

Реферат

Детков Никита Сергеевич, «Влияние label smoothing'а псевдо-размеченных данных на обучение свёрточных нейронных сетей», работа содержит: стр. 37, рис. 10, табл. 7, библ. назв. 19.

Ключевые слова: рак кожи, меланома, свёрточные нейронные сети, компьютерное зрение, машинное обучение, глубокое обучение, label smoothing, псевдо-разметка, pseudo-labeling, under-sampling.

Цель работы заключается в построении модели машинного обучения, определяющей меланому по изображению родинки или повреждения кожи, а также исследовании применимости методов label smoothing'а к псевдо-размеченным данным для улучшения работы данной модели.

В течение работы было поставлено множество экспериментов и проверен ряд гипотез, которые помогли прийти к тому, что даже в задаче такой сложности, при наличии достаточного количества негативных факторов, применяя вышеупомянутые методы, можно добиться лучших результатов.

Содержание

РЕФЕРАТ.....	1
СОДЕРЖАНИЕ.....	2
ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ.....	3
ВВЕДЕНИЕ.....	6
ОСНОВНАЯ ЧАСТЬ.....	8
1. ОПИСАНИЕ ДАТАСЕТА.....	8
1.1 ОБЗОР ДОСТУПНЫХ ФАЙЛОВ.....	8
1.2 ОБЗОР ИЗОБРАЖЕНИЙ.....	9
2. МЕТРИКА КАЧЕСТВА.....	10
3. ПОСТРОЕНИЕ МОДЕЛИ ОПРЕДЕЛЕНИЯ МЕЛНОМЫ.....	11
3.1 ОБЗОР ДАННЫХ И СТРАТЕГИЯ РЕШЕНИЯ.....	11
3.2 ОБРАБОТКА ДАННЫХ.....	16
3.3 РАЗБИЕНИЕ ДАННЫХ ДЛЯ ОБУЧЕНИЯ.....	17
3.4 АУГМЕНТАЦИЯ.....	18
3.5 ЭКСПЕРИМЕНТЫ ОБУЧЕНИЯ.....	19
3.6 АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ.....	21
4. ИССЛЕДОВАНИЕ ВЛИЯНИЯ LABEL SMOOTHING'А.....	25
4.1 ЧТО ТАКОЕ PSEUDO-LABELING.....	25
4.2 ЧТО ТАКОЕ LABEL SMOOTHING.....	25
4.3 ПРИМЕНИМОСТЬ К ЗАДАЧЕ.....	26
4.3.1 ПРОБЛЕМА ВЫРАВНИВАНИЯ РАСПРЕДЕЛЕНИЙ.....	26
4.3.2 ОПРЕДЕЛЕНИЕ «СМЯГЧАЮЩИХ» ФУНКЦИЙ.....	27
4.3.2 ОПРЕДЕЛЕНИЕ ДОЛИ НАБОРА ДАННЫХ ДЛЯ ОБУЧЕНИЯ.....	28
4.3.2 СТРАТЕГИЯ FINE-TUNING'А.....	28
4.4 ПРОВЕДЕНИЕ ЭКСПЕРИМЕНТОВ.....	29
4.5 АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ.....	29
5. ТЕХНОЛОГИЧЕСКИЙ СТЕК.....	31
ЗАКЛЮЧЕНИЕ.....	32
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	33
ПРИЛОЖЕНИЯ.....	35

Обозначения и сокращения

Искусственный персепtron (Artificial perceptron) – модель, описывающая восприятие, реакцию и обработку информации мозгом с точки зрения математики и компьютерных вычислений.

Свёрточная нейронная сеть (Convolutional Neural Network) – функция, принимающая на вход тензор и совершающая преобразования с помощью свёрток, функций активаций и других математических функций, также может определяться как многослойный персептрон.

Искусственный нейрон (Artificial neuron) – один из узлов нейронной сети, который представлен в виде математической функции по подобию нейрона – электрически возбудимой клетки нервной системы.

Функция потерь (Loss function) – функция, описывающая ошибку между действительным и предсказанным значениями.

Метод обратного распространения ошибки (Backpropagation algorithm) – итеративный метод вычисления градиента, который используется при обновлении весов нейронной сети с целью минимизации его функции потерь. Применим только в случае дифференцируемости функций активаций нейронов и других компонентов нейронной сети.

Обучение нейронной сети (Neural network training) – процесс обучения нейронной сети, который возможен благодаря методу обратного распространения ошибки.

Fine-tuning нейронной сети (Neural network fine-tuning) – процесс дообучения нейронной сети, при котором обычно обучаю не все слои, а лишь часть.

Инференс нейронной сети (Neural network inference) – процесс получения результата работы сети.

Бинарная классификация (Binary classification) – задача классификации объекта на два класса.

Under-Sampling – процесс изменения распределения объектов разных классов путём уменьшения количества объектов превалирующего класса.

Label Smoothing – процесс «сглаживания» дискретных значений меток классов в непрерывную величину.

Стратификация (Stratification) – процесс выборки множества из генеральной совокупности, при котором внутри выбранного множества сохраняется распределение признака максимально близко к генеральной совокупности.

Псевдо-разметка (Pseudo-labeling) – разметка классов, полученная от нейронной сети, путём инференса на данных с неизвестной разметкой.

Аугментация (Augmentation) – метод преобразования данных для увеличения обобщающей способности модели, которая будет обучаться на них.

Test Time Augmentation (TTA) – аугментация, применяемая к тестовым данным для увеличения точности предсказания.

Ансамбль моделей (Models ensemble) – множество моделей, которые могут выполнять одну задачу и объединять свои предсказания.

Оптимизатор (Optimizer) – алгоритм градиентного спуска. К примеру, стохастический градиентный спуск.

Скорость обучения (Learning rate) – параметр оптимизатора, определяющий величину шага по направлению вектора градиента.

Планировщик скорости обучения (Learning rate scheduler) – “планировщик”, определяющий изменение скорости обучения во время обучения.

Обучающая выборка (Train set) – выборка данных, для которой известна разметка.

Валидационная выборка (Validation set) – выборка данных, для которой известна разметка, используемая для валидации качества обучения. Обычно является подвыборкой обучающей выборки.

Тестовая выборка (Test set) – выборка данных, для которой не известна разметка.

Отложенная выборка (Hold-out set) – выборка данных, для которой известна разметка, используемая для построения ансамблей и других способов смешивания результатов работ моделей.

Фолды (Folds) – непересекающиеся блоки, на которые может быть разбита обучающая выборка.

Out Of Fold (OOF) Predictions – предсказания для тестовой выборки, данные моделями после обучения на всех, кроме одного, фолдах.

Threshold – порог, граница принятия решений.

Таргет, целевая переменная, метка класса – синонимы, обозначающие значение, которое нужно предсказать.

Введение

Рак кожи – наиболее распространенный тип рака. Меланома, в частности, является причиной 75% случаев смерти от рака кожи, несмотря на то, что она является наименее распространенным видом рака кожи. По оценкам Американского общества по борьбе с раком, в 2020 году будет диагностировано более 100 000 новых случаев меланомы. Также ожидается, что от этого заболевания умрут почти 7000 человек. Как и в случае с другими видами рака, раннее и точное обнаружение, в основном благодаря научным данным, может сделать лечение более эффективным.

В настоящее время дерматологи оценивают каждую родинку пациента, чтобы выявить внешние поражения или "уродливых утят", которые, скорее всего, являются меланомами. Существующие ИИ подходы к искусственному осмотру не учитывают должным образом эту клиническую картину. Дерматологи могли бы повысить свою диагностическую точность, если бы алгоритмы обнаружения учитывали контекст изображения внутри одного и того же пациента для определения того, какие изображения представляют собой меланому. В случае успеха классификаторы были бы более точными и могли бы лучше поддерживать работу дерматологических клиник.

Меланома – смертельная болезнь, но, если ее обнаружить рано, большинство меланом можно вылечить с помощью небольшой операции. Инструменты анализа изображений, автоматизирующие диагностику меланомы, повысят точность диагностики у дерматологов. Лучшее выявление меланомы дает возможность положительно повлиять на жизни миллионов людей. [1]

Первой задачей этого проекта стоит выявление меланомы в изображениях родинок и поражений кожи. В частности, будут использоваться изображения кожи одного и того же пациента и определяться, какие из них могут представлять собой меланому.

Второй частью является исследование возможности улучшения этого решения с применением малоизученной техники *label smoothing*'а псевдо-размеченных данных, которой и посвящена тема работы. Это техника, которая позволяет помочь в калибровке уверенности предсказания математической модели, добавляя в неё большую неуверенность и, таким образом, улучшить её обобщающую способность на данных из реального мира с неидеальной разметкой и данными для обучения и инференса. Одной из таких актуальных задач реального мира и является определение меланомы по фотографии родинки или поражения кожи.

Данный метод может быть применим к задаче такого рода, потому что добиться унификации способа съёмки крайне сложно: всегда будут разные фотокамеры, условия съёмки и прочее. Также меланома имеет свойство развиваться, и её визуальные признаки могут выражаться по-разному в разные этапы заболевания, а в этих данных, как и во многих последующих, которые будут собираться, признак «есть меланома или нет» будет принимать значение только «Да» и «Нет». Таким образом, важно попытаться применить данный метод и узнать результаты при различных подходах.

Основная часть

1. Описание датасета

Архив ISIC содержит самую большую общедоступную коллекцию дермоскопических изображений поражений кожи с контролем качества.

Изображения предоставляются в формате DICOM. Он доступен с помощью общедоступных библиотек, таких как *pydicom*, и содержит как изображения, так и метаданные. Это широко используемый формат данных медицинской визуализации.

Изображения также предоставляются в формате JPEG и TFRecord (в каталогах *jpeg* и *tfrecords* соответственно). Изображения в формате TFRecord были изменены до единого размера 1024x1024. Метаданные представляются также вне формата DICOM, в файлах CSV. Всего изображений 44 108, из них 33 126 в обучающей выборке и 10 982 в тестовой.

1.1 Обзор доступных файлов

- *train.csv* – обучающая выборка;
- *test.csv* – тестовая выборка.

Обозначения колонок:

- *image_name* – уникальный идентификатор, указывает на наименование DICOM изображения;
- *patient_id* – уникальный идентификатор пациента;
- *sex* – пол пациента (значение пусто, если неизвестно);
- *age_approx* – примерный возраст пациента на время съёмки (около 20, 25, и т.д.);
- *anatom_site_general_challenge* – местоположение на теле;
- *diagnosis* – детальный диагноз (только в обучающей выборке);
- *benign_malignant* – индикатор злокачественности;
- *target* – бинаризированная версия целевой переменной (0 и 1).

1.2 Обзор изображений

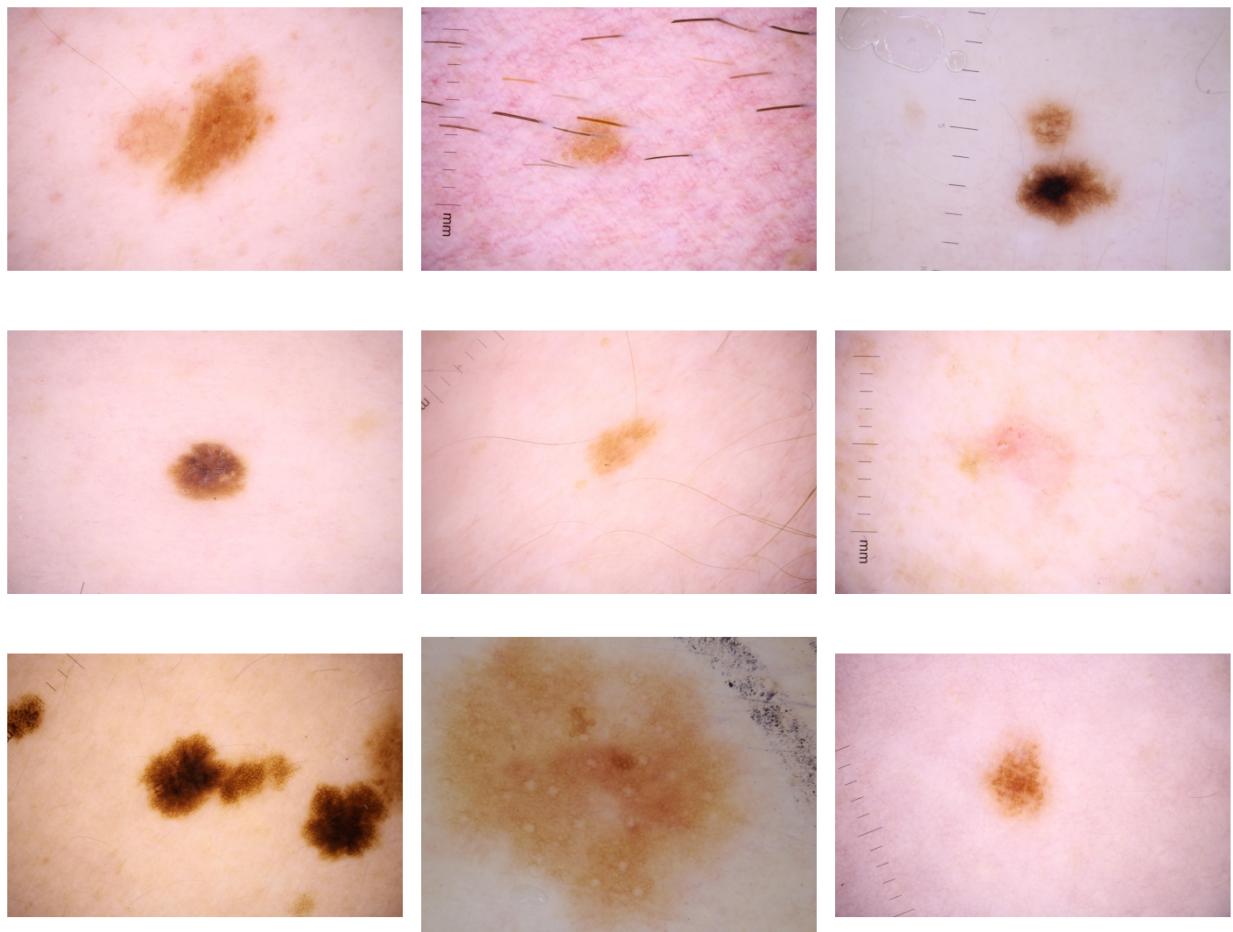


Рисунок 1.2.1 – Случайные изображения

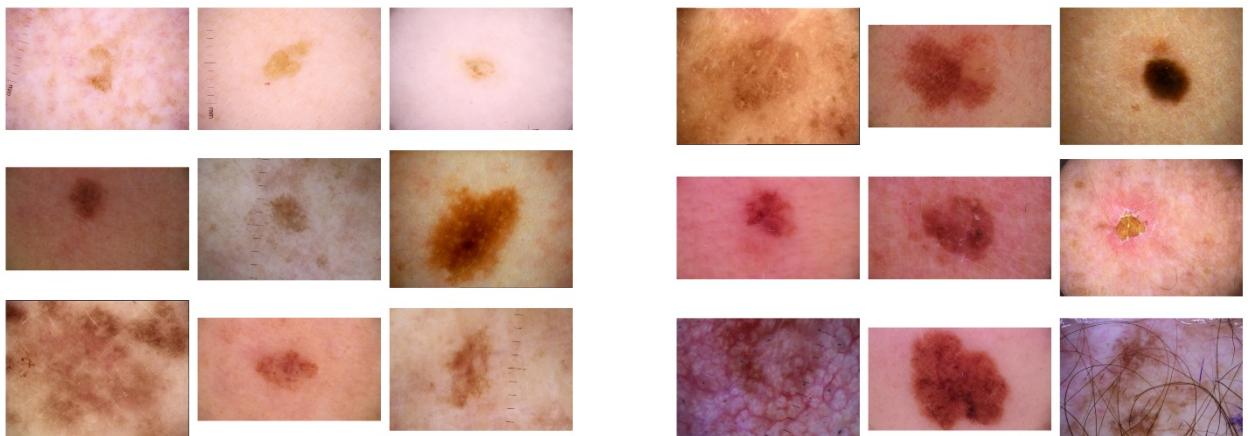


Рисунок 1.2.2 – Изображения с меткой доброкачественности



Рисунок 1.2.3 – Изображения с меткой злокачественности

Заметьте, что изображения имеют разное разрешение и форму.

2. Метрика качества

Метрика в задаче машинного обучения с учителем оценивает качество работы алгоритма, сверяя предсказания и истинные метки.

В данной задаче метрикой является ROC-кривая (receiver operating characteristic), а именно, AUC (Area Under ROC Curve) – площадь под ROC-кривой [2]. ROC-кривая строится как график с осями TPR (True Positive Rate) и FPR (False Positive Rate) [3] с рассчитыванием значений этих статистических величин, исходя из поочерёдного использования различных threshold'ов.

Метрика имеет следующее визуальное представление (площадь по голубой линией):

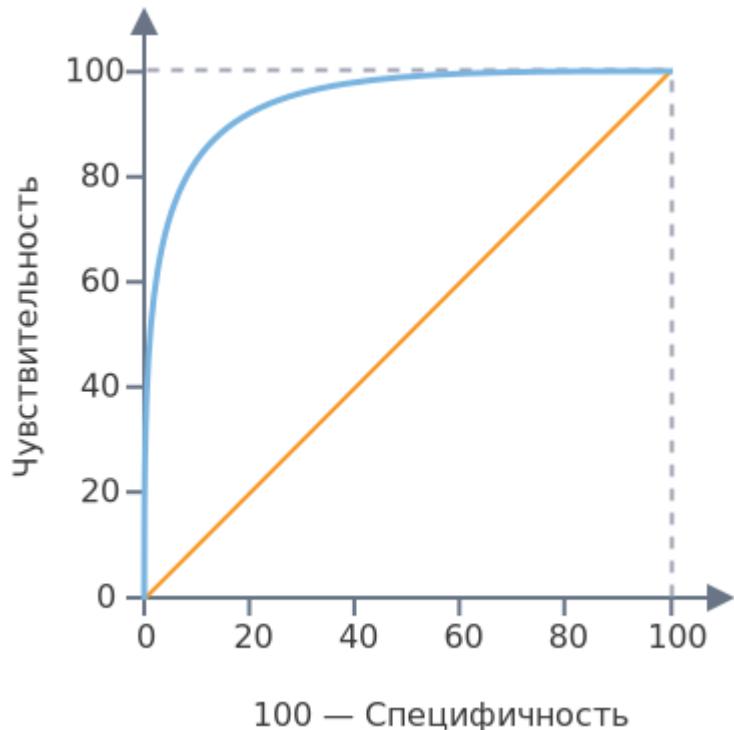


Рисунок 2.1 – график ROC кривой

3. Построение модели определения меланомы

3.1 Обзор данных и стратегия решения

Приведённый в этом заголовке исполняемый код является упрощенным, чтобы не перегружать документ. Полноценный же код вы можете посмотреть в репозитории к работе, ссылка на который есть в конце Заключения.

Для начала, считаем данные в файлах CSV:

```
train =  
pd.read_csv('..../input/train.csv')  
test =  
pd.read_csv('..../input/test.csv')
```

С самого начала смотрим на признак target и с помощью применения функции `value_counts()` к признаку target видим, что изображений с злокачественной опухолью всего 584, в то время как без неё 32 543:

```
train['target'].value_count  
s()
```

Также узнаём, что в трейне 1077 мужчин и 977 женщин, а в тесте 364 мужчин и 326 женщин:

```
train.groupby('patient_id')  
['sex'].first().value_counts()  
test.groupby('patient_id')  
['sex'].first().value_counts()
```

Было бы интересно узнать, раз нам дан признак age_approx, сколько лет наблюдались пациенты, содержащиеся в данных. Признак представлен дискретными значениями, кратными 5, так что можно лишь примерно сказать, сколько было человеку, у которого этот признак равен, скажем, 30.

Для начала посмотрим на то, в каком возрасте вообще находятся пациенты, у которых делали снимки родинок: как доброкачественных, так и злокачественных:

```
import seaborn as sns  
  
sns.countplot(train['age_approx'],  
hue=train['benign_malignant'],  
palette=['g', 'r'])
```

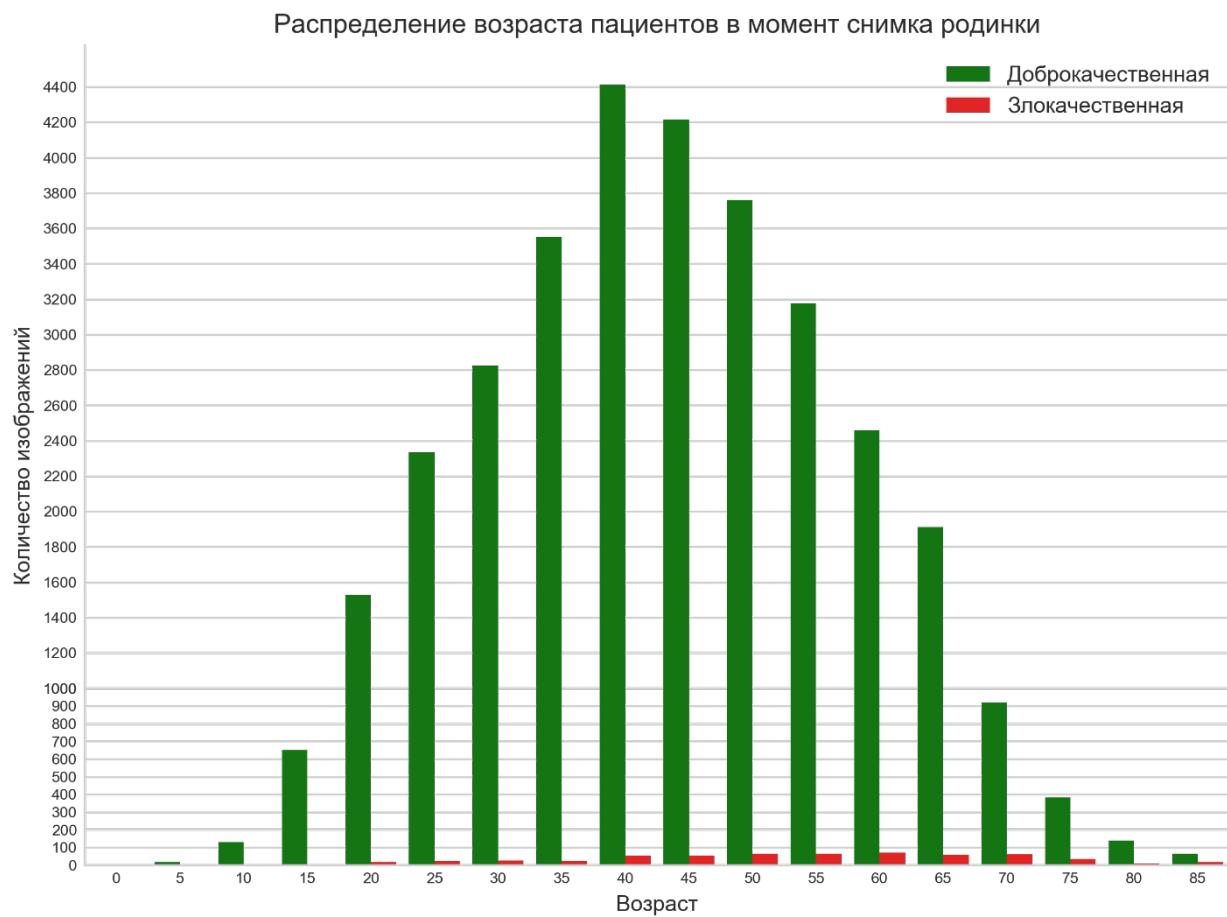


Рисунок 3.1.1 – Распределение возраста пациента в момент снимка родинки

По графику можно увидеть, что злокачественные родинки встречаются чаще у взрослых людей, что, кажется, логично с точки зрения биологии.

```
train.groupby(['patient_id'])['age_approx'].nunique()
           .value_counts()
           .plot(kind
= 'bar')
```

Также мне стало интересно, как долго могли наблюдаваться эти пациенты, поэтому я посмотрел на количество разных *age_approx* у пациентов:

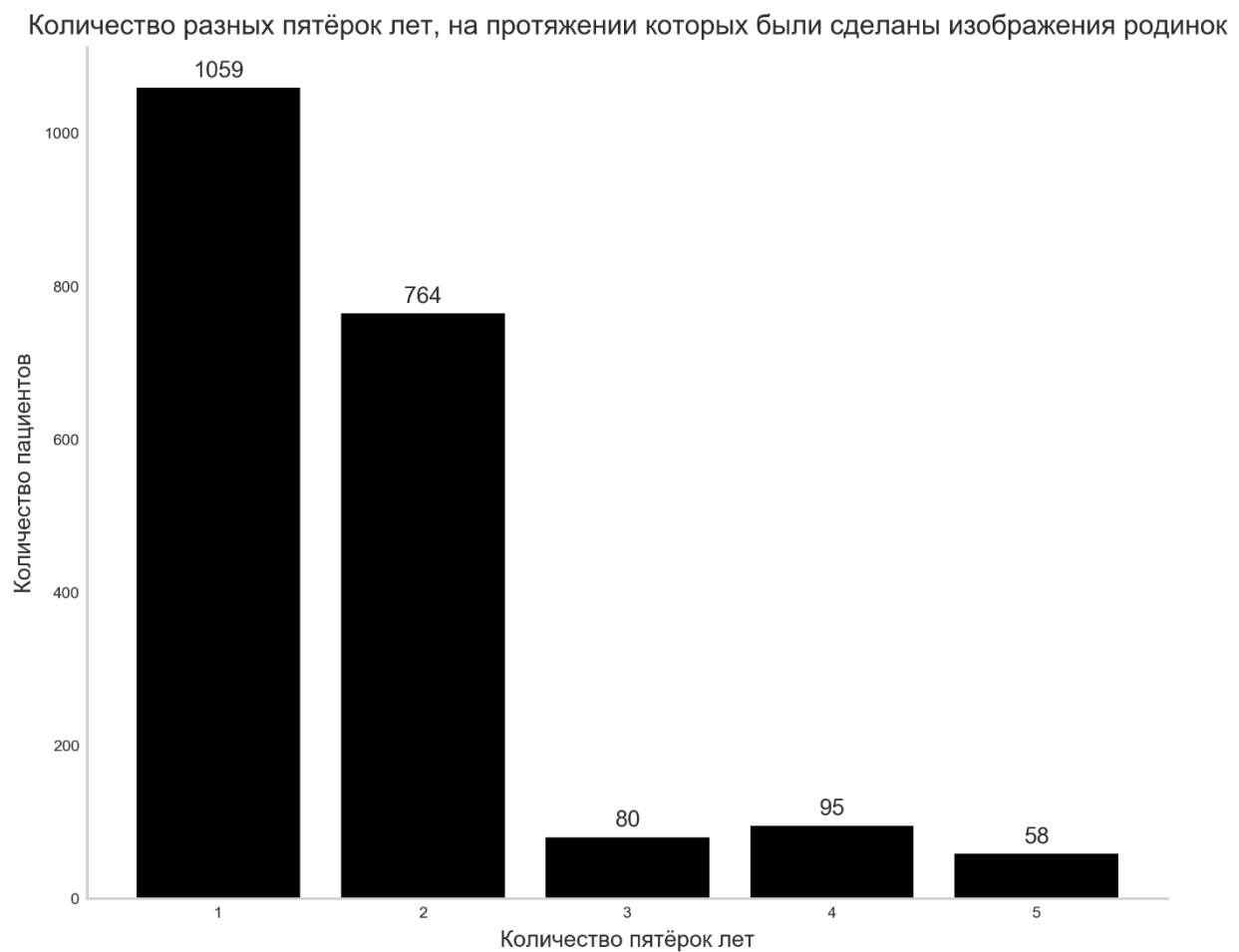


Рисунок 3.1.2 – количество «пятёрок» лет, на протяжении которых были сделаны изображения родинок

Несложно заметить, что большинство пациентов наблюдались в течение максимум 5 лет, в то время как самый долго наблюдающийся пациент мог быть на наблюдении больше 25 лет. При этом, лишь 11% пациентов наблюдались как минимум 10 лет.

Кстати, довольно интересно, а сколько вообще изображений есть по каждому из пациентов? Узнать точное число по пациенту не сложно:

```
train.groupby(['patient_id']).size()
```

Оказывается, самое небольшое – это два изображения, и они есть только у один пациента, у остальных же от трёх. Максимальное количество – 115 у четверых пациентов.

Узнаем среднее:

```
train.groupby(['patient_id']).size().mean()
```

Среднее количество – 16 изображений, но хочется поближе узнать статистики по этому измерению, и чтобы их получить в визуальном представлении, нам может помочь диаграмма размаха [4]:

```
import matplotlib.pyplot as plt  
  
plt.boxplot(train.groupby(['patient_id']).size(),  
            showfliers=False, vert=False)
```

Было решено убрать немногочисленные выбросы, так как они будут лишь портить картину.

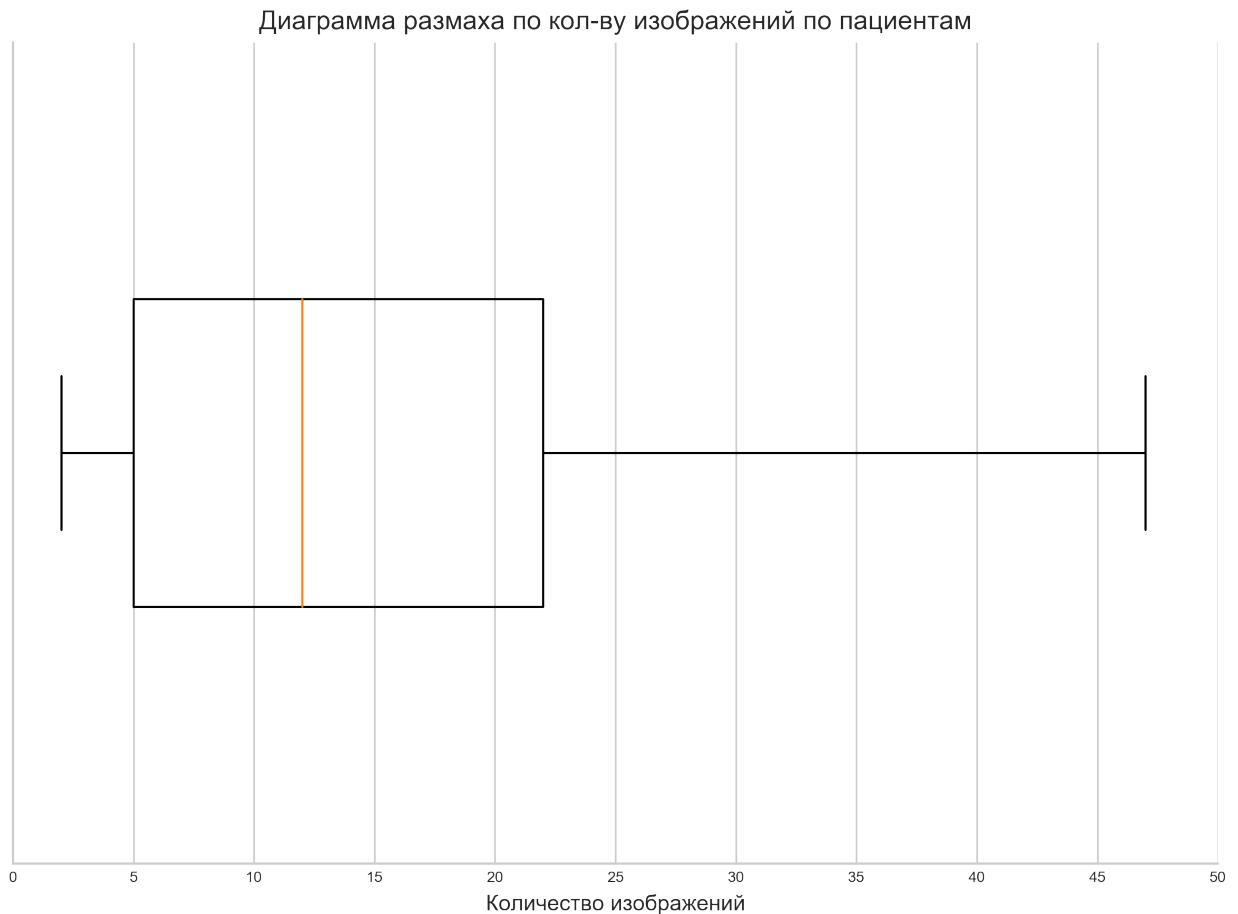


Рисунок 3.1.3 – Диаграмма размаха по количеству изображений по пациентам

Медианное количество изображений по пациенту равно 12, а 75-ый перцентиль равен 22. Можно сделать вывод, что люди либо единоразово фотографировали родинки у пациентов, либо поддерживали присмотр за их развитием с годами.

Для закрепления понимания, ещё одна визуализация:

```
import matplotlib.pyplot as plt
```

```

image_freq_per_patient =
train.groupby(['patient_id']).size()
plt.hist(image_freq_per_patient,
         image_freq_per_patient.nunique(),
color='black')

```

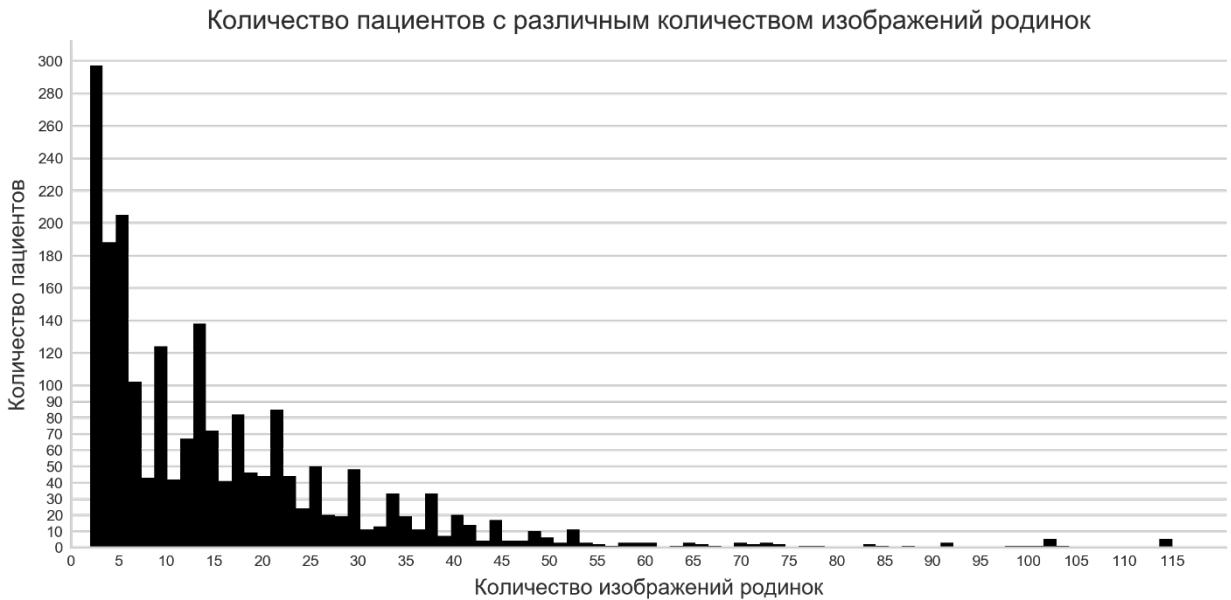


Рисунок 3.1.4 – Количество пациентов с различным количеством изображений родинок

Теперь можно посмотреть на признак *anatom_site_general_challenge*, который говорит нам о расположении родинки на теле:

```

import matplotlib.pyplot as plt
import numpy as np
site_vs_diagnosis = (train.groupby(
    ['anatom_site_general_challenge',
     'benign_malignant'])
    .count()['patient_id'])
labels =
(train['anatom_site_general_challenge']
     .value_counts(normalize=True)
     .sort_values().index)
benign_data = site_vs_diagnosis[0:12:2]
malignant_data = site_vs_diagnosis[1:12:2]
x = np.arange(len(labels))

width = 0.35
fig, ax = plt.subplots(figsize=(8,4))
ax.bar(x-width/2, benign_data, width,
color='g')
ax.bar(x+width/2, malignant_data, width,
color='r')

```

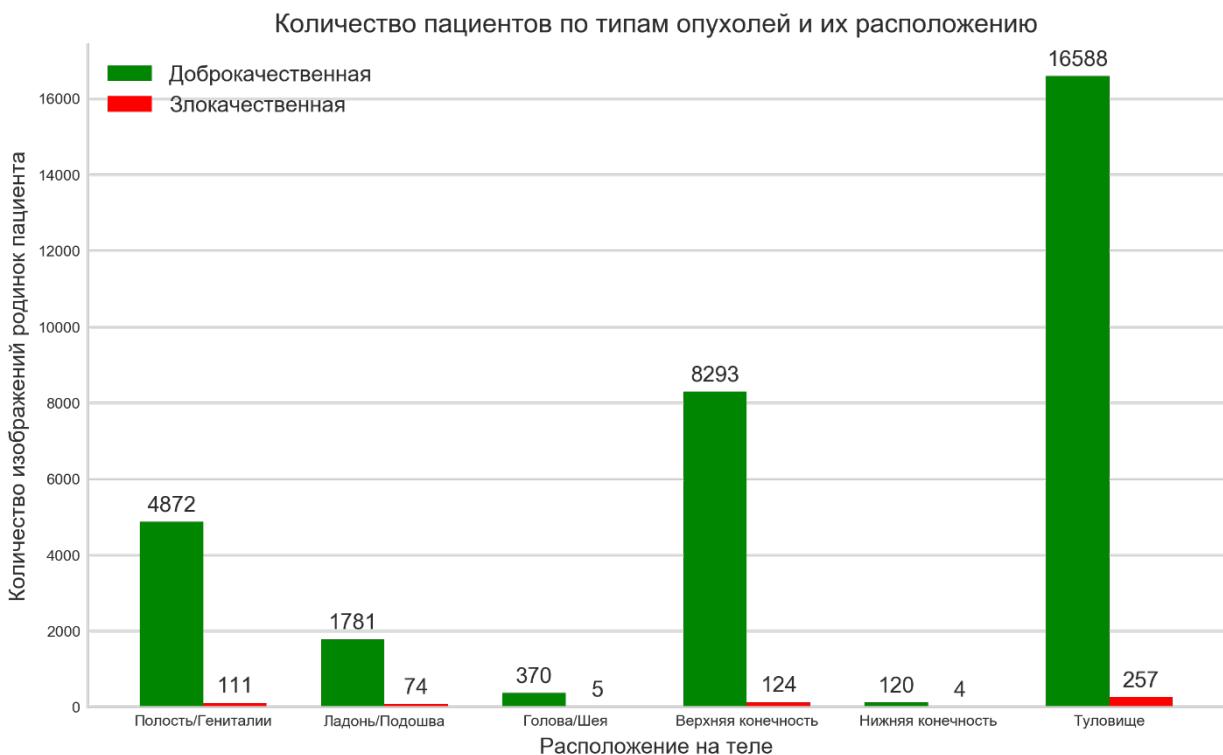


Рисунок 3.1.5 – Количество пациентов по типам опухолей и их расположению

К сожалению, из представленного графика нельзя почертнуть ничего, кроме знания о численных значениях: никаких открытий в этом признаке нет.

Полученная информация определённо дала нам понимание того, с какими данными мы имеем дело, и поэтому можно переходить к решению задачи. А какой именно задачи? С учётом всего вышеописанного, мы видим, что перед нами стоит задача несбалансированной бинарной классификации изображений с метаданными о пациентах.

Задачи, связанные с изображениями, обычно решаются с применением свёрточных нейронных сетей, тем же путём пойду и я. Изначально было решено использовать только изображения и данные об их принадлежности пациентам, хотя, очевидно, остальные табличные данные тоже могут помочь. Поэтому речь здесь пойдёт о построении свёрточной нейронной сети, принимающей на вход только изображение. Сейчас будут описаны шаги, по которым мы шли, чтобы получить решение и начать проводить исследование, которому посвящена, но не ограничивается ею, тема данной работы.

3.2 Обработка данных

Самыми первыми действиями были скачивание и обработка датасета. Обработка заключается в очевидном уменьшении разрешения по причине того, что исходные изображения имели разрешение даже до 4000x6000 пикселей. Вопрос в том, можно ли не

квадратные изображения преобразовывать в квадратный формат с точки зрения изменения формы дефекта кожи на изображении? Нельзя ли будет по измененному изображению предположить, что оно имеет другой класс, нежели до преобразования? Ответ можно дать, узнав, какие характеристики родинки могут говорить о злокачественности, а они, применимо к данной задаче, примерно следующие: асимметричность, нерегулярность, неровность, зубчатость или размытость границ, а также наличие нескольких цветов [5]. Таким образом, можно сделать вывод, что мы не можем применять эластичные трансформации. Далее эти знания помогут нам при составлении корректных аугментаций.

Соответственно, можно либо просто привести в квадратную форму с уменьшением размера, либо из изображений, которые имеют не квадратную форму, вырезать 2 пересекающихся квадрата и изменять уже их размер. Второй способ сложен с точки зрения имплементации и может быть некорректен с точки зрения присвоения меток класса, так как может оказаться так, что на одном из квадратов не будет вообще ничего. В итоге, мы сделали 2 датасета, состоящие из изображений в разрешении 512x512 и 256x256.

3.3 Разбиение данных для обучения

При обучении нужны данные для валидации, на которых можно оценивать качество работы модели, поэтому стандартной техникой является разбиение данных на N фолдов. После этого поочерёдно производится обучение на разных наборах фолдов и кроссвалидация [6].

В данном случае данные несбалансированные относительно меток класса: злокачественность отмечена на 1.76% всех изображений в трейне. Поэтому стоит исследовать стратифицированное разбиение на фолды, то есть в полученных фолдах распределение целевой переменной будет максимально приближено к распределению всего трейна.

С другой стороны, у каждого пациента есть хотя бы 2 изображения в трейне, и, при этом из 2056 пациентов, у 427 есть изображения с и без меланомы. Соответственно, имеет смысл делать разбиение с непересекающимися группами пациентов. Если переводить на понятный язык, то это задаёт условие того, чтобы каждый пациент полностью содержался только в одном фолде. Что интересно, при таком типе разбиения, сохраняется близкое к трейну распределение таргета.

Также для того чтобы иметь возможность для тестирования гипотез и проверки идей, нужно выделить hold-out сет, который бы “не видела” нейросеть при обучении, для которого были бы известны метки класса. Соответственно, нужно взять его из трейна. Мы решили взять 10% всего трейна под hold-out сет, а остальные 90% для обучения и для разбиения по фолдам. Естественно, этот hold-out сет должен иметь то же распределение, что и трейн.

Таким образом, были созданы оба типа разбиения с выделением 10% под hold-out сета (по соответствующему правилу разбиения). Ниже представлены распределения таргета по фолдам в обоих разбиениях и ситуация в трейне.

Таблица 3.3.1 – Распределение таргета в трейне

	Все данные
% класса 0	98.237034
% класса 1	1.762966

Таблица 3.3.2 – Распределение таргета при разбиении со стратификацией

	Фолд 1	Фолд 2	Фолд 3	Фолд 4	Фолд 5	Hold-out
% класса 0	98.2391	98.2391	98.2388	98.2388	98.2388	98.2196
% класса 1	1.7608	1.7608	1.7611	1.7611	1.7611	1.7803

Таблица 3.3.3 – Распределение таргета при разбиении по группам

	Фолд 1	Фолд 2	Фолд 3	Фолд 4	Фолд 5	Hold-out
% класса 0	98.4236	98.4739	98.0543	97.9033	98.3227	98.2498
% класса 1	1.5763	1.5260	1.9456	2.0966	1.6772	1.7501

Как мы видим, при разбиении по группам, распределение недопредставленного класса может варьироваться в границах до 20% от трейна.

3.4 Аугментация

Для того чтобы улучшить обобщающую способность сети (generalization), необходимо обучать её на различных вариациях предсказываемого класса. Для задач

компьютерного зрения этими различными вариациями могут быть изображения исследуемого объекта с разных ракурсов, в разных состояниях и т.д.

Как было замечено в п. “Обработка данных”, в данной задаче подходят только аффинные трансформации, то есть все виды поворотов, отзеркаливаний, сдвигов, приближений и т.д. Также существует множество вариантов аугментаций, связанных с изменением цвета, яркости, контрастности и другими параметрами изображений. Применимо к этой задаче, можно сказать, что изменять исходные цвета нежелательно, так как одним из основных признаков меланомы является её цвет. При этом, изображения делались в разных условиях, среди которых освещённость, разные параметры фотокамеры, тряска рук и множество других факторов. Исходя из этого, мы решили, что в данном случае можно менять яркость, контраст и освещенность.

Окончательный список аугментаций, используемых для обучения, получился следующим:

```
import albumentations as A

A.Compose([
    A.OneOf([
        A.ShiftScaleRotate(rotate_limit=90,
p=1.0),
        A.HorizontalFlip(p=1.0),
        A.VerticalFlip(p=1.0),
        A.RandomRotate90(p=1.0),
    ], p=0.5),

    A.OneOf([
        A.RandomBrightness(p=1.0),
        A.RandomBrightnessContrast(p=1.0),
        A.RandomGamma(p=1.0),
    ], p=0.5)
])
```

Также существует такая техника, как Test Time Augmentation, или просто ТТА. По названию становится понятно, что она применяется во время инференса для улучшения качества предсказания. Суть проста: во время предсказания на вход нейросети подаётся не картинка, а её аугментированные состояния, по каждому из которых делается предсказание. Далее, эти предсказания усредняются или смешиваются каким-либо другим образом.

Нами были выбраны повороты на 0, 90, 180 и 270 градусов и поворот по горизонтали, а позже усреднение предсказаний с них.

3.5 Эксперименты обучения

Для ведения проводимых экспериментов и повышения воспроизводимости результатов было сделано следующее:

- Зафиксированы сиды (случайные состояния) для всех функций, в которых есть процесс генерации случайных чисел, в том числе для вычислений на GPU;
- Каждый процесс обучения сопровождается конфигурационным файлом формата YAML, в котором содержатся все параметры эксперимента;
- В каждом из экспериментов сохранялись веса обученных сетей и файлы с предсказанием тестовой выборки, а также вычислялось значение метрики на hold-out выборке.

Для работы по проекту было решено использовать свёрточные нейронные сети архитектуры EfficientNet [7]. Спустя несколько экспериментов и сравнений стало понятно, что тип EfficientNet-B3, предобученный на датасете ImageNet [8], а именно, с помощью т.н. noisy student [9], является самым оптимальным. Функцией потерь была выбрана бинарная кросс-энтропия [10], а оптимизатором AdamW [11, 12]. Планировщиком был выбран ReduceLROnPlateau, но в нескольких экспериментах присутствовал StepLR – оба присутствуют в фреймворке PyTorch [13]. Полный список гиперпараметров и алгоритмов можно найти в репозитории. Обучалось 5 OOF моделей, предсказания которых на teste усреднялись. При этом, сами эти предсказания были получены с помощью TTA. Предсказания для hold-out сеты тоже были получены с помощью TTA каждой моделью отдельно, а затем усреднялись.

Всего было полностью проведено 7 больших экспериментов обучения, сводка по которым содержится в таблице ниже.

Таблица 3.5.1 – результаты и условия проведения экспериментов обучения

Номер Эксперимента	Используемое разбиение	Разрешение изображений	Размер батча	Кол-во эпох	Начальный learning rate	Метрика на 30% теста ¹	Метрика на hold-out сете
2	Страт.	512x512	10	15	0.0003	0.901	0.8263
3	Страт.	256x256	32	20	0.001	0.868	0.7252
4	Группы	256x256	32	20	0.001	0.884	0.8620

¹Предсказания на тест загружались на Kaggle, во время соревнования там показана метрика по 30% предсказаний только с 3 цифрами после запятой.

5	Группы	512x512	10	25	0.001	0.891	0.8892
6	Группы	512x512	10	25	0.0003	0.914	0.9040
7	Страт.	512x512	10	25	0.0003	0.911	0.8529

Несколько комментариев о принятых решениях в плане смены гиперпараметров:

- Смена разрешения изображений с 512x512 на 256x256 и обратно связана с тем, что на втором разрешении процесс обучения длится примерно в 3-4 раза быстрее, но, при этом, получаются более плохие результаты. То есть, если эксперимент с 512x512 длился от 12 часов, то с 256x256 мог уложиться и в 4;
- Переход от стратифицированного разбиения к разбиению по группам и обратно связан с тем, что идея о разбиении по группам появилась лишь во время проведения четвертого эксперимента, в то время как в ходе седьмого эксперимента стояла только цель проверить результат;
- Изменение количества эпох непрерывно связано с ранней остановкой при отсутствии улучшений, ReduceLROnPlateau с его сменой скорости обучения и с его фактором, на который умножается скорость обучения;

Также было проведено очень много небольших, одно-фолдовых экспериментов, исходя из которых подбирались и другие гиперпараметры. Самый первый большой эксперимент пропущен, так как после рефакторинга кода не удалось восстановить решение.

Из всего вышеупомянутого можно установить, что наилучшей стратегией для обучения в данных условиях будет обучение на изображениях в большом разрешении, начиная с learning rate'а порядка десятитысячных, scheduler'ом ReduceLROnPlateau и оптимизатором AdamW.

3.6 Анализ полученных результатов

В таблице 3.5.1 представлены значения метрик, но однозначно сказать, какая модель объективно, с точки зрения метрики, лучше, мы не можем. Также нельзя выбрать, какую модель следовало бы брать в дальнейшем для проведения основного исследования. Для этого необходимо провести более глубокое исследование предсказаний теста, обращая внимание на условия обучения и генеральную совокупность.

Для начала, можно взглянуть на распределение таргета и сравнить его с трейном, делая логичное, и, скорее всего, верное предположение о том, что баланс классов там такой же. Напомним, что в трейне к классу 0 относится **98.237034%** данных, а к 1 - **1.762966%**.

Таблица 3.6.1 – Распределение таргета в предсказании разных экспериментов

Номер эксперимента	2	3	4	5	6	7
% класса 0	99.4810	99.9454	99.9454	99.9545	99.9180	99.7359
% класса 1	0.5190	0.0546	0.0546	0.0455	0.0820	0.2641

Результаты, которые видны в таблице, в явном виде дают нам понимание того, что ни одно из распределений не близко к распределению генеральной совокупности, то есть, трейна. Смотря на 6-ой эксперимент, становится удивительно, что с таким распределением классов у него лучшие результаты на тесте и hold-out'e – это может говорить о том, что все предсказания класса 1 должны быть корректны и должны принимать примерно одно значение (исходя из того, как строится ROC кривая). Также внимание привлекают 3-ий, 4-ый и 5-ый эксперименты с критически низким процентом предсказаний класса 1. Если у 2-го эксперимента – с наивысшим процентом – он ниже процента генеральной выборки более, чем в 3 раза, то для 3-го, 4-го и 5-го он ниже более, чем в 30 раз. Такие модели использоваться дальше не будут как минимум из-за этого пункта. 7-ой же эксперимент лежит примерно посередине между всеми параметрами, поэтому для дальнейшего рассмотрения оставим только 2-ой, 6-ой и 7-ой.

Сейчас необходимо узнать уверенность предсказаний моделей, а для этого можно посмотреть сколько предсказаний лежит в разных интервалах от 0 до 1 в абсолютном и процентном количествах. Также, если считаем, что класс 0 ставится в случае, когда предсказание не больше 0.5, а иначе – 1, то хочется понять, какая доля значений «своего» класса лежит в разных интервалах. Всё это было объединено в таблице ниже:

Таблица 3.6.2 – Детальное распределение таргета предсказаний разных экспериментов

Номер эксперимента	2			6			7			
	Интервал	N Абс.	N %	N % класса	N Абс.	N %	N % класса	N Абс.	N %	N % класса
[0,00, 0,05]	1005 7	91,5 7	92,05 9		957 9	87,2 3	87,30 87,30	1018 5	92,7 4	92,99 92,99
(0,05, 0,10]	373	3,40	3,41		780	7,10	7,11	381	3,47	3,48

(0.10, 0.15]	169	1,54	1,55	256	2,33	2,33	149	1,36	1,36
(0.15, 0.20]	113	1,03	1,03	136	1,24	1,24	83	0,76	0,76
(0.20, 0.50]	213	1,94	1,95	222	2,02	2,02	155	1,41	1,42
(0.50, 0.80)	52	0,47	91,23	9	0,08	100,0	29	0,26	100,0
[0.80, 0.85)	4	0,04	7,02	0	0,00	0,00	0	0,00	0,00
[0.85, 0.90)	1	0,01	1,75	0	0,00	0,00	0	0,00	0,00
[0.90, 0.95)	0	0,00	0,00	0	0,00	0,00	0	0,00	0,00
[0.95, 1.00]	0	0,00	0,00	0	0,00	0,00	0	0,00	0,00

Сразу бросается в глаза то, что у 6-го и 7-го экспериментов крайне низкая уверенность в предсказаниях класса 1, в то время как у 2 эксперимента она не сильно, но выше, к тому же таких предсказаний в принципе больше. Опять же, если смотреть на 6-ой, то становится неясным то, почему он получил наилучшие значения метрик, в отличии от 2-го и 7-го. Мы только можем привести некоторые различия в их обучении:

1. 5 моделей из ансамбля 6-го эксперимента обучались в среднем 8.2 эпохи, в то время как модели из ансамбля 2-го и 7-го экспериментов обучались в среднем 14 эпох;
2. 6-ой эксперимент базировался на разбиении фолдов по группам, в то время как 2-ой и 7-ой на стратифицированном разбиении;
3. Также параметр ранней остановки у 6-го эксперимента был выставлен 4 эпохи, в то время как у 2-го и 7-го 3 и 5 соответственно.

Последнее, что не было до сих пор использовано – это предсказания на hold-out сет, который, между прочим, может дать нам большее понимание вследствие того, что там мы знаем истинные значения и можем наблюдать то, на каких конкретно сэмплах ошибаются модели. Идеально подойдёт для этого график распределения предсказаний, полученный с помощью Kernel Density Estimation [14], он представлен ниже:

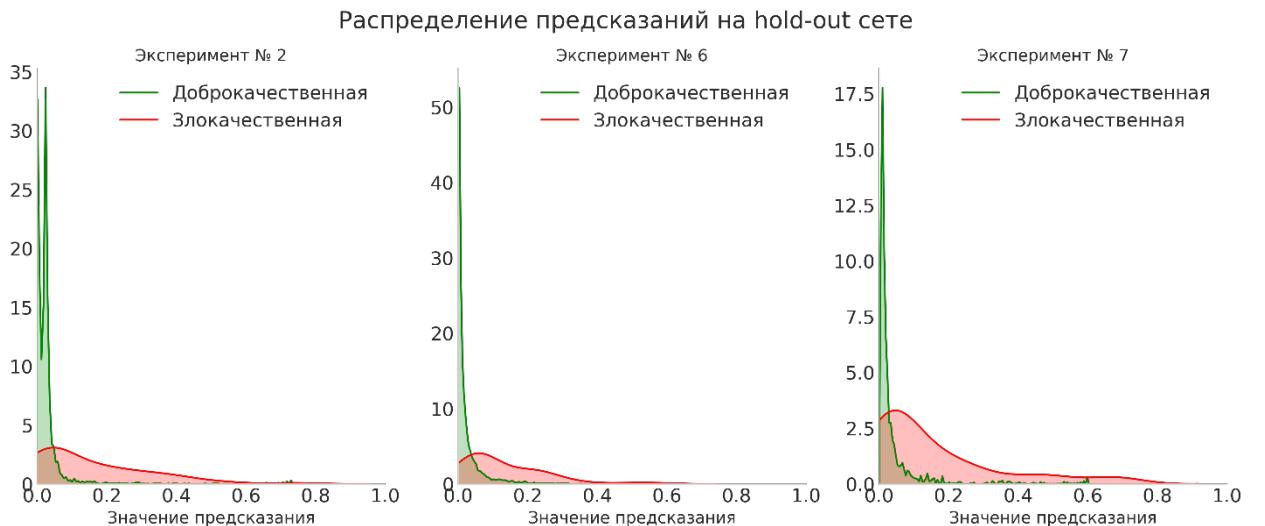


Рисунок 3.6.1 – Распределения предсказаний на hold-out сете

По графику видно, что распределения 2-го и 7-го экспериментов очень похожи друг на друга, но 2-ой менее уверен в классе 0, чем первый – это выражено тем, что пик с левой стороны отдалён от границы нуля.

Из всего вышеизложенного мы можем сделать вывод о том, что, возможно, модели 6-го эксперимента недообучились в способности отличать первый класс, и делают это редко, но метко, в то время как модели из 2-го и 7-го экспериментов переобучились и определяют сравнительно большое количество сэмплов нулевого класса неверно.

Обдумав и проанализировав все преимущества и недостатки, нами было принято решение взять модели и предсказание 2-го эксперимента как основные для проведения исследования.

Дальнейшая работа будет продолжена в рамках борьбы с осложняющим фактором в виде экстремально несбалансированного датасета и в рамках попытки использовать исходные и сглаженные псевдо-размеченные данные для fine-tuning'а нейросети.

4. Исследование влияния Label Smoothing'a

4.1 Что такое Pseudo-Labeling

Pseudo-labeling – разметка классов, полученная от нейронной сети, путём инференса на данных с неизвестной разметкой [15]. Она может применяться для дальнейшего дообучения нейронной сети или просто для разметки данных с некоторой заданной уверенностью. Данный способ успешно применяется как в исследовательских проектах [16], так и в различных соревнованиях по машинному обучению, достаточно почитать обзоры решений задач и убедиться в том, что почти всегда обучения на псевдо-размеченных данных является преимуществом и улучшает обобщающую способность моделей. Более того, автор и сам использовал эту технику и получал лучший результат, чем до применения.

4.2 Что такое Label Smoothing

Label Smoothing – это процесс «сглаживания» дискретных значений меток классов в непрерывную величину [17, 18, 19]. Если говорить про то, как была представлена эта идея изначально [17], то давайте представим функцию активации предпоследнего слоя нейронной сети, решающей задачу классификации:

$$p_k = \frac{e^{(x^T w_k)}}{\sum_{l=1}^L e^{(x^T w_l)}}$$

, где

p_k – вероятность того, что модель присвоит k -ый класс,

w_k – веса и сдвиг последнего слоя,

x – вектор, содержащий значения активаций в предпоследнем слое.

Затем мы используем кросс-энтропию как функцию потерь и пытаемся минимизировать ее, чтобы максимизировать логарифм правдоподобия.

Проблема в дискретных, «жёстких» метках. Модель должна производить большое значение логита² для правильной метки. Это способствует тому, что различия между самым большим логитом и всеми остальными становятся большими, а это, в сочетании с ограниченным градиентом, снижает способность модели адаптироваться, в результате

² Логитом называется вещественное значение метки, предсываемое моделью.

чего модель слишком уверена в своих прогнозах. Это, в свою очередь, может привести к переобучению³.

Но это не значит, что такие «жёсткие» метки – это плохие метки. Всё зависит от того, какие цели вы преследуете. Если вы хотите просто максимизировать правдоподобие, то дискретные метки – не плохой выбор, и мы знаем и видели, что это работает. Но если вашим мотивом является создание устойчивой модели, то это может помочь.

Таким образом, в оригинальной работе вводится сглаживающий параметр α , который изменяет таргет следующим образом:

$$y_k^{LS} = y_k(1 - \alpha) + \frac{\alpha}{K}$$

Теперь, вместо того чтобы минимизировать кросс-энтропию с дискретными, «жёсткими» метками, мы минимизируем кросс-энтропию с вещественными, «мягкими» метками.

Это оригинальный подход, и за 5 лет с момента публикации он претерпел множество изменений и модификаций. В принципе, label smoothing'ом сейчас называется почти любое действие, которые «смягчает» метки классов. В данном исследовании будет использоваться иной подход, о котором будет рассказано дальше.

4.3 Применимость к задаче

Для начала стоит заметить, что успешные статьи с хорошими результатами по теме label smoothing'a улучшают и так отличные результаты, в нашем же случае, задача крайне трудная и исходно, отличного решения у нас нет. Нам необходимо понять, какие действия необходимо предпринять, чтобы хоть как-то приблизить наши условия к достаточным для получения достоверных результатов.

4.3.1 Проблема выравнивания распределений

Проблема «недобора» меток класса 1, которая была описана ранее, является большим препятствием к fine-tuning'у моделей, потому как если они и до этого имели плохую предсказательную способность, то предоставив для обучения данные, в которых ещё и в несколько раз меньше представителей класса 1, чем было до этого, то результаты лучше не станут.

³ Переобучение – явление, при котором модель слишком сильно приспособилась к данным, на которых обучалась и не способна обобщать данные, которые ей не были доступны во время обучения.

Поэтому нам необходимо искусственно решить проблему с «выравниванием» распределения таргета в предсказанном teste и исходном трейне какой-то пост-обработкой предсказанных меток. Очень общим названием, который отвечает нашим требованиям, является under-sampling.

Under-Sampling – это процесс уменьшения представителей одного или нескольких классов в данных. В нашем случае необходимо каким-то образом уменьшить количество представителей класса 0 в предсказаниях теста таким образом, чтобы «выровнять» распределения. Нами было придумано два пути: это случайный выбор подвыборки нужного размера, либо выбор подвыборки таким образом, чтобы в разбитом на интервалы отрезке от 0 до 0.5 сохранилось распределение – стратифицированный under-sampling.

Также можно подойти к этому вопросу не со стороны удаления представителей превалирующего класса, а со стороны изменения границы присвоения метки. Понять это можно так: на текущий момент граница присвоения метки – это 0.5: если предсказание больше 0.5, то оно округляется до 1, и если не меньше 0.5, то до 0. А что, если сдвинуть эту границу? Очевидно, количество представителей классов будет меняться. Поэтому, как третий вариант решения проблемы с распределением таргета, предлагается как бы «набирать» представителей класса 0 и присваивать их классу 1, пока распределение классов не достигнет необходимого значения. Таким образом, мы не уменьшаем количество данных для fine-tuning'a, а просто переприсваиваем некоторое количество меток.

Итого, мы имеем 3 способа решения вышеобозначенной проблемы.

4.3.2 Определение «смягчающих» функций

Теперь необходимо определить, какие функции «смягчения» логитов нам необходимо попробовать. Определённо, чтобы было с чем сравнивать, мы должно ввести функцию, которая ничего не делает, то есть отображает логит сам в себя. Далее, исключительно ради эксперимента и понимания действия на результат, хочется немного смягчить метки и отобразить их из 0 и 1 в значения 0.05 и 0.95 соответственно. Точно также с отображением в 0.10 и 0.90.

Следующие функции же будут более разумными и основаны на следующей гипотезе: а что, если можно повлиять на общую уверенность нейросети в предсказании выделенного класса, занизв исходное значение метки, и максимизировав все остальные метки? Это действие позволяет приблизить превалирующий класс к границе присвоения метки, следовательно, у большего количества сэмплов будет шанс «перевалить» за 0.5.

Это звучит логично, поэтому хотелось бы попробовать этот подход по отношению к классу 0. Таким образом, мы добавляем ещё 3 функции, отображающие 0 и 1 в 0.05 и 1, 0.10 и 1 и 0.20 и 1 – то есть оставляем значение метки класса 1, равного 1, но увеличиваем значение метки класса 0, приближая его к границе присвоения метки, чтобы улучшить обобщающую способность модели.

4.3.2 Определение доли набора данных для обучения

Так как мы в данном эксперименте не располагаем исходно очень устойчивой и качественной моделью, то имеет смысл не доверяться полностью предсказаниям модели, а брать лишь те предсказания, в которых она уверена больше всего.

Нами было решено сделать 3 набора данных, состоящих из 50, 80 и 100 процентов данных. Когда мы говорим о N % данных, к примеру, 50%, мы говорим о том, чтобы взять по 50% самых увереных предсказаний по всем классам. Следовательно, с одной стороны, мы уменьшаем количество данных для обучения, а с другой, увеличиваем их надёжность.

Также появляется вопрос о том, какое разбиение данных на фолды выбирать. Так как второй эксперимент проводился на стратифицированных фолдах, то и эксперименты с fine-tuning'ом будут проводиться разбиением на стратифицированные фолды.

4.3.2 Стратегия Fine-Tuning'a

Однозначного ответа на то, какой тип fine-tuning'a лучше, нет. Как это часто бывает в глубоком обучении, всё познаётся в экспериментах, но так как мы имеем ограниченное время и вычислительные ресурсы, то выберем лишь две стратегии обучения.

Некоторые варианты того, как можно дообучаться: дообучать всю сеть, либо её последний или последние несколько слоёв, дообучиваться на трейне с добавлением псевдо-разметки, или только на псевдо-разметке (в таком случае придётся делать несколько экспериментов с балансировкой функции потерь), валидироваться на псевдо-разметке или нет, менять ли параметры обучения на более регуляризованные во избежание быстрого переобучения, или нет, и много других вариантов.

Нами же было выбрано 2 стратегии: обучение всей сети и её головы⁴, как 2 разных подхода. Остальные параметры обучения одинаковы: обучение без валидации на псевдо-разметке в течение 3 эпох с теми же гиперпараметрами, что и во время обучения.

⁴ Голова – синоним для последнего слоя нейросети.

4.4 Проведение экспериментов

Суммируя всё вышеописанное, мы имеем 3 способа выравнивания распределений, 6 «смягчающих» функций, 3 разных доли данных и 2 типа fine-tuning'a. Итого, получается $3 \times 6 \times 3 \times 2 = 108$ уникальных экспериментов.

Под конец проекта было проведено 105 экспериментов. 3 не проведённых были следующими: обучение всей сети на 100% данных с функцией отображения «в самих себя» по всем трём методам борьбы с неподходящим распределением.

4.5 Анализ полученных результатов

В соответствии с таблицей Приложения.1, в которой находятся результаты всех экспериментов, мы готовы предоставить основные выводы по проведённым экспериментам. Заметим, что все выводы будут предоставляться, исходя из значения метрики на hold-out сети. Напомним, что значение метрики 2-го эксперимента равно 0.8263.

Для начала, первый вопрос: помогла ли псевдо-разметка? Да, в 2 из 3 случаев. Если мы берём «сырую» псевдо-разметку, то в зависимости от методы борьбы с распределением, метрика принимает следующие значения: 0.8313, 0.8259, 0.8387. Стратифицированный метод показал результаты хуже, чем без fine-tuning'a.

Далее, лучше дообучать только последний слой или всю сеть? Исходя из средних, значительный перевес у обучения последнего слоя: 0.7712 у всей сети против 0.8333 у головы сети. Это критическая разница, и мы не можем ответить на вопрос, почему получился такой результат, кроме предположения о переобучении. Что интересно, если оставить в результатах только обучение с головой, то ответ на предыдущий пункт изменяется с 2/3 случаев до 3/3 случаев. Поэтому дальше мы будем рассматривать статистики только с обучением головы сети.

Теперь, какой процент наиболее уверенных предсказаний лучше брать? Среднее значение метрик оказалось следующим: 0.8001 у 100%, 0.8485 у 80% и 0.8520 у 50%. Следовательно, можно сделать вывод, что использование от 50 до 80 % наиболее уверенных предсказаний по отдельно взятым классам может значительно улучшить результаты.

Если посмотреть на результаты по функции «смягчения», то лучше всего оказалась функция, которая оставляла данные как есть, среднее значение метрики с ней оказалось равным 0.8439, далее идет функция $(0, 1) \rightarrow (0.05, 1)$ с 0.8353 и далее до 0.8275 . Довольно

интересный результат, по поводу которого можно сделать предположение о том, что псевдо-разметка оказалась недостаточно точной, чтобы функции смогли совершить теоретически задуманное. И из-за неточно качественной псевдо-разметки они просто сильнее зашумляли таргет.

В заключение хочется сказать, что из 54 экспериментов с обучением только головы, 40 получили лучшее значение метрики на hold-out сете лучше, чем у исходного эксперимента. А также, что 14 из 14 экспериментов с худшим значением, брали 100% данных.

Таким образом, исходя из всего вышеописанного, мы делаем вывод, что даже в такой сложной задаче, при всех отрицательно влияющих факторах, предложенные методы всё равно позволяют улучшить результаты работы нейронной сети.

5. Технологический стек

Используемые мощности:

- Компьютер: NVIDIA RTX 2080Ti 11Gb VRAM, Intel i7-9700k, 32Gb RAM, Ubuntu;
- Ноутбук: NVIDIA GeForce MX230 2Gb VRAM, Intel i7-10510U, 16Gb RAM, Ubuntu;
- Google Colab: NVIDIA Tesla K80 12Gb VRAM / NVIDIA Tesla P100 16Gb VRAM, Intel Xeon, 13Gb RAM, дистрибутив Linux.

Используемые программы:

- GitKraken Pro в паре с github.com как система контроля версий;
- VS Code, Jupyter Lab с Jupyter Notebook'ами для написания кода.

Используемые технологии разработки:

- Основной язык программирования - Python;
- Основной фреймворк для глубокого обучения - PyTorch;
- Скрипты на bash;
- Конфигурационные файлы в формате YAML;
- Описание хода решения, идей, гипотез и ведение журнала в MARKDOWN файлах;
- Хранение табличных данных в CSV;
- Библиотеки pandas, numpy, sklearn, albumentations, ttach, opencv-python, tqdm, PIL, multiprocessing и другие.

Заключение

В ходе выполнения работы была изучена информация о меланоме, поражениях кожи и о других связанных с данной областью вопросах. Далее с помощью полученных знаний были подобраны корректные аугментации и преобразования данных, отражающие ситуации из реального мира для обучения нейросети. Также был проведён ряд экспериментов, связанных с представлением данных, разными архитектурами нейросетей, выбором правильным подходов и алгоритмов оптимизации и их гиперпараметров.

В данном проекте была открыта дорога к исследованию таких сложных тем, как экстремально несбалансированные классы, калибровка уверенности модели, pseudo-labeling, label smoothing и другие. Поэтому основным достижением работы можно считать исследование крайне малоизученной области и её актуальность в рамках текущего развития нейронных сетей и появления новых задач, которые им нужно будет устойчиво решать.

Все наработки и кодовая база для воспроизведения результатов могут быть найдены в репозитории проекта на GitHub <https://github.com/detkov/LSoPLD>.

Список использованных источников

1. Описание меланомы. URL:<https://www.kaggle.com/c/siim-isic-melanoma-classification>.
2. ROC кривая. URL:https://en.wikipedia.org/wiki/Receiver_operating_characteristic.
3. Матрица ошибок. URL:https://en.wikipedia.org/wiki/Confusion_matrix.
4. Диаграмма размаха. URL:https://en.wikipedia.org/wiki/Box_plot.
5. Сигналы и признаки рака кожи меланома. URL:<https://www.cancer.org/cancer/melanoma-skin-cancer/detection-diagnosis-staging/signs-and-symptoms.html>.
6. Кросс-валидация. URL:[https://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](https://en.wikipedia.org/wiki/Cross-validation_(statistics)).
7. Mingxing Tan, Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv preprint arXiv:1905.11946, 2019.
8. Датасет ImageNet. URL:<http://www.image-net.org/>.
9. Qizhe Xie, Minh-Thang Luong, Eduard Hovy, Quoc V. Le. Self-training with Noisy Student improves ImageNet classification. arXiv preprint arXiv:1911.04252, 2019.
10. Бинарная кросс-энтропия и log-loss. URL:<https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a>.
11. Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980, 2014.
12. Ilya Loshchilov, Frank Hutter. Decoupled Weight Decay Regularization. arXiv preprint arXiv:1711.05101, 2017.
13. Пакет PyTorch , реализующий различные оптимизационные алгоритмы. URL:<https://pytorch.org/docs/stable/optim.html>.
14. Kernel Density Estimation. URL: https://en.wikipedia.org/wiki/Kernel_density_estimation.
15. Lee, Dong-Hyun. (2013). Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. ICML 2013 Workshop : Challenges in Representation Learning (WREPL).
16. Longlong Jing, Yingli Tian. Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey. arXiv preprint arXiv:1902.06162, 2019.
17. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. arXiv preprint arXiv:1512.00567, 2015.

18. Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, Geoffrey Hinton. Regularizing Neural Networks by Penalizing Confident Output Distributions. arXiv preprint arXiv:1701.06548, 2017.
19. Rafael Müller, Simon Kornblith, Geoffrey Hinton. When Does Label Smoothing Help? arXiv preprint arXiv:1906.02629, 2019.

Приложения

Метод выравнивания распределения таргета	% используемых данных	№ функции смягчения	Обучение только последнего слоя	Hold-Out AUC-ROC
rdf_bnd	50	1	true	0.855
rnd_undspml	50	4	true	0.8541
rnd_undspml	50	5	true	0.8539
str_undspml	50	5	true	0.8538
rnd_undspml	50	3	true	0.8538
str_undspml	50	4	true	0.8536
str_undspml	80	4	true	0.8535
str_undspml	50	3	true	0.8535
rnd_undspml	50	6	true	0.8533
str_undspml	50	2	true	0.8533
rnd_undspml	50	1	true	0.8533
rnd_undspml	50	2	true	0.8532
str_undspml	50	6	true	0.8532
str_undspml	80	3	true	0.853
str_undspml	80	1	true	0.851
rdf_bnd	80	4	true	0.8507
str_undspml	80	6	true	0.8507
rdf_bnd	50	4	true	0.8505
str_undspml	80	5	true	0.8504
rdf_bnd	50	3	true	0.8504
str_undspml	50	1	true	0.8504
str_undspml	80	2	true	0.85
rdf_bnd	80	3	true	0.85
rnd_undspml	80	4	true	0.8499
rnd_undspml	80	3	true	0.8496
rdf_bnd	50	5	true	0.8492
rnd_undspml	80	5	true	0.8492
rnd_undspml	80	2	true	0.8483
rdf_bnd	50	2	true	0.8478
rdf_bnd	80	1	true	0.8474
rnd_undspml	80	6	true	0.8473
rdf_bnd	80	5	true	0.8453
rdf_bnd	80	2	true	0.8439
rdf_bnd	50	6	true	0.8437
rnd_undspml	80	1	true	0.8424
rdf_bnd	80	6	true	0.8405
rnd_undspml	100	1	true	0.8387
rdf_bnd	100	1	true	0.8313

rdf_bnd	100	3	true	0.8309
rdf_bnd	100	4	true	0.8307
				0.8263
str_undspml	100	1	true	0.8259
rdf_bnd	80	5	false	0.8215
rdf_bnd	100	5	true	0.82
rdf_bnd	50	4	false	0.8153
rdf_bnd	100	2	true	0.8129
rnd_undspml	50	2	false	0.8105
rdf_bnd	80	1	false	0.8081
rdf_bnd	80	4	false	0.8078
rdf_bnd	50	5	false	0.8034
rdf_bnd	50	3	false	0.803
str_undspml	80	4	false	0.8027
rdf_bnd	80	2	false	0.8021
str_undspml	80	1	false	0.8015
rdf_bnd	80	6	false	0.7998
rdf_bnd	50	2	false	0.7995
rnd_undspml	50	1	false	0.7987
rdf_bnd	50	1	false	0.7987
rdf_bnd	100	2	false	0.7979
rnd_undspml	100	4	true	0.7968
rdf_bnd	50	6	false	0.7961
rdf_bnd	100	6	true	0.796
rnd_undspml	100	3	true	0.7938
rdf_bnd	100	3	false	0.7917
rdf_bnd	100	5	false	0.7901
rnd_undspml	80	2	false	0.79
rnd_undspml	50	6	false	0.7885
rnd_undspml	100	6	true	0.7883
str_undspml	50	4	false	0.7879
str_undspml	50	2	false	0.7874
str_undspml	80	3	false	0.7867
str_undspml	50	3	false	0.7859
rnd_undspml	100	5	true	0.785
rnd_undspml	80	1	false	0.7837
rdf_bnd	80	3	false	0.7835
str_undspml	80	6	false	0.7832
str_undspml	50	5	false	0.7825
rnd_undspml	50	3	false	0.7796
rnd_undspml	100	2	true	0.7791
str_undspml	100	4	true	0.7778
rnd_undspml	50	5	false	0.7773
rnd_undspml	100	2	false	0.7771
str_undspml	100	5	false	0.7762
str_undspml	100	3	true	0.7759
str_undspml	100	6	true	0.7748
str_undspml	100	3	false	0.7748

rnd_undspml	80	3	false	0.7728
str_undspml	100	5	true	0.7726
str_undspml	80	2	false	0.7711
str_undspml	100	2	true	0.7711
rnd_undspml	50	4	false	0.7696
rdf_bnd	100	4	false	0.7678
rnd_undspml	100	4	false	0.7678
rnd_undspml	80	5	false	0.7665
str_undspml	100	2	false	0.7663
str_undspml	100	4	false	0.7662
str_undspml	50	6	false	0.7623
str_undspml	80	5	false	0.7623
rnd_undspml	80	6	false	0.7612
str_undspml	50	1	false	0.7267
rnd_undspml	100	5	false	0.7221
rdf_bnd	100	6	false	0.6946
rnd_undspml	100	3	false	0.6907
rnd_undspml	80	4	false	0.6548
rnd_undspml	100	6	false	0.6426
str_undspml	100	6	false	0.5725

Таблица Приложения.1 – полная таблица с результатами экспериментов, где в колонке «Метод выравнивания распределения таргета» значение «rdf_bnd» – это метод со сдвигом границы присвоения метки класса, «rnd_undspml» – метод случайного under-sampling'a, «str_undspml» – метод стратифицированного under-sampling'a, а «№ функции смягчения» соответствует порядковому номеру появлению функций в тексте работы. Стока только со значением метрики показывает исходный эксперимент.