МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра биомедицинской информатики

ИССЛЕДОВАНИЕ СВЯЗИ СВОЙСТВ СГЕНЕРИРОВАННЫХ МЕДИЦИНСКИХ ИЗОБРАЖЕНИЙ И ХАРАКТЕРИСТИК ВХОДНОГО СЛУЧАЙНОГО ВЕКТОРА

Курсовая работа

Синявского Тимура Владимировича

студента 3 курса

специальность «информатика»

Научный руководитель:

доцент кафедры БМИ

Ковалёв В.А.

Минск

2020

Оглавление

Часть 1. Постобработка. 3
Глава 1. Постановка задачи. 3
1.1 Генеративно-состязательные сети. 3
1.2 Формулировка задачи. 5
Глава 2. Решение задачи с помощью сегментации. 6
2.1 Задача сегментации 6
2.2 Применение сегментации к поставленной задаче. 8
2.3 Формирование признаков для отсеивания негодных снимков. 8
2.4 Дерево решений и логистическая регрессия 9
2.5 Построение модели для отсеивания негодных снимков. 10
2.6 Итоговый пайплайн отбора снимков. 11
Глава 3. Другой подход к решению задачи 12
3.1 Transfer learning. 12
3.2 Grad-CAM. 13
3.3 Применение transfer learning и Grad-CAM для детектирования. 14
Заключение. 15
Часть 2. Предобработка. 16
1. Описание данных. 16
2. Предварительный анализ данных. 17
3. Анализ порогов классификации. 19
4. Модели классификации 20
5. Выбор порога классификации. 24
6. Время генерации снимков. 25
Заключение. 25
Список использованных источников 26
Приложения 27

Часть 1. Постобработка.

Глава 1. Постановка задачи.

1.1 Генеративно-состязательные сети.

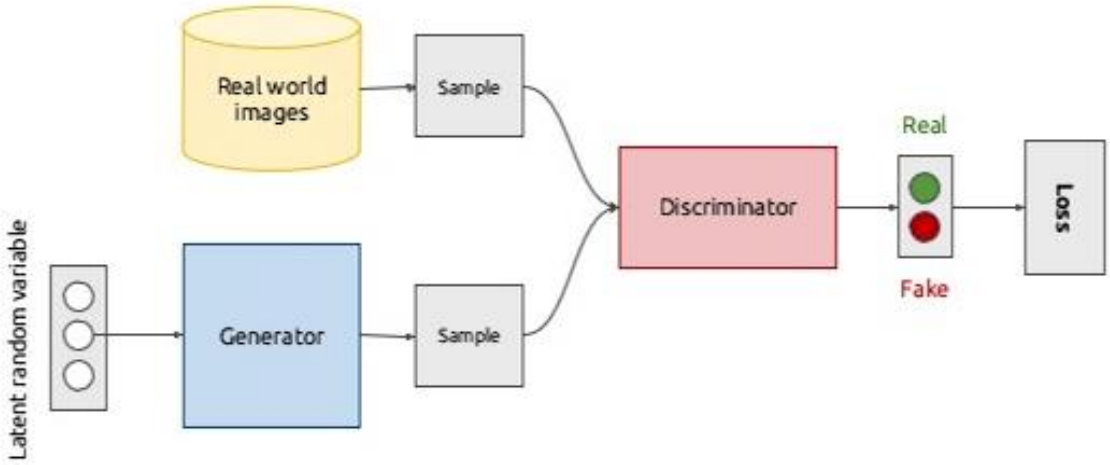
Генеративно-состязательная сеть (Generative Adversarial Network, GAN) [1] – это алгоритм глубокого обучения без учителя. GAN применяется для генерации изображений, музыки, речи, видео, трёхмерных моделей и др.

Структура GAN.

GAN состоит из двух взаимодействующих нейронных сетей:

Генератор (generator) – принимает на вход случайный элемент латентного пространства и генерирует образец. Цель генератора – научиться создавать образцы, похожие на образцы из обучающей выборки (но не повторяющие их) и достаточно реалистичные для того, чтобы «обмануть» сеть-дискриминатор. Таким образом, задача генератора – не копировать уже существующие образцы, а создавать изображения с распределением настолько близким к распределению исходных данных, что дискриминатор не сможет их различить.

Дискриминатор (discriminator) – принимает на вход образец и оценивает реалистичность относительно не сгенерированных образцов. Цель дискриминатора – научиться отличать синтезированные генератором образцы от настоящих образцов из обучающей выборки. Другими словами, дискриминатор решает задачу бинарной классификации для поступающего ему на вход образца: определяет, является ли он «подлинным» или сгенерированным. При этом дискриминатор адаптируется к тому, что генератор создаёт всё более реалистичные образцы.



1 GAN architecture.

Обучение GAN.

Принцип работы GAN иногда описывается с помощью метафор: первая сеть представляется как фальсификатор, а вторая – как эксперт, который сравнивает подделки с подлинником и указывает фальсификатору, какие детали в его работе реалистичны, а какие нет. Со временем фальсификатор учится создавать всё более реалистичные подделки, а эксперт становится всё более опытным в различении подделок. Генератор и дискриминатор соревнуются друг с другом, то есть между ними возникает антагонистическая игра.

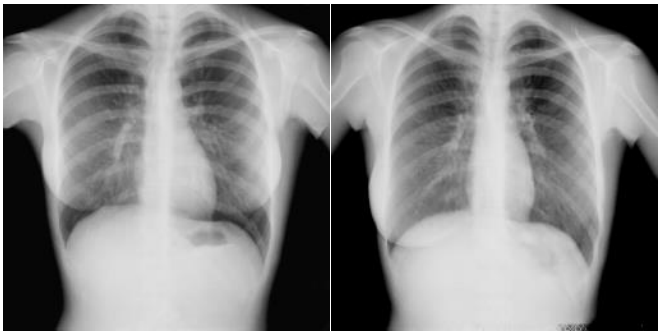
Генератор получает случайные вектора и генерирует образцы. Далее эти образцы наряду с реальными образцами из обучающей выборки подаются дискриминатору. А он возвращает вероятности принадлежности образцов к классам “сгенерированных” и “реальных”. Исходя из ошибок дискриминатора генератор модифицируется так, чтобы дискриминатор не смог отличить реальные образцы от сгенерированных. А после дискриминатор модифицируется так, чтобы снова отличать сгенерированное. То есть генератор и дискриминатор обучаются отдельно, но в рамках одной сети. Таким образом целью генератора является повысить процент ошибок дискриминатора, а целью дискриминатора является наоборот улучшение точности распознавания. Постепенно искусственные образцы на выходе генератора становятся всё более качественными.

На сегодняшний день GANы сильно развиваются, появляется много модификаций [2]. Одно из последних многочисленных достижений: StyleGAN [3] генерирует фотографии несуществующих людей в высоком разрешении [4].

1.2 Формулировка задачи.

Дана модель GAN, которая генерирует рентгеновские снимки грудной клетки мужчин и женщин размера 256 на 256 пикселей. Но женские снимки иногда генерируются несимметричные в области груди. Цель курсового проекта – научиться отсеивать такие снимки.

Пример хорошей (слева) и плохой (справа) генерации:



Глава 2. Решение задачи с помощью сегментации.

2.1 Задача сегментации.

Сегментация - это процесс разделения изображения на несколько сегментов (множество пикселей, также называемых суперпикселями). Альтернативное определение сегментации: процесс присвоения меток каждому пикселю изображения таким образом, что пиксели с одинаковыми метками имеют общие визуальные характеристики. Цель сегментации заключается в упрощении изображения, чтобы его было легче анализировать. Сегментация изображений обычно используется для того, чтобы выделить объекты и границы на изображениях. Результатом сегментации изображения является множество сегментов, которые вместе покрывают всё изображение, или множество контуров, выделенных из изображения.

Существуют следующие алгоритмы сегментации [5]:

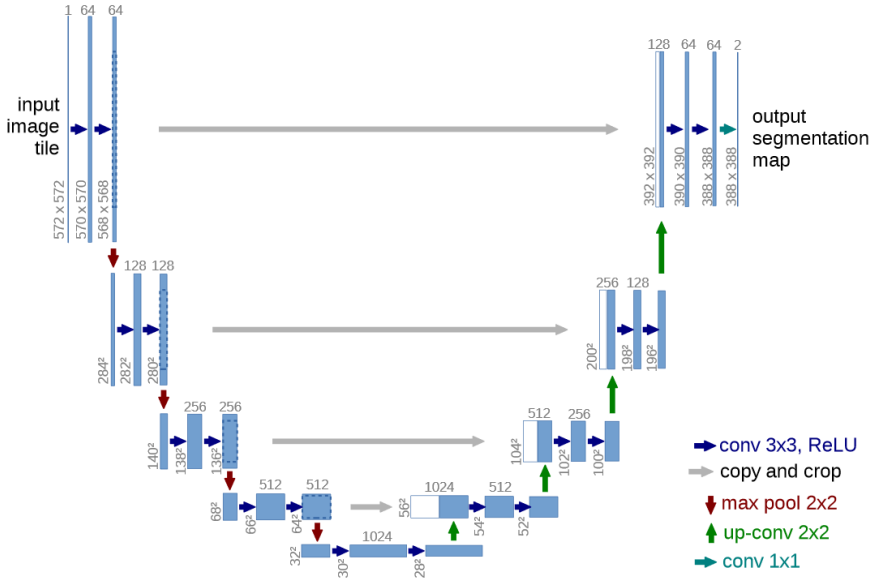
- Сегментация по регионам (пороговая сегментация, сегментация регионального роста) - разделяет объекты на разные регионы на основе некоторых пороговых значений.

- Сегментация с обнаружением краев (операторы Собеля и Лапласа) - использует прерывистые локальные элементы изображения для обнаружения краев и, следовательно, определения границы объекта.

- Сегментация на основе кластеризации - делит пиксели изображения на однородные кластеры.

- Сегментация с использованием нейронных сетей.

U-net [6] – это одна из нейросетевых архитектур используемых для сегментации. Впервые данная архитектура была применена к сегментации медицинских изображений, но позже так же распространилась и на другие виды изображений.



2 U-net architecture.

U-net состоит из двух частей. Первая часть – кодирующая часть (encoder). Она используется для захвата контекста изображения. Обычно кодировщик состоит из последовательности свёрточных слоёв и пулингов. Вторая часть сети – декодирующая часть (decoder). Декодер чаще всего симметричен кодировщику и обеспечивает точную локализацию на выходе. Таким образом U-net – это полностью свёрточная сеть (Fully Convolutional Network, FCN), то есть она содержит только свёрточные слои и не содержит полносвязных слоёв. Если задача – выделение одного объекта, то есть сегментация на два сегмента: “искомый объект” и “другое”, то на выходе модель каждому пикселю сопоставляет вероятность принадлежности к сегменту “искомый объект”. Так как задачу можно переформулировать в задачу классификации каждого пикселя, при обучении в качестве функции ошибки чаще всего используется бинарная кросс энтропия (Binary Cross Entropy, BCE)

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\big(p(y_i)\big) + (1 - y_i) \cdot \log\,(1 - p(y_i))$$

Метрики для оценки модели сегментации:

- Попиксельная точность – доля верно предсказанных пикселей.

- Intersection over union (IoU, коэффициент Жаккара):

$$IoU = \frac{S_{pred} \cap truth}{S_{pred} \cup truth}$$

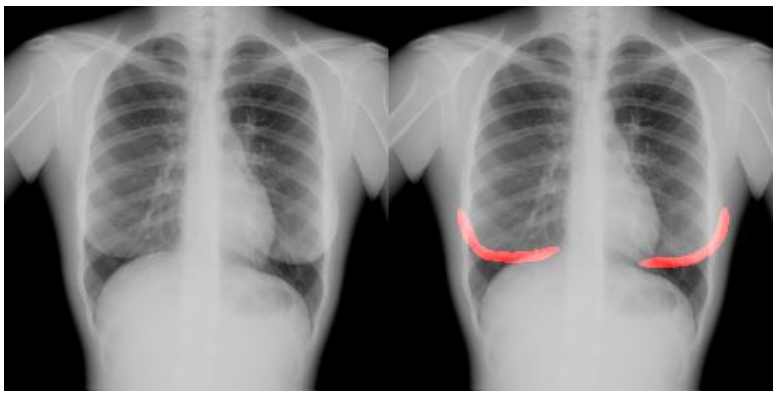
- Dice coefficient (F1 score, коэффициент Сёренсена):

$$DC = \frac{2 * S_{pred \cap truth}}{S_{pred} + S_{truth}}$$

2.2 Применение сегментации к поставленной задаче.

Для дальнейшего отсеивания снимков была решена задача сегментации женской груди на рентген снимке. Для решения использовалась модель U-net. Для обучения U-net было размечено порядка трехсот сгенерированных женских снимков. На вход модель принимает снимок размеронсти 1x256x256 (1 – размерность калала свёртки, так как снимок подаётся в чёрно-белом формате), значения пикселей переведены в диапазон от нуля до единицы. На выходе получается тензор той же размерности. Каждое значие тензора - вероятность принадлежности соответствующего пикселя к маске. Итоговая маска (состоящая из нулей и единиц) получается бинаризацией по порогу равному 0.5.

Пример результата работы модели (слева вход, справа полученная маска, наложенная на исходный снимок):



2.3 Формирование признаков для отсеивания негодных снимков.

Признаки формируются на основе масок сегментации следующим образом:

1) Каждая маска разбивается на две равные части относительно центральной вертикальной оси снимка.
2) Правая половина отражается относительно вертикальной оси.
3) К каждой части применяется морфологическая операция dilate (операция увеличивает светлые области).
4) Считаются следующие коэффициенты (признаки) как для исходных половин, так и для половин после операции dilate:

$$IoU = \frac{S_{left \cap right}}{S_{left \cup right}}$$

$$\cos = \frac{left \cdot right}{\|left\| \, \|right\|}$$

$$area\ ratio = \left(\frac{S_{left}}{S_{left} + S_{right}} - \frac{1}{2}\right)^2$$

Таким образом для каждого снимка получим шесть признаков. Интуитивный смысл первых двух признаков – мера похожести левой и правой части маски, а значит и мера симметрии исходного снимка. Третий признак – наоборот.

2.4 Дерево решений и логистическая регрессия.

Дерево решений классифицирует данное наблюдение при помощи последовательности вопросов, в которой следующий заданный вопрос зависит от ответа на текущий. Структура дерева представляет собой «листья» и «ветки». На рёбрах («ветках») дерева решения записаны атрибуты, от которых зависит целевая функция, в «листьях» записаны значения целевой функции, а в остальных узлах - атрибуты, по которым различаются случаи. Чтобы классифицировать новый случай, надо спуститься по дереву до листа и выдать соответствующее значение.

В основе популярных алгоритмов построения дерева решений, лежит принцип жадной максимизации прироста информации - на каждом шаге выбирается тот признак, при разделении по которому прирост информации оказывается наибольшим. Дальше процедура повторяется рекурсивно, пока энтропия не окажется равной нулю или какой-то малой величине.

Плюсы дерева решений:

- Интерпретируемость.

- Быстрый процесс обучения и прогнозирования.

- Малое число гиперпараметров.

Минусы:

- Чувствительны к шумам во входных данных.

- Нестабильность - небольшие изменения в данных могут существенно изменять построенное дерево решений.

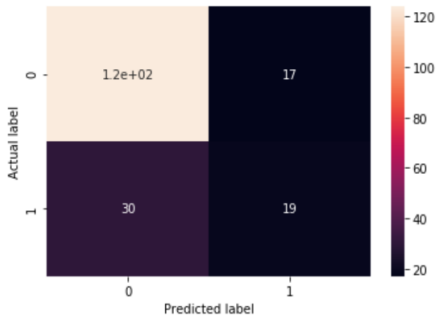
Логистическая регрессия – это статистическая модель, используемая для предсказания вероятности возникновения некоторого события путём подгонки данных к логистической кривой. Основная идея заключается в том, что

признаковое пространство может быть разделено гиперплоскостью на два полупространства, в каждом из которых прогнозируется одно из двух значений целевого класса. Логистическая регрессия выдает вероятности отнесения к разным классам. Плюсы логистической регрессии такие же, как и у дерева решений. Минус: плохо работает в задачах, в которых зависимость ответов от признаков нелинейная.

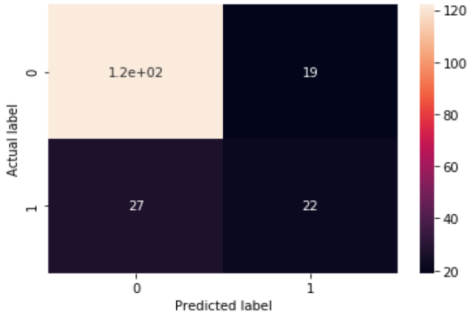
2.5 Построение модели для отсеивания негодных снимков.

Отсеивание плохих (несимметричных) снимков рассмотрим как задачу классификации: “хороший” или “плохой” снимок. Для решения были рассмотрены логистическая регрессия и дерево решений. На вход модели подаются признаки из п. 2.3. Для обучения этих алгоритмов было размечено 1000 снимков. Параметры моделей подбирались на кросс-валидации. Результаты:

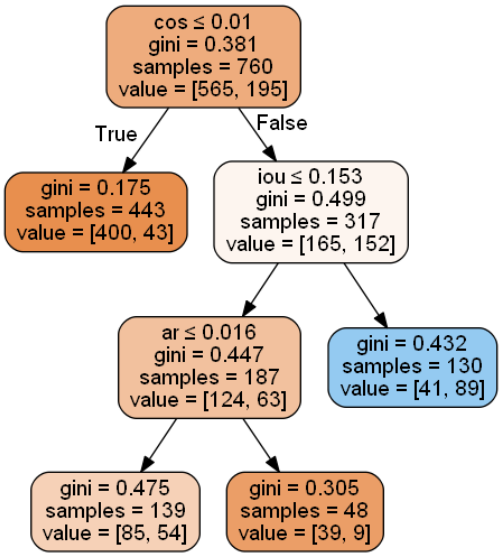
```
DecisionTreeClassifier - score_train: 0.817, score_test: 0.753
Test confusion matrix:
```




```
LogisticRegression - score_train: 0.807, score_test: 0.758
Test confusion matrix:
```



Визуализация обученного дерева решений (оранжевые узлы – плохие снимки, синие - хорошие):



Видно, что задача классификации решается не очень хорошо. От сюда следует, что признаков, полученных из масок сегментации, недостаточно. Но тем

не менее исходное соотношение плохих снимков к хорошим почти 3 к 1, а после отсеивания - 1 к 1.

2.6 Итоговый пайплайн отбора снимков.

1) Снимки прогоняются через модель сегментации U-net и получаются бинарные маски.

2) Каждой маске в соответствие ставятся признаки симметрии.

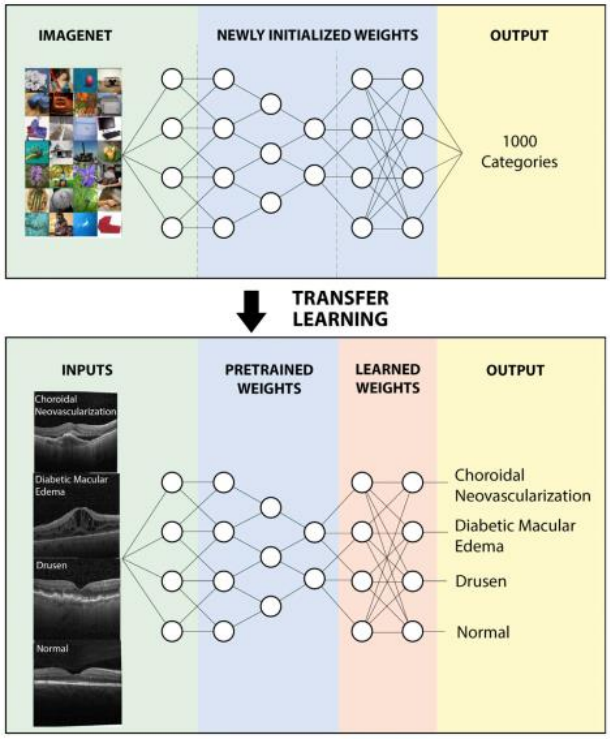
3) К признакам применяется классифицирующая модель, которая принимает решение о пригодности снимка.

Глава 3. Другой подход к решению задачи.

3.1 Transfer learning.

Transfer learning – это метод машинного обучения, в котором модель, разработанная для определенной задачи, используется как база для построения модели для другой задачи. В глубоком обучении предварительно обученные нейронный сети часто используются в компьютерном зрении и обработке языка. Это помогает сэкономить большие вычислительные и временные ресурсы для разработки моделей для этих задач с нуля. Так же это позволяет быстрее улучшать качество модели на поставленной задаче путем передачи знаний из связанной задачи, которая уже изучена. Но этот метод работает только если признаки, изученные в базовой задаче, являются достаточно обобщёнными, то есть подходящими как для базовой, так и для целевой задачи, а не специфичны для базовой задачи.

В компьютерном зрении одна из наиболее распространенных базовых задач - это классификация изображений. ImageNet - набор данных изображений с метками от 1 до 1000 (каждая метка – класс, например человек, машина и т.п.). Нейросетевые модели, обученные на этом датасете, часто используются для transfer learning. Примеры таких моделей: ResNet [7], VGG, Inception. Они обучаются днями/неделями на мощном оборудовании, что не всегда доступно обычному разработчику. Эти модели могут быть использованы как базовые для моделей, которые так же принимают на вход изображения. Признаки на ранних слоях сети более общие (например, контуры, формы и т.п.), а на поздних – более специфичные для базовой задачи. Поэтому зачастую при дообучении на новую задачу ранние слои базовой сети замораживаются (т.е. веса в этих слоях не обновляются при обучении), а поздние обновляются или вовсе инициализируются случайными значениями. Этот процесс называется fine-tuning.



В обработке языка используются эмбединги, которые отображают слова в многомерное векторное пространство. Примеры таких моделей: word2vec и GloVe.

Таким образом transfer learning даёт следующие преимущества:

- Результат получатся лучше.

- Существенно сокращается время, которое необходимо потратить на обучение модели.

- Позволяет использовать меньше данных для целевой задачи.

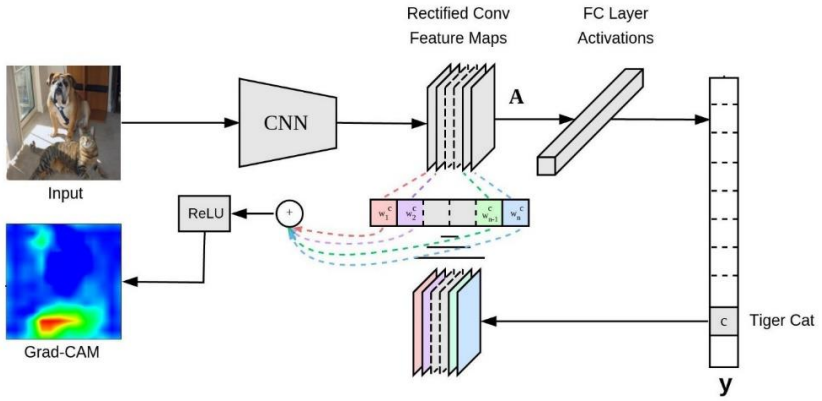
Но это выполняется при условии, что признаки, выученные на базовой задаче, применимы к поставленной задаче.

3.2 Grad-CAM.

Gradient-weighted Class Activation Mapping (Grad-CAM) [8] – алгоритм интерпретации решений свёрточных нейронных сетей за счёт выделения областей входного изображения, которые “важны” при формирования моделью ответа. Выделение получается в виде тепло карт (heat-map, localization map).

Результат получается следующим образом. Сначала считаются производные выходного нейрона требуемого класса y^c (c – индекс класса) по значениям feature map A_{ij}^k (обычно это выход последнего свёрточного слоя, имеем K карт, поэтому $k = \overline{1, K}$). Получаем K матриц производных такой же размерности как и A^k. Для каждой такой матрица считаем среднее значение её элементов и получаем веса α_k^c. Далее берем линейную комбинацию A^k с весами α_k^c и применяем функцию $ReLU(x) = \max(0, x)$. Получаем одну матрицу L^c, которую растягиваем до размеров входного изображения с помощью интерполяции или других методов и накладывает на входное изображение.

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}} \qquad L_{\text{Grad-CAM}}^c = ReLU\underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$



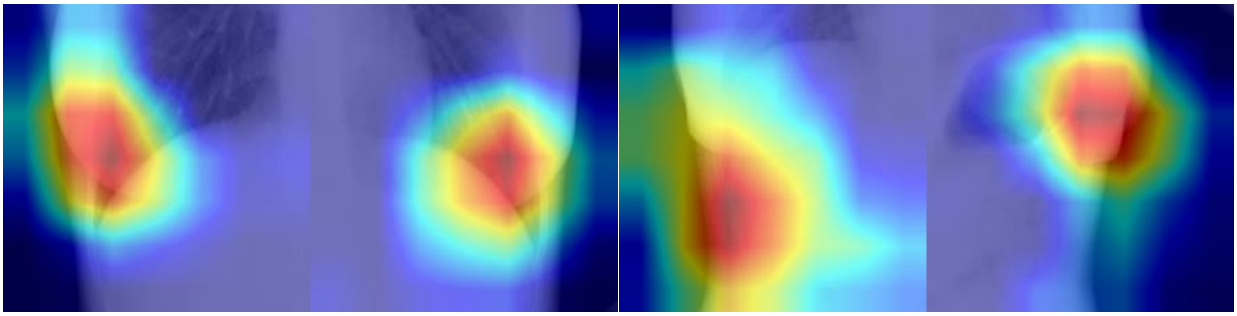
Работу нейронных сетей часто ассоциируют с черным ящиком. Данный метод является очень важным для понимания решения алгоритма, так как позволяет проинтерпретировать решение широкого класса свёрточных нейронных сетей (CNN).

3.3 Применение transfer learning и Grad-CAM для детектирования.

Идея заключается в следующем: обучим нейросетевой классификатор для определения пола человека по рентген снимку. А потом посмотрим на что сеть обращает внимание при определении пола методом Grad-CAM.

В качестве классификатора использовалась предобученная на ImageNet модель ResNet-18. Все слои кроме последних были заморожены. А последние полносвязные слои были заменены на один слой с одним выходным нейроном и логистической функцией активации. Сеть выдаёт как ответ единицу, если на снимке женщина, и ноль, если мужчина. При обучении использовалось порядка 200 снимков женщин и столько же мужских снимков. Использовался следующий препроцессинг: снимки обрезались сверху, так как хотелось бы, чтоб сеть делала решение на основе грудной области, а не шеи или плеч. Так же на вход сети подавались левые и правые половины снимков по отдельности, чтобы избежать ответа сети только по одной половине в случае подачи целого снимка. После обучения сеть показала точность 90%.

Применение Grad-CAM к обученной сети дало следующие результаты:



К сожалению, полученные хитмапы получаются довольно зашумлёнными. Например, так как в обучающей выборке на женских снимках иногда фигурировала рука, опёртая на бок, а среди мужских снимков такого не наблюдается, при получении хитмапа алгоритм Grad-CAM с большой вероятностью подсветит руку. Таким образом извлечение информации о положении груди или симметрии рентген снимка затрудняется.

Заключение.

Таким образом были рассмотрены различные подходы к задаче детектирования на рентгенографических изображениях. И был разработан пайплайн на основе детектирования с помощью сегментации, который улучшает генерацию предоставленной генеративно состязательной сети путём отсеивания некачественных изображений по признакам асимметрии детектированных областей.

Часть 2. Предобработка.

В первой части были рассмотрены методы получения признаков для улучшения качества фильтрации изображений после генерации. Наряду с этими признаками есть и другие. Идея следующей работы заключается в том, чтобы предсказать эти признаки по случайному вектору до генерации изображения и генерировать только заведомо хорошие снимки, тем самым сэкономив время генерации.

1. Описание данных.

Датасет содержит 60000 записей.

Каждая запись содержит информацию об одном сгенерированном снимке, которая включает в себя:

- group - половозрастная группа (одна из шести возможных: 18_24m, 18_24f, 38_44m, 38_44f, 58_64m, 58_64f). Каждая группа генерируется отдельной моделью GAN.
- valid – прошел (1) или нет (0) фильтрацию снимок. Чтобы пройти фильтрацию снимок должен иметь все положительные признаки, описанные ниже. Далее под словосочетанием хороший снимок, подразумевается, что у него признак valid равен единице.

Признаки для фильтрации:

- score_shape.
- score_abnormal – к снимку применяется нейронная сеть для детектирования болезней, которая принимает снимок и возвращает вероятность присутствия аномалии.
- score_maxheat – с помощью нейронной сети из предыдущего пункта получается тепло карта аномалии. Данный признак — это максимальное число на этой тепло карте.
- score_symmetry – dice score относительно левого и правого легких.
- score_left2right – отношение площади левого легкого к правому.

Все величины признаков нормализованы таким образом, что если значение больше нуля, то снимок хороший. То есть порог прохождения фильтрации по отдельному признаку – 0. Если все признаки больше нуля, то снимок проходит фильтрацию и признак valid равен 1, иначе 0.

Так же для каждого снимка содержится случайный вектор размерности 128, из которого было сгенерировано изображение. То есть сгенерированный снимок — это функция от этого случайного вектора представленная в виде генератора модели GAN. А так как признаки фильтрации — это функция от изображения, то они так же являются функцией от исходного случайного вектора. Таким образом задача сводится к аппроксимации функции признаков по случайному вектору.

2. Предварительный анализ данных.

Статистика по признакам фильтрации:

	valid	score_shape	score_abnormal	score_maxheat	score_symmetry	score_left2right
count	60000.000000	60000.000000	60000.000000	60000.000000	60000.000000	60000.000000
mean	0.043317	-0.163849	0.003217	0.170201	-1.441515	-0.299450
std	0.203571	0.221203	0.647530	0.590855	2.381669	10.611759
min	0.000000	-1.496000	-1.000000	-1.000000	-37.000000	-2582.800000
25%	0.000000	-0.302000	-0.620000	-0.199000	-1.902000	-0.702250
50%	0.000000	-0.142000	0.058000	0.359000	-0.732000	0.041000
75%	0.000000	-0.004000	0.620000	0.654000	0.032000	0.552000
max	1.000000	0.489000	0.988000	0.999000	1.000000	1.000000

Корреляция между признаками:

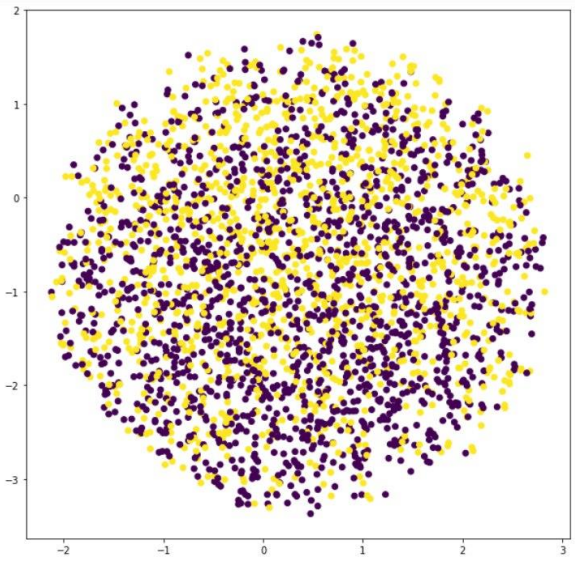
	valid	score_shape	score_abnormal	score_maxheat	score_symmetry	score_left2right
valid	1.000000	0.259087	0.216339	0.152048	0.170441	0.017922
score_shape	0.259087	1.000000	0.321058	0.280105	0.271647	0.038377
score_abnormal	0.216339	0.321058	1.000000	0.567984	0.240788	0.022292
score_maxheat	0.152048	0.280105	0.567984	1.000000	0.093406	0.003736
score_symmetry	0.170441	0.271647	0.240788	0.093406	1.000000	0.059781
score_left2right	0.017922	0.038377	0.022292	0.003736	0.059781	1.000000

Процентное соотношение хороших и плохих изображений по группам:

```
group   valid
18_24f  0        0.9397
        1        0.0603
18_24m  0        0.9306
        1        0.0694
38_44f  0        0.9637
        1        0.0363
38_44m  0        0.9678
        1        0.0322
58_64f  0        0.9589
        1        0.0411
58_64m  0        0.9794
        1        0.0206
```

Наблюдается сильный дисбаланс данных, то есть плохих снимков гораздо больше, чем хороших. В этом и заключается проблема: для генерации достаточного количества хороших снимков, требуется предварительная генерация очень большого количества снимков и потом их фильтрация. Например, для генерации 100 хороших снимков, надо сначала сгенерировать приблизительно 2000, что является ресурсозатратно.

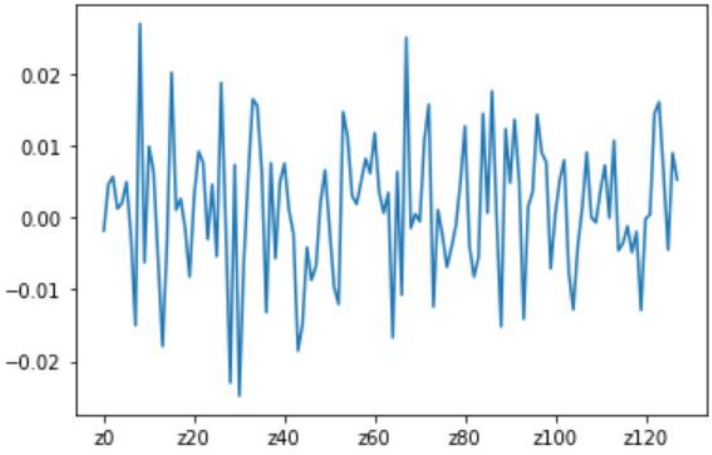
Отсортируем все данные по определенному признаку, затем выберем первые k записей и последние k записей. Таким образом получим “самые хорошие и самые плохие” записи по выбранному признаку. Возьмём случайные вектора этих записей и применим алгоритм снижения размерности (UMAP) с выходной размерностью равной двум. Отрисуем получившиеся вектора, раскрасив в два цвета плохие и хорошие снимки. Получится следующий график:



Видно, что плохие и хорошие снимки сильно перемешаны. Похожая картина наблюдалась для всех признаков и половозрастных групп. Стоит отметить, что в данном случае применение алгоритмов снижения размерности отрабатывает не лучшим образом, так как данные получены из многомерного равномерного распределения.

Возникло предположение, что в окрестности случайного вектора, из которого генерируется хорошее изображение, будет находиться также хороший для генерации вектор. Тогда хорошо бы отработал kNN с k=1, но это опроверглось, классификация такой моделью отработала плохо.

Корреляция между одним из признаков и компонентами случайного вектора:



Видно, что линейной зависимости нет.

Ввиду сложности зависимости признаков от исходного вектора, упростим задачу и не будем строить регрессию. Будем решать задачу классификации: больше или меньше нуля, то есть прошло ли изображение фильтрацию по признаку или нет.

3. Анализ порогов классификации.

Матрица ошибок (confusion matrix) — это квадратная матрица, которая позволяет визуализировать результаты работы алгоритма классификации. Каждая строка матрицы представляет экземпляры размеченных классов, в то время как каждый столбец представляет экземпляры предсказанных моделью классов.

TN – true negative (количество верно предсказанных плохих снимков),

FP – false positive (количество плохих снимков, которые предсказаны как хорошие),

FN – false negative (количество хороших снимков, которые предсказаны как плохие),

TP – true positive (количество верно предсказанных хороших снимков).

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

Величины, связанные с матрицей ошибок:

$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$, доля годных снимков после фильтрации.

$\text{True positive rate } (TPR, \text{recall}) = \frac{TP}{TP+FN}$, доля хороших снимков, допущенных фильтрацией, от всех хороших.

$False\ positive\ rate\ (FPR) = \frac{FP}{FP+TN}$, доля плохих снимков, допущенных фильтрацией, от всех плохих.

Кривая ошибок (ROC, Receiver Operating Characteristic curve) – график позволяющий визуализировать модель бинарной классификации путём варьирования порога классификации. Данный метод работает только для моделей, которые выдают вероятность принадлежности к классу, а не дискретный ответ принадлежит или нет. График строится следующим образом: выбираются все возможные пороги классификации, затем для каждого порога строится матрица ошибок и находятся TPR и FPR. Таким образом для каждого порога получим точку с двумя значениями. Отобразив все точки на плоскости получим неубывающую кривую.

Аналогичным образом строится Precision-Recall curve.

Данные графики используются для выбора порога классификации, наиболее подходящего под задачу. Например, при определении болезни лучше выявить всех больных и случайно классифицировать здорового, как больного, чем хорошо классифицировать здоровых и ошибаться на больных. В этом примере Precision и FPR могут быть низкими, но TPR должен быть высоким.

В рассматриваемой задаче фильтрации снимков хотелось бы повысить Precision и TPR, а FPR понизить.

4. Модели классификации.

Были рассмотрены как классические модели классификации машинного обучения, так и модели нейронных сетей. Все модели получают на вход случайный вектор размерности 128 и предсказывают вероятность того, что один из описанных признаков фильтрации больше нуля. При обучении моделей использовалась схема кроссвалидации с тремя разбиениями.

Использовались следующие классические модели:

- Метод k ближайших соседей (kNN) с k равным 1, 3, 10 и 30.

- Метод опорных векторов (SVC) с линейным, гауссовым и полиномиальным ядрами.

- Логистическая регрессия (LogReg) со значениями L2 регуляризации равными 1e-2, 1e-1, 1e0 и 1e1.

- Решающее дерево с максимальной глубиной 5 и 10

- Случайный лес.

Результаты экспериментов с различными моделями для признака valid:

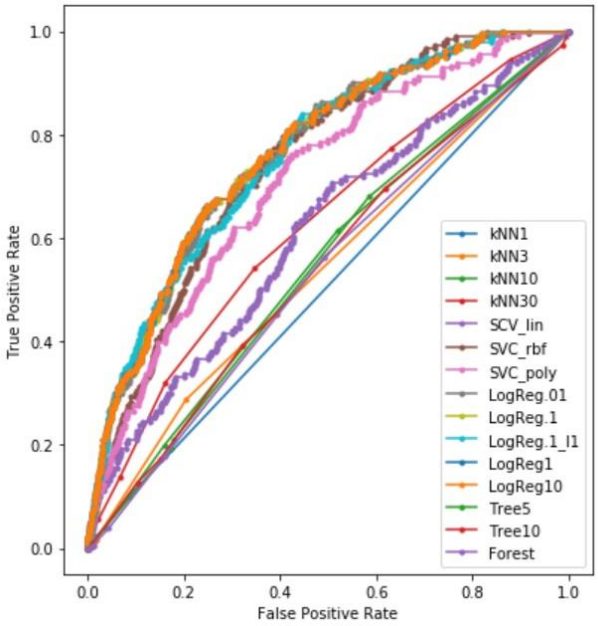
valid - precision

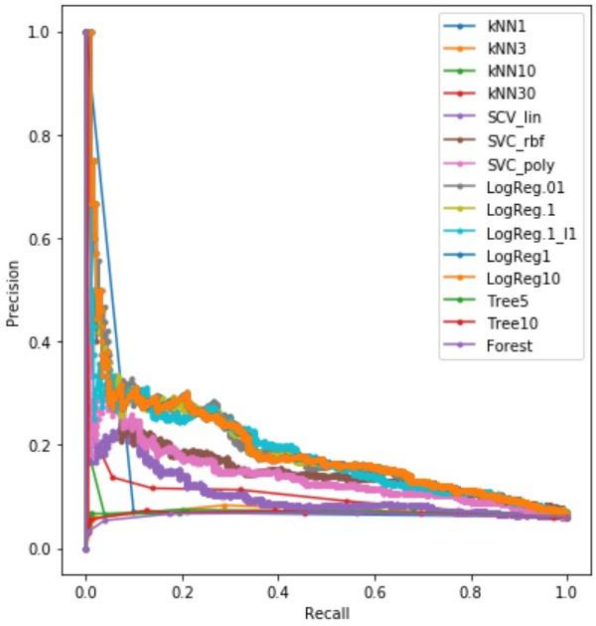
	18_24m	18_24f	38_44m	38_44f	58_64m	58_64f
kNN1	0.0989172	0.0595212	0.0639862	0.0733601	0.0247663	0.0755616
kNN3	0.12373	0.0830179	0.0938645	0.0682143	0	0.115618
kNN10	0	0	0	0	0	0
kNN30	0	0	0	0	0	0
SCV_lin	0	0	0	0	0	0
SVC_rbf	0	0	0	0	0	0
SVC_poly	0	0	0	0	0	0
LogReg.01	0.2	0	0	0	0	0
LogReg.1	0.4	0.44	0.2	0.1	0	0.1
LogReg.1_l1	0.2	0.2	0	0	0	0
LogReg1	0.344444	0.4	0.333333	0.09	0.05	0.316667
LogReg10	0.33873	0.378968	0.357143	0.107143	0.08	0.282857
Tree5	0.0869048	0.02	0.0533333	0	0	0.025
Tree10	0.0861738	0.0316585	0.0385714	0.0299842	0	0.0176923
Forest	0	0	0	0	0	0

valid - roc_auc

	18_24m	18_24f	38_44m	38_44f	58_64m	58_64f
kNN1	0.518646	0.499903	0.519762	0.522774	0.502834	0.520649
kNN3	0.529414	0.526228	0.528797	0.522826	0.499909	0.5358
kNN10	0.569038	0.564511	0.561761	0.539981	0.522257	0.577312
kNN30	0.622291	0.61825	0.581953	0.602178	0.588318	0.645839
SCV_lin	0.562919	0.611376	0.658443	0.604752	0.680804	0.60397
SVC_rbf	0.732893	0.73577	0.736201	0.768364	0.751652	0.754788
SVC_poly	0.693439	0.695288	0.698326	0.708741	0.668986	0.716538
LogReg.01	0.7333	0.756979	0.763791	0.779645	0.773556	0.763873
LogReg.1	0.732599	0.757519	0.764218	0.781131	0.778616	0.76225
LogReg.1_l1	0.729201	0.751877	0.749135	0.757719	0.732014	0.751761
LogReg1	0.732223	0.757425	0.76342	0.781091	0.776594	0.76105
LogReg10	0.732178	0.75741	0.763345	0.781033	0.776302	0.760839
Tree5	0.559802	0.539724	0.527501	0.571854	0.509806	0.512334
Tree10	0.551249	0.535545	0.53975	0.571306	0.525519	0.56282
Forest	0.545853	0.518648	0.511295	0.528033	0.5016	0.52213

Кривые построены для признака valid группы 18_24f (кривые для других групп находятся в приложении, как и результаты по всем признакам для всех групп и моделей):





На некоторых графиках есть участки с очень высоким TPR при не таком высоком FPR. Это значит, что можно отсеить некоторую долю (около 30%) плохих снимков, почти не трогая хорошие, взяв соответствующий порог классификации.

Видно, что логистическая регрессия показывает наилучшие результаты в данном случае. Так как эта модель разделяет данные гиперплоскостью, вероятно получается так, что она отсекает ту часть пространства случайных векторов, где меньше доля хороших снимков.

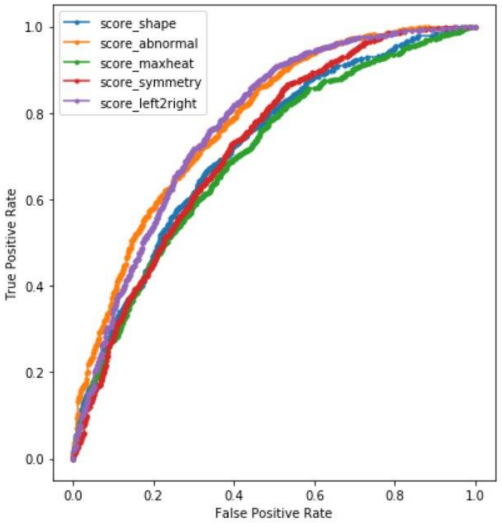
Результаты работы логистической регрессии для каждой половозрастной группы для признака valid. Порог выбран как $argmax(TPR - FPR)$. В ~2 раза увеличивается доля хороших снимков (precision):

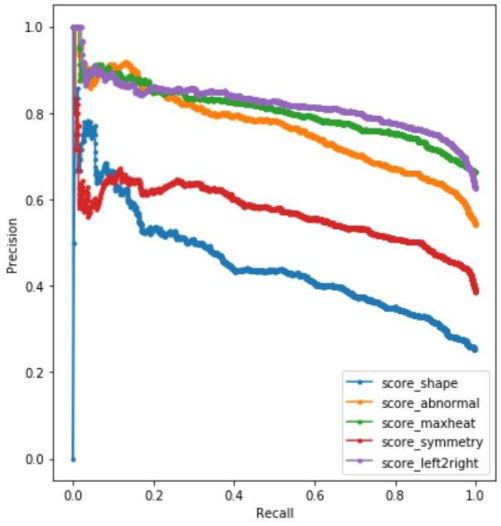
```
valid 18_24m
tpr: 0.7355769230769231
fpr: 0.4294412607449857
precision: 0.11250925240562547
init_precision: 0.06933333333333333
[[1593 1199]
 [  56  152]]

valid 18_24f
tpr: 0.7513812154696132
fpr: 0.3291947499113161
precision: 0.12699905926622765
init_precision: 0.060333333333333336
[[1891  928]
 [  46  135]]

valid 38_44m
tpr: 0.8041237113402062
fpr: 0.42094385118842575
precision: 0.059276366435719784
init_precision: 0.03233333333333333
[[1681 1222]
 [  20   77]]

valid 38_44f
tpr: 0.8256880733944955
fpr: 0.3427879626426842
precision: 0.0824074074074074
init_precision: 0.036333333333333336
[[1900  991]
 [  20   89]]

valid 58_64m
tpr: 0.6774193548387096
fpr: 0.2590197413206263
precision: 0.05112219451371571
init_precision: 0.020666666666666667
[[2177  761]
 [  21   41]]

valid 58_64f
tpr: 0.7235772357723578
fpr: 0.33229058046576293
precision: 0.0842911877394636
init_precision: 0.041
[[1921  956]
 [  35   88]]
```

Архитектура полносвязной нейронной сети (Dense – полносвязный слой, на последнем слое применяется функция активации сигмоида, чтобы предсказывать вероятность):

```
model.add(Dense(256, input_dim=128, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(128, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(BatchNormalization())

model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Кривые и матрицы ошибок (для порога равного 0.5) для нейронной сети для разных признаков группы 18_24f:






```
Confusion matrix:
 [[1288  219]
 [ 297  196]]
Precision: 0.472289156626506

score_abnormal:
Confusion matrix:
 [[735 246]
 [334 685]]
Precision: 0.7357679914070892

score_maxheat:
Confusion matrix:
 [[ 312  367]
 [ 312 1009]]
Precision: 0.7332848837209303

score_symmetry:
Confusion matrix:
 [[968 270]
 [321 441]]
Precision: 0.620253164556962

score_left2right:
Confusion matrix:
 [[505 247]
 [296 952]]
Precision: 0.7939949958298582
```

На других группах модель ведёт себя схожим образом (графики находятся в приложении).

Заметного превосходства нейронной сети над классическими моделями не наблюдается, поэтому в дальнейшем будет рассматриваться логистическая регрессия.

Так же был проведён следующий эксперимент. Для каждого признака обучается модель. Порог классификации выбирался таким образом, что TPR=0.95. Дальше признак valid предсказывался как 1, если все признаки фильтрации больше нуля, и как 0 иначе. Но результаты оказались приблизительно такими же, как и если предсказывать признак valid без промежуточных признаков.

5. Выбор порога классификации.

Эксперименты с различными порогами на группе 18_24m с признаком valid, модель – логистическая регрессия:

```
score: valid, group: 18_24m, threshold=argmax(tpr-fpr), valid
threshold: 0.05264174816025844
tpr: 0.7536231884057971
fpr: 0.4006514657980456
precision: 0.12351543942992874
init_precision: 0.0696969696969697
[[1656 1107]
 [  51  156]]

score: valid, group: 18_24m, threshold=argmax(tpr-fpr), test
threshold: 0.05264174816025844
tpr: 0.75
fpr: 0.4479166666666667
precision: 0.06521739130434782
init_precision: 0.04
[[53 43]
 [ 1  3]]

score: valid, group: 18_24m, threshold=argmax(2*tpr-fpr), valid
threshold: 0.02539035344676367
tpr: 0.9227053140096618
fpr: 0.6478465436120159
precision: 0.09641595153962645
init_precision: 0.0696969696969697
[[ 973 1790]
 [  16  191]]

score: valid, group: 18_24m, threshold=argmax(2*tpr-fpr), test
threshold: 0.02539035344676367
tpr: 1.0
fpr: 0.6666666666666666
precision: 0.058823529411764705
init_precision: 0.04
[[32 64]
 [ 0  4]]

score: valid, group: 18_24m, threshold=argmax(tpr-2*fpr), valid
threshold: 0.10663928188074857
tpr: 0.5072463768115942
fpr: 0.169019182048498
precision: 0.18356643356643357
init_precision: 0.0696969696969697
[[2296  467]
 [ 102  105]]

score: valid, group: 18_24m, threshold=argmax(tpr-2*fpr), test
threshold: 0.10663928188074857
tpr: 0.5
fpr: 0.14583333333333334
precision: 0.125
init_precision: 0.04
[[82 14]
 [ 2  2]]

score: valid, group: 18_24m, threshold=arg(tpr==0.95), valid
threshold: 0.019090226875107585
tpr: 0.9565217391304348
fpr: 0.7408613825551936
precision: 0.08819599109131403
init_precision: 0.0696969696969697
[[ 716 2047]
 [   9  198]]

score: valid, group: 18_24m, threshold=arg(tpr==0.95), test
threshold: 0.019090226875107585
tpr: 1.0
fpr: 0.75
precision: 0.05263157894736842
init_precision: 0.04
[[24 72]
 [ 0  4]]

score: valid, group: 18_24m, threshold=argmax(precision), valid
threshold: 0.27958700398479464
tpr: 0.12560386473429952
fpr: 0.026782482808541442
precision: 0.26
init_precision: 0.0696969696969697
[[2689   74]
 [ 181   26]]

score: valid, group: 18_24m, threshold=argmax(precision), test
threshold: 0.27958700398479464
tpr: 0.0
fpr: 0.03125
precision: 0.0
init_precision: 0.04
[[93  3]
 [ 4  0]]
```

Эксперименты провелись на валидационной и тестовой выборках. В качестве тестовой выборки были взяты 100 снимков их которых только 4 являлись хорошими.

Варианты выбора порога классификации:

- $argmax(TPR - FPR)$ –precision увеличивается приблизительно в два раза.

- $argmax(2 * TPR - FPR)$ – precision увеличивается меньше, но при этом хорошие снимки почти не теряются при фильтрации, отсеиваются только плохие.

- $argmax(TPR - 2 * FPR)$ – precision увеличивается в три раза, но и всего снимков проходит мало, как плохих, так и хороших. Уменьшается их разнообразие.

- $\arg(TPR = 0.95)$ - precision увеличивается совсем незначительно, хорошие снимки не теряются при фильтрации, только плохие.

- $argmax(precision)$ – фильтрацию проходит слишком мало снимков, на тестовой выборке не проходит ни одного.

Таким образом приемлемыми можно считать первые два варианта.

6. Время генерации снимков.

t - время затраченное на генерацию всех снимков.

t_0- время генерации одного снимка.

n – количество всех снимков.

k – количество хороших снимков.

$$t = t_0 n$$

$$k = n * precision$$

$$t = \frac{t_0 k}{precision}$$

t_1-время генерации без фильтрации.

t_2-время генерации с фильтрацией.

$$\frac{t_1}{t_2} = \frac{precision_2}{precision_1}$$

То есть время генерации обратно пропорционально доле хороших снимков (precision).

В примерах из предыдущего пункта, если порог выбирать как $argmax(TPR - FPR)$, то время сократится в 1.8 раз. Если $argmax(2 * TPR - FPR)$, то в 1.4 раза, но будет больше разнообразие снимков.

Заключение.

Введя дополнительную предобработку случайных векторов перед подачей их в генератор сети GAN, удалось увеличить долю сгенерированных хороших снимков в полтора-два раза и соответственно ускорить время генерации снимков во столько же раз.

Список использованных источников

[1] “Generative Adversarial Nets” [Electronic resource]. - https://arxiv.org/pdf/1406.2661.pdf

[2] “The GAN Zoo” [Electronic resource]. - https://github.com/hindupuravinash/the-gan-zoo

[3] “A Style-Based Generator Architecture for Generative Adversarial Networks” [Electronic resource]. - https://arxiv.org/pdf/1812.04948.pdf

[4] [Electronic resource]. - https://thispersondoesnotexist.com/

[5] “Image Segmentation Algorithms Overview” [Electronic resource]. - https://arxiv.org/ftp/arxiv/papers/1707/1707.02051.pdf

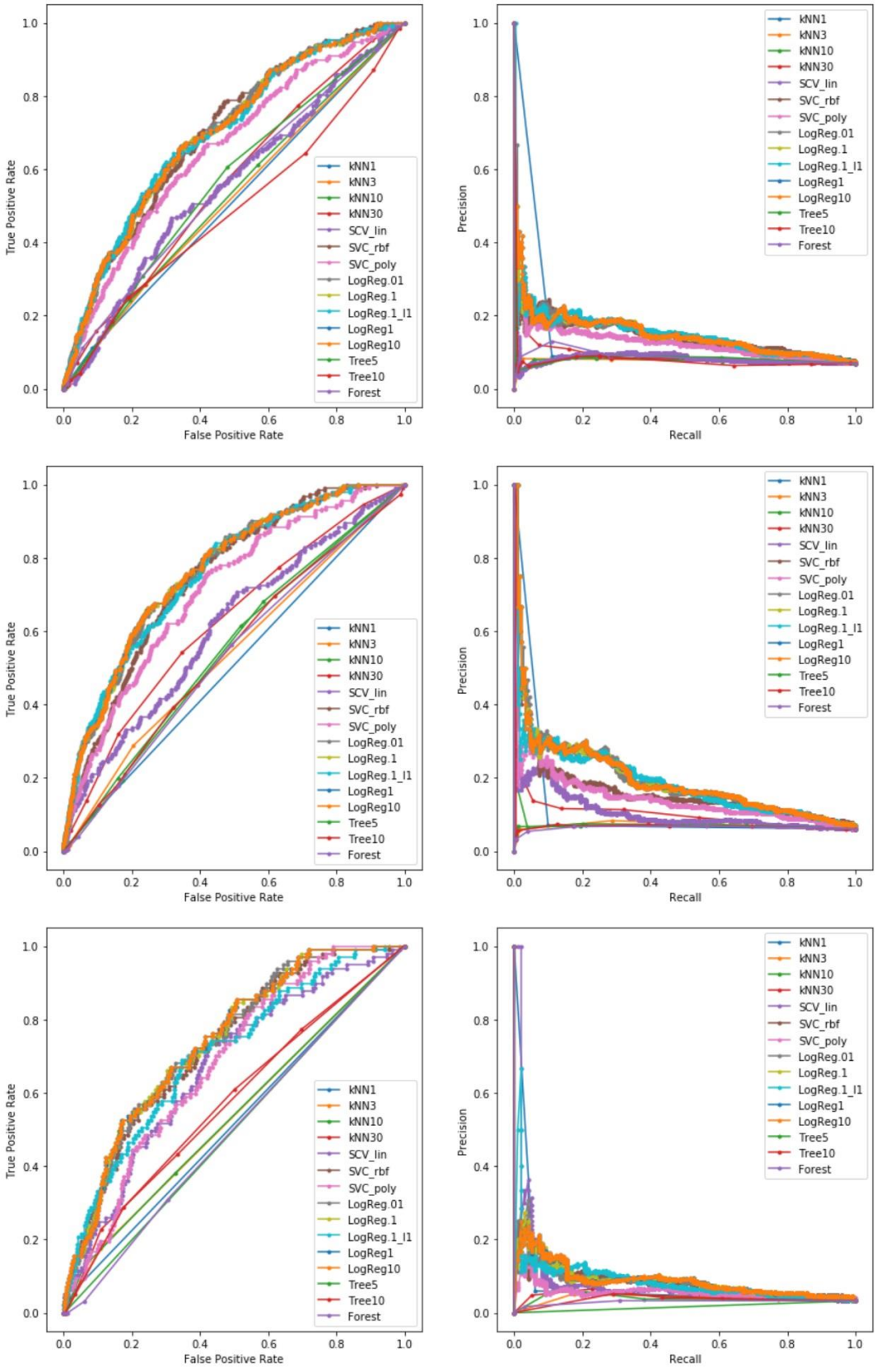
[6] “U-Net: Convolutional Networks for Biomedical Image Segmentation” [Electronic resource]. - https://arxiv.org/pdf/1505.04597.pdf

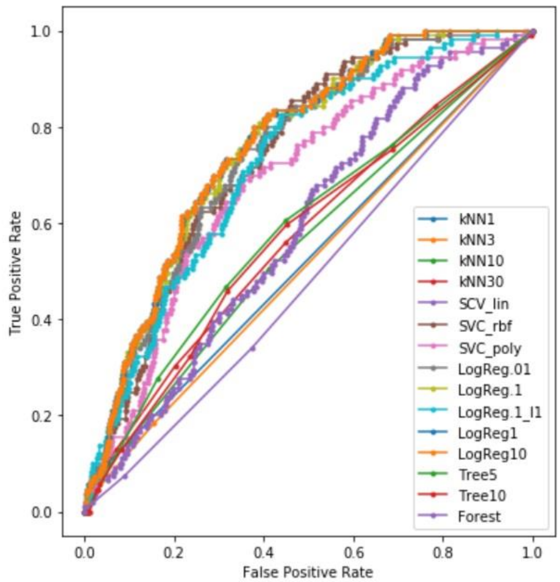
[7] “Deep Residual Learning for Image Recognition” [Electronic resource]. - https://arxiv.org/pdf/1512.03385.pdf

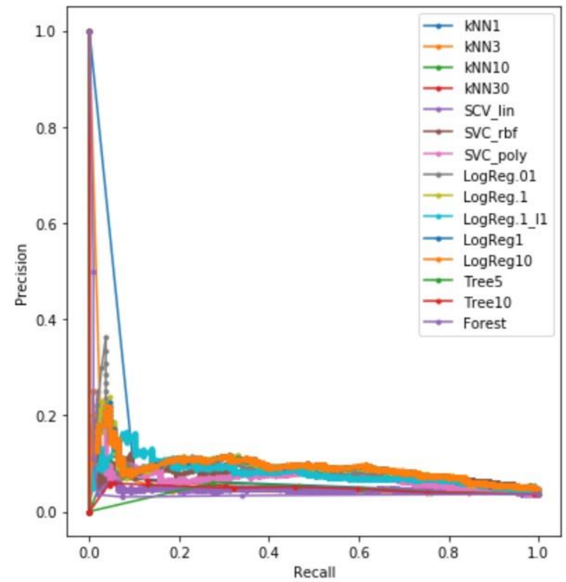
[8] “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization” [Electronic resource]. - https://arxiv.org/pdf/1610.02391.pdf

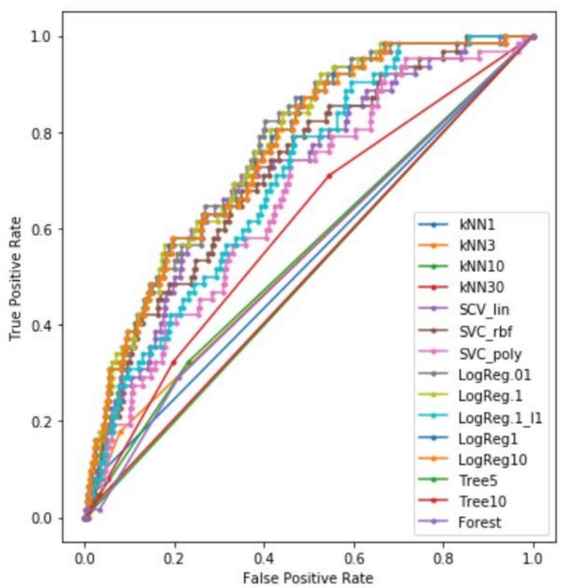
Приложения

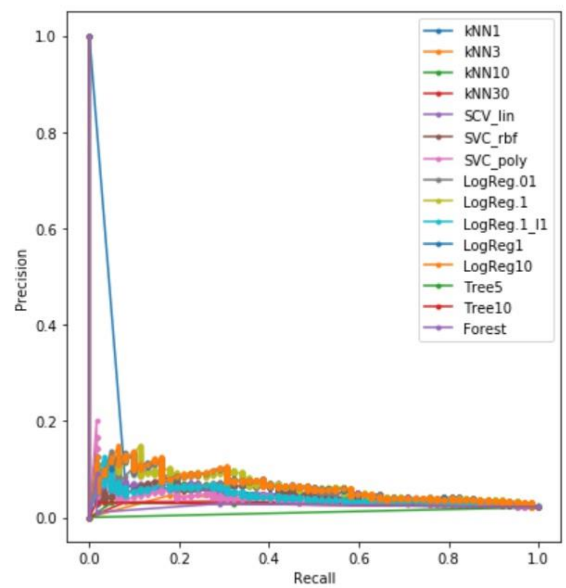
Кривые построены для признака valid по всем группам:

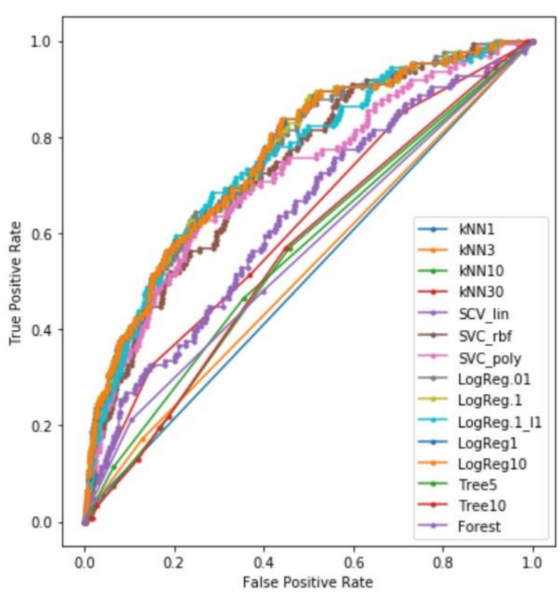


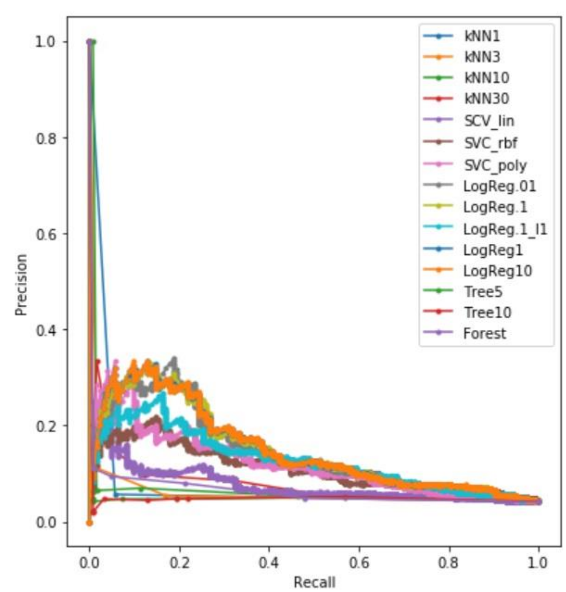

True Positive Rate
False Positive Rate
kNN1
kNN3
kNN10
kNN30
SCV_lin
SVC_rbf
SVC_poly
LogReg.01
LogReg.1
LogReg.1_l1
LogReg1
LogReg10
Tree5
Tree10
Forest


Precision
Recall
kNN1
kNN3
kNN10
kNN30
SCV_lin
SVC_rbf
SVC_poly
LogReg.01
LogReg.1
LogReg.1_l1
LogReg1
LogReg10
Tree5
Tree10
Forest

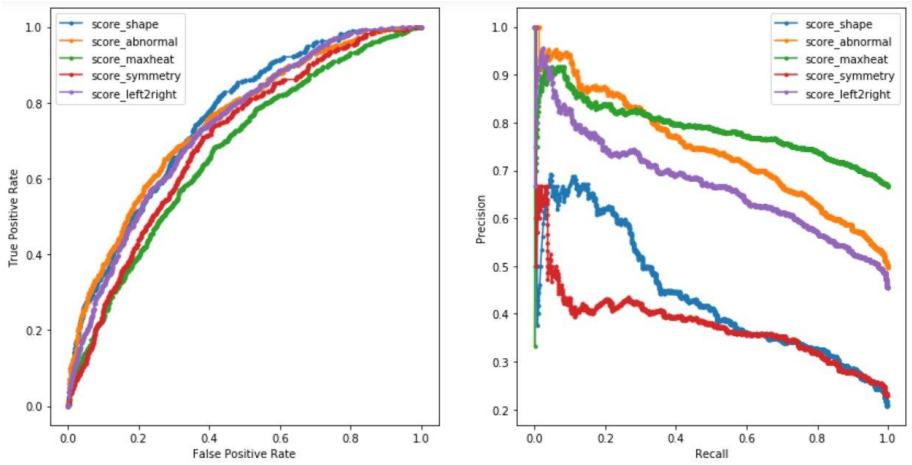

True Positive Rate
False Positive Rate
kNN1
kNN3
kNN10
kNN30
SCV_lin
SVC_rbf
SVC_poly
LogReg.01
LogReg.1
LogReg.1_l1
LogReg1
LogReg10
Tree5
Tree10
Forest


Precision
Recall
kNN1
kNN3
kNN10
kNN30
SCV_lin
SVC_rbf
SVC_poly
LogReg.01
LogReg.1
LogReg.1_l1
LogReg1
LogReg10
Tree5
Tree10
Forest


True Positive Rate
False Positive Rate
kNN1
kNN3
kNN10
kNN30
SCV_lin
SVC_rbf
SVC_poly
LogReg.01
LogReg.1
LogReg.1_l1
LogReg1
LogReg10
Tree5
Tree10
Forest

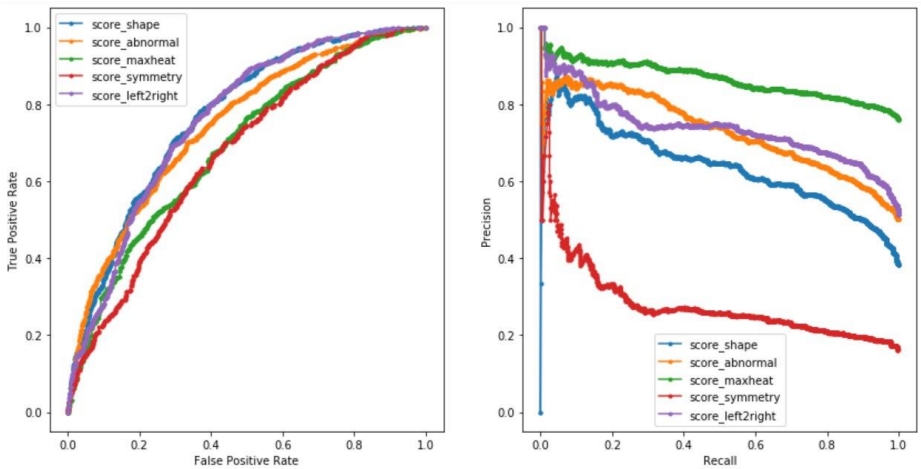

Precision
Recall
kNN1
kNN3
kNN10
kNN30
SCV_lin
SVC_rbf
SVC_poly
LogReg.01
LogReg.1
LogReg.1_l1
LogReg1
LogReg10
Tree5
Tree10
Forest

Кривые и матрицы ошибок для нейронной сети для группы 38_44m:




```
score_shape
confusion_matrix
 [[1265  331]
 [ 278  126]]
precision_score
 0.27571115973741794

score_abnormal
confusion_matrix
 [[523 488]
 [444 545]]
precision_score
 0.5275895450145208

score_maxheat
confusion_matrix
 [[227 442]
 [343 988]]
precision_score
 0.6909090909090909

score_symmetry
confusion_matrix
 [[1179  367]
 [ 322  132]]
precision_score
 0.26452905811623245

score_left2right
confusion_matrix
 [[598 518]
 [431 453]]
precision_score
 0.466529351184346
```

Кривые и матрицы ошибок для нейронной сети для группы 58_64f:




```
score_shape
confusion_matrix
 [[779 468]
 [415 338]]
precision_score
 0.41935483870967744

score_abnormal
confusion_matrix
 [[537 469]
 [398 596]]
precision_score
 0.5596244131455399

score_maxheat
confusion_matrix
 [[ 125  356]
 [ 315 1204]]
precision_score
 0.7717948717948718

score_symmetry
confusion_matrix
 [[1385  295]
 [ 253   67]]
precision_score
 0.1850828729281768

score_left2right
confusion_matrix
 [[527 457]
 [513 503]]
precision_score
 0.5239583333333333
```

Нейронная сеть со слоями от одного до четырёх на признаке valid для всех групп:

```
1: Sequential([
    Dense(1, input_dim=128, activation='sigmoid'),
]),
2: Sequential([
    Dense(128, input_dim=128, activation='relu'),
    Dense(1, activation='sigmoid'),
]),
3: Sequential([
    Dense(128, input_dim=128, activation='relu'),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid'),
]),
4: Sequential([
    Dense(128, input_dim=128, activation='relu'),
    Dense(128, activation='relu'),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid'),
]),
```

	group	score	n_layers	precision	recall	accuracy	roc_auc
0	18_24m	valid	1	0.089791	0.327014	0.722399	0.570652
1	18_24m	valid	2	0.277778	0.002880	0.930300	0.722847
2	18_24m	valid	3	0.355556	0.015823	0.930200	0.733771
3	18_24m	valid	4	0.083333	0.001437	0.930400	0.717585
4	18_24f	valid	1	0.076437	0.339967	0.714699	0.569140
5	18_24f	valid	2	0.833333	0.006633	0.940000	0.741619
6	18_24f	valid	3	0.595238	0.009950	0.939500	0.752014
7	18_24f	valid	4	0.333333	0.001658	0.939700	0.738711
8	38_44m	valid	1	0.041333	0.319892	0.738899	0.566943
9	38_44m	valid	2	0.000000	0.000000	0.967800	0.730904
10	38_44m	valid	3	0.000000	0.000000	0.967800	0.746964
11	38_44m	valid	4	0.333333	0.003115	0.967900	0.747139
12	38_44f	valid	1	0.045737	0.338843	0.719300	0.582363
13	38_44f	valid	2	0.000000	0.000000	0.963700	0.757105
14	38_44f	valid	3	0.000000	0.000000	0.963500	0.768112
15	38_44f	valid	4	0.000000	0.000000	0.963700	0.768849
16	58_64m	valid	1	0.030564	0.393365	0.730599	0.582893
17	58_64m	valid	2	0.000000	0.000000	0.979400	0.785995
18	58_64m	valid	3	0.000000	0.000000	0.979400	0.782045
19	58_64m	valid	4	0.000000	0.000000	0.979400	0.791534
20	58_64f	valid	1	0.044730	0.287105	0.720401	0.544279
21	58_64f	valid	2	0.000000	0.000000	0.958600	0.740036
22	58_64f	valid	3	0.000000	0.000000	0.958600	0.750019
23	58_64f	valid	4	0.000000	0.000000	0.958900	0.746812