

Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных
систем

Федор Игоревич Жилкин

Анализ решений задачи детекции лиц на
изображениях в сфере
киберкриминалистики

Курсовая работа

Научный руководитель:
доцент каф. СП СПбГУ, к.т.н. Ю. В. Литвинов

Санкт-Петербург
2020

Оглавление

Введение	3
1. Постановка задачи	5
2. Выбор набора данных	6
3. Выбор решений распознавания лиц	8
4. Схема тестирования	10
4.1. Метрика IoU	10
4.2. Классы TP, FP, FN	10
4.3. Метрики Recall, Precision, F-мера	11
4.4. Алгоритм тестирования	12
5. Тестируемые решения	14
5.1. Каскады Хаара	14
5.2. Гистограмма ориентированных градиентов	17
5.3. Single-Shot-Multibox Detector	18
5.4. Local Binary Patterns	19
5.5. Max-Margin Object Detection	20
5.6. Multi-Task Cascaded Neural Network	21
6. Результаты тестирования	24
6.1. Результаты лучших решений	24
6.2. Результаты решений, интегрированных под платформу .NET	25
6.3. Результаты тестирования скорости при работе на CPU .	25
6.4. Тест на поворот лица	26
6.5. Тест на окклюзию	27
7. Результаты	29
Список литературы	30

Введение

Наша безопасность напрямую связана с успехами в области технического прогресса, ведь благодаря новым технологиям нам становится проще пресекать преступления и изолировать людей, совершивших злодеяние. Одним из важнейших средств безопасности являются камеры видеонаблюдения на улицах, в метро и в других общественных местах. Важно понимать, что количество информации, поступающей с этих камер, растет с каждым днем и автоматизация обработки полученных видео и фотографий становится всё более необходима. Решить эту проблему помогает автоматическое распознавание лиц для дальнейшей их верификации (определение того, что два лица на разных фотографиях принадлежат одному и тому же человеку), кластеризации (разбиение лиц на фотографиях по группам, каждая из которых соответствует одному человеку) и классификации (выяснение, принадлежит ли данное лицо человеку, находящемся в базе данных).

Любая задача, связанная с лицами, начинается с их обнаружения на фотографиях или видео. Данная работа посвящена распознаванию лиц (face detection). В ней будут рассмотрены основные методы распознавания лиц, проведена сравнительная характеристика и будут выяснены лучшие решения этой задачи для работы в сфере криминалистики на основе двух ключевых факторов:

- эффективность решения в условиях слабой освещенности, наличия угла поворота лица и наличия вещей, частично закрывающих лицо;
- по данным компании BelkaSoft у криминалистов чаще всего нет графических ускорителей, а тяжелые решения (например сверточные глубокие нейронные сети) требуют большой вычислительной мощности, что ведет к долгой работе на CPU, что подчеркивает важность быстродействия готового продукта.

Данная работа выполнена совместно с компанией Belkasoft, специализирующейся на создании программного обеспечения в сфере кибер-

криминалистики. Кроме того данное исследование выполнялось при участии студента Олега Чернявского, который занимается кластеризацией лиц (face clustering).

1. Постановка задачи

Целью работы является обзор и сравнение уже существующих методов распознавания лиц на изображениях на основе факторов, приведенных во введении. По результатам исследования лучшее решение будет интегрировано в продукт BelkaSoft Evidence Center. Для успешного выполнения данной цели были поставлены следующие задачи:

- выбрать набор данных для тестирования решений;
- рассмотреть существующие решения детектирования лиц;
- создать тестирующую систему для сравнения решений;
- провести полное сравнение всех решений.

2. Выбор набора данных

Для задачи обнаружения лиц в сфере киберкриминалистики необходимо выбрать набор изображений, обладающий определенными свойствами:

- окклюзия (возникает в ситуации, когда лицо частично прикрыто);
- лица, расположенные под разными углами;
- разная освещенность лиц;
- make-up (лицо чем-то разукрашено);
- разный размер лиц на изображениях (маленькие – до 30x30 пикселей и большие – больше 300x300 пикселей);
- разные эмоции на лицах.

Существует достаточно много наборов данных, которые используются в задачах обнаружения лиц на изображениях. Вот некоторые из них:

- FDDB [3];
- Wider Face [11];
- MAFA [2];
- 4k face dataset [9].

Все эти наборы данных используются в разных задачах детектирования лиц. Но для решения задачи успешного детектирования лиц в сфере киберкриминалистики нам необходимо наличие изображений, удовлетворяющих вышеназванным свойствам. Поэтому был выбран набор данных Wider Faces [11]. Это размеченный набор изображений, т.е. каждое лицо на каждом изображении имеет свои координаты, которые записаны в отдельном файле. Более того этот набор данных полностью удовлетворяет вышеназванным свойствам. В нем содержится около 12000 изображений, разбитых на три уровня сложности по каждому из параметров.

Этот набор данных представляет особый интерес для криминалистов, так как чаще всего, изображения, полученные с камер видеонаблюдения имеют плохое качество, а лица на них могут быть чем-то прикрыты, плохо освещены или разрисованы. Примеры изображений из этого набора данных приведены на Рис.1.



Рис. 1: Набор данных Wider Faces, (источник: <https://arxiv.org/pdf/1511.06523.pdf>, дата обращения: 17.05.2020)

3. Выбор решений распознавания лиц

Выбирать решения будем, опираясь на следующие факторы:

- быстрота при работе на CPU;
- относительная популярность решений в индустрии;
- качество работы решений.

Поскольку продукт BelkaSoft Evidence Center работает под платформой .NET, то для тестирования были выбраны решения, которые уже реализованы под данной платформой и удовлетворяют вышеописанным факторам:

- каскады Хаара и LBP;
- гистограмма ориентированных градиентов (HOG);
- Single-Shot-Multibox Detector (SSD);
- FaceNet's Multi-Task Cascaded Neural Network (MTCNN);
- Maximum-Margin Object Detector (MMOD).

Также были выбраны три решения, показавшие лучшие результаты на выбранном наборе данных Wider Faces:

- Dual Shot Face Detector (DSFD);
- Selective Refinement Network for High Performance Face Detection (SRN);
- Accurate Face Detection for High Performance (AIInnoFace).

Последние три решения – специально обученные на этом же наборе изображений нейронные сети. Они имеют хорошие результаты (будут приведены ниже), но они не были протестированы на других наборах данных, поэтому объективную оценку этим решениям в контексте текущей задачи дать нельзя. Эти решения приведены лишь для сравнения

с результатами вышеописанных решений. В дальнейшем планируется реализовать эти решения под платформой .NET и протестировать на других наборах данных.

4. Схема тестирования

Окончательная оценка решений будет определяться двумя тестами:

- тест на точность решения на выбранном наборе данных;
- тест на скорость работы на CPU.

4.1. Метрика IoU

Введем метрику степени пересечения между двумя ограничивающими рамками. Это необходимо для того, чтобы понимать, правильно ли алгоритм обнаружил лицо на изображении. IoU (Intersection-over-Union) считается следующим образом:

$$IoU = \frac{\text{Intersection_Area}}{\text{Union_Area}}$$

Если IoU от набора реальных координат лица (*ground_truth*) и набора координат, полученного из алгоритма (*predicted_vector*), больше или равно 0.5, то говорим, что этот полученный набор координат – действительно правильно обнаруженное лицо. Примеры IoU метрик можно увидеть на Рис.2.

4.2. Классы TP, FP, FN

Для построения метрик введем классы True-Positive (TP), False-Positive (FP), False-Negative (FN). Как показано на Рис.3, у нас есть три варианта для результатов, полученных алгоритмом:

- при $IoU(\text{ground_truth}, \text{predicted_vector}) \geq 0.5$ – True-Positive (лицо обнаружили, как минимум, наполовину);
- при $IoU(\text{ground_truth}, \text{predicted_vector}) < 0.5$ – False-Positive (вектор, не являющееся ground-truth лицом);
- все остальные лица, не обнаруженные алгоритмом – False-Negative.

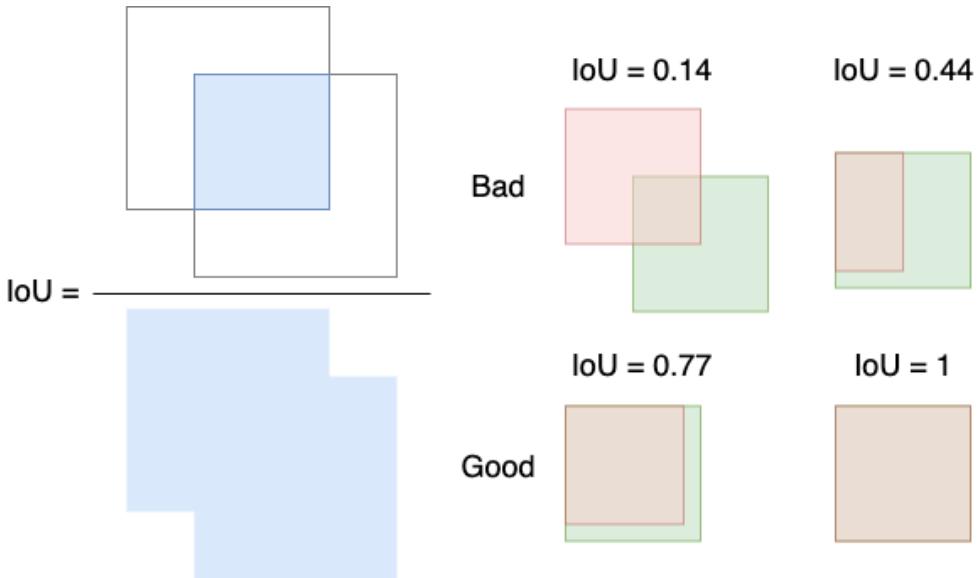


Рис. 2: Метрика Intersection-over-Union (источник: <https://habr.com/ru/company/jetinfosystems/blog/498294/>, дата обращения: 17.05.2020)

4.3. Метрики Recall, Precision, F-мера

Для оценки точности работы алгоритмов введем следующие метрики:

- $Recall = \frac{TP}{TP+FN}$;
- $Precision = \frac{TP}{TP+FP}$;
- $F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}$.

Recall показывает, какую долю лиц из всех лиц на изображении нашел алгоритм. Precision можно интерпретировать как долю лиц, названных детектором лицами и при этом действительно являющимися лицами. Если наш детектор будет обнаруживать все как лицо, то Recall будет стремиться к единице, а Precision – к нулю. Зеркальная ситуация (такой, в которой детектор ничего не обнаруживает) дает нам обратные результаты (Recall стремится к нулю, Precision – к единице). Получается, что невозможно оценить качество работы алгоритма только лишь по одному из этих параметров. Поэтому вводится метрика F-мера. Она представляет собой среднее гармоническое Precision и Recall. F-мера

51_Dresses_wearingdress_51_1030

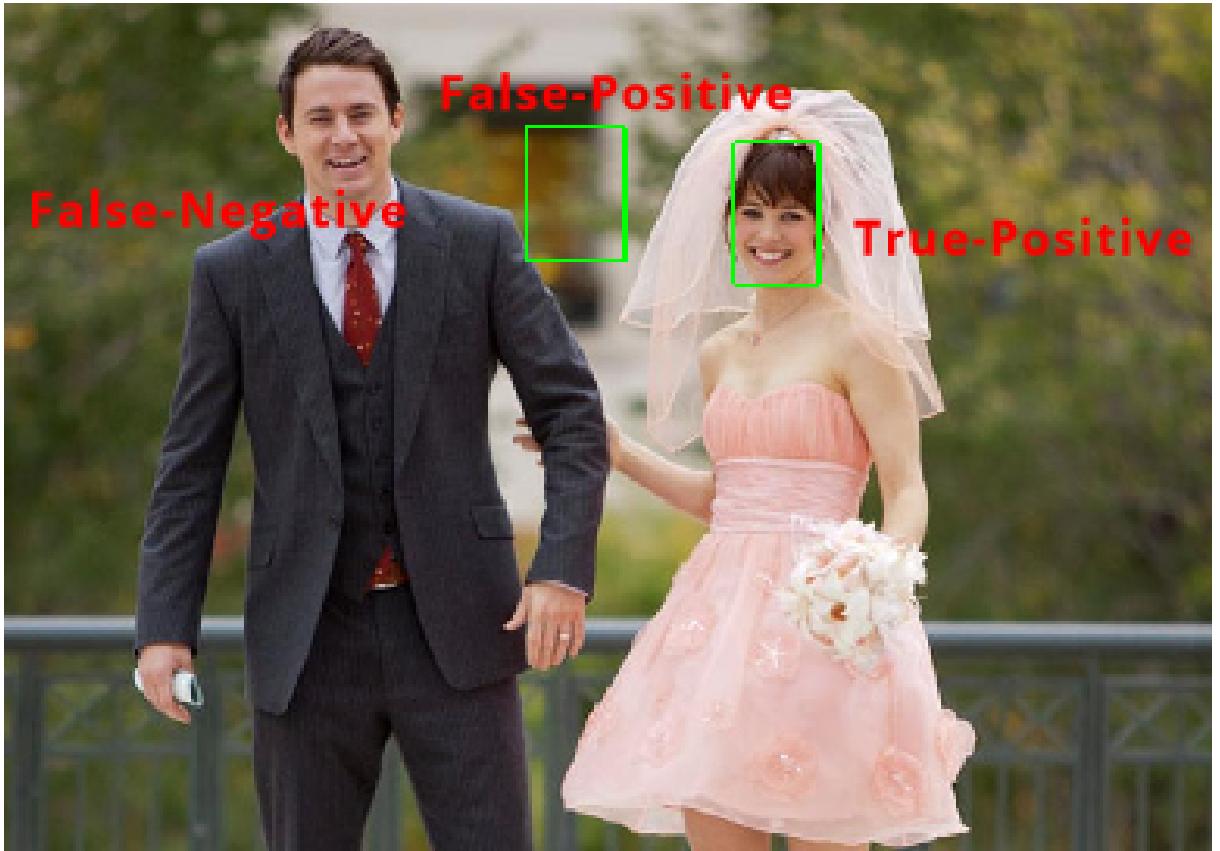


Рис. 3: Классы ТР, FP, FN (источник: Набор данных Wider Faces)

позволяет точно определить качество работающего алгоритма. Именно на эту метрику будем ссыльаться при тестировании наших решений.

4.4. Алгоритм тестирования

Для тестирования решений была выбрана легкая часть набора изображений по параметру Scale, где каждое изображение содержит лицо, размером не меньшие, чем 300x300 пикселей. Далее для каждого изображения получаем реальные координаты лиц (ground-truth векторы) и координаты, полученные с помощью решения (predicted векторы). Затем с помощью метрики IoU заполняем классы ТР, FP, FN. После вычисляем метрики Recall, Precision, F-мера. Схематично алгоритм тестирования изображен на Рис.4. Таким образом, первая часть тестиро-

вания закончена.

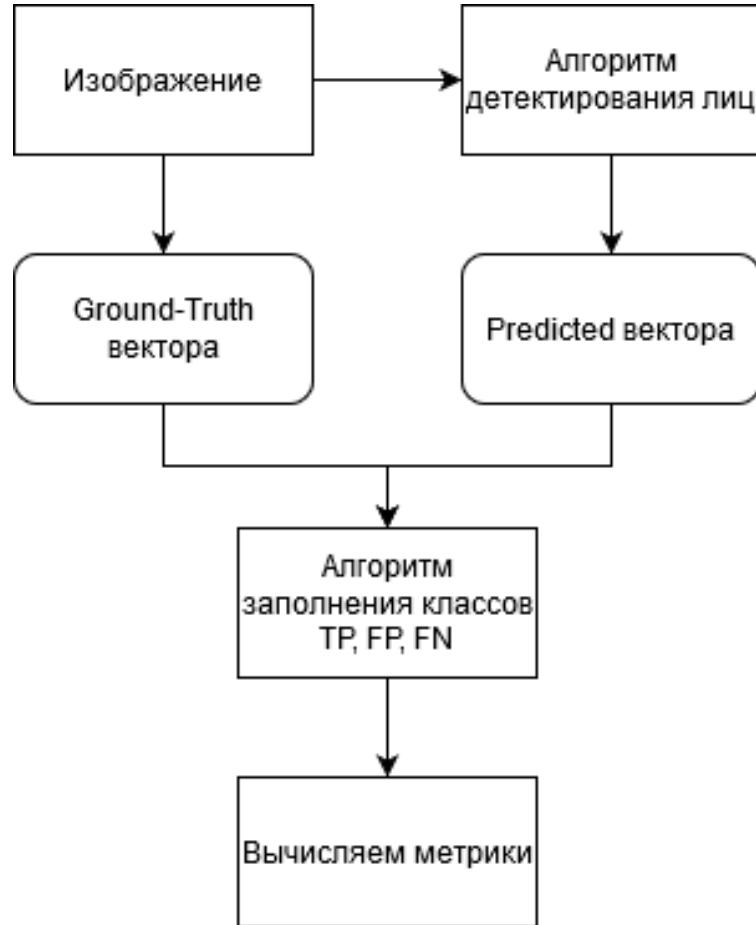


Рис. 4: Алгоритм тестирования решений

Вторая часть тестирования заключается в проверке скорости работы решения. Для этого возьмем изображение 300x300 пикселей и запустим на нем решение 10000 раз. Посчитаем среднюю скорость обработки изображения и дисперсию (минимальное и максимальное время). После этого выберем лучшее решение по этим двум тестам.

5. Тестируемые решения

Поскольку данная работа носит обзорный характер, мы не будем подробно рассматривать все шаги каждого алгоритма, а остановимся на самых важных, благодаря которым можно понять общую концепцию решения.

5.1. Каскады Хаара

Каскадный классификатор Хаара, впервые описанный в 2001 году в оригинальной статье [7], представляет собой особый случай ансамблированного обучения [15], называемый ускорением (boosting). Как правило, он основан на классификаторах Adaboost [14]. Существует алгоритм, называемый «алгоритмом Виолы-Джонса», который включает в себя все этапы, необходимые для обнаружения лица:

- выбор характеристик (признаков) Хаара;
- интегральное представление изображения;
- Adaboost Training;
- каскадный классификатор.

Рассмотрим некоторые шаги алгоритма:

Выбор характеристик Хаара.

Есть некоторые общие черты, которые мы находим на человеческих лицах:

- темная область глаз по сравнению с областью щек;
- яркая область переносицы по сравнению с областью глаз;
- какое-то конкретное расположение глаз, рта, носа.

Процесс извлечения этих характеристик продемонстрирован на Рис. 5.

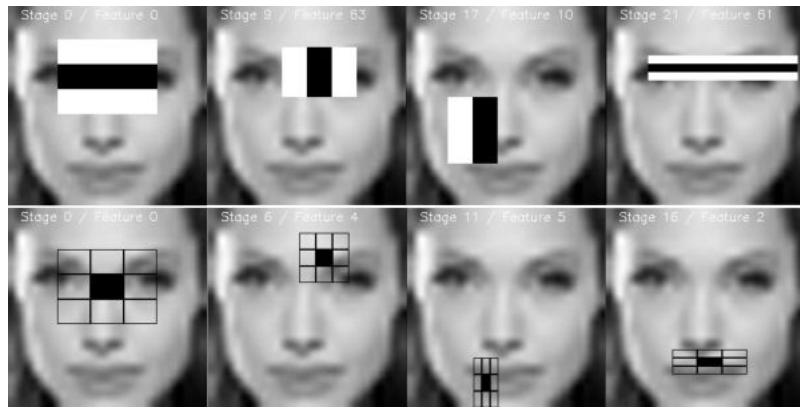


Рис. 5: Извлечение характеристик (feature extraction) (источник: <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf>, дата обращения: 17.05.2020)

В этом примере первая характеристика измеряет разницу в интенсивности между областью глаз и областью щек. Значение характеристики вычисляется путем суммирования пикселей в черной области и вычитания пикселей в белой области.

$$\text{RectangleFeature} = \sum \text{pixels}_{\text{blackarea}} - \sum \text{pixels}_{\text{whitearea}}$$

Затем мы применяем эту характеристику как сверточное ядро по всему нашему изображению. Вычислять прямоугольные характеристики, используя принцип интегрального изображения, получается намного быстрее, поэтому следующий шаг после выбора характеристик – интегральное представление изображения [8].

Существует несколько типов прямоугольников (изображены на Рис. 6), которые можно применять для извлечения характеристик Хаара.

Теперь, когда характеристики выбраны, мы применяем их к набору обучающих изображений, используя классификацию Adaboost, которая представляет собой набор слабых классификаторов, для создания точной ансамблированной модели (ansamble model).

Каскадный классификатор

На изображении большая часть изображения – это область без лица. Придавать одинаковую важность каждой области изображения не имеет смысла, поскольку мы должны сосредоточиться на областях, кото-

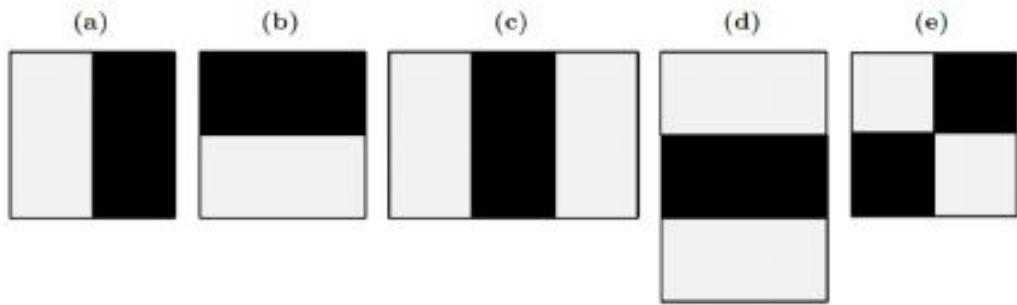


Рис. 6: Прямоугольники Хаара (Haar's rectangles) (источник: <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf>, дата обращения: 17.05.2020)

рые, скорее всего, содержат лицо. Ключевая идея состоит в том, чтобы не рассматривать области изображения, которые не содержат граней (областей резкой смены интенсивности цвета). Поскольку задача состоит в том, чтобы правильно идентифицировать лицо, мы хотим минимизировать количество ложных отрицательных результатов, при которых области, которые содержат лицо не были идентифицированы как области с лицом.

Ряд классификаторов применяется к каждой области изображения. Эти классификаторы являются простыми деревьями решений:

- если первый классификатор положительный, мы переходим ко второму;
- если второй классификатор положительный, мы переходим к третьему;
- ...

Любой отрицательный результат в некоторой точке приводит к отклонению области как потенциально содержащей лицо. Первоначальный классификатор исключает большинство отрицательных примеров при низких вычислительных затратах, а следующие классификаторы устраняют дополнительные отрицательные примеры, но требуют больших вычислительных усилий. Схематично этот процесс изображен на Рис.7.

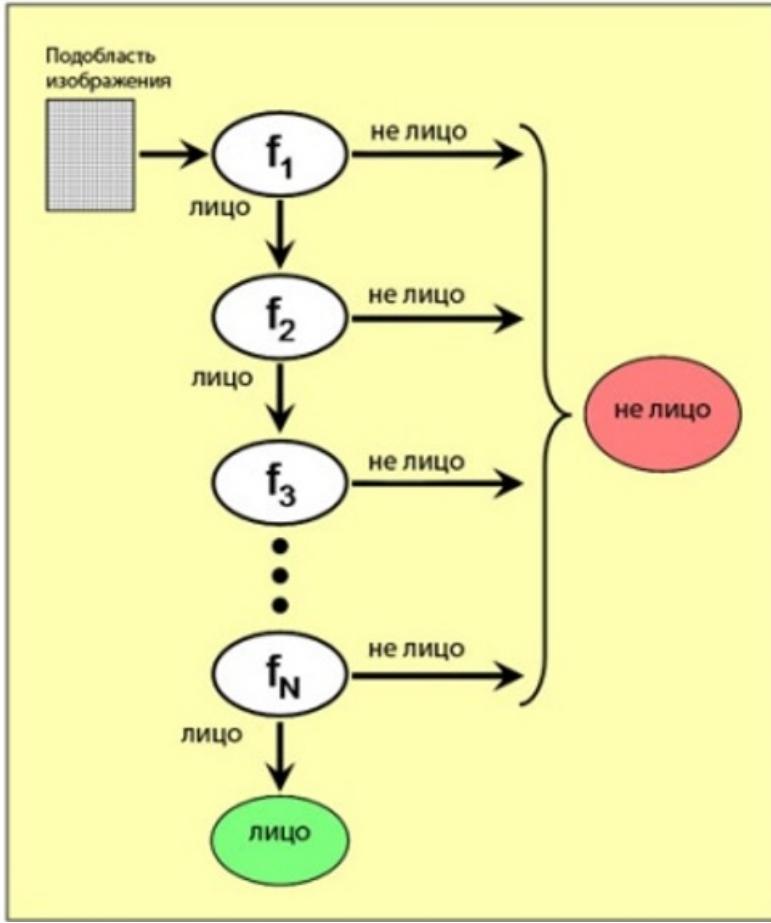


Рис. 7: Каскад (Haar's cascade) (источник: <https://habr.com/ru/post/208092/>, дата обращения: 17.05.2020)

5.2. Гистограмма ориентированных градиентов

Работу HOG [1] можно описать следующими шагами:

- предварительная обработка изображения, включая изменение размера и нормализацию цвета;
- вычисление вектора градиентов каждого пикселя, а также его величину и направление;
- Разделение изображения на множество ячеек размером 8x8 пикселей. В каждой ячейке значения магнитуд этих 64 ячеек сгруппированы и кумулятивно добавлены в 9 сегментов без учета направления ($0^\circ - 180^\circ$ вместо $0^\circ - 360^\circ$), то есть построена гистограмма направленных градиентов.

Конечным шагом в распознавании объектов с использованием HOG является классификация дескрипторов при помощи системы обучения с учителем [16]. Создатели алгоритма использовали метод опорных векторов (SVM, Support Vector Machine) [13].

5.3. Single-Shot-Multibox Detector

Данный метод, Single-Shot-Multibox Detector (SSD), впервые описанный в 2016 году в оригинальной статье [10], позволяет обнаруживать объекты на изображениях, используя только одну глубокую сверточную нейронную сеть. При использовании этого подхода выходное пространство границ обнаруженных объектов (space of bounding boxes) разбивается на набор базовых прямоугольников с различными соотношениями сторон. В качестве предсказания алгоритм выдает числа, определяющие степень уверенности в присутствии той или иной категории объектов в каждом из базовых прямоугольников. Также производится корректировка границ с целью лучшего покрытия объекта на изображении. В названии “Single Shot MultiBox Detector”:

- Single Shot означает, что задачи локализации и классификации объектов выполняются за один проход сети;
- MultiBox – методика поиска ограничивающего прямоугольника;
- Detector – нейронная сеть, которая работает как детектор объектов.

Рис.8 демонстрирует схему работы данного подхода. К входному изображению применяется некоторое количество сверточных слоев, причем часть из этих слоев взята из модели VGG16 [5]. Пространственная размерность изображения убывает после каждого слоя, доходя до единицы. Далее к промежуточным результатам применяется блок «Detector and classifier», на выходе которого имеем детекцию, т.е. прямоугольники с классом. Затем эти детекции подаются на вход алгоритму Fast Non-Maximum Suppression (Fast NMS) [4]. Этот алгоритм объединяет все прямоугольники для получения финального результата.

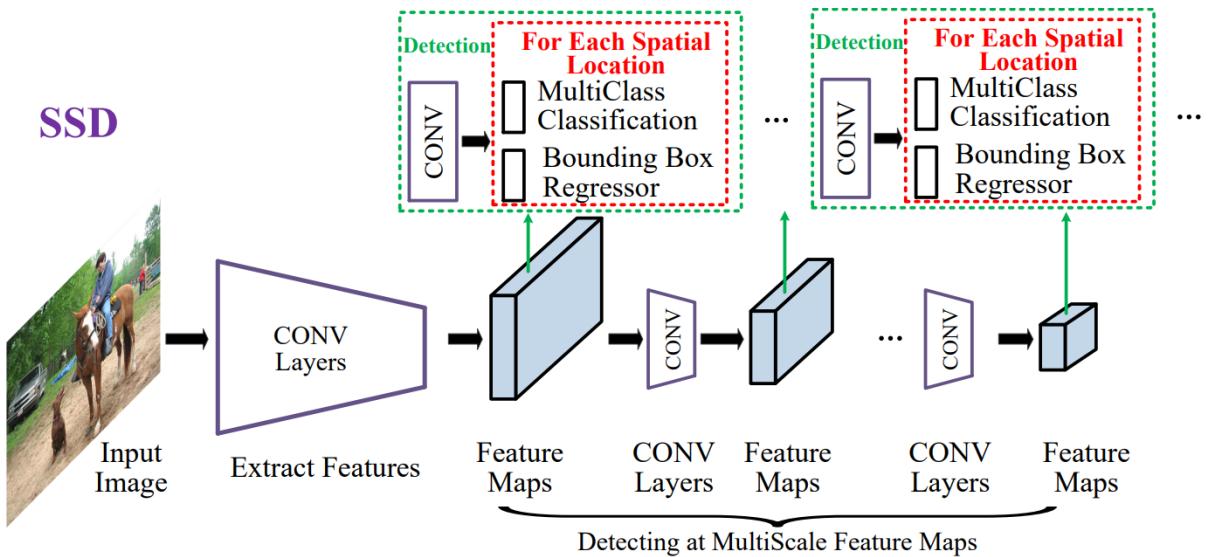


Рис. 8: Схема работы SSD (источник: http://dx.doi.org/10.1007/978-3-319-46448-0_2, дата обращения: 17.05.2020)

5.4. Local Binary Patterns

Local Binary Patterns (LBP) – простой оператор, используемый для классификации текстур. Впервые был описан в оригинальной статье [12] в 1996 году. Концепцию работы этого решения можно изложить следующим образом:

- выбираются радиус и количество точек (см. Рис.9). Далее будем считать LBP на основе восьми смежных точек;

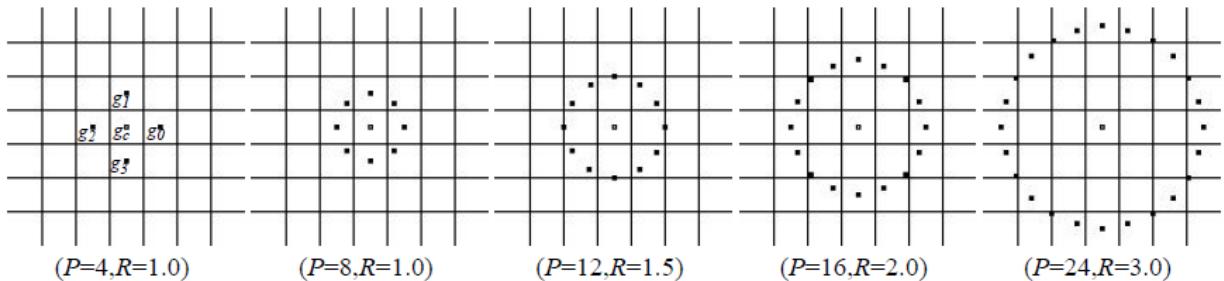


Рис. 9: Выбор точек для построения LBP (источник: <https://habr.com/ru/post/280888/>, дата обращения: 17.05.2020)

- далее необходимо пронумеровать выбранные точки;

- затем вычисляем разность значений яркости между каждым из крайних пикселей и центральным;
- если разность отрицательная (яркость убывает), записываем на месте соседа единицу, если неотрицательная – ноль;
- построим гистограмму по полученным клеткам (см. Рис.10).

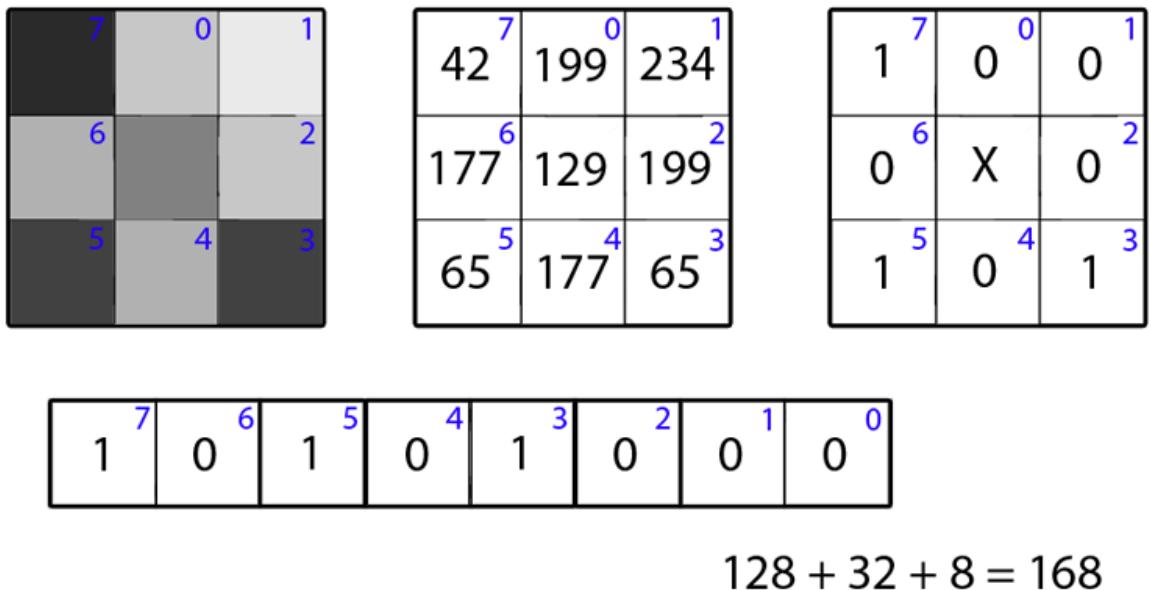


Рис. 10: Полученная гистограмма смены интенсивности в области (источник: <https://habr.com/ru/post/280888/>, дата обращения: 17.05.2020)

Полученное двоичное число называется Local Binary Pattern. По существу, биты дескриптора – это знаки производных яркости по направлениям. Последним шагом алгоритма будет подача полученных гистограмм на вход метода опорных векторов (SVM).

5.5. Max-Margin Object Detection

Большинство методов обнаружения объектов работают путем применения двоичного классификатора к определенным областям изобра-

жения, при этом обнаружение на перекрывающихся областях удаляется. Поскольку число возможных областей в наборах данных изображений даже умеренного размера чрезвычайно велико, классификатор обычно обучается только на подмножестве этих областей. Это позволяет избежать вычислительных трудностей при работе со всем набором областей, однако это приводит к неоптимальной производительности детектора. Метод Max-Margin Object Detection (MMOD) [6] не выполняет подвыборку, а оптимизирует все области. MMOD может быть использован для улучшения методов обнаружения объектов, таких как HOG или каскадный классификатор Хаара.

5.6. Multi-Task Cascaded Neural Network

Метод обнаружения и выравнивания лиц Multi-Task Cascaded Neural Network (MTCNN) описан в статье [17] в 2016 году. Конвейер каскадного метода обнаружения лиц делится на три основные стадии:

- используем сверточную сеть, сеть предложений (P-Net), чтобы получить области-кандидаты и их ограничивающие рамки. Затем мы используем оцененные векторы регрессии ограничивающего прямоугольника для калибровки кандидатов. После этого мы используем алгоритм Non-Maximum Suppression (Fast NMS) [4] для слияния сильно перекрывающихся кандидатов (см. Рис.11);

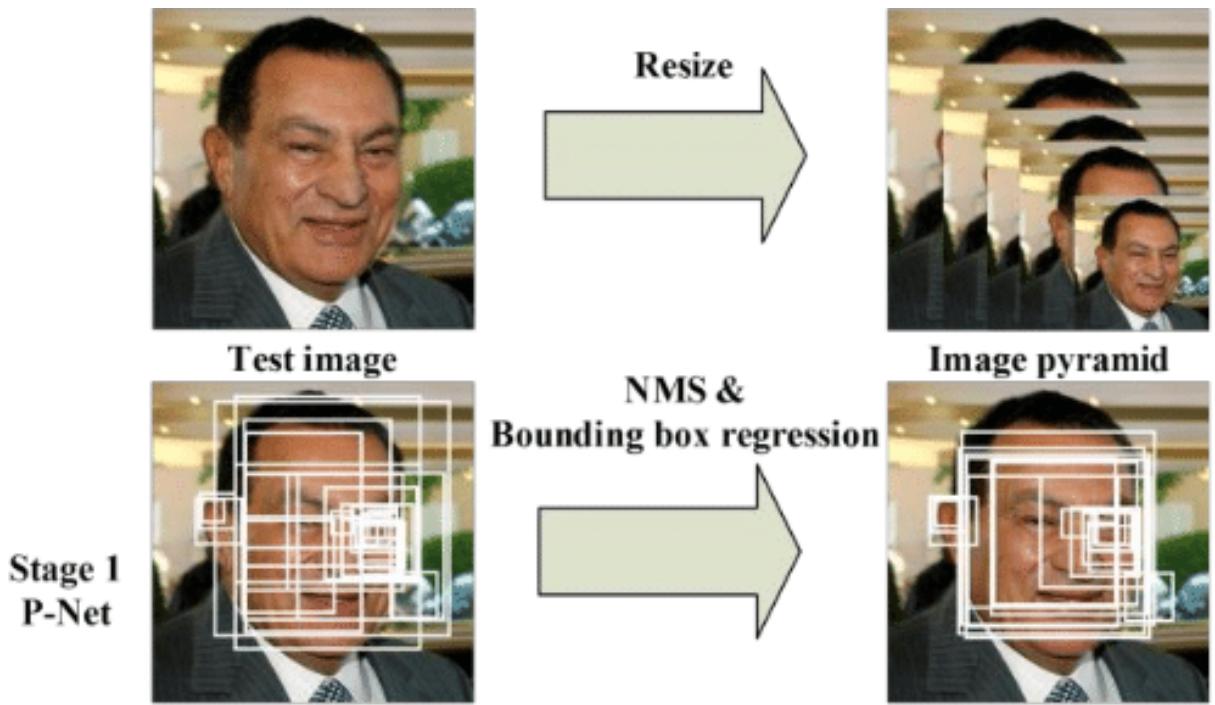


Рис. 11: Поиск областей-кандидатов (источник: <https://arxiv.org/ftp/arxiv/papers/1604/1604.02878.pdf>, дата обращения: 17.05.2020)

- все области-кандидаты подаются на другой слой CNN, называемый Refine Network (R-Net), который дополнительно отклоняет большое количество ложных кандидатов, выполняет калибровку с помощью регрессии ограничивающего прямоугольника. Далее происходит слияние кандидатов с помощью алгоритма NMS (см. Рис.12);

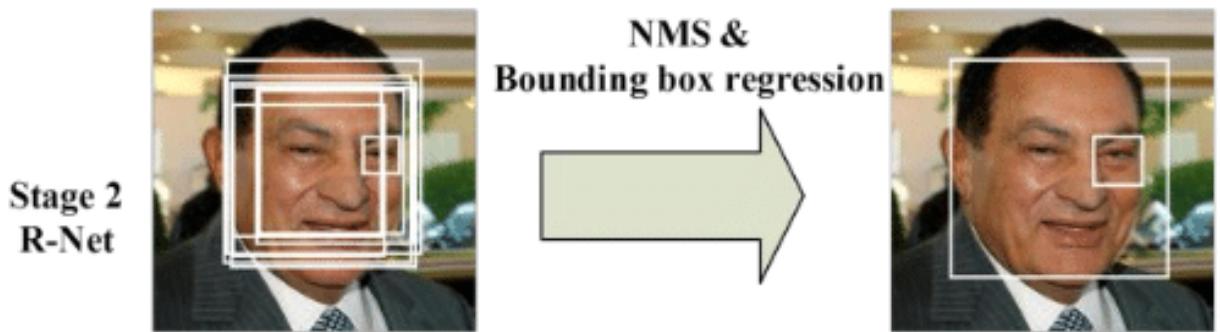


Рис. 12: Уточнение областей-кандидатов (источник: <https://arxiv.org/ftp/arxiv/papers/1604/1604.02878.pdf>, дата обращения: 17.05.2020)

- третья стадия похожа на вторую, но на этой стадии мы стремимся описать лицо более подробно. В частности, сеть будет выводить на экран пять лицевых ориентиров (нос, глаза, уголки рта) (см. Рис. 13).

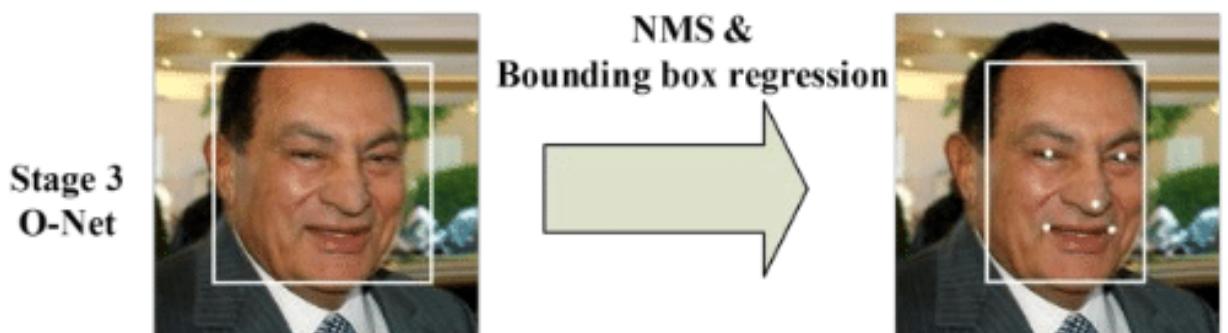


Рис. 13: Вывод окончательных границ и точек-ориентиров лица (источник: <https://arxiv.org/ftp/arxiv/papers/1604/1604.02878.pdf>, дата обращения: 17.05.2020)

6. Результаты тестирования

6.1. Результаты лучших решений

Рис. 14 демонстрирует результаты решений, показавших лучшие результаты на наборе данных Wider Faces. Видно, что F-мера трех лучших решений близка к единице, что говорит об очень неплохой точности алгоритмов. В дальнейшем планируется интегрировать эти решения под платформу .NET и протестировать на других наборах данных. Далее будем оценивать наши остальные решения, исходя из того, что знаем лучшие результаты.

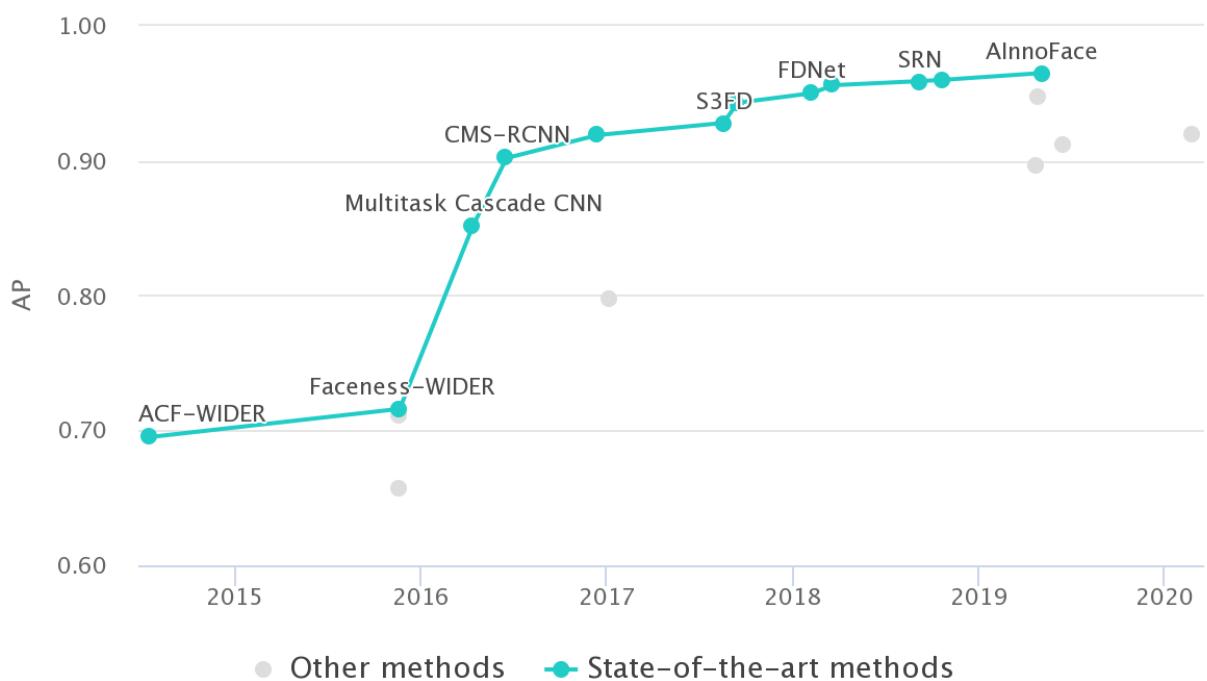


Рис. 14: График «Face Detection on Wider Faces» (источник: <https://paperswithcode.com/paper/wider-face-a-face-detection-benchmark>, дата обращения: 17.05.2020)

6.2. Результаты решений, интегрированных под платформу .NET

	Recall	Precision	F-Score
LBP	0.29	0.86	0.43
HOG	0.38	0.95	0.54
Haar	0.45	0.91	0.60
MMOD	0.52	0.86	0.65
SSD	0.82	0.66	0.73
MTCNN	0.86	0.89	0.87

Таблица 1: Результаты существующих решений под .NET

Как видно из таблицы 1, лучшие результаты показывают MTCNN и SSD. Однако SSD имеет достаточно низкий параметр Precision, что вызвано высоким содержанием False-Positive обнаружений.

6.3. Результаты тестирования скорости при работе на CPU

Скорость будем тестировать на изображении размером 300x300. Каждый алгоритм обработает изображение 10000 раз.

Hardware:

- Processor : Intel Core i7 6850K – 6 Core;
- RAM : 32 GB;
- GPU : NVIDIA GTX 1080 Ti with 11 GB RAM;
- OS : Ubuntu 16.04 LTS;
- Programming Language : C#.

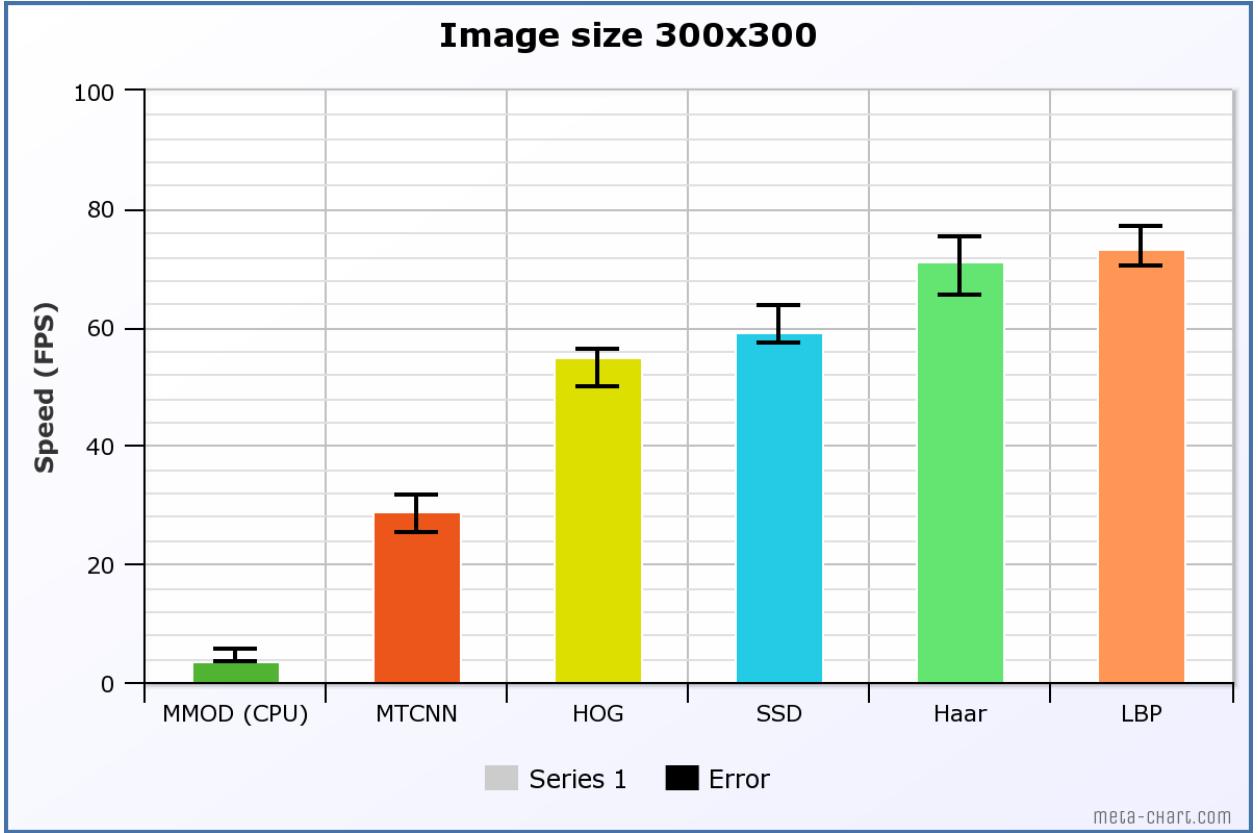


Рис. 15: Средняя скорость работы с минимальным и максимальным временем

Самое лучшее решение по свойству быстродействия – LBP. Также довольно быстро работают каскады Хаара, SSD и HOG.

6.4. Тест на поворот лица

Рис. 16 показывает как разные решения справляются с ситуацией, в которой лицо повернуто на некоторый градус. Лучше всего с этой задачей справляется SSD. Это решение обнаруживает лицо с углом поворота вплоть до 180° .

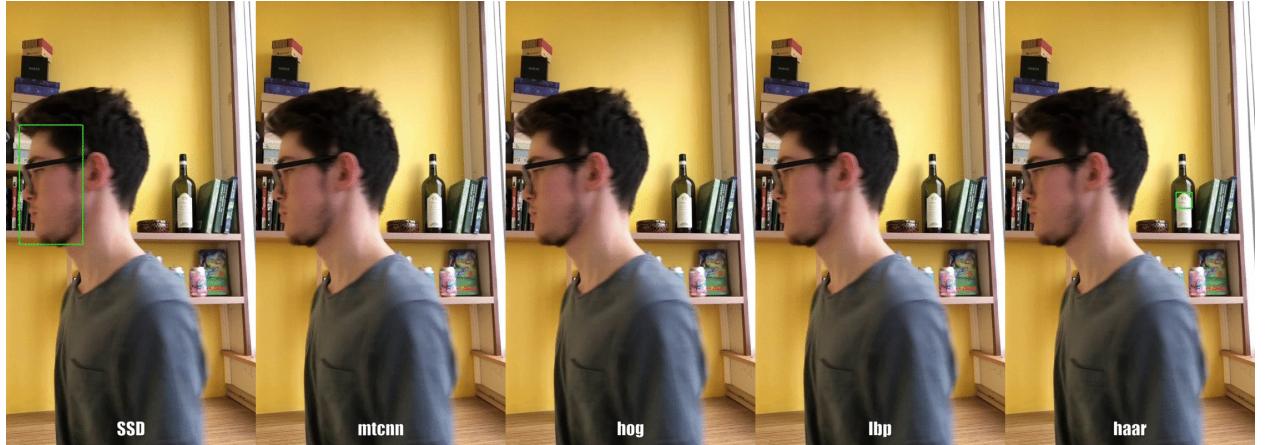


Рис. 16: Обнаружение лица под углом

6.5. Тест на окклюзию

Тест на окклюзию изображен на Рис. 17. Лучшие результаты показывают решения SSD и MTCNN. Они обнаруживают лицо даже, когда оно прикрыто наполовину.

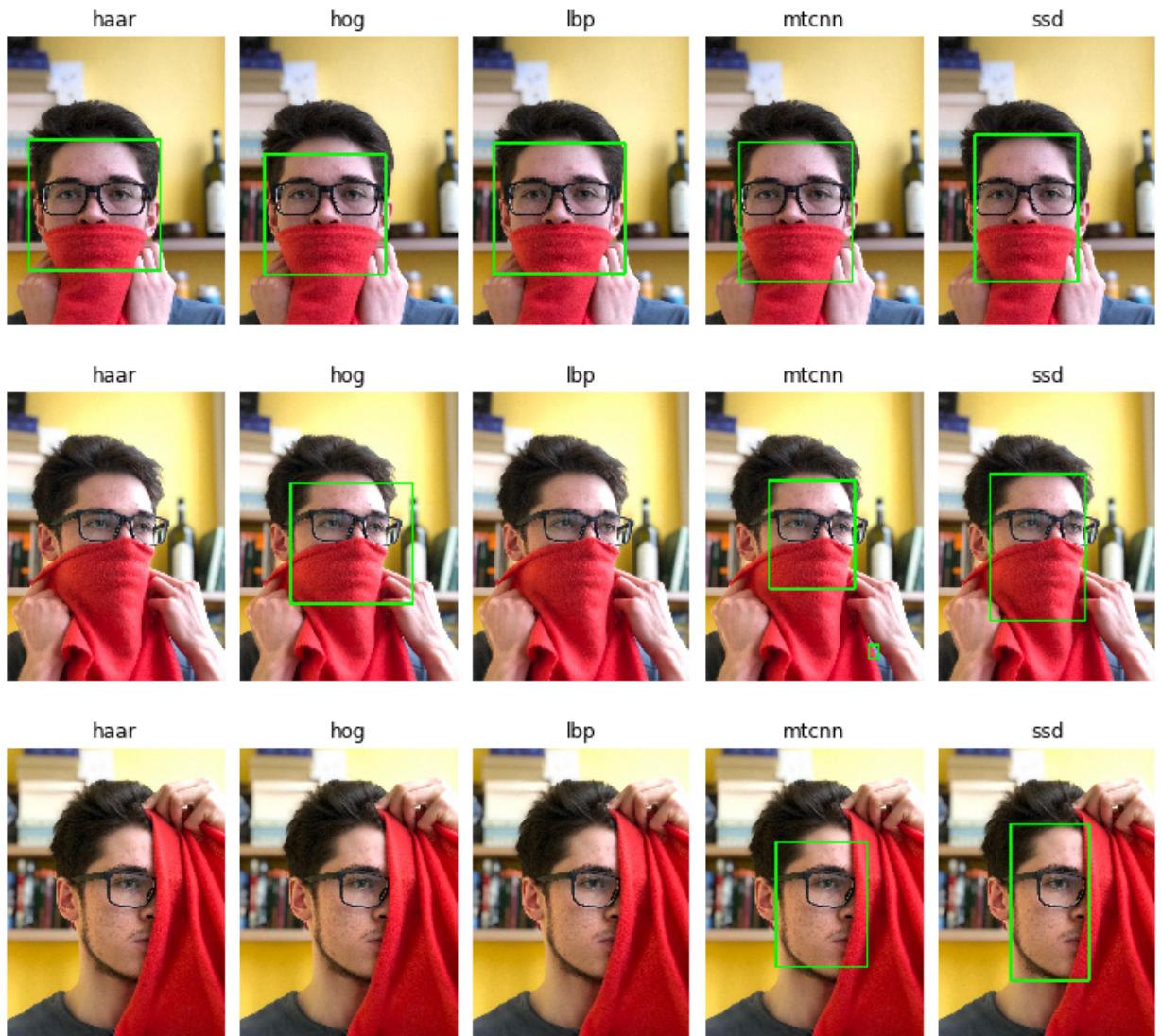


Рис. 17: Обнаружение частично прикрытого лица

7. Результаты

Исходя из поставленных целей и задач, были получены следующие результаты:

- создана система для тестирования различных решений;
- рассмотрены существующие решения детектирования лиц на изображениях;
- проведено полное сравнение всех решений и выбрано лучшее – Single-Shot-Multibox Detector;
- <https://github.com/Feodoros/BelkaFaces/blob/master/WiderFaces.ipynb>.

Список литературы

- [1] Dalal N., Triggs B. Histograms of oriented gradients for human detection // 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). — Vol. 1. — 2005. — June. — P. 886–893 vol. 1.
- [2] Detecting Masked Faces in the Wild With LLE-CNNs / Shiming Ge, Jia Li, Qiting Ye, Zhao Luo // The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
- [3] FDDB: A Benchmark for Face Detection in Unconstrained Settings : Rep. : UM-CS-2010-009 / University of Massachusetts, Amherst ; Executor: Vudit Jain, Erik Learned-Miller : 2010.
- [4] Jan Hosang Rodrigo Benenson Max Planck. Learning non-maximum suppression // <https://arxiv.org/>. — 2017. — URL: <https://arxiv.org/pdf/1705.02950.pdf> (online; accessed: 17.05.2020).
- [5] Karen Simonyan Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition // <https://arxiv.org/>. — 2015. — URL: <https://arxiv.org/pdf/1409.1556.pdf> (online; accessed: 17.05.2020).
- [6] King Davis E. Max-Margin Object Detection // <https://arxiv.org/>. — 2015. — URL: <https://arxiv.org/pdf/1502.00046.pdf> (online; accessed: 17.05.2020).
- [7] Paul Viola Michael Jones. Rapid Object Detection using a Boosted Cascade of Simple Features // <https://www.cs.cmu.edu/>. — 2001. — URL: <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf> (online; accessed: 11.12.2019).
- [8] Paul Viola Michael Jones. Robust real-time face detection // International Journal of ComputerVision, 57 (2004), pp. 137–154. — 2004. — URL: <http://dx.doi.org/10.1023/B:VISI.0000013087.49260.fb> (online; accessed: 11.12.2019).

- [9] SFace: An Efficient Network for Face Detection in Large Scale Variations / Jianfeng Wang, Ye Yuan, Gang Yu, Sun Jian. — 2018. — 04.
- [10] SSD: Single Shot MultiBox Detector / Wei Liu, Dragomir Anguelov, Dumitru Erhan et al. // Lecture Notes in Computer Science. — 2016. — P. 21–37. — URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.
- [11] Shuo Yang Ping Luo Chen Change Loy Xiaoou Tang. WIDER FACE: A Face Detection Benchmark // <https://arxiv.org/>. — 2015. — URL: <https://arxiv.org/pdf/1511.06523.pdf> (online; accessed: 17.05.2020).
- [12] T. Ojala M. Pietikäinen, Harwood D. Local Binary Patterns // A Comparative Study of Texture Measures with Classification Based on Feature Distributions. — Vol. 29. — 1996. — P. 51–59.
- [13] Wikipedia. Support vector machine // Википедия, свободная энциклопедия. — 1995. — URL: https://en.wikipedia.org/wiki/Support_vector_machine (online; accessed: 17.05.2020).
- [14] Wikipedia. AdaBoost // Википедия, свободная энциклопедия. — 2019. — URL: <https://en.wikipedia.org/wiki/AdaBoost> (online; accessed: 11.12.2019).
- [15] Wikipedia. Ensemble learning // Википедия, свободная энциклопедия. — 2019. — URL: https://en.wikipedia.org/wiki/Ensemble_learning (online; accessed: 17.05.2020).
- [16] Wikipedia. Supervised learning // Википедия, свободная энциклопедия. — 2019. — URL: https://en.wikipedia.org/wiki/Supervised_learning (online; accessed: 11.12.2019).
- [17] Xiang J., Zhu G. Joint Face Detection and Facial Expression Recognition with MTCNN // 2017 4th International Conference on

Information Science and Control Engineering (ICISCE). — 2017. —
P. 424–427.