

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 Программная реализация метода распознавания челюстно–лицевых костей	6
1.1 Средства реализации программного комплекса	6
1.1.1 Выбор языка программирования	6
1.1.2 Выбор библиотеки глубокого обучения	6
1.1.3 Выбор средства реализации машины опорных векторов	6
1.2 Реализация программного комплекса	6
1.2.1 Модель UNet	6
1.2.2 Тренировка модели	7
1.2.3 Классификация зубов при помощи машины опорных векторов	10
1.2.4 Выделение сегментированных участков	10
1.3 Результаты обучения модели	12
1.4 Примеры использования разработанного программного комплекса	13
1.4.1 Пример использования модуля модели UNet	13
1.4.2 Пример использования модуля пользовательского приложения	14
2 Исследования разработанного программного комплекса на применимость для снимков в разных проекциях и быстродействия	16
2.1 Сравнение применимости программного комплекса для снимков в разных проекциях	16
2.2 Сравнение применимости программного комплекса для снимков в разных оттенках	18
2.3 Сравнение времени работы приложения при различном разрешении исходного снимка	20
2.4 Оценка разработанного программного комплекса	22
ЗАКЛЮЧЕНИЕ	23
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	24
ПРИЛОЖЕНИЕ 1. Модуль модели UNet	25
ПРИЛОЖЕНИЕ 2. Модуль пользовательского приложения	32
ПРИЛОЖЕНИЕ 3. Утилитарный модуль	39

ВВЕДЕНИЕ

Согласно исследованиям [1][2], на момент 2018 года 10% всех несчастных случаев и обращений в отделения неотложной помощи приходятся на травмы челюстно–лицевой области головы человека, а 1% всех пластических операций — на контурную пластику (изменение формы) лица.

Для лечения лицевых повреждений требуется диагностика типа травмы, а также причины ее появления. Для диагностики обычно используются КТ и МРТ снимки головы. На основе полученных снимков можно выбрать план лечения.

Нередко при лечении челюстно–лицевых повреждений требуется хирургическое вмешательство, а проведение безоперационной контурной пластики дает непостоянный результат, требующий повторной процедуры спустя некоторое время. Поэтому пациенты часто прибегают к операции, чтобы добиться долгосрочных изменений [2].

Проведение операций по лечению дефектов челюстно–лицевого сустава может быть затруднительным ввиду недостаточных данных, полученных при сборе анамнеза [3]. Для уточнения и расширения показаний пациента можно воспользоваться компьютерными технологиями для распознавания челюстно–лицевых костей и их повреждений. Использование нейронных сетей глубокого обучения позволяет повысить эффективность определения травм лица, что в свою очередь снижает риск замедления лечения и выбора неправильного подхода к реабилитации [4].

Компьютерная диагностика также позволяет выявить дефекты в строении зубочелюстной системы или предотвратить развитие патологических отклонений. Кроме того, она может использоваться как вспомогательный инструмент при проектировании зубочелюстных протезов.

Цель работы — разработать метод распознавания челюстно–лицевых костей черепа по томографическим снимкам головы человека.

Для достижения поставленной цели потребуется:

- Разработать программный комплекс, реализующий интерфейс для взаимодействия с разработанным методом.
- Исследовать разработанный метод на применимость при работе с различными типами томографических снимков и при работе с различными про-

екциями одного снимка.

1. Программная реализация метода распознавания челюстно–лицевых костей

В данном разделе описываются средства реализации программного комплекса. Приводятся листинги программных компонентов, графики процесса обучения разрабатываемой нейронной сети.

1.1 Средства реализации программного комплекса

1.1.1 Выбор языка программирования

Для написания программного комплекса будет использоваться язык программирования Python [5].

Данный выбор обусловлен следующими факторами:

- широкий набор библиотек для работы с нейронными сетями;
- возможность тренировать нейронную сеть на графическом процессоре с использованием технологии CUDA [6];

1.1.2 Выбор библиотеки глубокого обучения

Для создания и обучения модели нейронной сети была выбрана библиотека tensorflow [7] версии 2.3.0. Выбор данной версии обусловлен тем, что версия 2.3.0 является последней версией с поддержкой CUDA 10, которая предоставляется на высоконагруженном кластере NVIDIA DGX2, на котором будет обучаться нейронная сеть.

Кроме того, tensorflow показал себя производительнее, чем pytorch, что является плюсом, так как тренировка модели с tensorflow займет меньше времени, чем с pytorch [8].

1.1.3 Выбор средства реализации машины опорных векторов

Для реализации классификатора машины опорных векторов будет использована библиотека Scikit Learn. Она предоставляет реализацию машины опорных векторов и требует от пользователя только данные для обучения [9].

1.2 Реализация программного комплекса

1.2.1 Модель UNet

Реализация модели построена на классической UNet модели с 23 сверточными слоями. В качестве функции активации на всех слоях, кроме последнего, используется ReLU. На последнем слое используется сигмоидальная функция активации. Она является более точной, чем ReLU, но менее быстрой. Именно поэтому для небольшого увеличения точности она используется именно на

последнем слое.

В листинге 5 (приложение 1) приведена реализация модели, отвечающей за сегментацию изображения.

На выходе модель предоставляет маску для изображения, по которой происходит дальнейшее выделение. Примеры таких масок приведены на рисунках 1.1 и 1.2 для зубов и для нижней челюсти соответственно.

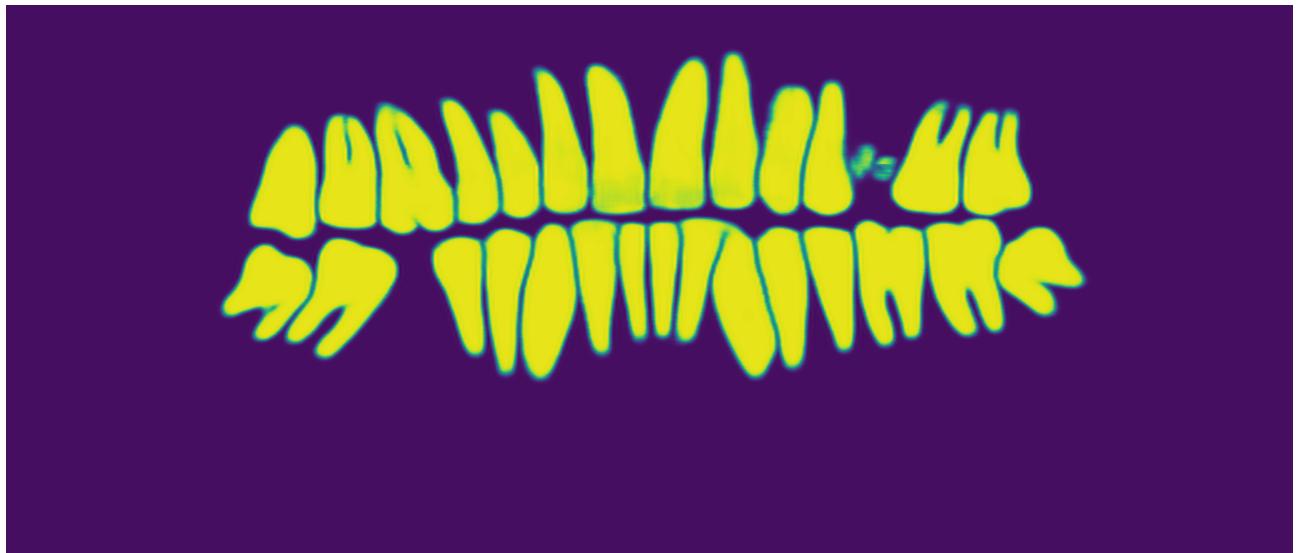


Рис. 1.1: Результат работы модели (зубной состав)

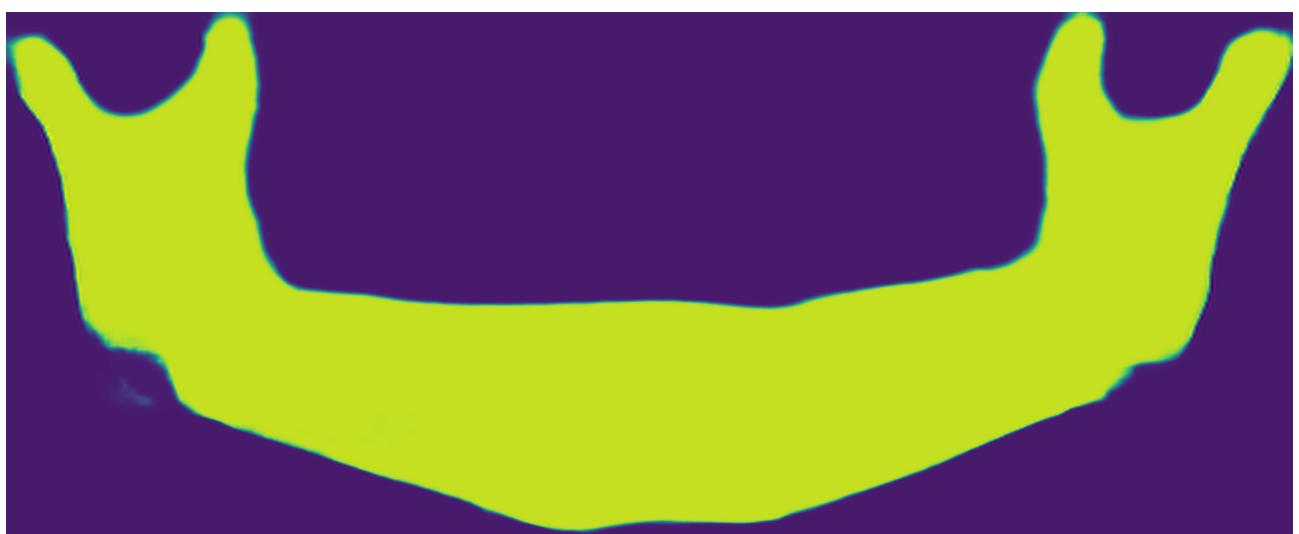


Рис. 1.2: Результат работы модели (нижняя челюсть)

1.2.2 Тренировка модели

Перед тренировкой модели необходимо произвести конвертацию исходных изображений и их масок к размеру 512×512 пикселей, как этого требует модель.

Кроме того, в процессе обучения, тренировочные данные подвергаются преобразованиям, таким как обрезка, изменение контраста, поворот, отражения, применение Гауссова шума и прочие. Это нужно для расширения набора тестовых данных и повышения точности модели.

В качестве функции оптимизации для модели будет использована функция Adam (адаптивная оценка момента). Adam — один из самых эффективных алгоритмов оптимизации в обучении нейронных сетей. Он сочетает в себе идеи среднеквадратичного распространения корня (RMSProp) и оптимизатора импульса. Вместо того чтобы адаптировать скорость обучения параметров на основе среднего первого момента (среднего значения), как в RMSProp, Adam также использует среднее значение вторых моментов градиентов. В частности, алгоритм вычисляет экспоненциальное скользящее среднее градиента и квадратичный градиент.

Обучение проводилось на графическом процессоре NVIDIA Tesla V100 с 32 гигабайтами видеопамяти, что позволило существенно ускорить процесс обучения. Обучение одной эпохи заняло 21 секунду по сравнению с 32 минутами при обучении на процессоре Intel Core I7 6-th Gen.

В листинге 6 (приложение 1) приведена реализация процесса обучения.

На рисунках 1.3 — 1.5 приведены примеры тренировочных данных для модели: снимок лица, маска зубов и маска нижней челюсти соответственно.

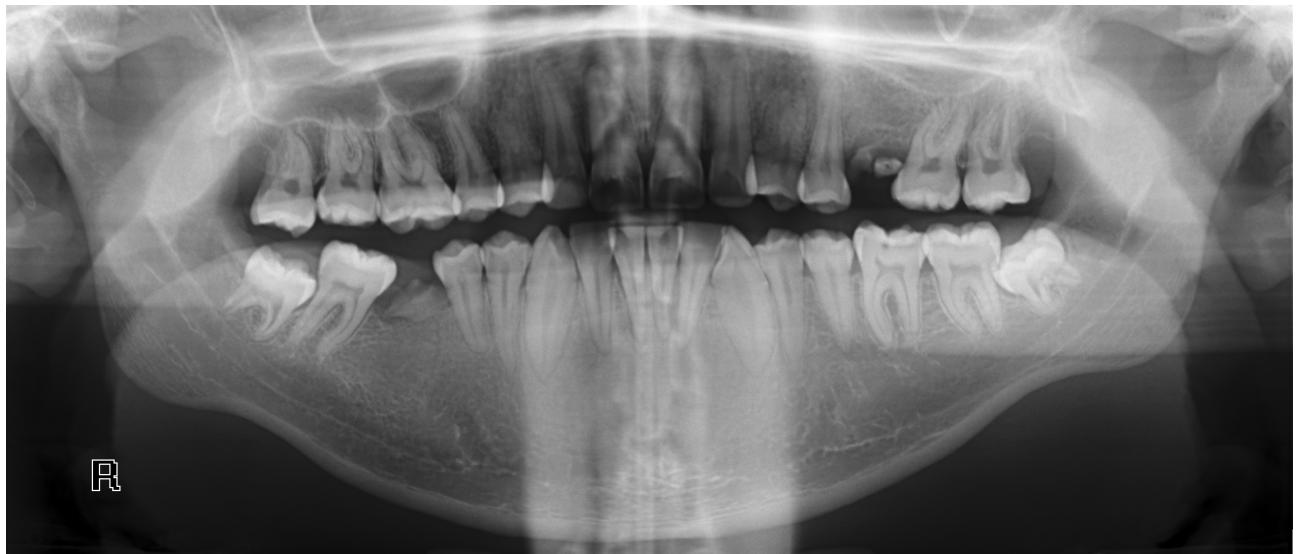


Рис. 1.3: Пример тренировочных данных. Томографический снимок

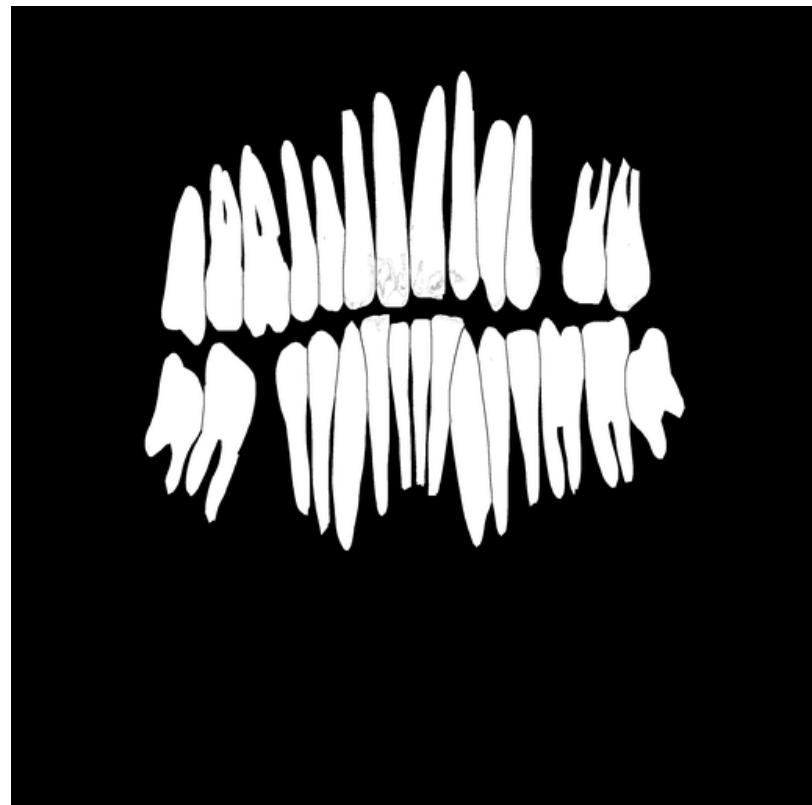


Рис. 1.4: Пример тренировочных данных. Маска зубов



Рис. 1.5: Пример тренировочных данных. Маска нижней челюсти

Тренировка проходила в течение 200 эпох (итераций) для обеспечения стабильного результата точности и потерь.

1.2.3 Классификация зубов при помощи машины опорных векторов

Для реализации машины опорных векторов, используемый датасет был размечен в мануальном режиме. Разметка производилась путем вычисления размеров зубов снимков и выделения полученных размеров в четыре группы: моляры, премоляры, улыки и резцы. При формировании векторов особенностей учитывались два фактора: ширина и высота зуба. Вычислить порядковый номер зуба в ряду не представляется возможным ввиду представления полученных при сегментации данных.

В листинге 11 (приложение 2) приведена реализация и обучение машины опорных векторов.

1.2.4 Выделение сегментированных участков

Для выделения сегментированных (и классифицированных) участков используется метод ССА (от англ. Connected Components Analysis — анализ связанных компонентов). Данный анализ проводится при помощи выделения на изображении объектов переднего плана (сегментированных участков) и последующего анализа полученного участка. Анализ проводится на основе различных факторов, таких как размер, абсолютное и относительное расположение объекта [10].

В листинге 11 (приложение 2) приведена реализация анализа связанных компонентов с их последующим выделением.

На рисунках 1.6 — 1.9 представлены изображения с выделенными сегментированными участками.

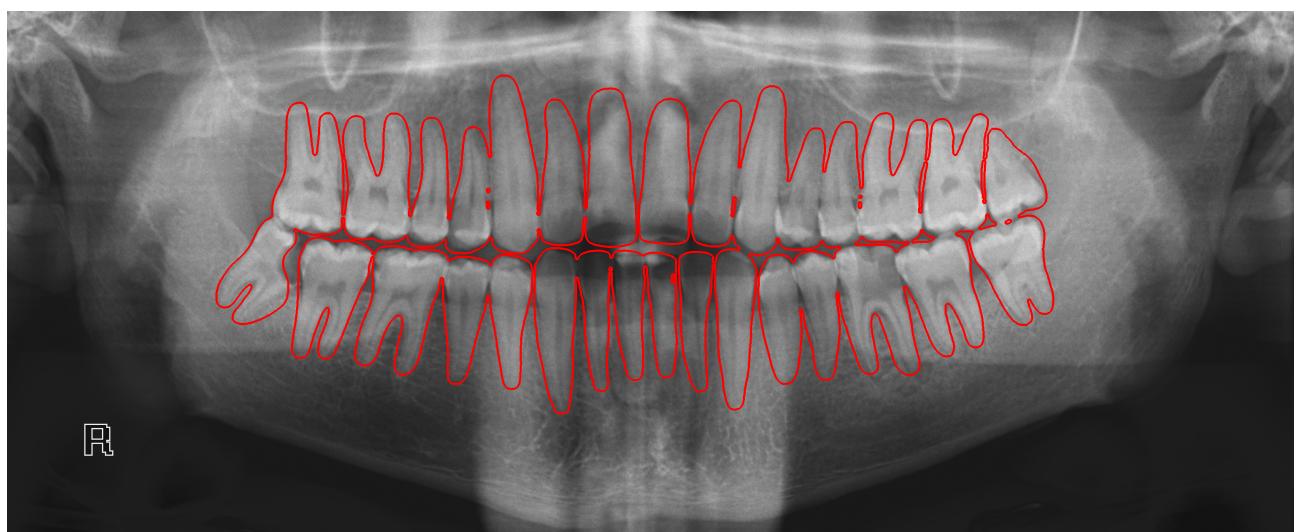


Рис. 1.6: Результат работы семантической сегментации (зубной состав)

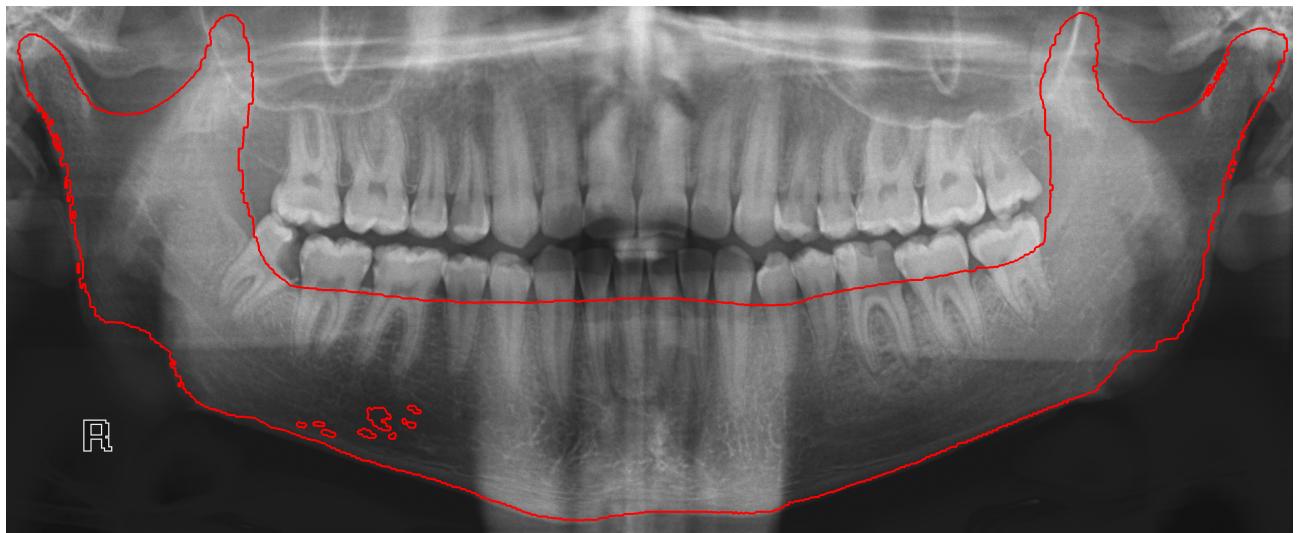


Рис. 1.7: Результат работы семантической сегментации (нижняя челюсть)

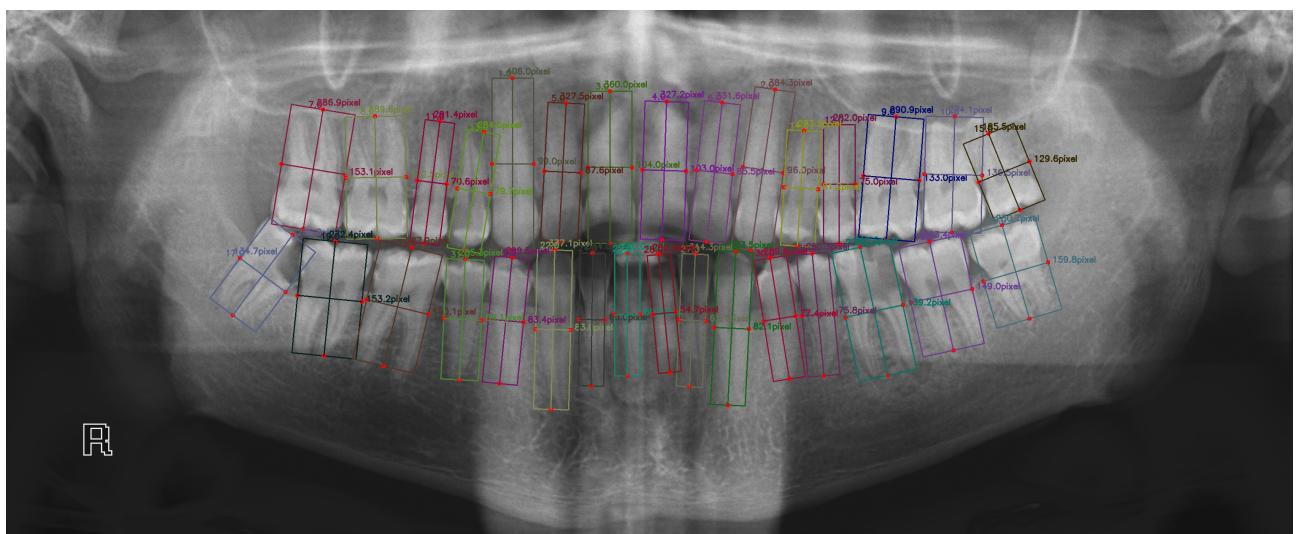


Рис. 1.8: Результат работы сегментации экземпляров

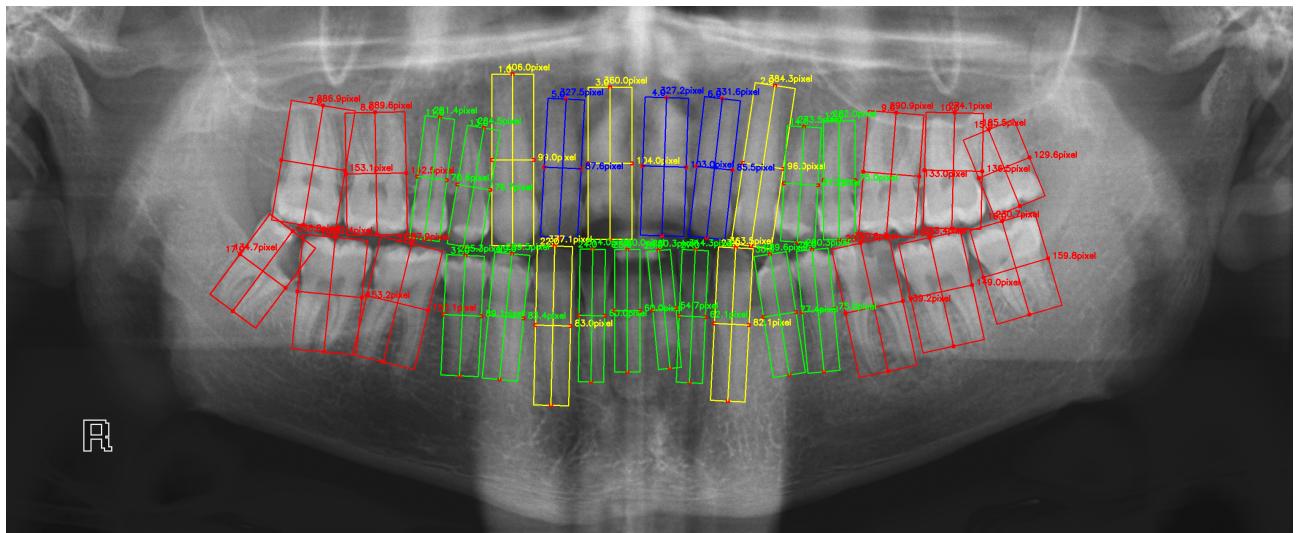


Рис. 1.9: Результат работы сегментации экземпляров (с применением машины опорных векторов)

1.3 Результаты обучения модели

На рисунках 1.10 и 1.11 представлены результаты обучения модели. Точность модели составила 92 процента, а потери составили 4 процента. Стоит заметить, что уже на десятой итерации точность составила не менее 88 процентов.

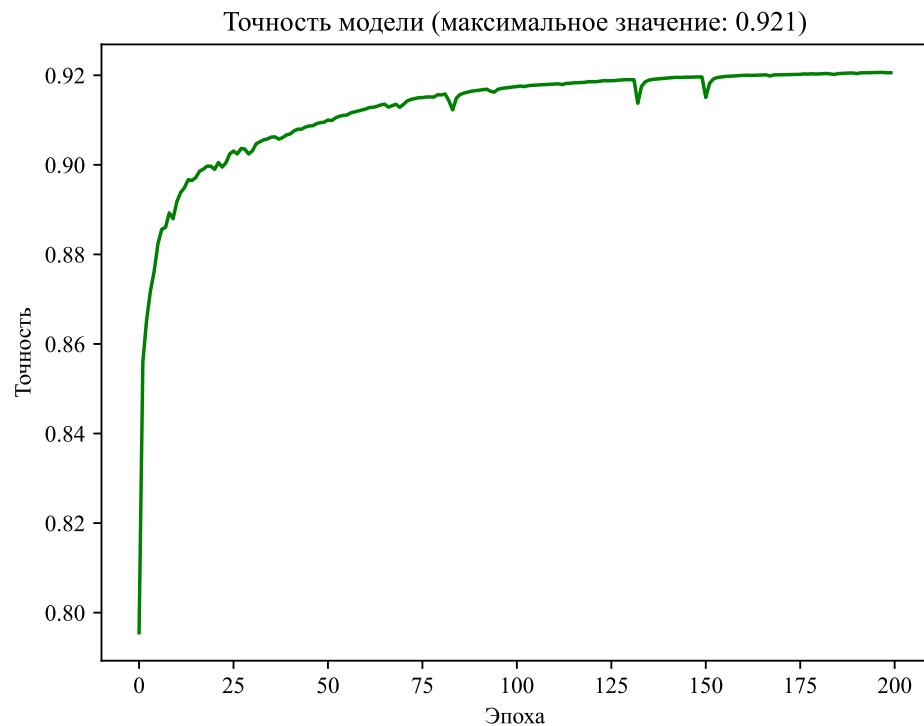


Рис. 1.10: Точность модели

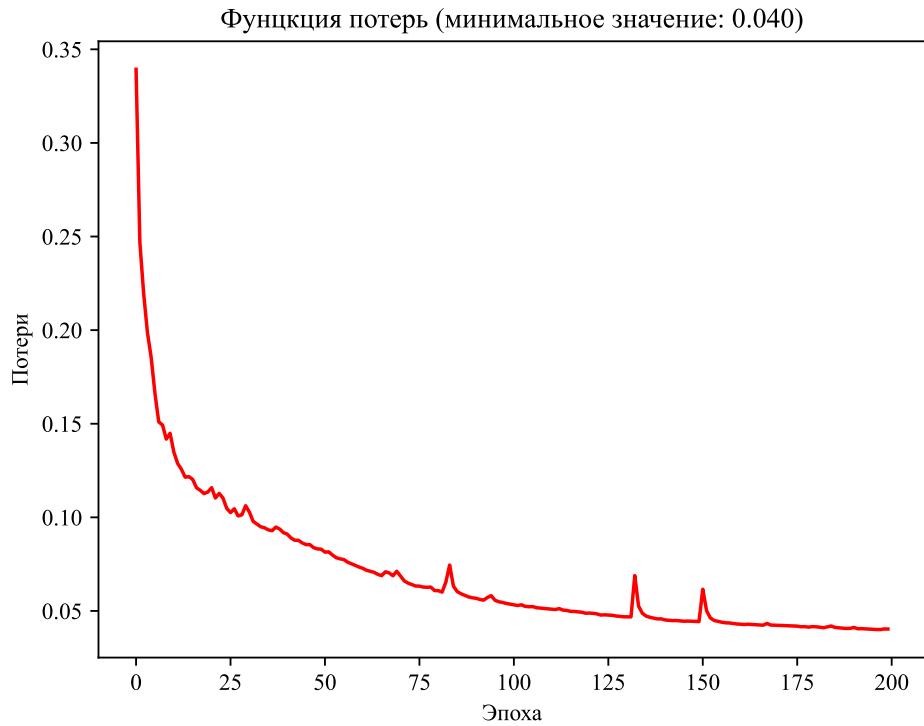


Рис. 1.11: Потери модели

1.4 Примеры использования разработанного программного комплекса

Оба модуля разработанного программного комплекса представляют собой консольные приложения без графического интерфейса, конфигурируемые при помощи аргументов командной строки.

1.4.1 Пример использования модуля модели UNet

В листинге 1 представлены параметры с которыми можно взаимодействовать с модулем обучения модели.

Листинг 1: Взаимодействие с модулем модели

```

1 python main.py -h
2
3 usage: main.py [-h] [-m] [-o OUTPUT] [-d DATA]
4
5 optional arguments:
6 -h, --help            show this help message and exit
7 -m                   Произвести тренировку челюстной сегментации
8 -o OUTPUT, --output OUTPUT Путь сохранения обученной модели
9 -d DATA, --data DATA Путь сохранения истории обучения

```

В листинге 2 представлены пример запуска модуля для обучения зубной сегментации.

Листинг 2: Запуск обучения

```
1 python main.py -o trained/teeth.h5 -d trained/teeth.hist
2
3 Extracting data/dataset.zip...
4 Extracted to data
5 Training teeth segmentation
6 Extracting masks/teeth.zip...
7 Extracted to masks/teeth
8 2022-05-24 01:51:26.391184: I tensorflow/core/platform/cpu_feature_guard.cc
   :151] This TensorFlow binary is optimized with oneAPI Deep Neural Network
   Library (oneDNN) to use the following CPU instructions in performance-
   critical operations: SSE4.2 AVX AVX2 FMA
9 To enable them in other operations, rebuild TensorFlow with the appropriate
   compiler flags.
10 Epoch 1/200
11 1/73 [...........................] - ETA: 33:51 - loss: 0.8040 - accuracy:
   0.2244
12 ...
```

1.4.2 Пример использования модуля пользовательского приложения

В листинге 3 представлены параметры с которыми можно взаимодействовать с модулем обучения модели.

Листинг 3: Взаимодействие с пользовательским приложением

```
1 python main.py -h
2
3 usage: main.py [-h] [-m MODEL] [-i IMAGE] [-o OUTPUT] [-c]
4
5 optional arguments:
6 -h, --help            show this help message and exit
7 -m MODEL, --model MODEL Путь к файлу с обученной моделью
8 -i IMAGE, --image IMAGE Путь к файлу снимка
9 -o OUTPUT, --output OUTPUT Директория для сохранения результатов
10 -c                  Произвести сегментацию по экземплярам
```

В листинге 4 представлены пример запуска приложения с включенной классификацией по типу зубов.

Листинг 4: Запуск приложения

```
1 python main.py -m ../train/trained/teeth.h5 -i ../train/data/Images/19.png -c
2
3 2022-05-24 01:57:50.542260: I tensorflow/core/platform/cpu_feature_guard.cc
   :151] This TensorFlow binary is optimized with oneAPI Deep Neural Network
   Library (oneDNN) to use the following CPU instructions in performance-
   critical operations: SSE4.2 AVX AVX2 FMA
```

- 4 To enable them **in** other operations, rebuild TensorFlow with the appropriate compiler flags.
- 5 Segmented teeth count **is** 31

Выход

Были описаны средства реализации программного комплекса. Приведены листинги реализации каждого компонента комплекса, примеры работы компонентов, их входные и выходные данные. Описаны технологии и методы, использовавшиеся при реализации. Представлены примеры взаимодействия с модулями.

2. Исследования разработанного программного комплекса на применимость для снимков в разных проекциях и быстродействия

В данном разделе проводится оценка качества разработанного программного комплекса. Описывается его применимость в различных ситуациях. Дается анализ достоинств и недостатков.

2.1 Сравнение применимости программного комплекса для снимков в разных проекциях

При обучении модели использовались снимки челюстно–лицевого става в анфас. Несмотря на это, обученную модель можно применять и на снимках в профиль.

На рисунках 2.1 и 2.2 приведены снимки одного черепа в разных проекциях. На данных снимках будет проведен эксперимент.



Рис. 2.1: Снимок анфас



Рис. 2.2: Снимок в профиль

На рисунках 2.3 и 2.4 соответственно приведены полученная маска и результат сегментации для снимка анфас. Как видно из снимков, никаких артефактов при проведении сегментации замечено не было.

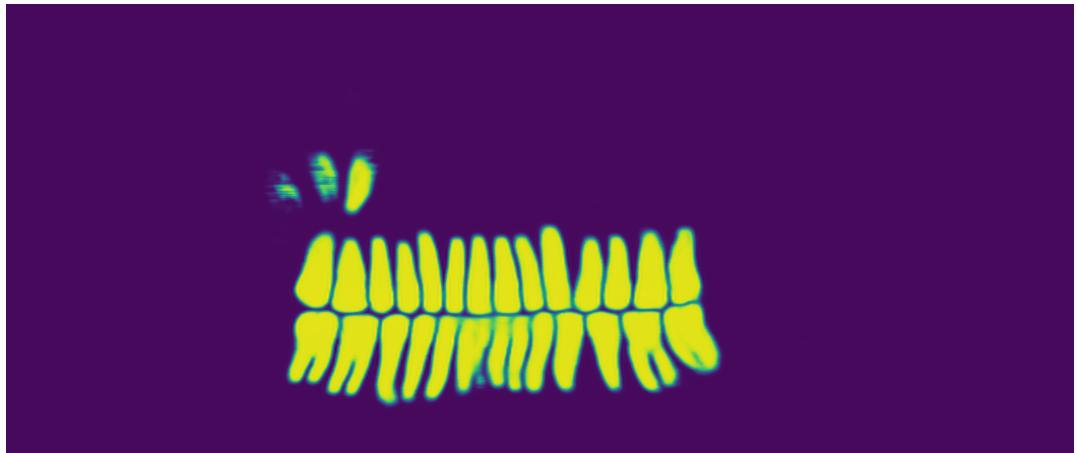


Рис. 2.3: Мaska для снимка анфас



Рис. 2.4: Сегментация для снимка анфас

На рисунках 2.5 и 2.6 соответственно приведены полученная маска и результат сегментации для снимка в профиль. Так же как и со случаем анфас, в данном случае никаких артефактов замечено не было, маска для зубного состава была составлена корректно, за исключением небольшой области в левой верхней части снимка. Эта область является своеобразной водяной знаком снимка, такое отклонение допустимо и не затрагивает зубной состав, то есть не препятствует дальнейшему анализу полученных результатов.

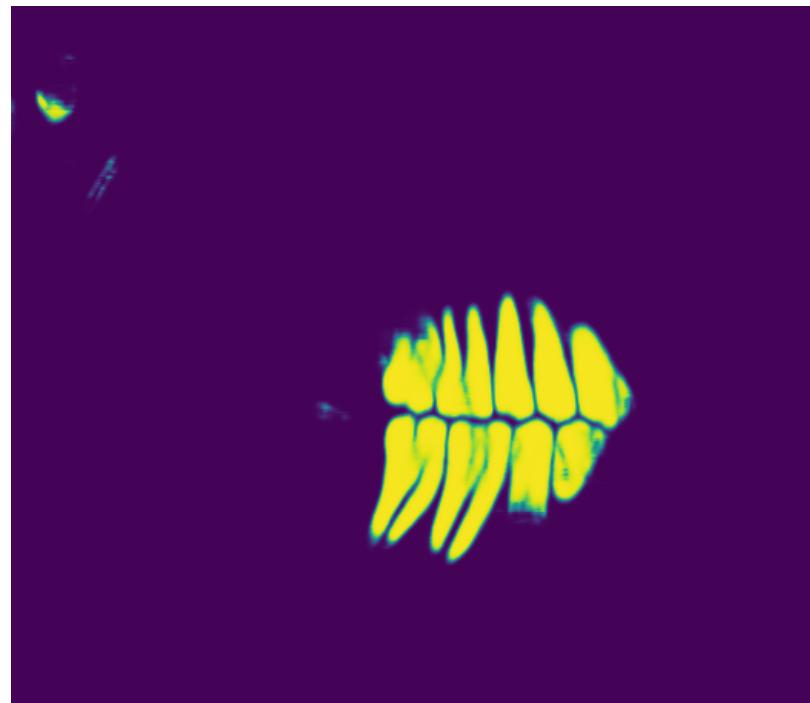


Рис. 2.5: Мaska для снимка в профиль

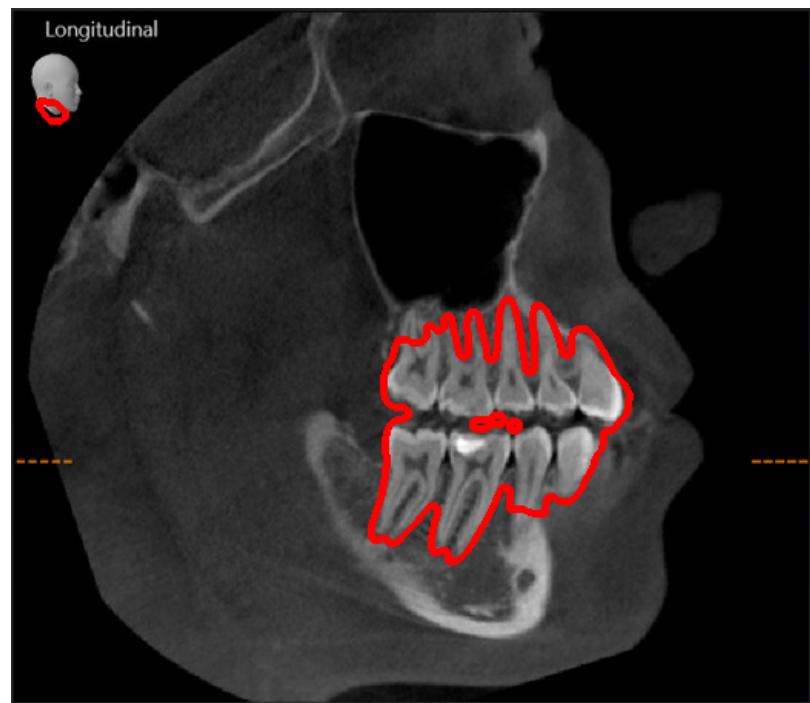


Рис. 2.6: Сегментация для снимка в профиль

2.2 Сравнение применимости программного комплекса для снимков в разных оттенках

В процессе обучения использовались снимки в черно–белой гамме. Помимо таких снимков, существуют аппараты, которые при снятии снимка сохраняют его в других цветовых тонах, например голубой вместо белого.

На рисунке 2.7 представлен пример такого снимка.

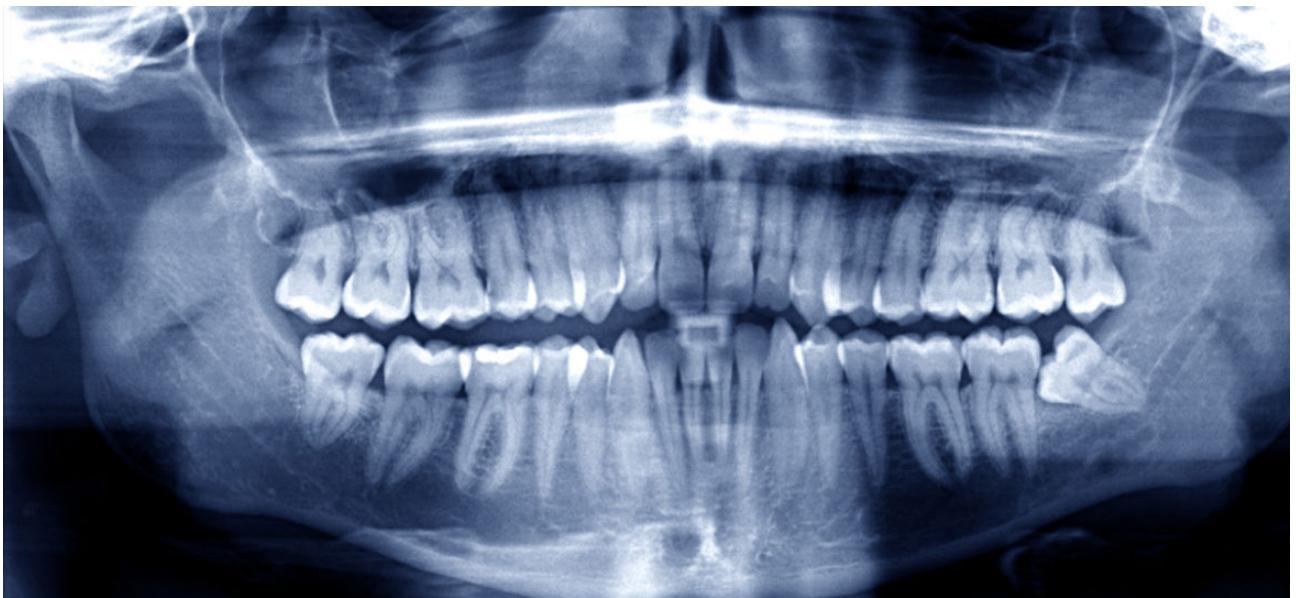


Рис. 2.7: Снимок в другой цветовой гамме

На рисунках 2.8 и 2.9 соответственно приведены полученная маска и результат сегментации для снимка в другой цветовой гамме. В целом сегментация проведена успешно, артефактов в щебном составе не наблюдается.

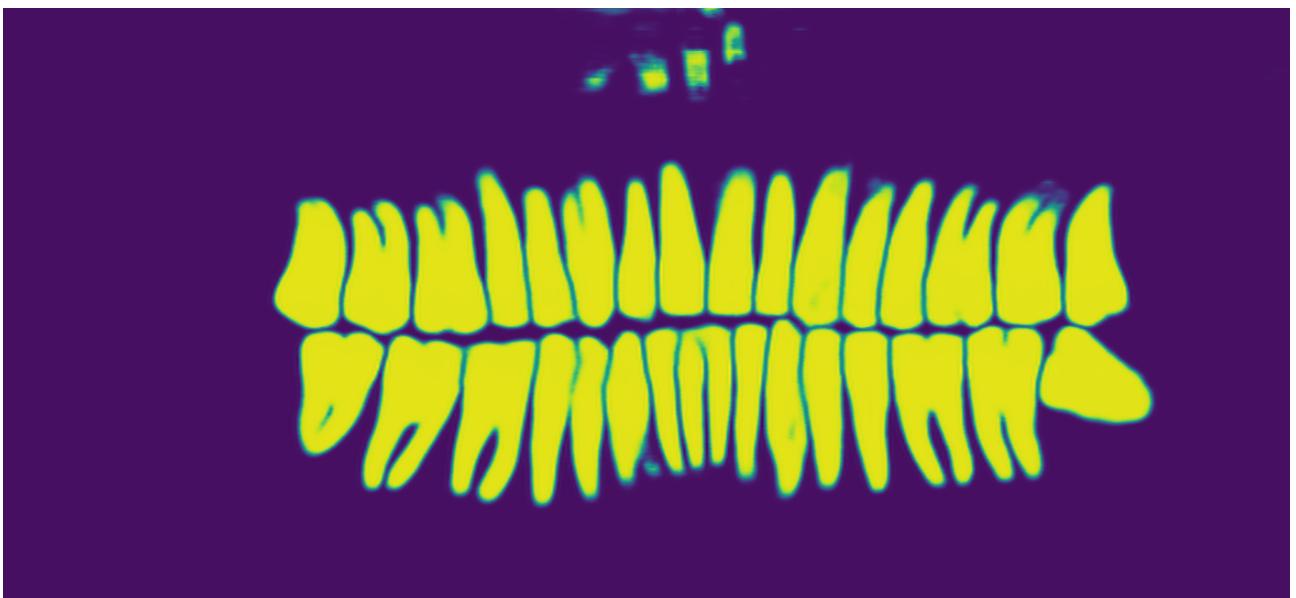


Рис. 2.8: Мaska для снимка в другой цветовой гамме

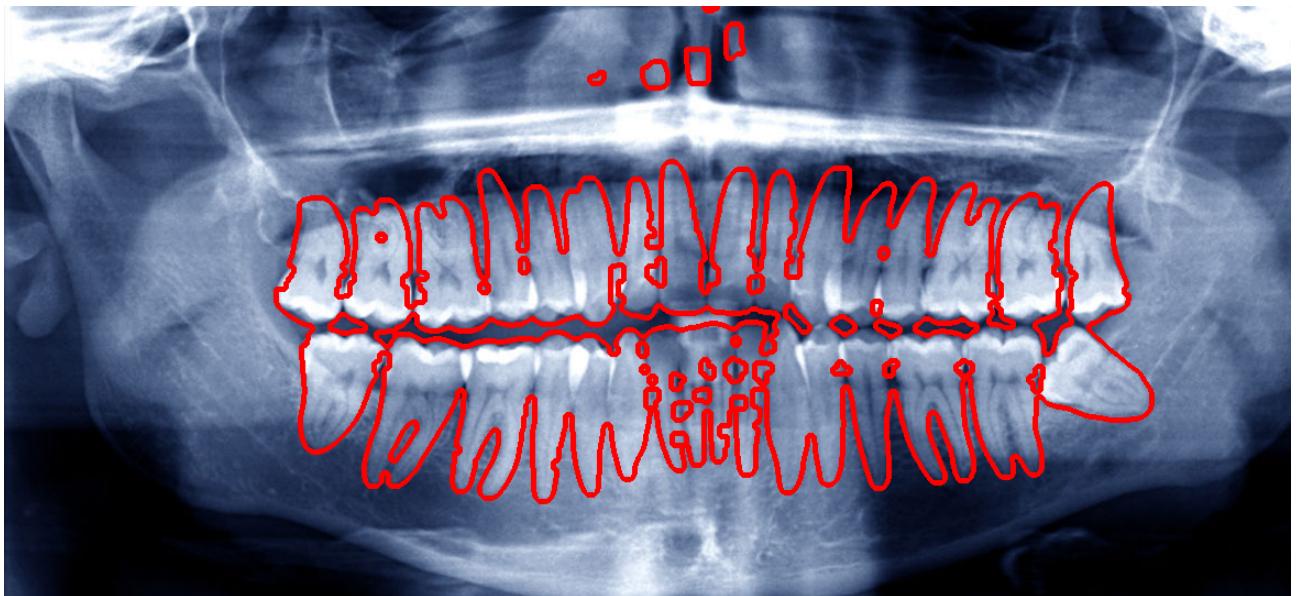


Рис. 2.9: Сегментация для снимка в другой цветовой гамме

2.3 Сравнение времени работы приложения при различном разрешении исходного снимка

Для проведения эксперимента будем использовать снимок, представленный на рисунке 2.10. Исходный снимок имеет разрешение 3050×1250 пикселей. Эксперимент будет проводиться на изображениях размером 100%, 75%, 50% и 25% от первоначального.

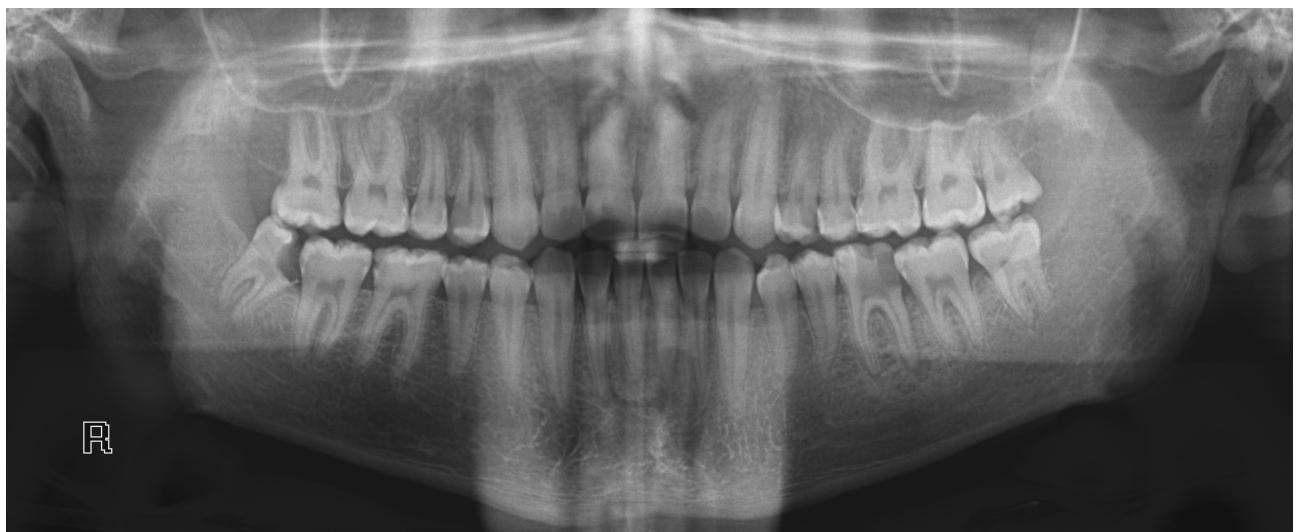


Рис. 2.10: Снимок для проведения эксперимента

В таблице 1 представлена зависимость времени работы приложения от размера исходного изображения. Также эта зависимость представлена на рисунке 2.11

Таблица 1: Время работы приложения в зависимости от размера исходного изображения

Режим работы приложения	100%, сек	75%, сек	50%, сек	25%, сек
Без классификации	14.4	12.68	11.48	10.46
С классификацией	17.5	13.18	11.48	10.46

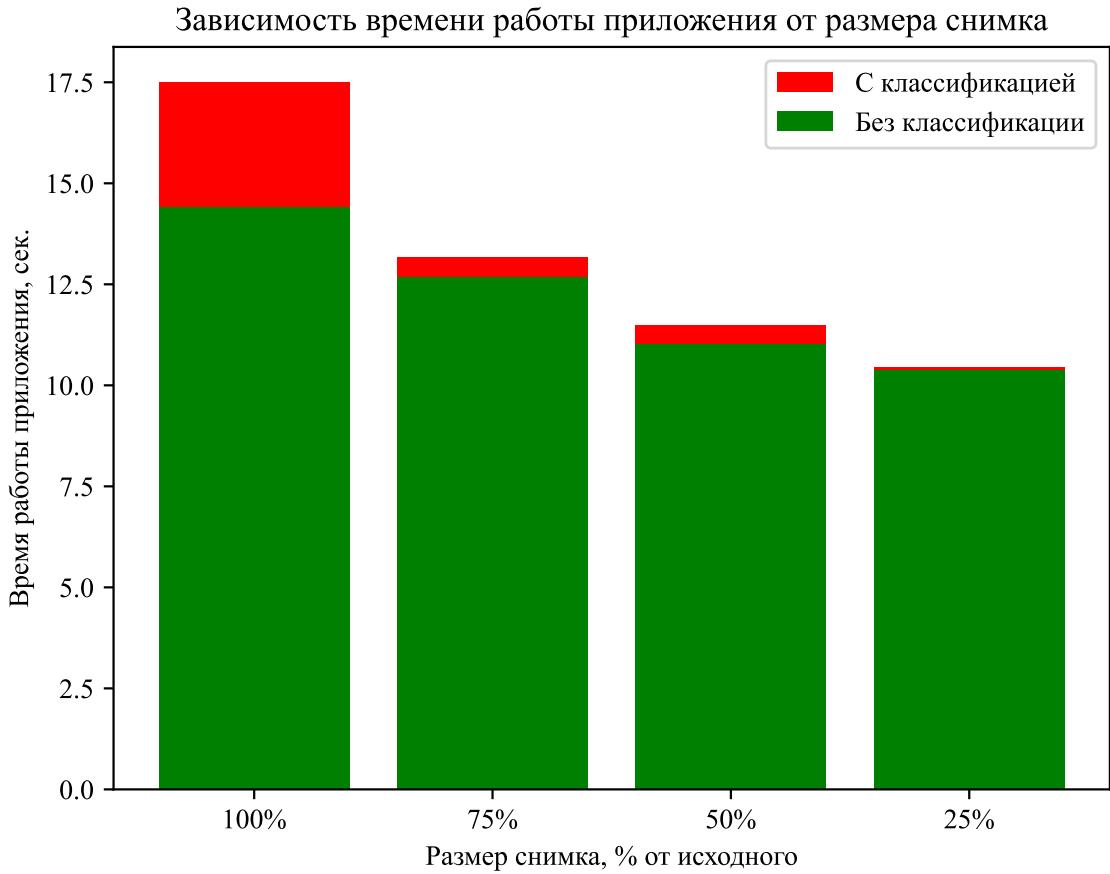


Рис. 2.11: Зависимость времени работы приложения от размера исходного изображения

При уменьшении размера изображения потеря в качестве не наблюдалась, маска для сегментации всегда выделялась одинаково четко, без артефактов.

Уменьшение изображения повлияло на корректность работы классификации при помощи машины опорных векторов. Так как вектор особенностей состоял из параметров ширины и высоты зуба в пикселях, для изображений другого размера эти данные не являются корректными. Это стоит учитывать при работе с приложением.

Кроме того, классификация с уменьшением изображения занимает меньше времени. Это связано с более низкой нагрузкой на машину опорных векторов, так как приходится обрабатывать векторы, содержащие в себе меньшие числовые значения параметров модели.

2.4 Оценка разработанного программного комплекса

У разработанного программного комплекса можно выявить следующие достоинства и недостатки.

Достоинства:

- Универсальность. Возможно использование при анализе снимков в разных проекциях, разных размеров и разных цветовых гаммах.
- Высокая точность. Отсутствие потери точности при сегментации при разных размерах исходного снимка.

Недостатки:

- Классификация. При уменьшении разрешения снимка наблюдаются ошибки классификации ввиду неполноты данных для обучения.

Вывод

Было проведено сравнение применимости разработанного программного комплекса при различных нетипичных сценариях. Протестировано быстродействие программы при использовании классификации и без ее использования. Представлены достоинства и недостатки программного комплекса.

ЗАКЛЮЧЕНИЕ

Был разработан программный комплекс, состоящий из двух модулей: модуль модели UNet и модуль пользовательского приложения. Приведены результаты обучения модели и показаны примеры использования разработанного программного комплекса.

Проведено исследование качества созданного программного комплекса. В результате исследования были выделены достоинства, среди которых универсальность и высокая точность, и недостатки, среди которых ошибки классификации при изменении размера изображения.

Таким образом, цель работы — разработать метод распознавания челюстно-лицевых костей черепа по томографическим снимкам головы человека, была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Список литературы

1. Epidemiological analysis of facial fractures / Deepalakshmi Tanthry, Aisha Nehla, Mahesh Santhraya [и др.] // International Journal of Otorhinolaryngology and Head and Neck Surgery. 2021. 06. Т. 7. с. 1176.
2. ISAPS. Survey: Aesthetic/Cosmetic procedures performed in 2018. 2018.
3. Facial fractures: classification and highlights for a useful report / Eva Roselló, Ana Granado, Miquel Garcia [и др.] // Insights into Imaging. 2020. 12. Т. 11.
4. Deep neural network improves fracture detection by clinicians / Robert Lindsey, Aaron Daluiski, Sumit Chopra [и др.] // Proceedings of the National Academy of Sciences. 2018. 10. Т. 115. с. 201806905.
5. Welcome to Python.org [Электронный ресурс]. Режим доступа: <https://www.python.org/> (дата обращения 18.05.2022).
6. What is CUDA | NVIDIA official blog [Электронный ресурс]. Режим доступа: <https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/> (дата обращения 18.05.2022).
7. Tensorflow [Электронный ресурс]. Режим доступа: <https://tensorflow.org/> (дата обращения 18.05.2022).
8. Performance Analysis of Deep Learning Libraries: TensorFlow and PyTorch / Felipe Florencio, Thiago Silva, Edward Ordóñez [и др.] // Journal of Computer Science. 2019. 05. Т. 15.
9. 1.4. Support Vector Machines | scikit-learn 1.1.0 documentation [Электронный ресурс]. Режим доступа: <https://scikit-learn.org/stable/modules/svm.html/> (дата обращения 18.05.2022).
10. Bailey Donald, Johnston Christopher, Ma Ni. Connected components analysis of streamed images. 2008. 10. С. 679 – 682.

ПРИЛОЖЕНИЕ 1. Модуль модели UNet

Листинг 5: Модель UNet

```
1 from tensorflow.keras.layers import (BatchNormalization, Conv2D,
2                                         Conv2DTranspose, Dropout, Input,
3                                         MaxPooling2D, concatenate)
4 from tensorflow.keras.models import Model
5
6
7 def UNet(input_shape=(512, 512, 1), last_activation='sigmoid'):
8     inputs = Input(shape=input_shape)
9
10    conv1 = Conv2D(32, (3, 3), activation='relu', padding='same',
11                  kernel_initializer='he_normal')(inputs)
12    d1 = Dropout(0.1)(conv1)
13    conv2 = Conv2D(32, (3, 3), activation='relu', padding='same',
14                  kernel_initializer='he_normal')(d1)
15    b = BatchNormalization()(conv2)
16
17    pool1 = MaxPooling2D(pool_size=(2, 2))(b)
18    conv3 = Conv2D(64, (3, 3), activation='relu', padding='same',
19                  kernel_initializer='he_normal')(pool1)
20    d2 = Dropout(0.2)(conv3)
21    conv4 = Conv2D(64, (3, 3), activation='relu', padding='same',
22                  kernel_initializer='he_normal')(d2)
23    b1 = BatchNormalization()(conv4)
24
25    pool2 = MaxPooling2D(pool_size=(2, 2))(b1)
26    conv5 = Conv2D(128, (3, 3), activation='relu', padding='same',
27                  kernel_initializer='he_normal')(pool2)
28    d3 = Dropout(0.3)(conv5)
29    conv6 = Conv2D(128, (3, 3), activation='relu', padding='same',
30                  kernel_initializer='he_normal')(d3)
31    b2 = BatchNormalization()(conv6)
32
33    pool3 = MaxPooling2D(pool_size=(2, 2))(b2)
34    conv7 = Conv2D(256, (3, 3), activation='relu', padding='same',
35                  kernel_initializer='he_normal')(pool3)
36    d4 = Dropout(0.4)(conv7)
37    conv8 = Conv2D(256, (3, 3), activation='relu', padding='same',
38                  kernel_initializer='he_normal')(d4)
39    b3 = BatchNormalization()(conv8)
40
41    pool4 = MaxPooling2D(pool_size=(2, 2))(b3)
42    conv9 = Conv2D(512, (3, 3), activation='relu', padding='same',
43                  kernel_initializer='he_normal')(pool4)
44    d5 = Dropout(0.5)(conv9)
```

```

45     conv10 = Conv2D(512, (3, 3), activation='relu', padding='same',
46                     kernel_initializer='he_normal')(d5)
47     b4 = BatchNormalization()(conv10)
48
49     conv11 = Conv2DTranspose(512, (4, 4), activation='relu', padding='same',
50                             strides=(2, 2), kernel_initializer='he_normal')(
51         b4)
52     x = concatenate([conv11, conv8])
53     conv12 = Conv2D(256, (3, 3), activation='relu', padding='same',
54                     kernel_initializer='he_normal')(x)
55     d6 = Dropout(0.4)(conv12)
56     conv13 = Conv2D(256, (3, 3), activation='relu', padding='same',
57                     kernel_initializer='he_normal')(d6)
58     b5 = BatchNormalization()(conv13)
59
60     conv14 = Conv2DTranspose(256, (4, 4), activation='relu', padding='same',
61                             strides=(2, 2), kernel_initializer='he_normal')(
62         b5)
63     x1 = concatenate([conv14, conv6])
64     conv15 = Conv2D(128, 3, activation='relu', padding='same',
65                     kernel_initializer='he_normal')(x1)
66     d7 = Dropout(0.3)(conv15)
67     conv16 = Conv2D(128, 3, activation='relu', padding='same',
68                     kernel_initializer='he_normal')(d7)
69     b6 = BatchNormalization()(conv16)
70
71     conv17 = Conv2DTranspose(128, (4, 4), activation='relu', padding='same',
72                             strides=(2, 2), kernel_initializer='he_normal')(
73         b6)
74     x2 = concatenate([conv17, conv4])
75     conv18 = Conv2D(64, (3, 3), activation='relu', padding='same',
76                     kernel_initializer='he_normal')(x2)
77     d8 = Dropout(0.2)(conv18)
78     conv19 = Conv2D(64, (3, 3), activation='relu', padding='same',
79                     kernel_initializer='he_normal')(d8)
80     b7 = BatchNormalization()(conv19)
81
82     conv20 = Conv2DTranspose(64, (4, 4), activation='relu', padding='same',
83                             strides=(2, 2), kernel_initializer='he_normal')(
84         b7)
85     x3 = concatenate([conv20, conv2])
86     conv21 = Conv2D(32, (3, 3), activation='relu', padding='same',
87                     kernel_initializer='he_normal')(x3)
88     d9 = Dropout(0.1)(conv21)
89     conv22 = Conv2D(32, (3, 3), activation='relu', padding='same',
90                     kernel_initializer='he_normal')(d9)
91
92     outputs = Conv2D(1, (1, 1), activation=last_activation, padding='same',

```

```

87                 kernel_initializer='he_normal')(conv22)
88 model = Model(inputs=inputs, outputs=outputs)
89
90 return model

```

Листинг 6: Обучение модели

```

1 import argparse
2 import os
3 import pickle
4
5 import albumentations as A
6 import cv2
7 import numpy as np
8 from dataset import download_dataset, prepare_dataset, prepare_masks
9 from imageprep import pre_images
10 from maskprep import pre_masks
11 from model import UNet
12 from tensorflow.keras.models import save_model
13
14
15 def main(model_path, history_path, trainTeeth):
16     if not os.path.exists('data/dataset.zip'):
17         download_dataset('data')
18
19     prepare_dataset('data/dataset.zip', 'data')
20
21     if trainTeeth:
22         print('Training teeth segmentation')
23         prepare_masks('masks/teeth.zip', 'masks/teeth')
24         y_train = pre_masks('masks/teeth')
25     else:
26         print('Training mandibles segmentation')
27         prepare_masks('masks/mandibles.zip', 'masks/mandibles')
28         y_train = pre_masks('masks/mandibles')
29
30     x_train = pre_images((512, 512), 'data/Images')
31
32     x_train = np.float32(x_train/255)
33     y_train = np.float32(y_train/255)
34
35     aug = A.Compose([
36         A.OneOf([A.RandomCrop(width=512, height=512),
37                  A.PadIfNeeded(min_height=512, min_width=512, p=0.5)], p=0.4),
38         A.RandomBrightnessContrast(brightness_limit=0.25, contrast_limit
39             =0.25, p=0.5),
40         A.Compose([A.RandomScale(scale_limit=(-0.15, 0.15), p=1,
41             interpolation=1),

```

```

40             A.PadIfNeeded(512, 512, border_mode=cv2.BORDER_CONSTANT),
41             A.Resize(512, 512, cv2.INTER_NEAREST), ], p=0.5),
42             A.ShiftScaleRotate(shift_limit=0.325, scale_limit=0.15, rotate_limit
43             =15,
44             border_mode=cv2.BORDER_CONSTANT, p=1),
45             A.Rotate(15, p=0.5),
46             A.Blur(blur_limit=1, p=0.5),
47             A.Downscale(scale_min=0.15, scale_max=0.25, always_apply=False, p
48             =0.5),
49             A.GaussNoise(var_limit=(0.05, 0.1), mean=0, per_channel=True,
50             always_apply=False, p=0.5),
51             A.HorizontalFlip(p=0.25),
52         ])
53
54     x_train1 = np.copy(x_train)
55     y_train1 = np.copy(y_train)
56
57     count = 0
58
59     while(count < 4):
60         x_aug = np.copy(x_train1)
61         y_aug = np.copy(y_train1)
62
63         for i in range(len(x_train1)):
64             augmented = aug(image=x_train1[i, :, :, :], mask=y_train1[i, :, :
65             :, :])
66             x_aug[i, :, :, :] = augmented['image']
67             y_aug[i, :, :, :] = augmented['mask']
68             x_train = np.concatenate((x_train, x_aug))
69             y_train = np.concatenate((y_train, y_aug))
70
71         if count == 9:
72             break
73
74         count += 1
75
76     mem_to_free = [x_aug, x_train, y_train, y_aug, y_train1, x_train1,
77     augmented]
78
79     for mem in mem_to_free:
80         del mem
81
82
83     model = UNet(input_shape=(512, 512, 1), last_activation='sigmoid')
84     model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['
85     accuracy'])
86     data = model.fit(x_train, y_train, batch_size=8, epochs=200, verbose=1)
87
88     data_file = open(history_path, 'wb')
89     pickle.dump(data.history, data_file)
90     data_file.close()

```

```

82
83     save_model(model, model_path)
84
85
86 if __name__ == '__main__':
87     parser = argparse.ArgumentParser()
88     teeth = True
89     parser.add_argument('-m', action='store_const', default=teeth, const=False,
90                         teeth,
91                         help='Произвести тренировку челюстной сегментации')
92     parser.add_argument('-o', '--output', default='trained/bonerecon.h5',
93                         help='Путь сохранения обученной модели')
94     parser.add_argument('-d', '--data', default='trained/bonerecon.hist',
95                         help='Путь сохранения истории обучения')
96
97     args = parser.parse_args()
98
99
100    main(args.output, args.data, args.m)

```

Листинг 7: Загрузка данных для обучения

```

1 import os
2 from io import BytesIO
3 from zipfile import ZipFile
4
5 import requests
6
7 DATASET_LINK = 'https://md-datasets-cache-zipfiles-prod.s3.eu-west-1.
8     amazonaws.com/hxt48yk462-1.zip'
9
10 def download_dataset(path):
11     print(f'Downloading dataset to {path}...')
12     r = requests.get(DATASET_LINK)
13     z = ZipFile(BytesIO(r.content))
14     z.extractall(path)
15     os.rename(f'{path}/{z.namelist()[0]}', f'{path}/dataset.zip')
16
17     print(f'Downloaded to {path}/dataset.zip')
18
19
20 def prepare_dataset(path='data/dataset.zip', extract_to='data'):
21     print(f'Extracting {path}...')
22
23     if not os.path.exists(extract_to):
24         os.mkdir(extract_to)
25
26     ZipFile(path).extractall(extract_to)

```

```

27     print(f'Extracted to {extract_to}')
28
29
30 def prepare_masks(path='masks/masks.zip', extract_to='data'):
31     print(f'Extracting {path}...')
32
33     if not os.path.exists(extract_to):
34         os.mkdir(extract_to)
35
36     ZipFile(path).extractall(extract_to)
37     print(f'Extracted to {extract_to}')

```

Листинг 8: Предобработка снимков

```

1 import os
2
3 import numpy as np
4 from image import convert_one_channel
5 from natsort import natsorted
6 from PIL import Image
7
8
9 def pre_images(resize_shape, path):
10     dirs = natsorted(os.listdir(path))
11     img = Image.open(f'{path}/{dirs[0]}')
12     images = convert_one_channel(np.asarray(img.resize((resize_shape),
13             Image.ANTIALIAS)))
14
15     for i in range(1, len(dirs)):
16         img = Image.open(f'{path}/{dirs[i]}')
17         img = img.resize((resize_shape), Image.ANTIALIAS)
18         img = convert_one_channel(np.asarray(img))
19         images = np.concatenate((images, img))
20
21     images = np.reshape(images, (len(dirs), resize_shape[0], resize_shape[1],
22         1))
23
24     return images

```

Листинг 9: Предобработка масок

```

1 import os
2
3 import numpy as np
4 from image import convert_one_channel
5 from natsort import natsorted
6 from PIL import Image
7

```

```
8
9 def pre_masks(path='masks'):
10     dirs = natsorted(os.listdir(path))
11     masks = img = Image.open(f'{path}/{dirs[0]}')
12     masks = convert_one_channel(np.asarray(masks))
13
14     for i in range(1, len(dirs)):
15         img = Image.open(f'{path}/{dirs[i]}')
16         img = convert_one_channel(np.asarray(img))
17         masks = np.concatenate((masks, img))
18
19     masks = np.reshape(masks, (len(dirs), 512, 512, 1))
20
21     return masks
```

ПРИЛОЖЕНИЕ 2. Модуль пользовательского приложения

Листинг 10: Пользовательское приложение

```
1 import argparse
2 import os
3
4 import cv2
5 import matplotlib.pyplot as plt
6 import numpy as np
7 from cca import analyze
8 from image import convert_one_channel, convert_rgb
9 from PIL import Image
10 from tensorflow.keras.models import load_model
11
12
13 def main(model, image, output_dir, use_svm):
14     model = load_model(model)
15     img = Image.open(image)
16     img = np.asarray(img)
17
18     img_cv = convert_one_channel(img)
19     img_cv = cv2.resize(img_cv, (512, 512), interpolation=cv2.INTER_LANCZOS4)
20     img_cv = np.float32(img_cv/255)
21     img_cv = np.reshape(img_cv, (1, 512, 512, 1))
22
23     predicted = model.predict(img_cv)[0]
24     predicted = cv2.resize(
25         predicted, (img.shape[1], img.shape[0]), interpolation=cv2.
26         INTER_LANCZOS4)
27
28     if not os.path.exists(output_dir):
29         os.mkdir(output_dir)
30
31     plt.imsave(f'{output_dir}/predicted.png', predicted)
32
33     thresh = np.uint8(predicted*255)
34     thresh = cv2.threshold(thresh, thresh=0, maxval=255, type=cv2.
35     THRESH_BINARY+cv2.THRESH_OTSU)[1]
36     kernel = (np.ones((5, 5), dtype=np.float32))
37     thresh = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=1)
38     thresh = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel, iterations=1)
39
40     cnts = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
41     [0]
```

```

39     output = cv2.drawContours(convert_rgb(img), cnts, -1, (255, 0, 0), 3)
40     plt.imsave(f'{output_dir}/segmented.png', output)
41
42     img = cv2.imread(image)
43     predicted = cv2.imread(f'{output_dir}/predicted.png')
44     predicted = cv2.resize(predicted, (img.shape[1], img.shape[0]),
45                           interpolation=cv2.INTER_LANCZOS4)
46
47     cca, teeth_count = analyze(img, predicted, 3, 2)
48     plt.imsave(f'{output_dir}/segmented_cca.png', cca)
49     if use_svm:
50         svm, _ = analyze(img, predicted, 3, 2, use_svm)
51         plt.imsave(f'{output_dir}/segmented_svm.png', svm)
52         print(f'Segmented teeth count is {teeth_count}')
53
54 if __name__ == '__main__':
55     parser = argparse.ArgumentParser()
56     parser.add_argument('-m', '--model', default='trained/bonerecon.h5', help=
57                         ='Путь к файлу с обученной моделью')
58     parser.add_argument('-i', '--image', default='data/file.jpeg', help='
59                         Пусть к файлу снимка')
60     parser.add_argument('-o', '--output', default='predicted', help='
61                         Директория для сохранения результатов')
62     use_svm = False
63     parser.add_argument('-c', action='store_const', default=use_svm, const=
64                         not use_svm,
65                         help='Произвести сегментацию по экземплярам')
66
67     args = parser.parse_args()
68
69     main(args.model, args.image, args.output, args.c)

```

Листинг 11: Анализ связанных компонентов и машина опорных векторов

```

1 import cv2
2 import numpy as np
3 from imutils import perspective
4 from scipy.spatial import distance as dist
5 from sklearn import svm
6
7 MOLARES_COLOR = [255, 0, 0]
8 PREMOLARES_COLOR = [0, 255, 0]
9 CANINOS_COLOR = [255, 255, 0]
10 INCISIVOS_COLOR = [0, 0, 255]
11
12 COLORS = {
13     0: MOLARES_COLOR,

```

```

14     1: PREMOLARES_COLOR,
15     2: CANINOS_COLOR,
16     3: INCISIVOS_COLOR
17 }
18
19
20 def midpoint(ptA, ptB):
21     return ((ptA[0] + ptB[0]) * 0.5, (ptA[1] + ptB[1]) * 0.5)
22
23
24 def classifier():
25     X = [
26         # Molares
27         [286.9, 153.1],
28         [289.6, 142.5],
29         [236.8, 134.7],
30         [272.4, 153.2],
31         [287.9, 150.1],
32         [290.9, 133],
33         [274.1, 136.5],
34         [195.5, 129.6],
35         [310.6, 139.2],
36         [263.4, 149],
37         [230.7, 159.8],
38
39         [338.3, 144.5],
40         [309.5, 137.3],
41         [277.6, 149.5],
42         [365.9, 153.1],
43         [375.7, 147.6],
44         [351.4, 113.6],
45         [256, 133],
46         [283.3, 182.2],
47         [368.4, 153.7],
48         # Premolares
49         [281.4, 70.6],
50         [284.5, 79.7],
51         [285.3, 89.1],
52         [289.5, 83.4],
53         [273.5, 81.2],
54         [289.6, 77.4],
55         [290.3, 75.8],
56         [282, 75],
57         # Caninos
58         [406, 99],
59         [384.3, 96],
60         [377.1, 83],
61         [363.5, 82.1],

```

```

62      [433.6, 111],
63      [350.7, 88.4],
64      [388.8, 101.1],
65      [363.9, 91.2],
66      # Incisivos
67      [327.5, 87.6],
68      [360, 104],
69      [327.2, 103],
70      [331.6, 86.5],
71      [314, 60],
72      [288, 60],
73      [280, 64],
74      [314.3, 62.1],
75  ]
76
77  Y = [
78      # Molares
79      0,
80      0,
81      0,
82      0,
83      0,
84      0,
85      0,
86      0,
87      0,
88      0,
89      0,
90
91      0,
92      0,
93      0,
94      0,
95      0,
96      0,
97      0,
98      0,
99      0,
100     # Premolares
101     1,
102     1,
103     1,
104     1,
105     1,
106     1,
107     1,
108     1,
109     # Caninos

```



```

155     unique_labels = np.unique(labels)
156     for label in unique_labels:
157         if label == 0:
158             continue
159
160         mask = np.zeros(thresh.shape, dtype="uint8")
161         mask[labels == label] = 255
162
163         cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.
164         CHAIN_APPROX_SIMPLE)[0][0]
165         c_area = cv2.contourArea(cnts)
166
167         if c_area > 2000:
168             count += 1
169
170         rect = cv2.minAreaRect(cnts)
171         box = cv2.boxPoints(rect)
172         box = np.array(box, dtype="int")
173         box = perspective.order_points(box)
174
175         if use_svm:
176             tooth = rect[1]
177             if tooth[1] > tooth[0]:
178                 tooth = [tooth[1], tooth[0]]
179             color = COLORS[clf.predict([tooth])[0]]
180         else:
181             intcolor = (list(np.random.choice(range(150), size=3)))
182             color = [int(intcolor[0]), int(intcolor[1]), int(intcolor[2])]
183
184             cv2.drawContours(oimage, [box.astype("int")], 0, color, 2)
185             (tl, tr, br, bl) = box
186
187             (tltrX, tltrY) = midpoint(tl, tr)
188             (blbrX, blbrY) = midpoint(bl, br)
189
190             (tblbLX, tblbLY) = midpoint(tl, bl)
191             (trbrX, trbrY) = midpoint(tr, br)
192
193             cv2.circle(oimage, (int(tltrX), int(tltrY)), 5, (255, 0, 0), -1)
194             cv2.circle(oimage, (int(blbrX), int(blbrY)), 5, (255, 0, 0), -1)
195             cv2.circle(oimage, (int(tblbLX), int(tblbLY)), 5, (255, 0, 0), -1)
196             cv2.circle(oimage, (int(trbrX), int(trbrY)), 5, (255, 0, 0), -1)
197             cv2.line(oimage, (int(tltrX), int(tltrY)), (int(blbrX), int(blbrY)),
198             color, 2)
199             cv2.line(oimage, (int(tblbLX), int(tblbLY)), (int(trbrX), int(trbrY)),
200             color, 2)
201
202             dA = dist.euclidean((tltrX, tltrY), (blbrX, blbrY))

```

```
200     dB = dist.euclidean((tblX, tblY), (trbrX, trbrY))
201     pixelsPerMetric = 1
202
203     dimA = dA * pixelsPerMetric
204     dimB = dB * pixelsPerMetric
205
206     cv2.putText(oimage, "{:.1f}pixel".format(dimA), (int(tltx - 15),
207                                         int(tlty - 10)), cv2.FONT_HERSHEY_SIMPLEX, 0.65, color,
208)
209     cv2.putText(oimage, "{:.1f}pixel".format(dimB), (int(trbx + 10),
210                                         int(trby)), cv2.FONT_HERSHEY_SIMPLEX, 0.65, color, 2)
211     cv2.putText(oimage, "{:.1f}{}".format(label), (int(tltx - 35),
212                                         int(tlty - 5)), cv2.FONT_HERSHEY_SIMPLEX, 0.65, color,
213)
214
215     return oimage, count
```

ПРИЛОЖЕНИЕ 3. Утилитарный модуль

Листинг 12: Конвертирование каналов изображения

```
1 import cv2
2
3
4 def convert_one_channel(img):
5     if len(img.shape) > 2:
6         return cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
7
8     return img
9
10
11 def convert_rgb(img):
12     if len(img.shape) == 2:
13         return cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
14
15     return img
```

Листинг 13: Подготовка челюстных масок

```
1 import os
2
3 from natsort import natsorted
4 from PIL import Image
5
6
7 def main():
8     images = natsorted(os.listdir('data/Segmentation1'))
9
10    for img in images:
11        oimg = Image.open(f'data/Segmentation1/{img}')
12        grayed = oimg.convert('L')
13        bw = grayed.point(lambda x: 0 if x < 40 else 255, '1')
14        bw.save(f'masks/mandibles/{img}')
15
16    images = natsorted(os.listdir('masks/mandibles'))
17
18    for img in images:
19        oimg = Image.open(f'masks/mandibles/{img}')
20        oimg = oimg.resize((512, 512))
21        oimg.save(f'masks/mands/{img}')
22
23
24 if __name__ == '__main__':
25     main()
```

Листинг 14: Визуализация метрик

```
1 import pickle
2
3 import matplotlib.pyplot as plt
4 import numpy as np
5
6
7 def main():
8     with open('../train/trained/teeth.hist', 'rb') as f:
9         data = pickle.load(f)
10
11     plt.rcParams['font.family'] = 'Times New Roman'
12
13     plt.figure(0)
14     plt.plot(data['accuracy'], color='green')
15     plt.title(f"Точность модели максимальное ( значение: {max(data['accuracy'])} )")
16     plt.ylabel('Точность')
17     plt.xlabel('Эпоха')
18     plt.savefig('accuracy.svg', format='svg')
19
20
21     plt.figure(1)
22     plt.plot(data['loss'], color='red')
23     plt.title(f"Функция потерь минимальное ( значение: {min(data['loss'])} )")
24     plt.ylabel('Потери')
25     plt.xlabel('Эпоха')
26     plt.savefig('loss.svg', format='svg')
27
28     x = ['100%', '75%', '50%', '25%']
29     y1 = [17.5, 13.18, 11.48, 10.46]
30     y2 = [14.4, 12.68, 11.03, 10.37]
31
32     plt.figure(2)
33     plt.bar(x, y1, color='red')
34     plt.bar(x, y2, color='green')
35     plt.title('Зависимость времени работы приложения от размера снимка')
36     plt.xlabel('Размер снимка, % от исходного')
37     plt.ylabel('Время работы приложения, сек.')
38     plt.legend(['С классификацией', 'Без классификации'])
39     plt.savefig('run.svg', format='svg')
40
41 if __name__ == '__main__':
42     main()
```