

Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных
систем

Жилкин Федор Игоревич

Анализ решений задачи детекции лиц на
изображениях в сфере
киберкриминалистики

Курсовая работа

Научный руководитель:
к. т. н., доц. Литвинов Ю. В.

Санкт-Петербург
2020

Оглавление

Введение	3
1. Постановка задачи	5
2. Набор данных Wider Faces	6
3. Выбор решений распознавания лиц	7
4. Схема тестирования	8
4.1. Метрика IoU	8
4.2. Классы TP, FP, FN	9
4.3. Метрики Recall, Precision, F-мера	10
4.4. Алгоритм тестирования	10
5. Тестируемые решения	11
5.1. Каскады Хаара	11
5.2. Гистограмма ориентированных градиентов	15
5.3. Single-Shot-Multibox Detector	15
5.4. Local Binary Patterns	16
5.5. Max-Margin Object Detection	18
5.6. Multi-Task Cascaded Convolutional Networks	18
6. Результаты тестирования	21
6.1. Результаты лучших решений	21
6.2. Результаты решений, интегрированных под платформу .NET	21
6.3. Результаты тестирования скорости при работе на CPU .	22
6.4. Тест на поворот лица	22
6.5. Тест на окклюзию	23
7. Результаты	24
Список литературы	25

Введение

С продвижением технического прогресса связана безопасность людей, ведь благодаря новейшим технологиям стало куда легче и быстрее пресекать преступления, изолировать людей, совершивших злодеяния. Одно из важнейших обстоятельств является появление камер видеонаблюдения на улицах, в метро и общественных местах. Таким образом службы безопасности разных стран могут отслеживать преступников для их дальнейшей изоляции. Но количество информации, поступающей с этих камер растет с каждым днем и становится необходимым автоматизация обработки полученных видео и фотографий. Решить эту проблему помогает автоматическое распознавание лиц для дальнейшей их верификации (определение, 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 классификация (выяснение, принадлежит ли данное лицо человеку, находящемся в базе данных).

Любая задача, связанная с лицами, начинается с их обнаружением на фотографиях или видео. Данная работа посвящена как раз распознаванию лиц (face detection). В ней будут рассмотрены основные методы распознавания лиц, проведена сравнительная характеристика и будут выяснены лучшие решения 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;
- так как нет гарантии, что криминалисты будут использовать графические ускорители при работе с программой, а тяжелые решения (например сверточные глубокие нейронные сети) требуют большой вычислительной мощности, что ведет к долгой работе на CPU, появляется немаловажный аспект быстродействия готового продукта;

Данная работа проводится совместно с компанией Belkasoft, специализирующейся на создании программного обеспечения в сфере киберкrimиналистики. Также в данной работе принимает участие Чернявский Олег, который занимается кластеризацией лиц (face clustering).

1. Постановка задачи

Целью работы является обзор и сравнение существующих методов распознавания лиц на изображениях на основе факторов, приведенных в введении. Также 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. Набор данных Wider Faces

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

- окклюзия (когда лицо частично чем-то прикрыто)
- лица, расположенные под разным углом
- 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

Исходя из факторов, приведенных выше был выбран набор данных Wider Faces [8]. Это размеченный набор изображений, т.е. каждое лицо на каждом изображении имеет свои координаты, которые записаны в отдельном файле. Примеры изображений из этого набора данных приведены на Рис.1.



Рис. 1: Набор данных Wider Faces

3. Выбор решений распознавания лиц

Поскольку продукт BelkaSoft Evidence Center работает под платформой .NET, то для тестирования были выбраны решения, которые уже реализованы под данной платформой и имеют хорошую скорость при работе на CPU:

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

Также были выбраны три лучшие 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 вектора реальных координат лица (*ground_truth*) и вектора, полученного из алгоритма (*predicted_vector*), больше или равно 0.5, то говорим, что этот полученный вектор правильно обнаружил лицо. Примеры IoU метрик можно увидеть на Рис.2.

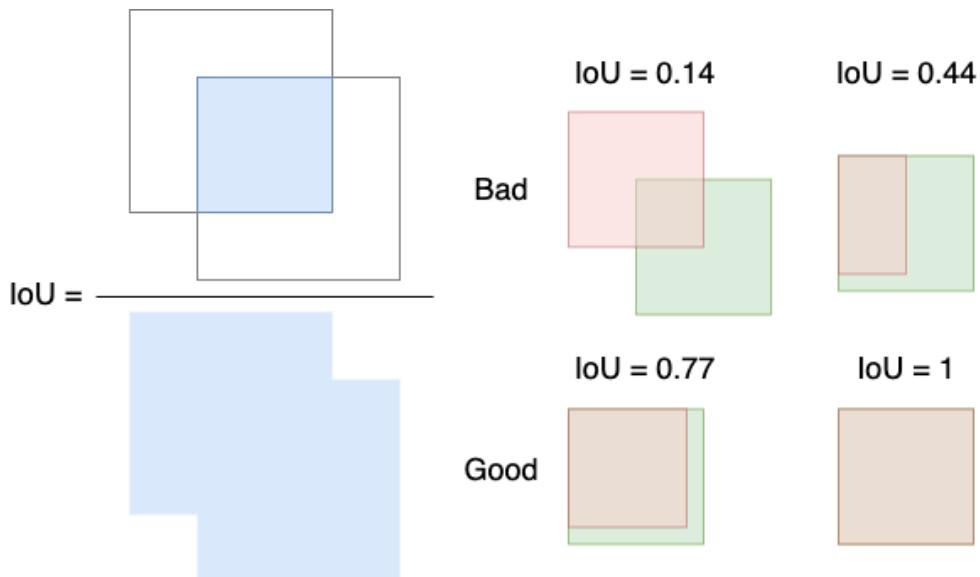


Рис. 2: Метрика Intersection-over-Union

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 $IoU(ground_truth, predicted_vector) \geq 0.5$ – True-Positive (правильно обнаруженное лицо)
- при $IoU(ground_truth, predicted_vector) < 0.5$ – False-Positive (вектор, не являющееся ground-truth лицом)
- все остальные вектора, не обнаруженные алгоритмом – False-Negative



Рис. 3: Классы TP, FP, FN

4.3. Метрики Recall, Precision, F-мера

Для оценки точности работы алгоритмов введем следующие метрики:

- $Recall = \frac{TP}{TP+FN}$ – полнота
- $Precision = \frac{TP}{TP+FP}$ – точность
- $F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}$ – среднее гармоническое Precision и Recall

4.4. Алгоритм тестирования

Для тестирования решений была выбрана легкая часть набора изображений по параметру Scale, где каждое изображение содержит лицо не меньшее, 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 TP, FP, FN. После вычисляем метрики Recall, Precision, F-мера. Схематично алгоритм тестирования изображен на Рис.4. Первая часть тестирования закончена.

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

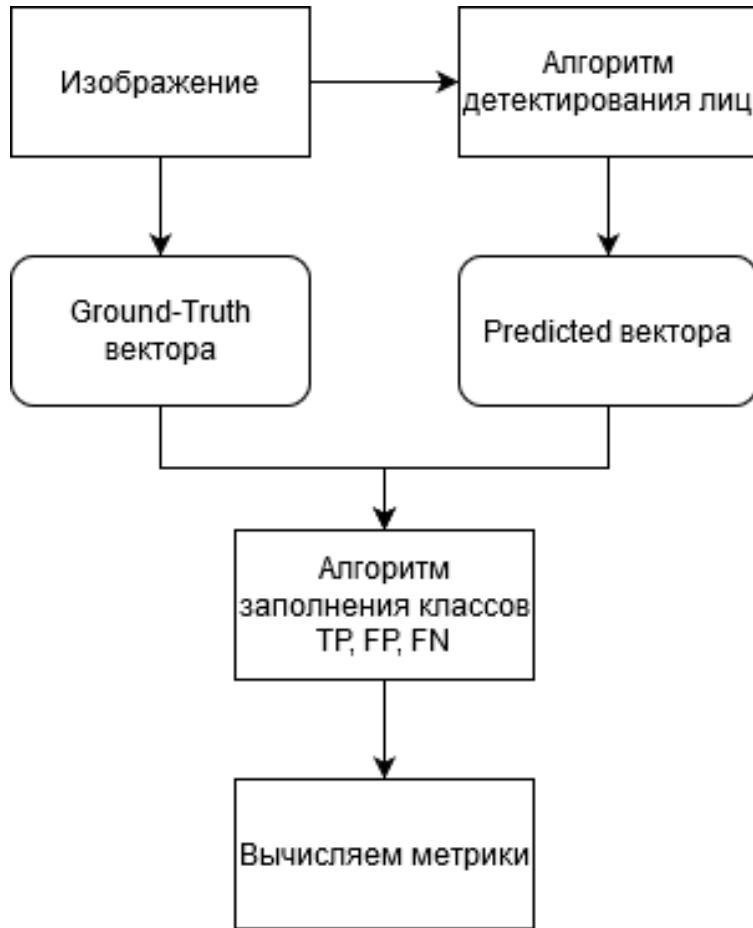


Рис. 4: Алгоритм тестирования решений

5. Тестируемые решения

Поскольку данная работа носит обзорный характер, то мы не будем подробно рассматривать все шаги каждого алгоритма, а остановимся лишь 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 [5], представляет собой особый случай ансамблированного обучения [13], называемый ускорением (boosting). Как правило он основан на классификаторах Adaboost [12]. Существует алгоритм, называемый «Средой обнаружения объектов Viola – Jones» (Viola–Jones object detection framework), который включает в себя все этапы, необ-

ходимые для обнаружения лица:

- выбор характеристик (признаков) Хаара, особенности, полученные из вейвлетов Хаара [10].
- интегральное представление изображения
- Adaboost Training
- каскадный классификатор

Рассмотрим некоторые шаги алгоритма:

Выбор характеристик Хаара.

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

- область темных глаз по сравнению с щеками
- яркая область переносицы по сравнению с глазами
- какое-то конкретное расположение глаз, рта, носа и т.д.

Процесс извлечения этих характеристик продемонстрирован на Рис. 5.

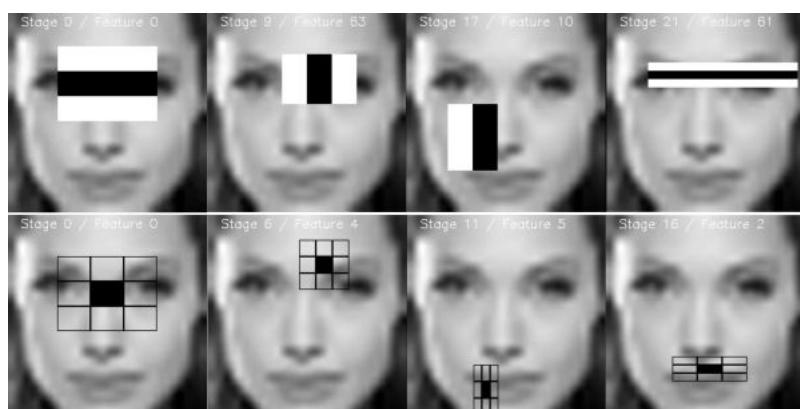


Рис. 5: Извлечение характеристик (feature extraction)

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, намного быстрее, поэтому следующий шаг после выбора характеристик – это интегральное представление изображения [6].

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

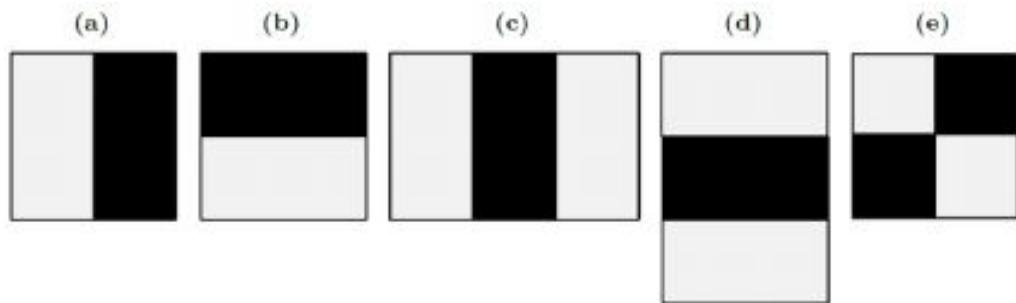


Рис. 6: Прямоугольники Хаара (Haar's rectangles)

Теперь, когда характеристики выбраны, мы применяем их к набору обучающих изображений, используя классификацию Adaboost, которая представляет собой набор слабых классификаторов, для создания точной модели ансамбля (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, которые, 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, то есть области, которые содержат лицо и не были идентифицированы как 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.

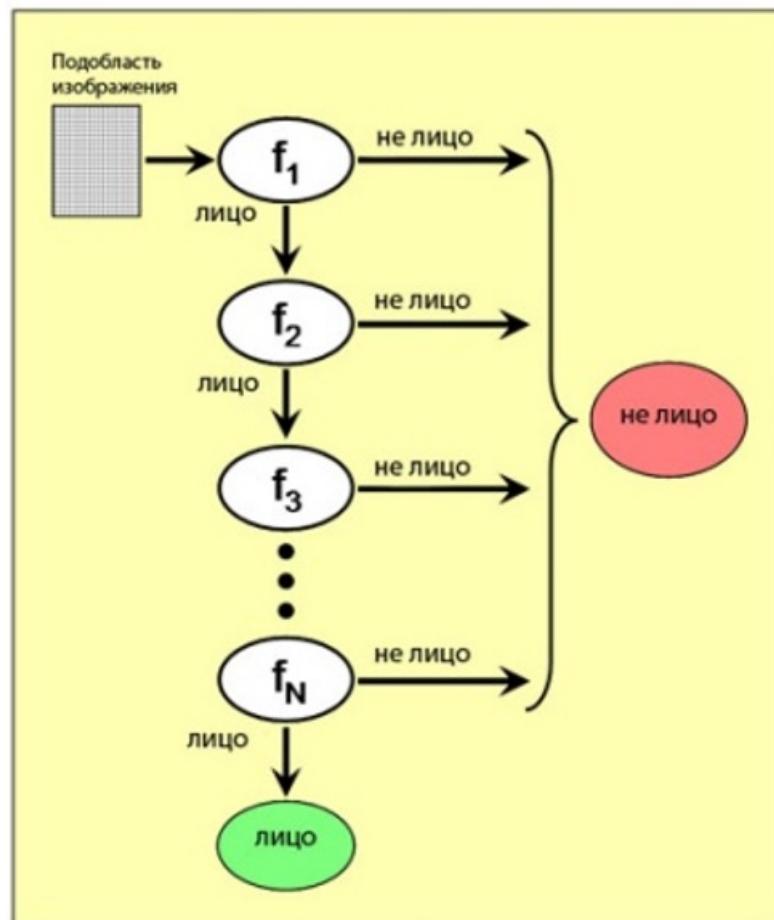


Рис. 7: Каскад (Haar's cascade)

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 [14]. Создатели алгоритма использовали метод опорных векторов (SVM, Support Vector Machine) [11].

5.3. Single-Shot-Multibox Detector

Данный метод, Single-Shot-Multibox Detector(SSD), впервые описан в 2016 в оригинальной статье [7], 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 – так называется методика поиска ограничивающего прямоугольника
- Detector – нейронная сеть работает как классификатор (детектор) объектов

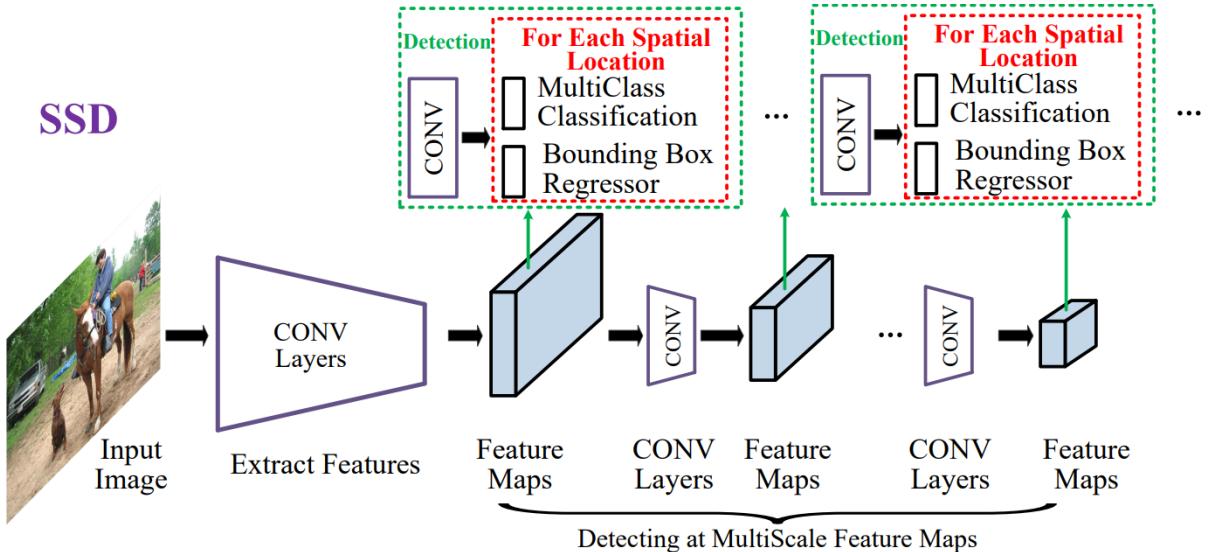


Рис. 8: Схема работы SSD

Рис.8 демонстрирует схему работы данного подхода. К входному изображению применяется некоторое количество сверточных слоев, причем часть из этих слоев взято из модели VGG16 [3]. Пространственная размерность изображения убывает после каждого слоя, доходя до единицы. Далее к промежуточным feature-map'ам применяется блок «Detector and classifier», на выходе которого имеем детекцию, т.е. прямоугольники с классом. Затем эти детекции подаются вход алгоритма Fast Non-Maximum Suppression (Fast NMS) [2]. Этот алгоритм объединяет все прямоугольники для получения финального результата.

5.4. Local Binary Patterns

Local Binary Patterns (LBP) – простой оператор, используемый для классификации текстур. Впервые был описан в оригинальной статье [9] в 1996 году. Концепцию работы решения можно изложить следующим образом:

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

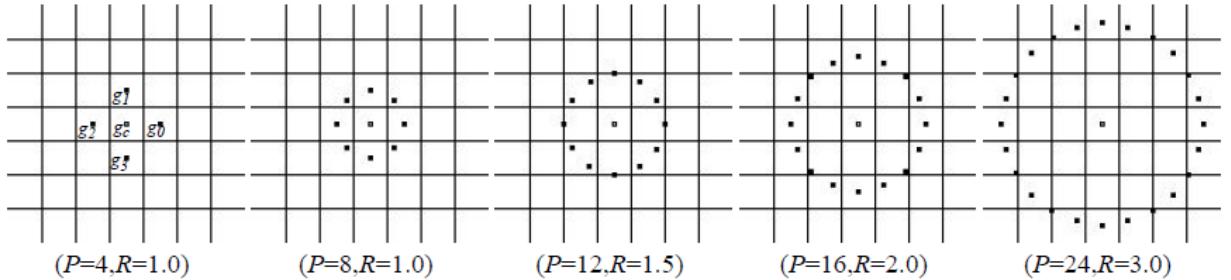


Рис. 9: Выбор точек для построения LBP

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

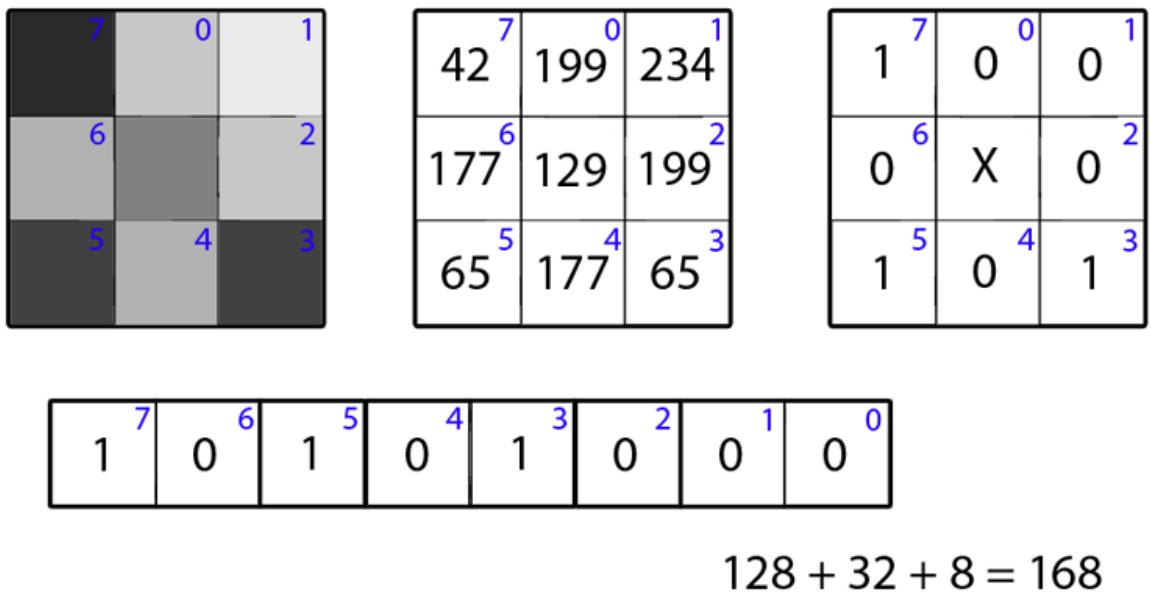


Рис. 10: Полученная гистограмма смены интенсивности в области

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 изображения, 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 не оптимальной производительности детектора. Метод Max-Margin Object Detection (MMOD) [4] не выполняет подвыборку, а оптимизирует все области. MMOD может быть использован для улучшения любого метода обнаружения объектов, который является линейным по изученным параметрам, таким как HOG или каскадный классификатор Хаара.

5.6. Multi-Task Cascaded Convolutional Networks

Метод обнаружения и выравнивания лиц Multi-Task Cascaded Convolutional Networks (MTCNN) описан в статье [15] в 2016 году. Водопровод каскадного метода обнаружения лиц делится на три основные стадии:

- используем сверточную сеть, называемой сетью предложений (R-Net), чтобы получить области-кандидаты и их ограничивающие рамки регрессионных векторов. Затем мы используем оцененные векторы регрессии ограничивающего прямоугольника для калибровки кандидатов. После этого мы используем алгоритм Non-Maximum Suppression (Fast NMS) [2] для слияния сильно перекрывающихся кандидатов (см. Рис.11)

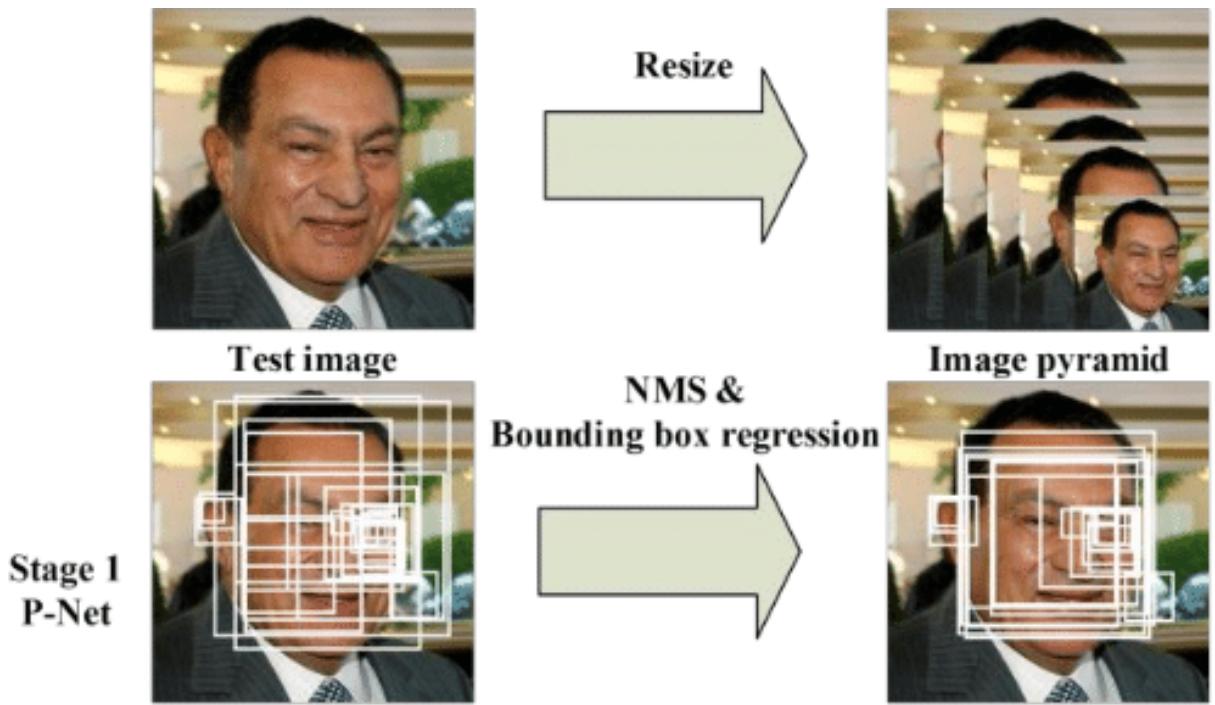


Рис. 11: Поиск областей-кандидатов

- 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)

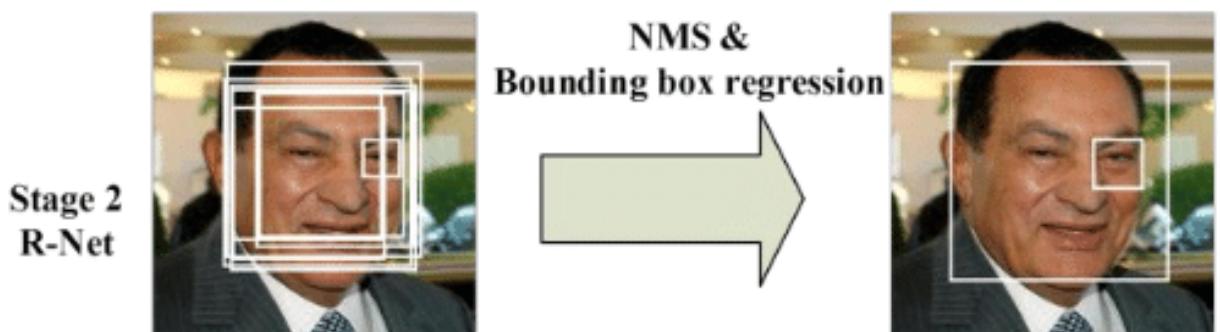


Рис. 12: Уточнение областей-кандидатов

- 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)

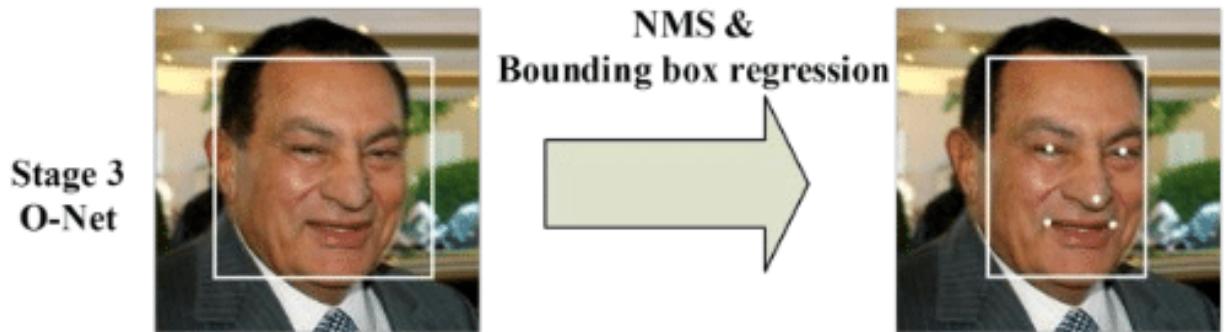


Рис. 13: Вывод окончательных границ и точек-ориентиров лица

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 перетестировать на других наборах данных. Далее будем оценивать наши остальные решения, исходя из того, что знаем лучшие результаты.

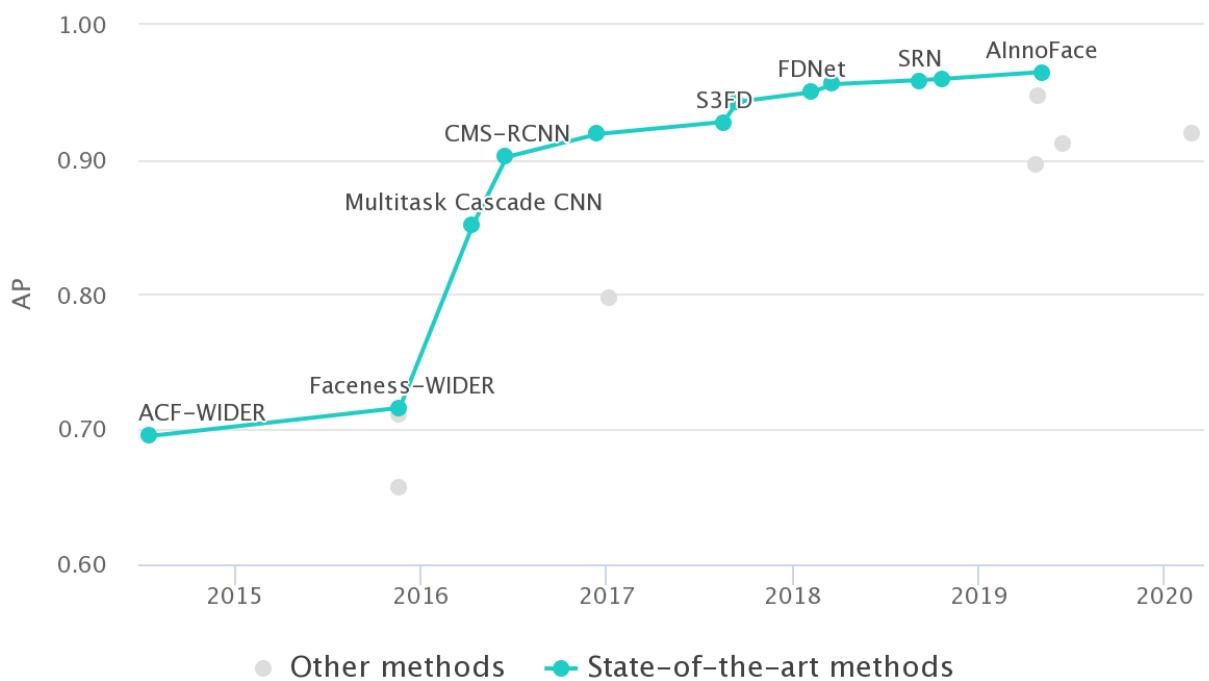


Рис. 14: График «Face Detection on Wider Faces»

6.2. Результаты решений, интегрированных под платформу .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.

	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

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#

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

6.4. Тест на поворот лица

Рис. 16 показывает как разные решения справляются со случаем, когда лицо повернуто на некоторый градус. Лучшего всего с этой задачей справляется SSD. Это решение обнаруживает лицо с углом поворота вплоть до 180°.



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

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. 16: Обнаружение лица под углом

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>



Рис. 17: Обнаружение чем-либо прикрытоого лица

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] 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).
- [3] 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).

- [4] King Davis E. Max-Margin Object Detection // <https://arxiv.org/>. — 2015. — URL: <https://arxiv.org/pdf/1502.00046.pdf> (online; accessed: 17.05.2020).
- [5] 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).
- [6] Paul Viola Michael Jones. Robust real-time face detection // International Journal of Computer Vision, 57 (2004), pp. 137–154. — 2004. — URL: <http://dx.doi.org/10.1023/B:VISI.0000013087.49260.fb> (online; accessed: 11.12.2019).
- [7] 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.
- [8] 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).
- [9] 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.
- [10] Wikipedia. Haar wavelet // Википедия, свободная энциклопедия. — 1910. — URL: https://en.wikipedia.org/wiki/Haar_wavelet (online; accessed: 17.05.2020).
- [11] Wikipedia. Support vector machine // Википедия, свободная энциклопедия. — 1995. — URL: https://en.wikipedia.org/wiki/Support_vector_machine (online; accessed: 17.05.2020).

- [12] Wikipedia. AdaBoost // Википедия, свободная энциклопедия. — 2019. — URL: <https://en.wikipedia.org/wiki/AdaBoost> (online; accessed: 11.12.2019).
- [13] Wikipedia. Ensemble learning // Википедия, свободная энциклопедия. — 2019. — URL: https://en.wikipedia.org/wiki/Ensemble_learning (online; accessed: 17.05.2020).
- [14] Wikipedia. Supervised learning // Википедия, свободная энциклопедия. — 2019. — URL: https://en.wikipedia.org/wiki/Supervised_learning (online; accessed: 11.12.2019).
- [15] 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.