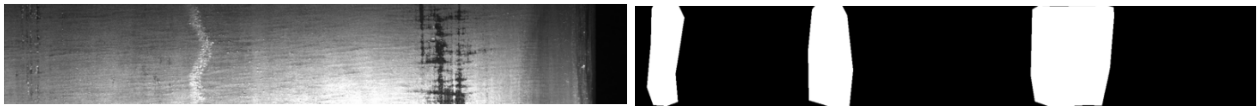

Рисунок 9 – пример разметки, не учитывающей форму дефект

1.1.2. Подготовка данных к обучению

Подготовка данных к обучению состоит из нескольких этапов. Первый из них – преобразование csv-файла к более удобному формату. Теперь если на картинке больше одного дефекта, то ей соответствует ровно одна строка, а в столбцах с классами и масками находятся списки, содержащие все классы и маски, содержащиеся на данном изображении. Результат представлен на Рисунке 10. Такой формат удобен не только для дальнейшей подготовки

данных, но и для продолжения их анализа. Второй – написание функции для декодирования масок дефектов.

	ImageId	ClassesList	EncodedPixels
0	0002cc93b.jpg	[1]	[29102 12 29346 24 29602 24 29858 24 30114 24 ...
1	0007a71bf.jpg	[3]	[18661 28 18863 82 19091 110 19347 110 19603 1...
2	000a4bcdd.jpg	[1]	[37607 3 37858 8 38108 14 38359 20 38610 25 38...
3	000f6bf48.jpg	[4]	[131973 1 132228 4 132483 6 132738 8 132993 11...
4	0014fce06.jpg	[3]	[229501 11 229741 33 229981 55 230221 77 23046...
5	0025bde0c.jpg	[3, 4]	[8458 14 8707 35 8963 48 9219 71 9475 88 9731 ...

Рисунок 10 – Результат преобразования csv-файла

Рассмотрим распределение данных по классам. На Рисунке 11 видно, что оно неравномерное. На снимках данного датасета значительно преобладают дефекты третьего класса, а второго наоборот в меньшинстве. Это необходимо учитывать при выборе моделей, их параметров и метрик оценки качества.

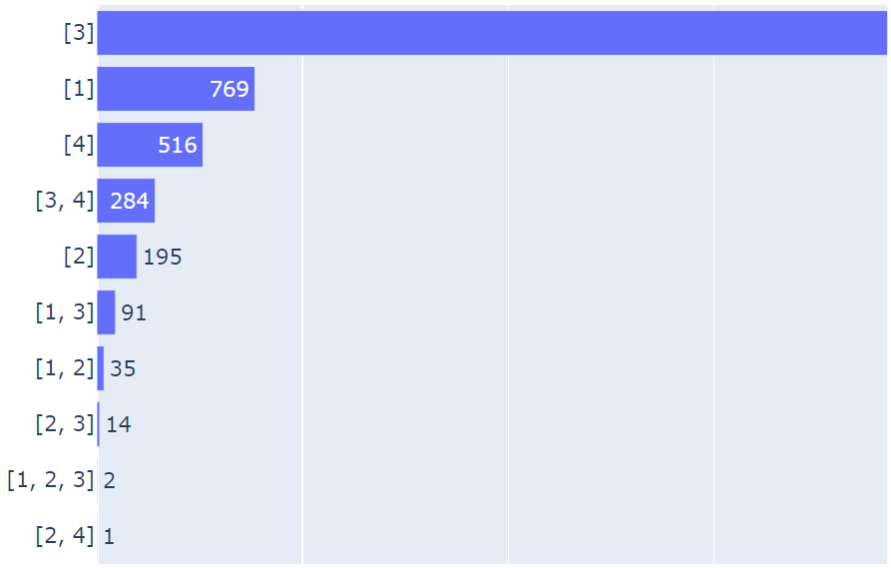

Рисунок 11 – Распределение по классам

Следующий этап – создание класса SteelDataset, его помощью удобнее организовать обработку данных к виду, необходимому для модели. В нем реализовано изменение размера изображения до 128×800, использование функции для декодирования масок, преобразование изображений и масок в тензо-

ры и реализация аугментации. В данном решении были применены следующие методы аугментации: отражение по горизонтали и по вертикали. Следует отметить, что данные преобразования были применены только к обучающей выборке.

Отдельно, в случае реализации нейронной сети Mask R-CNN, были дополнительно посчитаны ограничивающие рамки для дефектов и их площадь, что обусловлено архитектурой данной сети.

И последний этап подготовки данных – разделение получившегося датасета на обучающий и тестовый. Это делается с учетом распределения данных. Сначала распределяются изображения только с одним дефектом, для каждого класса дефекта 75% данных отводится для тренировочного датасета, а остальные для валидационного. Оставшиеся изображения, имеющие больше одного дефекта, делятся в том же отношении, но без учета номеров классов. Далее каждая из этих выборок делится на батчи. Это делается с помощью метода DataLoader используемой библиотеки. В данной работе размер батчей равен пяти.

1.2. Модели нейронных сетей

В данном разделе будут описаны детали архитектур и реализации выбранных моделей для решения задачи детектирования дефектов металлических пластин.

1.2.1. Модель Mask R-CNN

Архитектуру модели Mask R-CNN обычно делят на две части: backbone и head. Head представляет собой три независимых ветви, предназначенных для классификации, выделения масок объектов и ограничивающих рамок.

Backbone также имеет название RPN (Region Proposal Network). Она состоит из CNN (для выделения признаков), на вход которой подается изначальное изображение; выделения регионов кандидатов и процедуры RoIAlign, которая вычисляет матрицу признаков для каждого региона-кандидата. В качестве backbone, в данной работе предлагается использовать ResNet-50-FPN.

FPN (Feature Pyramid Networks) – метод улучшения качества детектирования объектов, учитывающий большой диапазон их возможных размеров. Его идея состоит в том, что CNN (в данном случае ResNet-50) рассматривается как сужающаяся пирамида. Проходя каждый ее уровень выделяемые карты признаков приобретают большую семантическую, обобщающую способность, но теряют разрешение. С помощью FPN предлагается объединять карты признаков вышележащего слоя с нижележащим путем увеличения размера первого и поэлементного суммирования их содержимого. Увеличение размерности происходит с помощью метода ближайших соседей. Архитектура FPN представлена на Рисунке 12.

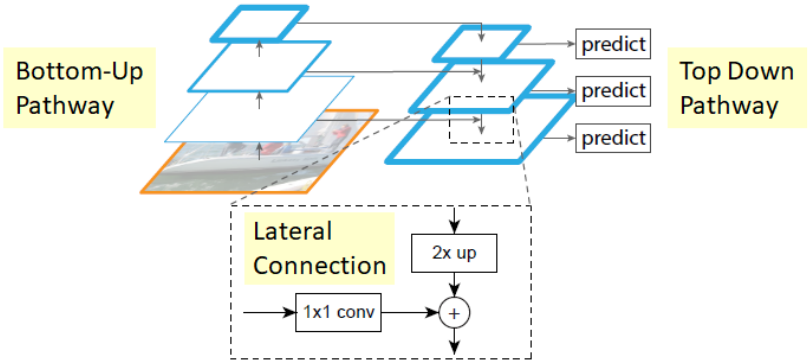

Рисунок 12 – Архитектура FPN

Поскольку у данной сети три выхода (предсказанные классы, ограничивающие рамки и сегментационные маски), необходимо обратить внимание на подсчет loss-функции, которая также включает в себя три компонента и вычисляется следующим образом:

$$L = L_{cls} + L_{box} + L_{mask}, \tag{2}$$

где L_{cls} и L_{box} определяются также как и в [2]:

$$L_{cls} = -lb[p_i p_i^* + (1 - p_i)(1 - p_i^*)], \tag{3}$$

lb – логарифмическая функция потерь, p_i – вероятность нахождения объекта в i якоре, p_i^* – правильный номер класса

$$L_{box} = R(t_i - t_i^*), \tag{4}$$

R – робастная функция потерь(smooth L_1), t_i – представляет 4 параметризованных координаты предсказанного вектора, соответствующего ограничивающей рамке, t_i^* – вектор, соответствующий целевой рамке.

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a, \tag{5}$$

$$t_w = \log(w/w_a), \quad t_h = \log(h/h_a), \tag{6}$$

$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a, \tag{7}$$

$$t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a), \tag{8}$$

x, y, w, h – обозначают центр, ширину и высоту ограничивающей рамки, x, x_a, x^* – обозначают предсказанные рамки, значение якорей и целевые рамки (аналогично для y, h, w)

L_{mask} определяются как средняя бинарная кросс-энтропия:

$$L_{mask} = -\frac{1}{x}\sum x_i \log p(x_i^*) + (1 - x_i) \log (1 - p(x_i^*)), \tag{9}$$

x – количество пикселей, x_i – метка категории, в которой находится пиксель, $p(x_i^*)$ – вероятность прогнозируемой категории x_i.

1.2.2. Модель U-Net

В качестве архитектуры модели U-Net была выбрана ее классическая интерпретация с фильтрами: 64, 128, 256, 512, 1024. Количество входных каналов было установлено равным трем, а выходных – четырем. Для реализации нейросети был написан класс с одноименным названием. Для удобства была написана функция описывающая кодирующий блок сужающейся части сети со следующей структурой:

- свертка 3×3;
- батч-нормализация;
- функция активации ReLU();
- свертка 3×3;
- батч-нормализация;
- функция активации ReLU();
- операция максимального объединения (2×2 степени 2).

А также для декодирующего блока расширяющейся части:

- свертка 2×2, которая уменьшает количество каналов свойств;
- объединение с обрезанной картой признаков пути сокращения;
- свертка 3×3;
- батч-нормализация;
- функция активации ReLU();
- свертка 3×3;
- батч-нормализация;
- функция активации ReLU().

U-Net с предобученным кодировщиком

Данная модель является модификацией предыдущей подобно LinkNet-34, о которой говорилось выше. Единственное отличие заключается в том, что в качестве кодировщика будет использована сеть ResNet с другим количеством слоев, предварительно обученная на датасете ImageNet. А именно будут использоваться версии сети с 18 (ResNet-18) слоями и 50 (ResNet-50). Декодировщик остается таким же как и в классической U-Net. Архитектура ResNet-18 представлена на Рисунке 13.

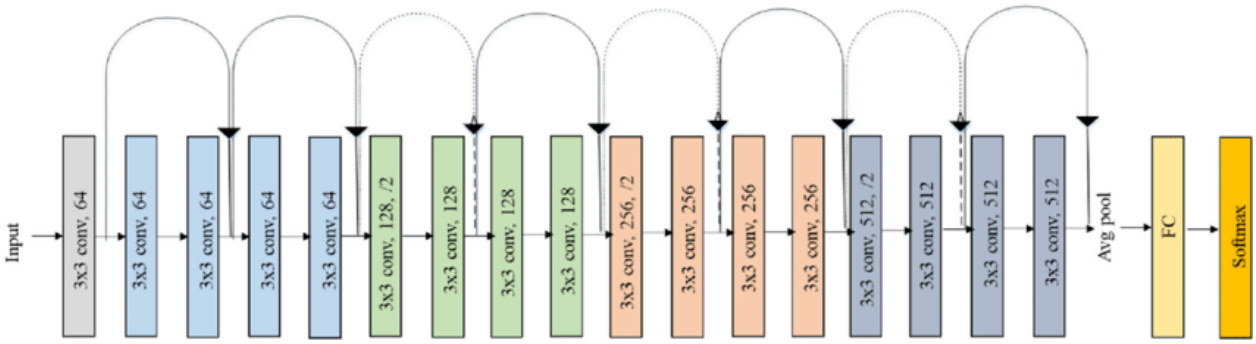

Рисунок 13 – Архитектура ResNet-18

1.2.3. Модель U-Net++

В качестве архитектуры U-Net++ (представлена на Рисунке 14.), так же как и для предыдущей модели была выбрана классическая интерпретация со следующими значениями фильтров: 64, 128, 256, 512, 1024.

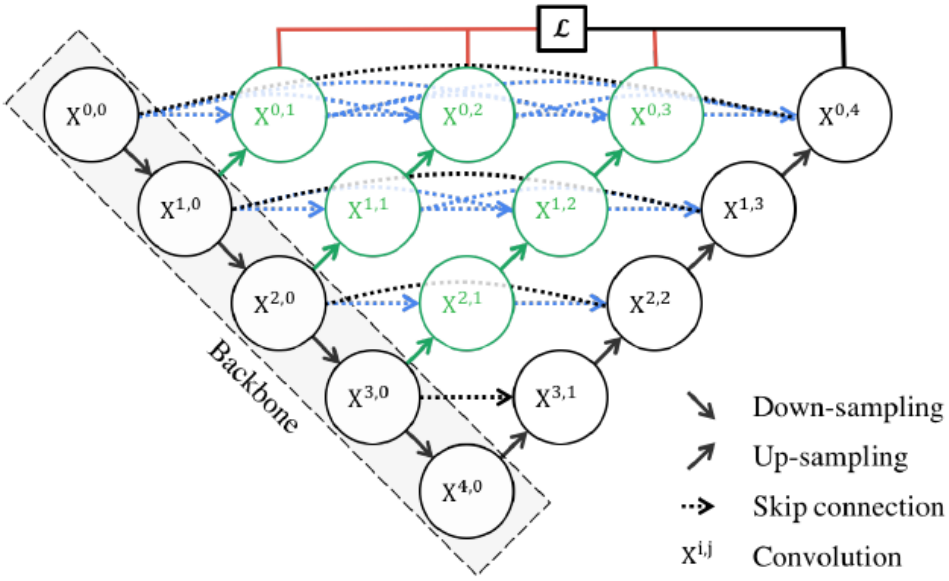

Рисунок 14 – Архитектура U-Net++[25]. Черным показана стандартный вид модели U-Net.

Реализация данной сети мало отличается от U-Net. Единственным изменением является то, что в декодирующих блоках расширяющейся части сети перед тем как производить объединение необходимо применить плотный блок свертки (Dense block). Все их свертки имеют ядро 3×3.

Эту процедуру можно представить в виде следующей формулы:

$$x^{i,j} = \begin{cases} H(x^{i-1,j}), & j = 0 \\ H([[x^{i,k}]_{k=0}^{j-1}, U(x^{i+1,j-1})]), & j > 0, \end{cases} \tag{10}$$

где $x^{i,j}$ – стек карт признаков, полученный из узла $X^{i,j}$ сети, i – номер слоя кодирующей части модели, j – номер свертки, $H()$ – операция свертки, $U()$ - функция активации, [] – означает слой объединения.

Схематически Dense block представлен на Рисунке 15. Важно отметить, что применяемые здесь skip connections отличаются от тех, которые используются в сети ResNet. Если в Residual block происходит суммирование, то здесь объединение карт признаков.

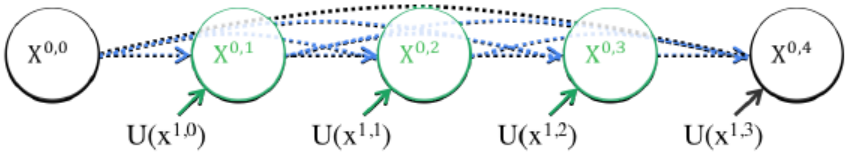

Рисунок 15 – Блок плотной свертки

Глава 2. Экспериментальная реализация

2.1. Mask R-CNN

В данном разделе будет использована модель Mask R-CNN для решения поставленной задачи. Перед началом обучения необходимо правильно настроить модель и подготовить данные. В предыдущей главе было рассказано в общих чертах о подготовке данных, а также было отмечено, что в случае данной модели подготовка немного отличается. Данные для обучения должны быть представлены в виде изображения, преобразованного в тензор, и словаря, содержащего данные о наличии и местонахождении дефектов на данном изображении. Словарь имеет следующую структуру:

- boxes – представляет собой список координат ограничивающих рамок для каждого дефекта, координаты имеют следующий вид: $[x_{min}, y_{min}, x_{max}, y_{max}]$;
- labels – список номеров классов дефектов, находящихся на изображении;
- image_id – идентификатор изображения;
- area – список площадей каждой ограничивающей рамки;
- iscrowd – устанавливается значение равное нулю для того, чтобы данные учитывались во время оценки;
- masks – список сегментационных масок каждого дефекта на изображении.

После подготовки данных необходимо загрузить нейронную сеть. В данном случае мы загружаем Mask R-CNN with backbone ResNet-50-FPN предварительно обученную на датасете COCO. Также необходимо настроить модель под данную задачу, для этого меняем количество классов модели на 5. Четыре класса из них обозначают дефекты из задачи, и один класс отводится под пространство снимка без дефектов.

В качестве оптимизатора при обучении данной модели используется SGD (стохастический градиентный спуск). Лучший результат был достигнут при обучении с начальным шагом градиента равным 0.001, использовании функции накопления импульса (для этого в параметрах SGD был установлен параметр $momentum = 0.9$) и использование L_2-регуляризации (был уста-

новлен параметр $weight\_decay = 0.0005$). Кроме того был использован планировщик, позволяющий менять параметры обучения, в данном случае он менял шаг градиентного спуска раз в 20 эпох, путем умножения его на 0.2. Обучение заняло 4 часа. Его результаты с данными параметрами представлены на Рисунке 16.

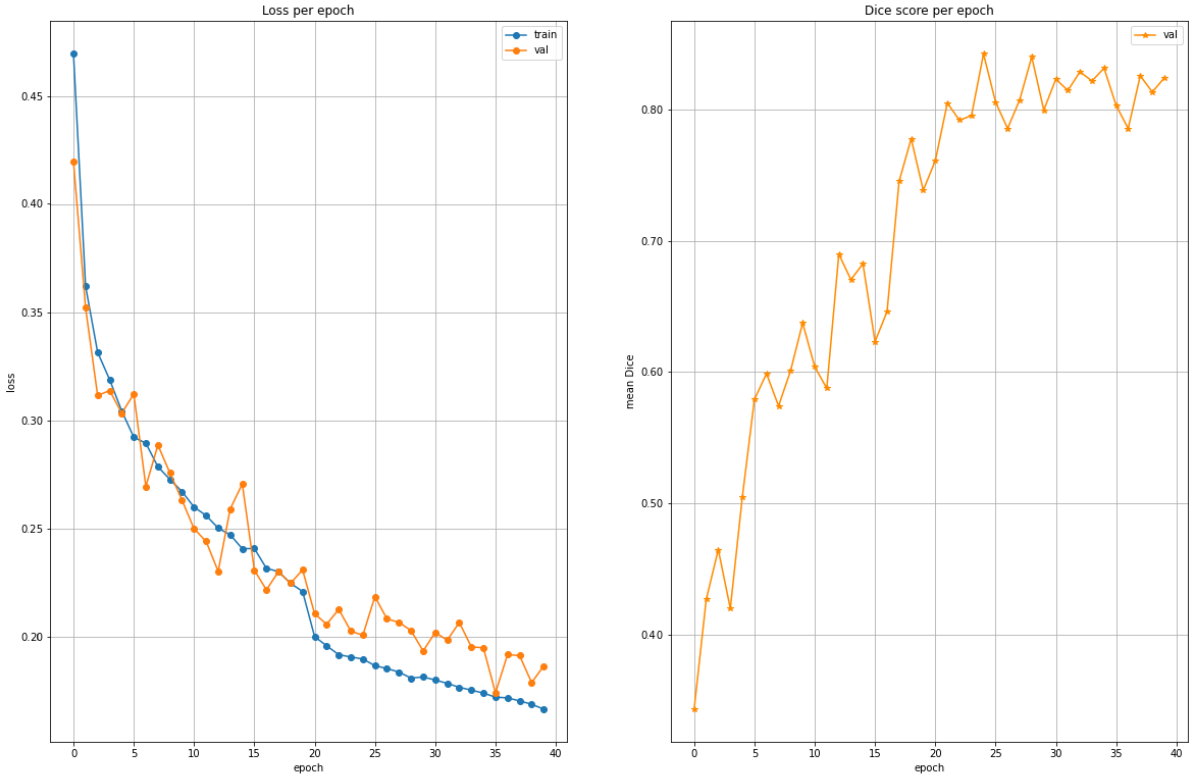

Рисунок 16 – Графики иллюстрирующие результаты обучения модели Mask R-CNN. На графике слева показано изменение значения loss по эпохам, а на графике справа – динамика значений Dice коэффициента

Поскольку при реализации данной модели был использован готовый фреймворк, в котором нет возможности посчитать динамику среднего Dice коэффициента, на правом графике представлено изменение значений, полученных лишь во время валидации.

2.2. U-Net

В данном разделе для решения поставленной задачи будет использована модель U-Net и ее модифицированная версия, предполагающая использование в качестве кодировщика предобученной сверточной нейронной сети.

Также как и в реализации предыдущей модели, первым делать следует обратить внимание на подготовку данных. Она будет одинакова для всех моделей данного раздела.

Целевая семантическая маска представляется в виде тензора с размерностью $[4, 128, 800]$. Таким образом каждому из классов дефектов сопоставляется его семантическая маска. Если на изображении отсутствует дефект того или иного класса, то ему сопоставляется маска, заполненная нулями.

Такой вид сегментирующих масок обусловлен выбором метода для подсчитывания значений loss. Для данной модели была выбрана функция бинарной перекрестной энтропии, реализованная в библиотеке Pytorch. Количество выходных каналов сетей было установлено равным четырем.

U-Net

При экспериментах с данной моделью лучший результат был получен при использовании оптимизатора AdamW. Он является модификацией оптимизатора Adam, который в свою очередь представляет из себя расширение стохастического градиентного спуска. AdamW исправляет алгоритм регуляризации L_2 своего предшественника путем уменьшения веса таким образом, что чем он больше, тем больший штраф на него накладывается.

Для данной модели были подобраны следующие параметры: начальный шаг градиента – $0,001$ и $weight\_decay = 0.0001$. Кроме того был использован планировщик для регулирования шага градиента, что позволяет более тонко настраивать модель. Таким образом, начиная с 35 эпохи был установлен шаг 0.0006, а с 43 эпохи 0.0001. Всего обучение длилось 53 эпохи. Результаты представлены на Рисунке 17.

Все обучение заняло 5.5 часов, и было достигнуто среднее значение Dice коэффициента равное 0.822. Были предприняты попытки продолжить обучение модели, но это не дало лучших результатов.

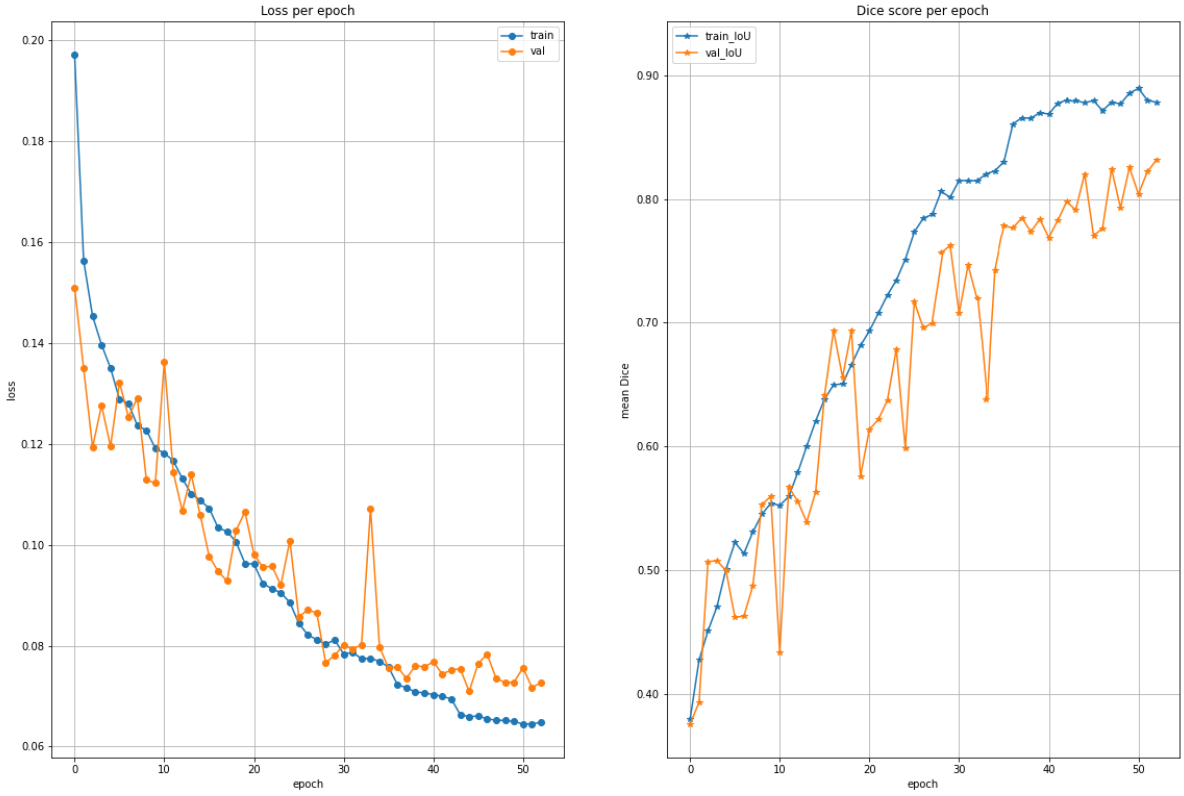

Рисунок 17 – Графики иллюстрирующие результаты обучения модели U-Net. На графике слева показано изменение значения loss по эпохам, а на графике справа – динамика значений Dice коэффициента

U-Net with ResNet-18 encoder

Также как и для предыдущей модели в качестве оптимизатора был выбран AdamW с начальным градиентным шагом $0,0004$ и $weight\_decay = 0.0001$. Был использован планировщик, начиная с эпохи номер 30, шаг градиента стал равен $0,0004$. Всего обучение длилось 40 эпох, что заняло около семи часов. Удалось достичь среднее значение Dice коэффициента равное 0.841. Эволюция значений loss и Dice коэффициента на обучающем и валидационном множествах представлена на Рисунке 18.

На графике, показывающем динамику среднего значения Dice коэффициента, видно, что после 30 эпохи значение установилось на определенном уровне, а следовательно обучение можно не продолжать. Была попытка еще больше уменьшить шаг градиента после 33 эпохи, но это не привело к улучшениям.

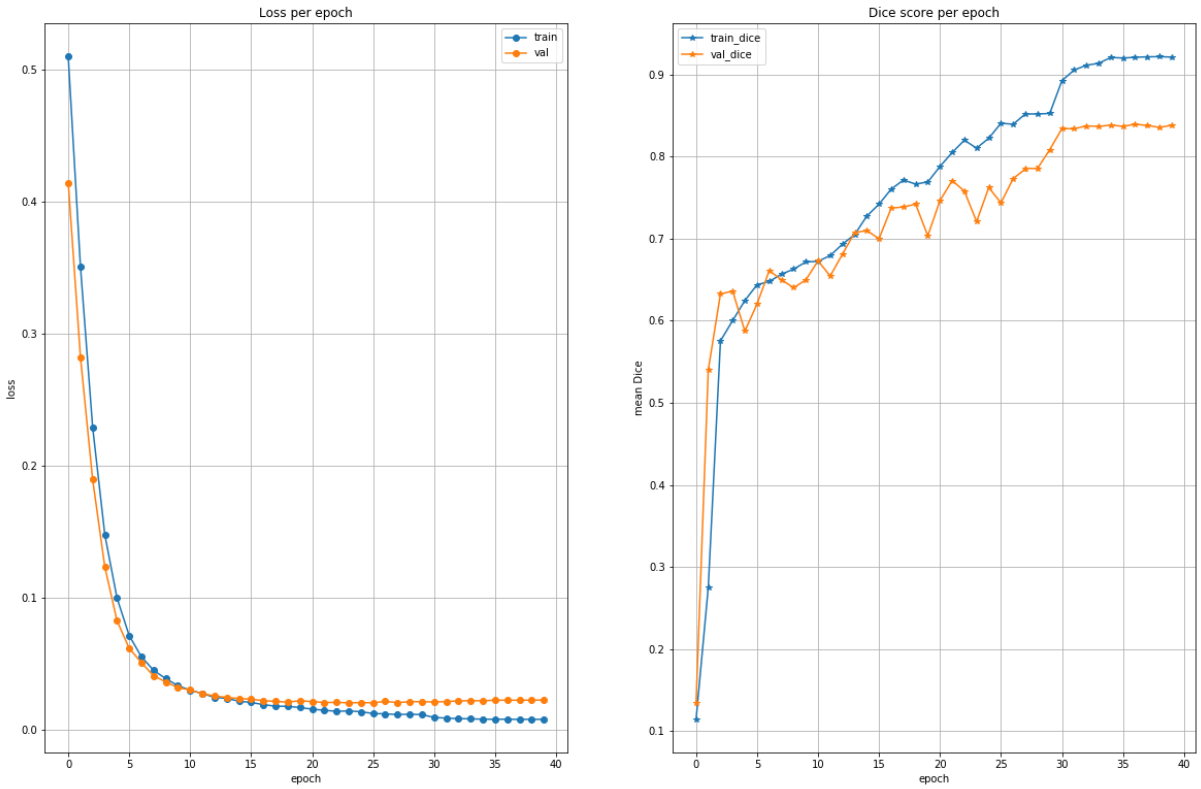

Рисунок 18 – Графики иллюстрирующие результаты обучения модели U-Net with ResNet-18 encoder. На графике слева показано изменение значения loss по эпохам, а на графике справа – динамика значений Dice коэффициента

U-Net with ResNet-50 encoder

Для U-Net with ResNet-50 encoder было проделано все аналогично предыдущей модели, за тем лишь исключением, что в качестве кодировщика использовалась модель с 50 слоями.

Для оптимизатора AdamW был подобран начальный градиентный шаг равный 0.0001, $weight\_decay = 0.0001$. С помощью планировщика, начиная с 20 шага, был установлен шаг градиента равным 0.00001. Спустя 35 эпох было завершено обучение данной модели. Это заняло 6.5 часов. Среднее значение Dice коэффициента при указанных параметрах оказалось равным 0.894. На Рисунке 19 представлены графики ошибки и средних значений Dice коэффициента на тренировочном и валидационном множествах.

Сравнивая графики, представленные в данном разделе, можно заметить, что при использовании предобученных кодировщиков, динамика значения loss не имеет таких скачков, как при обучении классической версии U-Net.

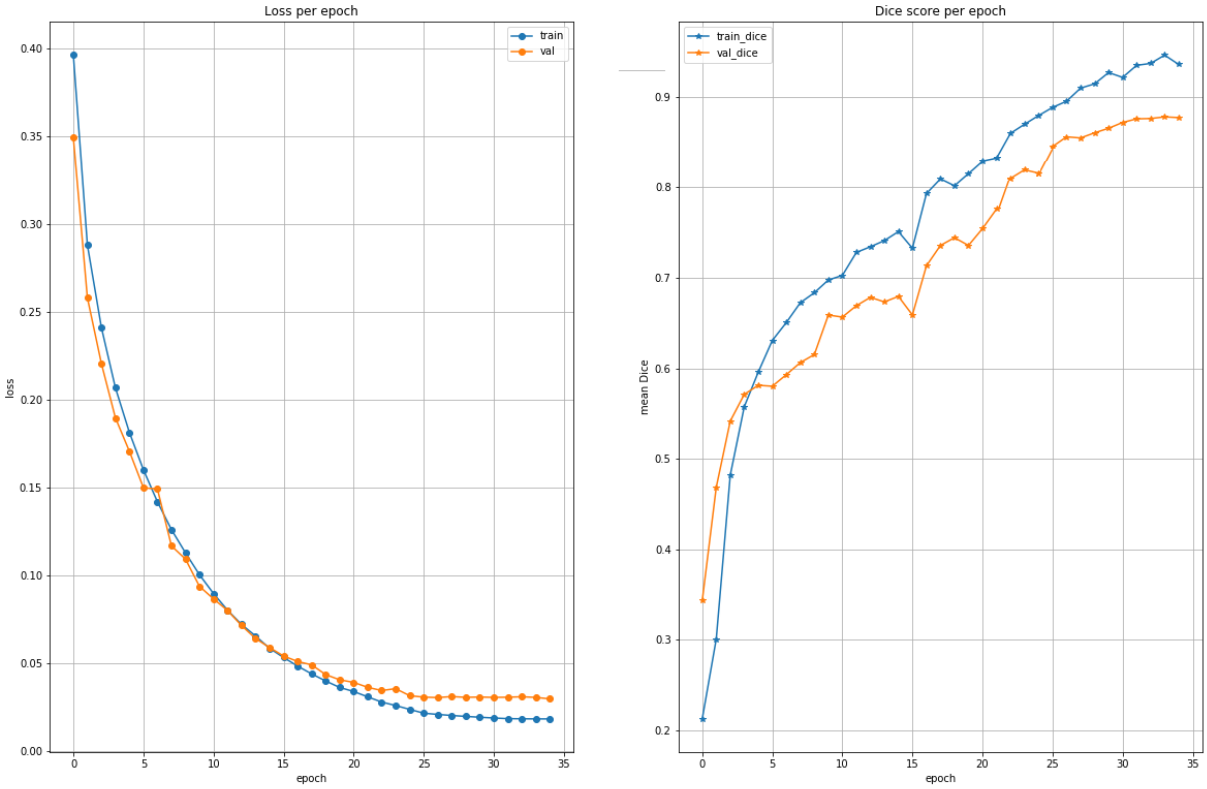

Рисунок 19 – Графики иллюстрирующие результаты обучения модели U-Net with ResNet-50 encoder. На графике слева показано изменение значения loss по эпохам, а на графике справа – динамика значений Dice коэффициента

2.3. U-Net++

Поскольку данная модель является модификацией U-Net, подготовка данных происходит аналогично предыдущему разделу. Также в роле оптимизатора выступает алгоритм AdamW. Наилучших результатов данной модели удалось достичь при следующих параметрах: начальный шаг градиента – $0,0001$ и $weight\_decay = 0.0001$. Так же как и во всех предыдущих случаях был использован планировщик. Таким образом после 20 эпохи был установлен шаг градиента равный 0,00001, а после 30 0,000001. Всего обучение длилось 40 эпох. На Рисунке 20 представлены результаты обучения алгоритма.

Поскольку алгоритм был усложнен добавлением плотных блоков свертки, время обучения увеличилось по сравнению с классической моделью U-Net и заняло порядка 11 часов.

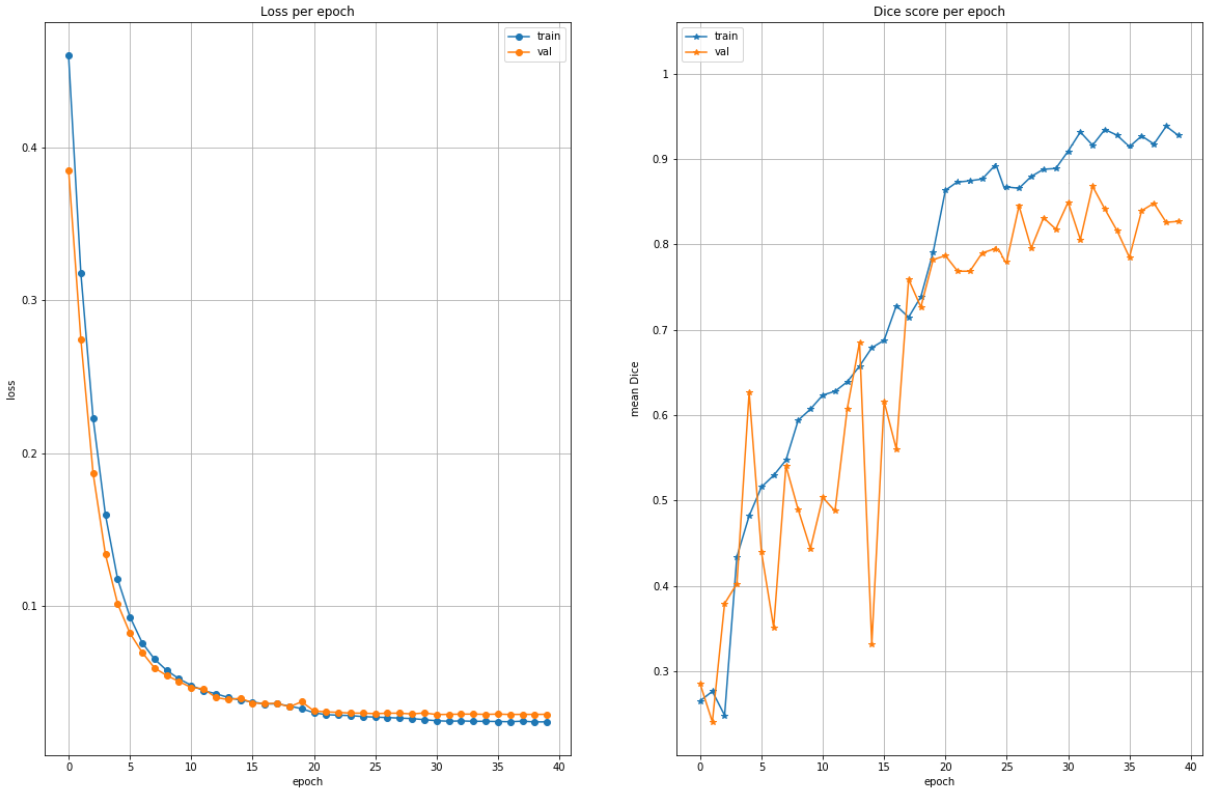

Рисунок 20 – Графики иллюстрирующие результаты обучения модели U-Net++. На графике слева показано изменение значения loss по эпохам, а на графике справа – динамика значений Dice коэффициента

2.4. Результаты

Результаты работы всех представленных алгоритмов для наглядности были занесены в Таблицу 1. Оценка качества работы производилась с помощью вычисления среднего значения Dice коэффициента, описанного ранее. Данная метрика показывает степень сходства двух масок на изображении.

Модель	**mean Dice coefficient**
MaskR-CNN	0.856
U-Net	0.822
U-Net with ResNet18 encoder	0.841
U-Net with ResNet50 encoder	0.894
U-Net++	0.837

Таблица 1 – Результаты работы алгоритмов

В среднем результаты моделей не сильно отличаются. Хуже всего показали себя модели в которых не использовался transfer learning (U-Net и U-Net++). Сравнивая их видно, что модифицированная модель справилась с решением данной задачи лучше, но разница результатов оказалась совсем небольшой.

Выводы

В данной работе стояла задача детектирования дефектов металлических пластин с помощью методов машинного обучения. Из всех приведенных алгоритмов лучше всего себя показал U-Net с предварительно обученным кодировщиком ResNet-50. После обучения он достиг 89% качества, что является довольно хорошим результатом, но для внедрения его в производство стоит найти метод, решающий задачу лучше.

Чтобы улучшить решение поставленной задачи можно попробовать использовать двухэтапные методы, в которых идет сначала классификация, а после сегментация или наоборот. Также ввиду того, что данные были несбалансированы, можно попробовать исправить это путем увеличения датасета и заново применить методы описанные в данной работе.

Заключение

В ходе работы были выполнены все поставленные задачи. Была проанализирована база данных. Рассмотрены различные подходы к решению задачи детектирования и выбрано несколько алгоритмов для реализации. Ими стали Mask R-CNN, U-Net, U-Net с предварительно обученным кодировщиком и U-Net++. Данные алгоритмы были изучены и реализованы на языке Python с использованием библиотеки Pytorch. Произведено обучение всех моделей с подбором параметров для улучшения качества их работы. Обученные модели были протестированы и вычислен средний Dice коэффициент для каждой из них. На основе вычисленного коэффициента было произведено сравнение работы алгоритмов.

Реализованные методы могут быть полезны при детектировании дефектов металлических пластин. В частности, показавший лучший результат в данной работе, алгоритм U-Net with ResNet encoder.

Список используемых источников

[1] He K. et al. Mask r-cnn //Proceedings of the IEEE international conference on computer vision. – 2017. – С. 2961-2969.

[2] Girshick R. Fast r-cnn //Proceedings of the IEEE international conference on computer vision. – 2015. – С. 1440-1448.

[3] Long J., Shelhamer E., Darrell T. Fully convolutional networks for semantic segmentation //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2015. – С. 3431-3440.

[4] Jaderberg M. et al. Spatial transformer networks //Advances in neural information processing systems. – 2015. – Т. 28.

[5] Chiao J. Y. et al. Detection and classification the breast tumors using mask R-CNN on sonograms //Medicine. – 2019. – Т. 98. – №. 19.

[6] Cao X. et al. Application of generated mask method based on Mask R-CNN in classification and detection of melanoma //Computer Methods and Programs in Biomedicine. – 2021. – Т. 207. – С. 106174.

[7] Nie S. et al. Inshore ship detection based on mask R-CNN //IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium. – IEEE, 2018. – С. 693-696.

[8] Guo F. et al. Automatic rail surface defects inspection based on Mask R-CNN //Transportation research record. – 2021. – Т. 2675. – №. 11. – С. 655-668.

[9] Ronneberger O., Fischer P., Brox T. U-net: Convolutional networks for biomedical image segmentation //International Conference on Medical image computing and computer-assisted intervention. – Springer, Cham, 2015. – С. 234-241.

[10] Long J., Shelhamer E., Darrell T. Fully convolutional networks for semantic segmentation //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2015. – С. 3431-3440.

[11] Munir K., Frezza F., Rizzi A. Brain tumor segmentation using 2D-UNET convolutional neural network //Deep Learning for Cancer Diagnosis. – Springer, Singapore, 2021. – С. 239-248.

[12] Sivagami S. et al. Unet architecture based dental panoramic image segmentation //2020 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET). – IEEE, 2020. – С. 187-191.

[13] Kumar S. N. et al. Lung Nodule Segmentation Using UNet //2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS). – IEEE, 2021. – Т. 1. – С. 420-424.

[14] Liu Z. et al. Computer vision-based concrete crack detection using U-net fully convolutional networks //Automation in Construction. – 2019. – Т. 104. – С. 129-139.

[15] Smith A. G. et al. Segmentation of roots in soil with U-Net //Plant Methods. – 2020. – Т. 16. – №. 1. – С. 1-15.

[16] Iglovikov V., Shvets A. Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation //arXiv preprint arXiv:1801.05746. – 2018.

[17] Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition //arXiv preprint arXiv:1409.1556. – 2014.

[18] Shvets A. A. et al. Automatic instrument segmentation in robot-assisted surgery using deep learning //2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). – IEEE, 2018. – С. 624-628.

[19] He K. et al. Deep residual learning for image recognition //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – С. 770-778.

[20] Shvets A. A. et al. Angiodysplasia detection and localization using deep convolutional neural networks //2018 17th IEEE international conference on machine learning and applications (ICMLA). – IEEE, 2018. – С. 612-617.

[21] Konovalov D. A. et al. Automatic segmentation of kidney and liver tumors in CT images.

[22] Natarajan V. A. et al. Segmentation of Nuclei in Histopathology images using Fully Convolutional Deep Neural Architecture //2020 International Conference on Computing and Information Technology (ICCIT-1441). – IEEE, 2020. – С. 1-7.

[23] Workman S. et al. Single image cloud detection via multi-image fusion //IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium. – IEEE, 2020. – С. 1468-1471.

[24] Arvind C. S. et al. Low-Altitude Unmanned Aerial Vehicle for Real-Time Greenhouse Plant Disease Monitoring Using Convolutional Neural Network //Soft Computing for Problem Solving. – Springer, Singapore, 2021. – С. 63-76.

[25] Zhou Z. et al. Unet++: A nested u-net architecture for medical image segmentation //Deep learning in medical image analysis and multimodal learning for clinical decision support. – Springer, Cham, 2018. – С. 3-11.

[26] Huang G. et al. Densely connected convolutional networks //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2017. – С. 4700-4708.

[27] Xu D. et al. Automatic segmentation of low-grade glioma in MRI image based on UNet++ model //Journal of Physics: Conference Series. – IOP Publishing, 2020. – Т. 1693. – №. 1. – С. 012135.

[28] Wang H., Li Y., Luo Z. An improved breast cancer nuclei segmentation method based on unet++ //Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence. – 2020. – С. 193-197.

[29] de Melo M. J. et al. Automatic segmentation of cattle rib-eye area in ultrasound images using the UNet++ deep neural network //Computers and Electronics in Agriculture. – 2022. – Т. 195. – С. 106818.

[30] Li Z. et al. Pneumothorax Image Segmentation and Prediction with UNet++ and MSOF Strategy //2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE). – IEEE, 2021. – С. 710-713.

[31] Jia Y., Liu L., Zhang C. Moon Impact Crater Detection Using Nested Attention Mechanism Based UNet++ //IEEE Access. – 2021. – Т. 9. – С. 44107-44116.

[32] Liao J. et al. Road Garbage Segmentation and Cleanliness Assessment Based on Semantic Segmentation Network for Cleaning Vehicles //IEEE Transactions on Vehicular Technology. – 2021. – Т. 70. – №. 9. – С. 8578-8589.