|  |  |
| --- | --- |
| **СОГЛАСОВАНО**  Доцент кафедры  ИАНИ ННГУ, к.ф.-м.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_ Д.А.Яшунин  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г. | **УТВЕРЖДАЮ**  Профессор кафедры  ИАНИ ННГУ, д.т.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Старостин  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г. |

**Отчет по НИР «Разработка и реализация программного обеспечения   
для распознавания лиц на групповых фотографиях»**

**(Шифр ПО «AFR»)**

Ответственный исполнитель

\_\_\_\_\_\_\_\_\_\_\_\_ М.М. Годовицын

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

**Н. Новгород 2019**

**Оглавление**

[1 Методы распознавания лиц 2](#_Toc5538641)

[2 Методы детекции лиц 12](#_Toc5538642)

[3 Feature extraction 23](#_Toc5538649)

[4 Методы выравнивания лиц 29](#_Toc5538650)

[5 Нефункциональные требования к идентификации лиц 35](#_Toc5538650)

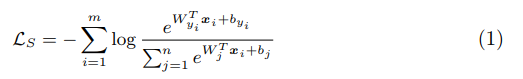
[6 Тестовый базис 37](#_Toc5538650)

1. Методы распознавания лиц

1.1 Функция потери

## Softmax

Одна из самых простых функций потерь есть softmax.



В формуле xi∈Rd обозначает признак, принадлежащий i-му элементу из датасета, yi – ground truth метка i-го элемента из датасета. d- это размер вектора признаков. Wj∈Rd обозначает j-й столбец весов W∈R(d × n) в последнем полносвязном слое, а b∈Rn– смещение, n–число классов.

## FaceNet (TripletLoss)

FaceNet использует особую функцию потерь называемую TripletLoss. Она минимизирует дистанцию между якорем и изображениями, которые содержат ту же личность, и максимизирует дистанцию между различными личностями.

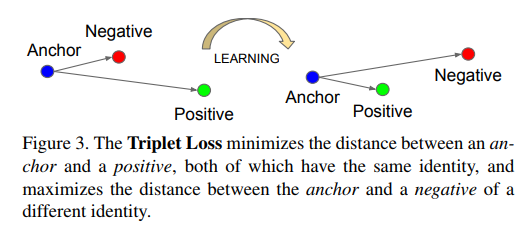
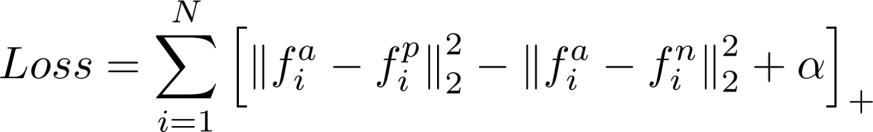


Рисунок 1. Потеря триплета минимизирует расстояние между якорем и позитивом, оба из которых являются идентичными, и максимизирует расстояние между якорем и негативом различной идентичности



* f(a) это энкодинг якоря
* f(p) это энкодинг похожих лиц (positive)
* f(n) это энкодинг непохожих лиц (negative)
* α– это константа, которая позволяет создать зазор между классами
* [sum]+– эквивалентноmax(0, sum)

## Center Loss

Функция потерь, основанная на softmax. CenterLoss потерь учитывает расстояния между объектами и расстояния между объектами и центром классов, пытаясь балансировать эти оба расстояния. В процессе обучения мы одновременно обновляем центр и минимизируем расстояния между элементами и их соответствующими классовыми центрами. Понятно, что потеря softmax вынуждает разделять признаки разных классов. Потеря центра притягивает к своим центрам признаки того же класса. При совместном обучении увеличиваются не только различия между классами, но и уменьшается внутриклассовое различие. Минус данной функции в том, что обновлять центр для более точного вычисления нужно довольно часто что иногда не совсем просто.



Cyi∈Rd обозначает центр классов глубоких признаков. В идеале, cyi должен быть обновлен по мере изменения вектора признаков каждого объекта.

Итоговая функция потерь с гиперпараметром лямбда:

C:\Users\TrueSkit\Desktop\centerloss21.PNG

, где Ls - softmax функция потерь.

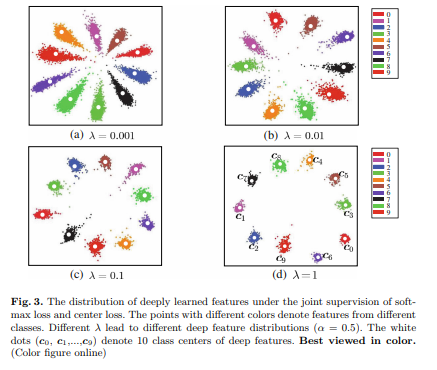


Рисунок 2. Распределение глубоко обученных признаков под совместным наблюдением потери Softmax и потери центра. Точки с разными цветами обозначают элементы из разных классов. Разные λ приводят к разным распределениям глубоких признаков (α = 0,5). Белые точки (c0, c1, ..., c9) обозначают 10 классов классов глубоких признаков.

## SphereFace (A-Softmax loss)

Исследователи решили для усиления различительной способности сети добавлять границу вокруг классов объектов, попадание в которую являлось определяющим при принятии решения о классификации объекта.  
Centerloss явно поощряет только внутриклассовую компактность.  
Модифицированый softmax(переход в угловое пространство) способна оптимизировать углы, позволяя CNN изучать углово-распределенные элементы.  
Благодаря оптимизации потерь A-Softmax области принятия решений становятся более разделенными, одновременно увеличивая межклассовую границу и уменьшая внутриклассовое угловое распределение.

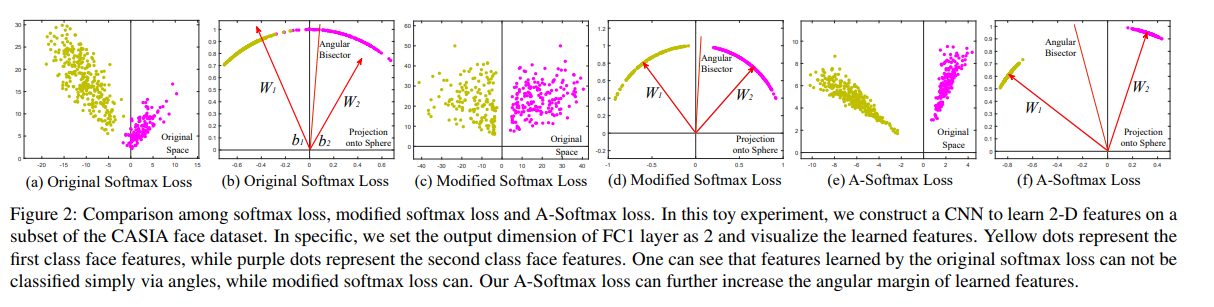
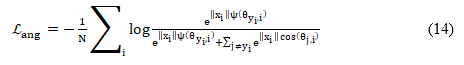


Рисунок 3. Сравнение потерь softmax, модифицированных потерь softmax и потерь A-Softmax. В этом игрушечном эксперименте мы строим CNN для изучения 2-D объектов на подмножестве набора данных casia face [8]. В частности, мы устанавливаем выходной размер слоя FC1 как 2 и визуализируем изученные объекты. Желтые точки представляют объекты первого класса, а фиолетовые - объекты второго класса. Можно видеть, что функции, изученные оригинальной потерей softmax, не могут быть классифицированы просто по углам, в то время как модифицированная потеря softmax может. Наша потеря A-Softmax может еще больше увеличить угловой запас изученных функций.

Функция потерь A-Softmax loss:

 гдеC:\Users\TrueSkit\Desktop\a-softmax2.png и θyij – угол между векторами признаков i-го примера и шаблона его истинного класса, m – параметр.

# C:\Users\TrueSkit\Desktop\a-softmax3.PNG

Рисунок 4. Геометрическая интерпретация евклидовой маржинальной потери (например, контрастная потеря, потеря триплета, потеря центра и т.д.), модифицированная потеря softmax и потеря A-Softmax. Первая строка-это ограничение 2D-объекта, а вторая строка-ограничение 3D-объекта. Оранжевая область указывает дискриминационное ограничение для класса 1, в то время как зеленая область для класса

## CosFace (Large margin cosine loss (LMCL))

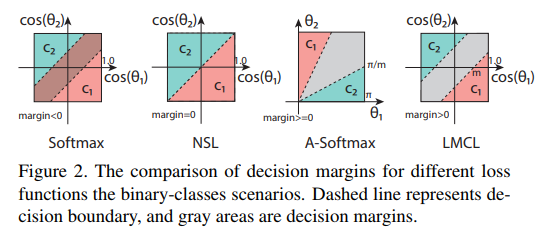
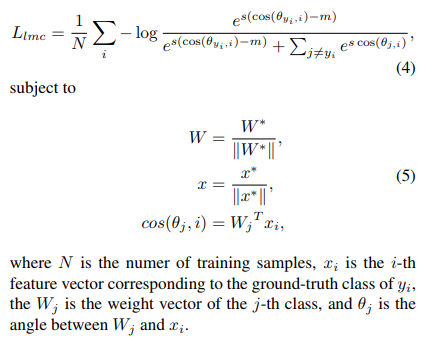
Данная функция потерь берет в себя идеи center loss, large margin softmax loss и angular softmax loss чтобы реализовать всё альтернативным подходом.  Чтобы минимальная внутриклассовая дисперсия и максимальная межклассовая дисперсия достигалась за счет нормализации и максимизации допустимого предела косинус-решения.  
Сначала они переформулируют потерю softmax как потерю косинуса с помощью L2, нормализуя как особенности, так и векторы веса, чтобы устранить радиальные отклонения, на основе которых вводится член m запаса косинуса, чтобы дополнительно максимизировать допустимый запас в угловом пространстве. Формально они определяют гиперпараметр m таким образом, чтобы граница решения определялась как cos (θ1) - m = cos (θ2), где θi - угол между признаком и весом класса i.  


Рисунок 5. Сравнение полей принятия решений для различных функций потерь в сценариях двоичных классов. Пунктирная линия представляет границу решения, а серые области-поля решения.

Функция потерь LMCL:



В a-softmax использовалось угловое пространствоcos(mθ1) = cos(θ2), что затрудняет оптимизацию из-за не монотонности. Что еще более важно, предел решения A-softmax зависит от θ, что приводит к разным запасам (margin) для разных классов. В результате в области принятия решений некоторые межклассовые признаки имеют больший запас, в то время как другие имеют меньший запас, что уменьшает способность различать.  
В отличие от A-Softmax, их подход определяет границы принятия решения в косинус-пространстве, что позволяет избежать эти недостатки.

## ArcFace(Additive Angular Margin Loss)

Создателями данной функции потерь предлагается новый способ введения границы между классами, при котором граница принятия решения о принадлежности объекта классу является окружностью фиксированного радиуса на поверхности гиперсферы единичного радиуса в пространстве признаков.  
  
**Sphereface** ввел важную идею углового запаса, их функция потерь требует ряда приближений для вычисления, что привело к нестабильному обучению сети. Чтобы стабилизировать тренировку, они предложили гибридную функцию потери, которая включает стандартную потерю softmax. Эмпирически, потеря softmax доминирует в процессе обучения, потому что мультипликативный угловой запас на основе целочисленных значений делает целевую логит-кривую очень стремительной и, следовательно, препятствует сходимости. **CosFace** напрямую добавляет штраф по косинусному краю к целевому логиту, который обеспечивает лучшую производительность по сравнению со **SphereFace**, но допускает гораздо более простую реализацию и избавляет от необходимости совместного наблюдения за потерями softmax.

Они предлагают **AdditiveAngularMarginLoss(ArcFace)**, чтобы дополнительно улучшить дискриминационную силу модели распознавания лиц и стабилизировать процесс обучения.

Как показано на рисунке 6, скалярное произведение между фичей DCNN и последним полносвязным слоем равно расстоянию косинуса после нормализации элемента и веса. Мы используем функцию арккосинус для вычисления угла между текущей фичей и целевым весом. После этого мы добавляем аддитивное угловое поле к целевому углу и снова получаем целевой логит с помощью функции косинуса. Затем мы масштабируем все логиты по фиксированной норме, и последующие шаги точно такие же, как в softmax функции потерь.

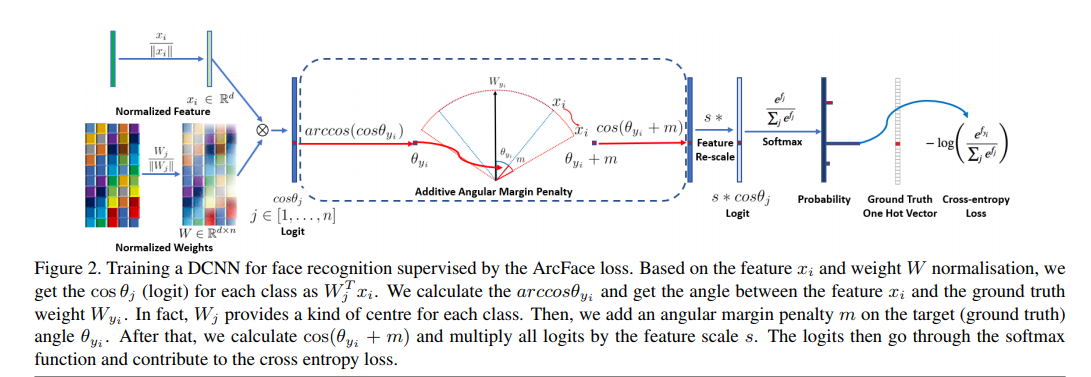


Рисунок 6. Тренировка CNN для распознавания лиц, контролируемого потерей лица дуги. Основываясь на функции xi и нормализации веса W, мы получаем cos θj (logit) для каждого класса как WT j x i. Мы вычисляем arccosθyi и получаем угол между функцией xi и весом земной истины Wyi. Фактически, Wj предоставляет своего рода центр для каждого класса. Затем мы добавляем угловой штрафной штраф m на целевой (наземный) угол θyi. После этого мы вычисляем cos (θyi + m) и умножаем все логиты на масштаб объектов s. Логиты затем проходят через функцию softmax и способствуют потере перекрестной энтропии.

Функция потерь Additive Angular Margin Loss:



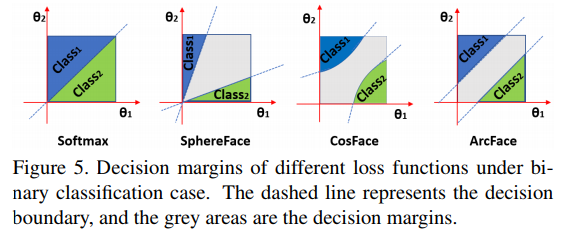


Рисунок 7. Поля принятия решений различных функций потерь в случае двоичной классификации. Пунктирная линия представляет границу решения, а серые области-поля решения.

1.2 Перенос обучения (Transfer Learning)

Технология переноса обучения позволяет использовать готовые нейронные сети для решения задач нового типа, не тех, для которых сети предварительно обучались.  
  
В качестве примера мы рассмотрим, как можно применить предварительно обученные нейронные сети для распознавания кошек и собак.  
  
С помощью технологии переноса обучения мы можем изменить архитектуру предварительно обученной сети таким образом, чтобы она подходила для решения нашей новой задачи. Измененная сеть затем обучается на новом наборе данных.

Нейронные сети, обученные для решения задач классификации изображений, состоят из двух частей:

* **Сверточная часть** используется для выделения характерных признаков из изображения.
* **Полносвязная часть** реализует классификацию - определяет, что за объект находится на изображении на основе признаков, которые извлекла сверточная часть.

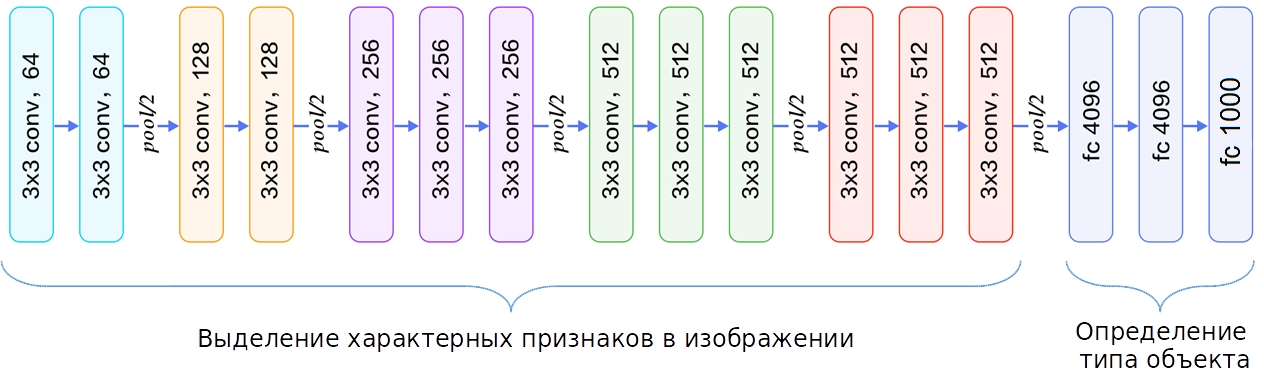
Идея переноса обучения заключается в следующем. Сверточная часть сети во время обучения учится выделять характерные признаки на изображениях. Если признаки получились достаточно общими, то мы можем взять их и применить для решения другой задачи классификации. Таким образом, мы *переносим* обучение сверточной части сети на новую задачу.  
  
Для реализации переноса обучения нам нужно заменить классификатор в предварительно обученной нейронной сети. Давайте рассмотрим как это сделать на примере сети VGG16. Эта сеть достаточно просто устроена и ее легко понять, но в то же время качество работы сети довольно высокое. Архитектура сети VGG16 показана на рисунке 1.  
  


Рисунок 8. Архитектура VGG16  
  
Сверточная часть сети VGG16 состоит из пяти каскадов свертки и подвыборки. В первых двух каскадах используются по два слоя свертки и слой подвыборки с выбором максимального значения (maxpooling). На трех следующих каскадах по три слоя свертки и один слой подвыборки. Размер ядер во всех слоях свертки 3х3.

Полносвязная часть сети VGG16 включает три уровня. На выходном уровне 1000 нейронов по количеству классов объектов. Используется формат one-hoten coding: значение только одного выходного нейрона должно быть близко к единице, остальные близки к нулю. Класс объекта на картинке соответствует нейрону, значение которого близко к единице. Перед выходным слоем в сети VGG16 еще два полносвязных слоя по 4096 нейронов.

На первом этапе необходимо убрать полносвязную часть из сети VGG16. Получится сеть как на рисунке 2.

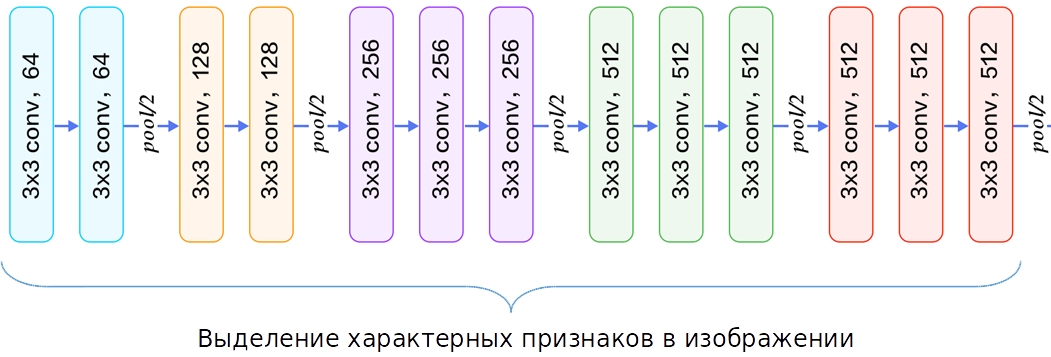


Рисунок 9. Архитектура VGG16 без полносвязного слоя

Второй этап: к сверточной части сети VGG16 добавляем новый классификатор (Рисунок 3).

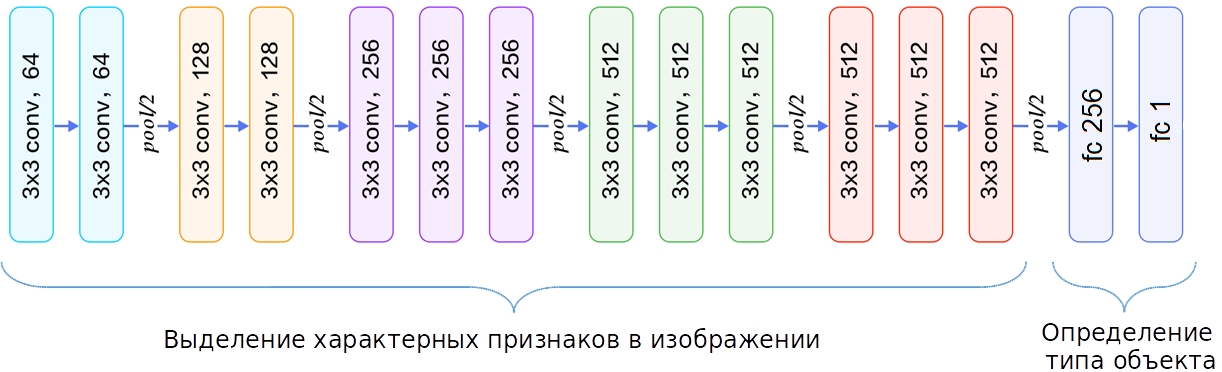


Рисунок 10. Архитектура VGG16 c новым классификатором (кошка или собака)

Новый классификатор устроен гораздо проще полносвязной части сети VGG16, т.к. нам нужно распознавать всего два класса объектов, а не 1000. На выходном слое один нейрон, что соответствует задаче бинарной классификации. Ноль на выходе из сети означает, что на фотографии кот, а единица - собака. Перед выходным слоем находится еще один полносвязный слой, в котором 256 нейронов. На вход этого слоя поступают данные из сверточной части сети VGG16.

На третьем этапе измененную сеть нужно обучить на новом наборе данных с фотографиями котов и собак.

В процессе обучения новой сети нужно запретить обучать сверточную часть сети, в противном случае веса в сети могут испортиться в процессе обучения с новым классификатором. В классификаторе, который мы добавим к сети, веса нейронов будут инициализированы случайными числами. Поэтому на первых этапах обучения значения ошибки на выходе из сети будет очень большим. По алгоритму обратного распространения ошибки сигнал об ошибке будет передаваться и в сверточную часть сети VGG16, из-за чего веса в ней могут испортиться.

2. Методы детекции лиц

# 2.1 Regional Convolution Neural Network (RCNN)

Рисунок 1. Схема работы RCNN.

**Входные данные.**

Область изображения, представленная вектором размерности 3x227x227, найденная селективным поиском.

**Выходные данные.**

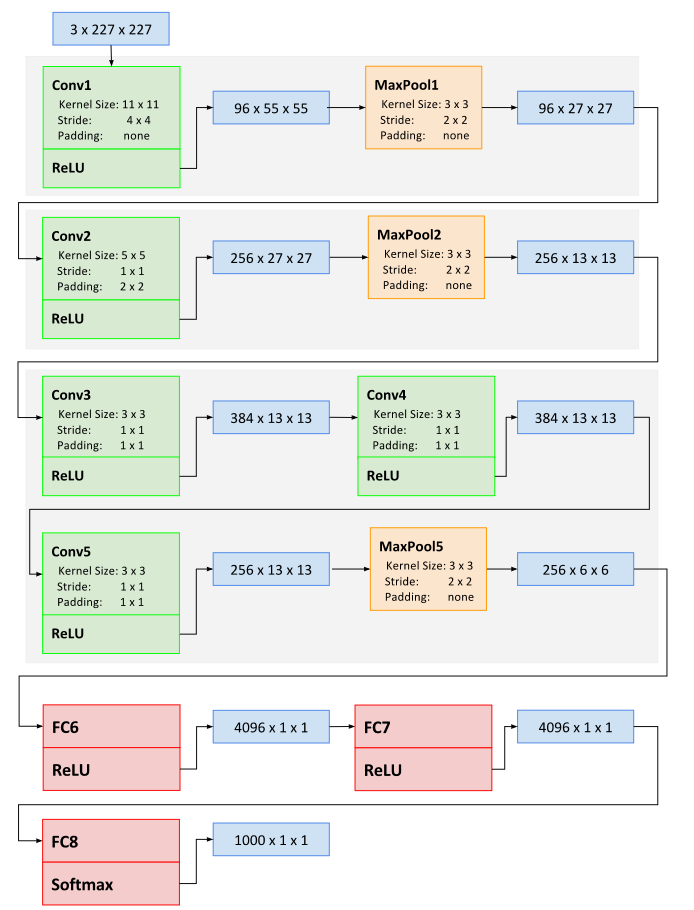
Координаты найденных ограничительных прямоугольников, содержащих искомый объект, и результаты их классификации.

Рисунок 2. Архитектура RCNN.

**Производительность.**

50 секунд на изображение (GPU)

**Алгоритм работы.**

С помощью селективного поиска на входном изображении выделяются области (ограничительные рамки), предположительно содержащие искомый объект (гипотезы). Похожие гипотезы объединяются вместе жадным алгоритмом. Далее трансформируются в квадрат (3x227x227) и подаются на вход свёрточной нейронной сети, которая генерирует карту признаков, представленную вектором (4096x1x1). Далее производится классификация либо с использованием метода опорных векторов, либо с помощью softmax слоя (в таком случае получим ускорение, но точность может снизиться в среднем на 4%). Потом с помощью линейного регрессора уточняется местоположение ограничительной рамки с помощью прямой регрессии координат.

**Комментарий.**

Занимает много времени: алгоритм поиска областей является фиксированным (нет элемента обучения).



Рисунок 3. Селективный поиск.

# 2.2 Fast Regional Convolution Neural Network (Fast RCNN)

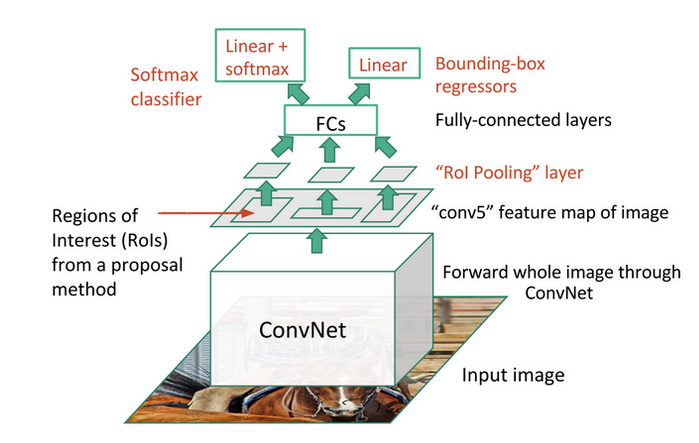
****

Рисунок 4. Схема работы Fast RCNN.

**Входные данные.**

Целое изображение (сеть самостоятельно формирует карту признаков).

**Выходные данные.**

Координаты найденных ограничительных прямоугольников, содержащих искомый объект, и результаты их классификации.

**Производительность.**

2 секунды на изображение (GPU)

Быстрее, потому что операция свёртки выполняется только один раз для каждого изображения, а не каждый раз для каждой области.

**Алгоритм работы.**

В свёрточную сеть подаются не гипотезы, а входное изображение для создания свёрточной карты признаков. Далее гипотезы проецируется на эту карту признаков и перемасштабируются (3x227x227). Далее следуют полносвязные слои, которые для каждой гипотезы вычисляют класс и смещение с помощью прямой регрессии координат ограничительных прямоугольников.

# 2.3 Faster Regional Convolution Neural Network (Faster RCNN)

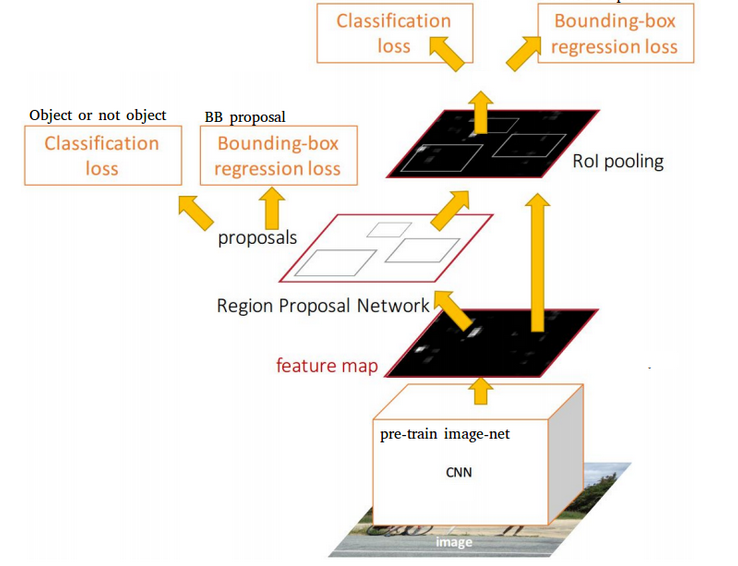


Рисунок 5. Схема работы Faster RCNN.

**Входные данные.**

Целое изображение.

**Выходные данные.**

Координаты найденных ограничительных прямоугольников, содержащих искомый объект, и результаты их классификации.

**Производительность.**

0.2 секунды на изображение.

**Алгоритм.**

В свёрточную сеть подаётся изображение, затем создаётся свёрточная карта признаков. Затем к этой карте применяются сверточный слой и два полносвязанных слоя для генерации гипотез (Regions Proposal Network). После этого полученные с её помощью ограничивающие прямоугольники используются для вырезания из исходной свёрточной карты соответствующих найденному объекту признаков.

Выходы же с этих слоёв базируются на так называемых *«якорях»*, представляющих собой k рамок для каждого положения скользящего окна, которое просматривает карту признаков. Якоря имеют разные размеры и соотношения сторон.

Слой RoI (Region of Interest) pooling принимает на вход карту признаков и гипотезы в координатах изображения. RoI преобразует координаты изображения в координаты на карте признаков. Полученный прямоугольник разбивается на сетку с заданными параметрами ( например, для VGG16 сетка 7x7), и затем производится max pooling по каждой из ячеек. Затем данные подаюся на softmax слой для оценки принадлежности преендента одному из классов объектов и слой регрессии для уточнения координат ограничительных рамок.

# 2.4 You Only Look Once (YOLO)

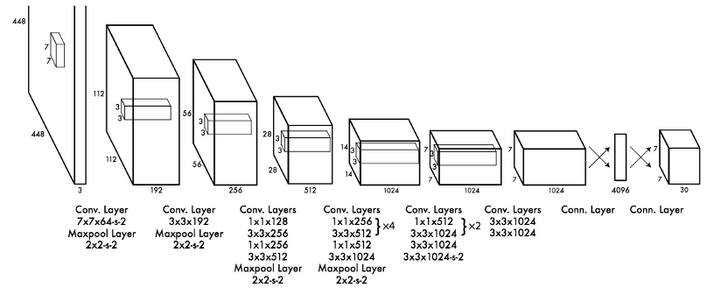


Рисунок 6. Архитектура YOLO.

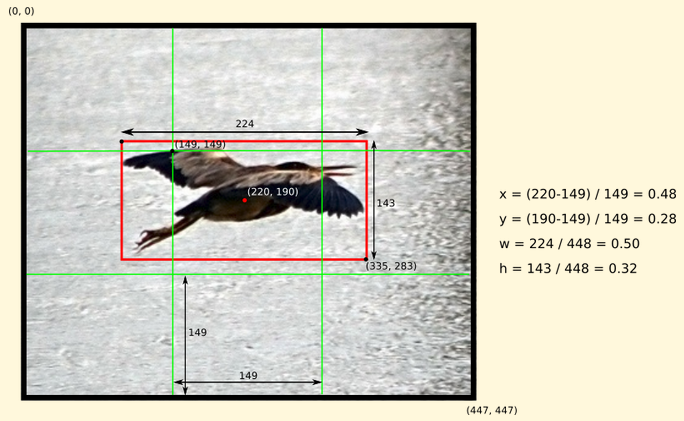
**Входные данные.**

Целое изображение.

**Выходные данные.**

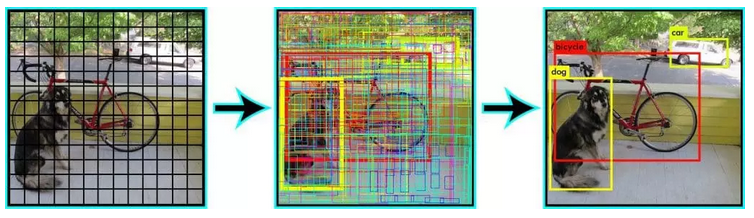
Тензор N\*N(A+B\*5), определяющий прогноз для каждой ячейки сетки, где A — оценка вероятности соответствия каждому классу, B — число рамок в ячейке, каждой рамке соответствуют 5 значений: координаты центра, ширина и высота (нормализованы в интервале [0,1]), а также её точность (=Pr(Object) \* IOU(pred, truth); если в рамке нет объектов, точность будет равна нулю). Подробнее об этом можно узнать, прочитав раздел «алгоритм».

Рисунок 7.Пример вычисления координат рамки для изображения 448x448 пикселей, N=3



**Производительность.**

55 кадров/с.



**Алгоритм.**

Делит изображение на сетку NxN, каждая ячейка сетки формирует B ограничительных рамок и значение её точности, т.е. насколько вероятно содержание в ней объекта (независимо от класса). Из сформированных NxNxB рамок часть может быть убрана из рассмотрения, если их точность не соответствует установленному порогу. Далее с помощью прямой регрессии координат уточняется положение ограничительных прямоугольников.

**Комментарий.**

YOLO Сталкивается с трудностями при детектировании объектов малого размера

Сеть имеет 24 свёрточных слоя и два полносвязных. Существует облегчённая её версия, Fast YOLO, с 9 свёрточными слоями и меньшим количеством фильтров. Работает она ещё быстрее (150 кадров/с), однако точность её значительно ниже (примерно на 8%).

**Модификации.**

1. YOLO v2 (YOLO9000)  
   Рассчитывается смещение *«якорей»* (описаны выше для Faster RCNN) с целью уточнения положения ограничительных прямоугольников.  
   Решена проблема детекции небольших объектов: в YOLOv2 изображение делится на сетку размером 13\*13.  
   Сеть YOLOv2 тренируется на изображениях с различным разрешением от 320\*320 до 608\*608, что позволяет находить объекты на изображениях с различными разрешениями с более высокой точностью.
2. YOLOv3  
   Используются логистические классификаторы вместо softmax слоя, что позволяет использовать мульти-метки (например, если для одного ограничительного прямоугольника вероятность, что внутри него содержится животное, будет равна 0.89, а вероятность того, что в нём содержится собака, равна 0.83, то для этого прямоугольника объект будет обозначен одновременно и как животное, и как собака)

# 2.5 Single-Shot Detector (SSD)

**Входные данные.**

Изображение фиксированного размера (300x300 для SSD300; максимальное разрешение — 512x512 для SSD512)

**Выходные данные.**

Карта объектов: каждое место этой карты хранит информацию об объекте (класс), который здесь содержится, и об ограничивающей рамке. Например, SSD512 выдаёт 7 карт объектов с разрешениями 64x64, 32x32, 16x16, 8x8, 4x4, 2x2, и 1x1. Для SSD300 информацию о выходных данных можно увидеть на рисунке ниже.

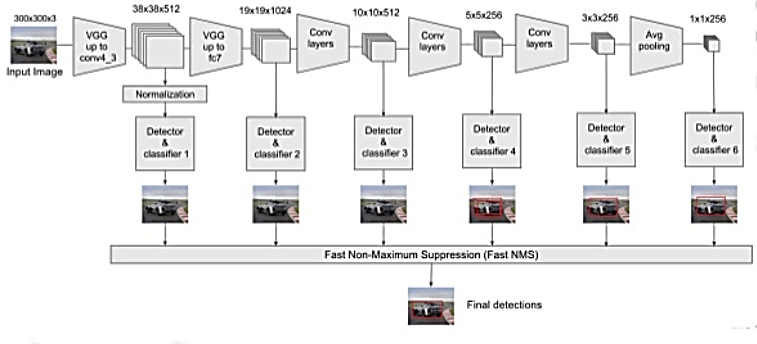


Рисунок 8. Схема работы SSD.

**Производительность.**

45 кадров/с.

**Алгоритм.**

Рассчитывается смещение *«якорей»* с целью уточнения положения ограничительных прямоугольников. Изображение обрабатывается сетью с различными фильтрами (10x10, 5x5, 3x3). Карты признаков, получаемые в разных частях сети, обрабатываются фильтрами 3x3 и формируют ограничительные рамки (подобно Faster RCNN). Каждая рамка имеет пять параметров: координаты центра, ширину, высоту, а также вектор соответствия каждому классу (подобно YOLO).

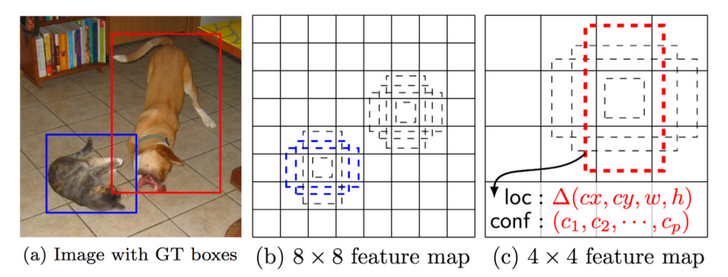


Рисунок 9. Поиск координат ограничительных прямоугольников с помощью якорей в SSD

# 2.6 Выводы

YOLO и SSD несомненно демонстрируют хорошее соотношение между скоростью и точностью, но содержат в себе существенный недостаток, который, возможно, не так уж и критичен для нашей задачи: трудности с обнаружением маленьких объектов. Пока не ясно, насколько далеки будут от камеры лица, которые необходимо будет обнаружить.

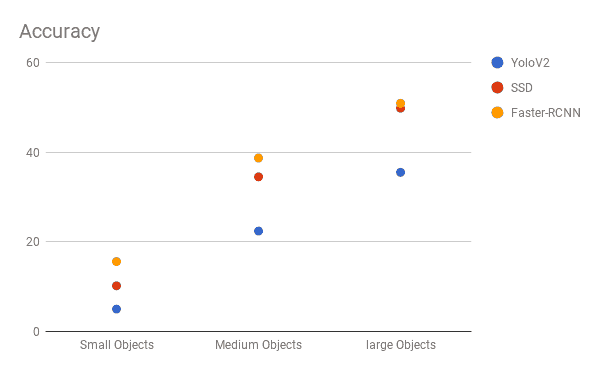
Faster RCNN пусть и работает медленнее, но показывает более высокую точность. Время обработки изображения этим алгоритмом вполне укладывается в ограничения.

Ниже приведены результаты сравнения описанных методов на датасете VOC2007 с помощью метрики mAP (Рис. 10) и на датасете COCO с помощью метрик precision, recall, IoU (Рис. 11).

Рисунок 10. Сравнение методов на датасете VOC2007.



Figure 11. Сравнение методов на датасете COCO.



3. Feature extraction

# 3.1 Модели сверточных сетей

## 3.1.1 VGG-16, 19

Модели от Оксфордской лаборатории, которые придерживаются цели исследования того, как глубина сверточной сети влияет на их точность в настройке распознавания больших изображений [4].

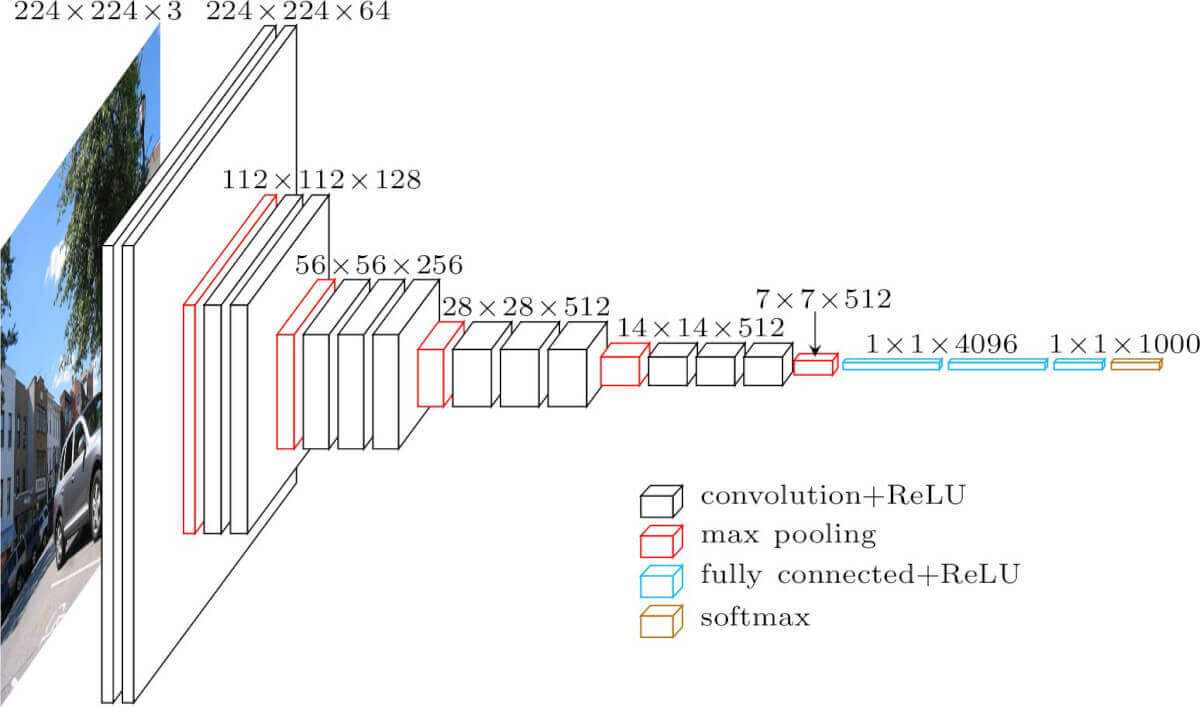


Рисунок3. Архитектура VGG-16

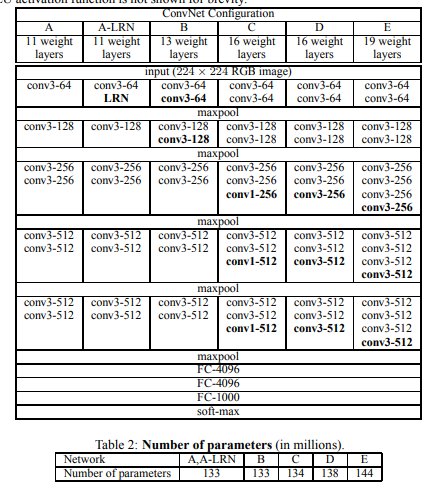


Рисунок 4.Различные конфигурации сетей VGG-16,19

performance: [5]

(вход: 224х224, выход:1000)

## 3.1.2 Inceptionv1,2,3

В первой версии модели исследователи Google руководствовались принципом увеличения ширины нейронной сети в каждом слоепротив увеличения глубины(слоев).Разработали идею Inceptionмодулей для расширения сети в ширину.

Использовались несколько каналов сверточных слоев для входа, результаты каналов объединялись и таким образом происходила фильтрация.

Первая версия модели (27 слоев) подвержена проблеме "исчезающего" градиента.

Во второй версии исследователи предложили ряд улучшений, которые повысили точность и уменьшили вычислительную сложность.

В третьей версии несколько улучшений, связанных с точностью и избегания перенасыщения.

(вход: 299х299, выход: 1000)

performance: [6]

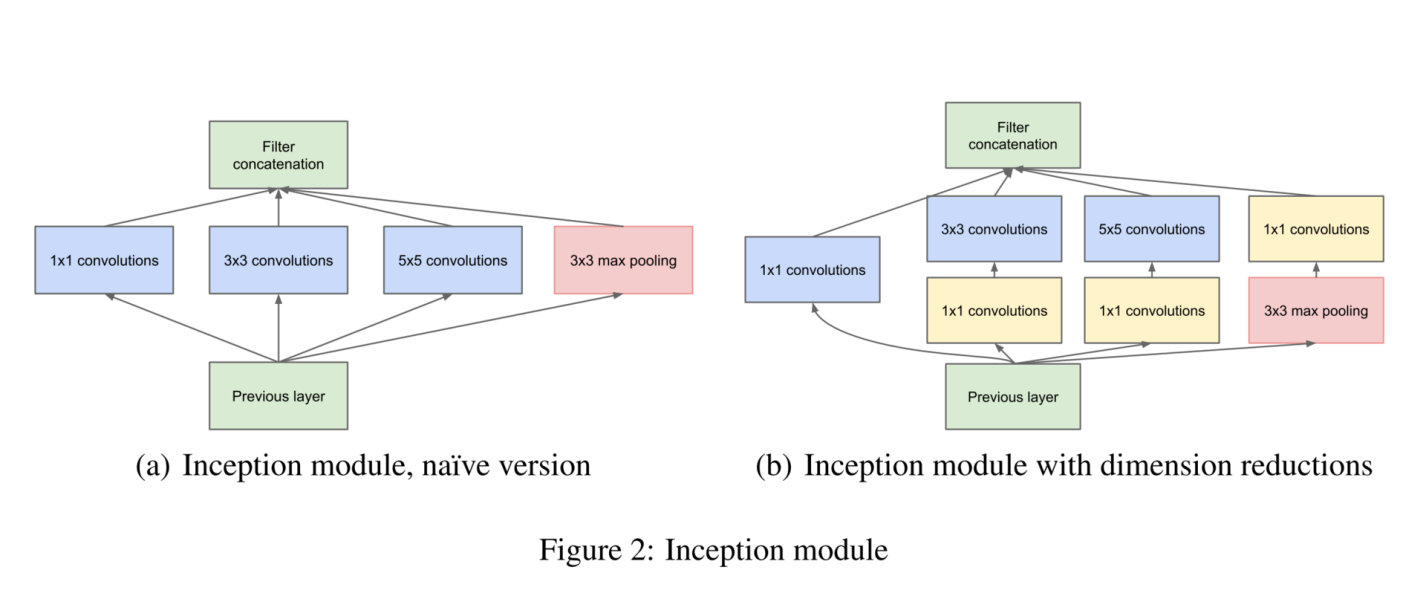


Рисунок 5.Модуль Inception

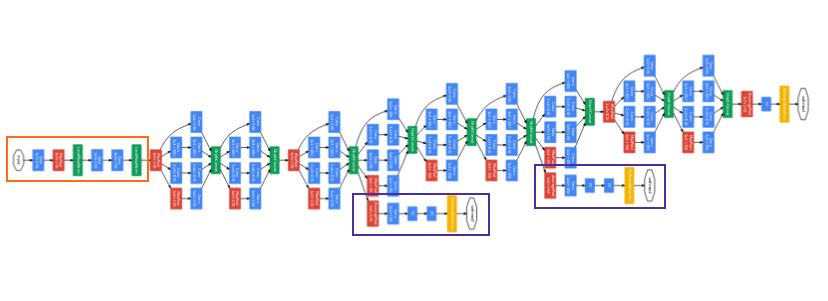


Рисунок 6.Архитектура Inception v1

## 3.1.3 ResNet \*

Достаточно глубокое семейство сетей (18, 34, 50, 101, 152 слоя(й)). Как известно большое количество слоев приводит к перенасыщению сети и к проблеме "исчезающего" градиента (Vanishing Gradient Problem). Создатели данной сети решили данную проблему с помощью так называемых shortcut-соединений, которые позволяют "пропускать" некоторые скрытые слои.

1) входные данные: (224, 224)

2) выходные данные: 1000-чи мерный вектор.

3) производительность (Рис. 1, 2)

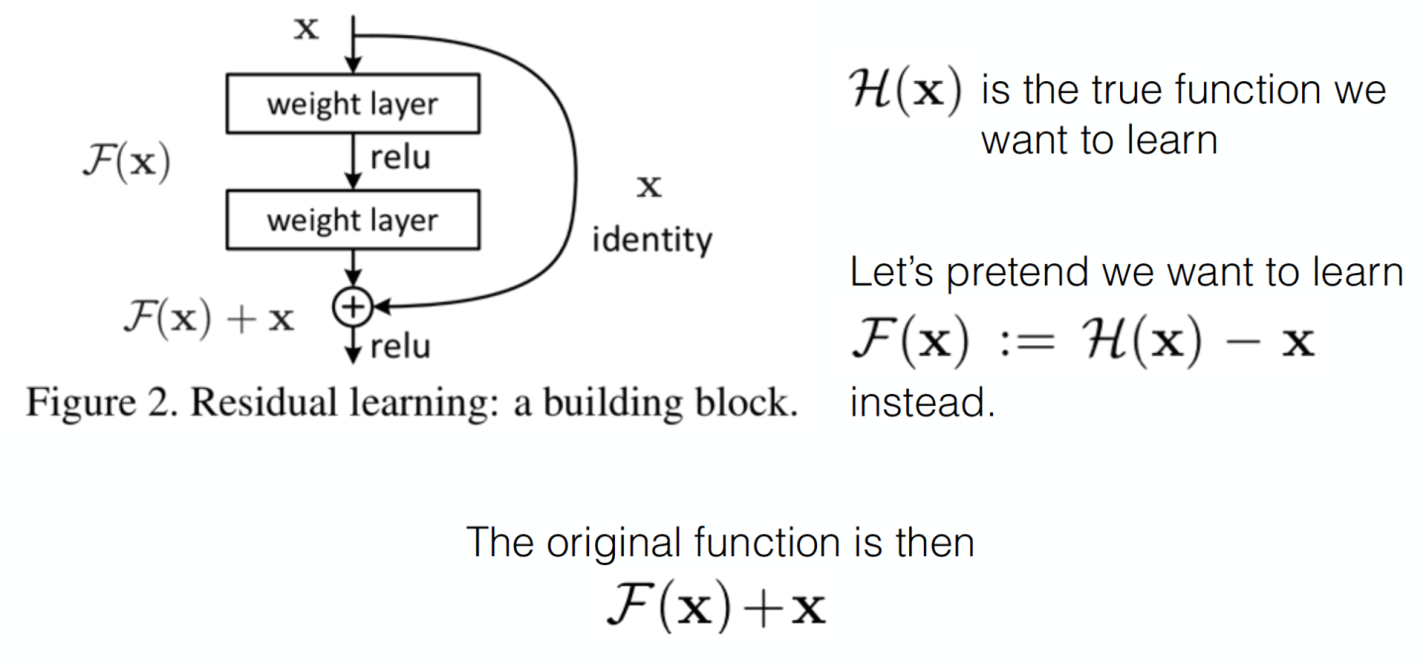


Рисунок7. Shortcut соединения

## 3.1.4 Inception ResNet v1,2

Была придумана исследователями Google, заимствовав идеи ResNet c shortcut-соединениями.

Результатом статьи становятся модели InceptionV4 и InceptionResNet.

InceptionV4 не особо отличается от предыдущих поколений, только фактом использования ResNet [3].

1) входные данные: (220, 220)

2) выходные данные: 1000-ми мерный фиче вектор.

3) производительность (Рис. 1, 2, 9)

## 3.1.5 OpenFace models (основанынаGoogle’s FaceNet [Inception ResNet])

Предлагают точности распознавания сравнимые с популярными сетями распознавания лиц.

Предлагают достаточно высокоуровневый API для feature extraction (и для всего пайплайна распознавания лиц).

Есть возможность заимпортить предобученые модели и использовать их без API:

1) входные данные: (96, 96)

2) выходные данные: 128-ми мерный фиче вектор.

3) производительность:

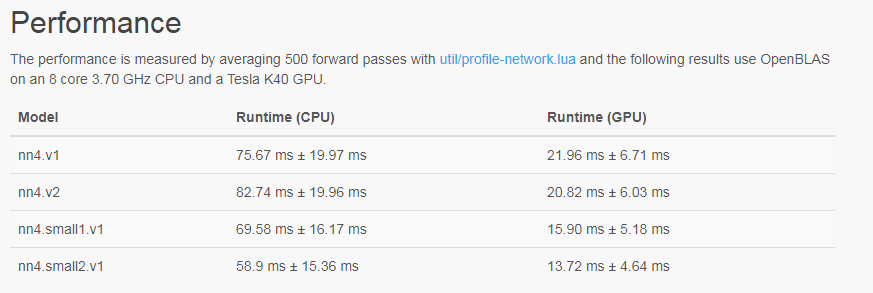


Рисунок 8.Производительность предобученых моеделей OpenFace [7]

## 3.1.6 Выводы

Существуют уже обученные реализации модели **Inception ResNet** с достаточно неплохой точностью ~99% на датасете CASIA-WebFace [1].

Кроме того библиотека dlib представляет весь требуемый пайплайн распознавания лиц, включая фиче экстрашен (dlib.face\_recognition\_model\_v1) на базе модели ResNet.

Для совсем высокоуровнего подхода можно использовать APIOpenFace.

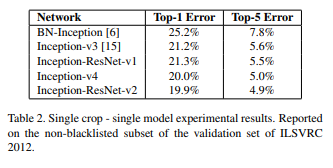


Рисунок 9. Результаты экспериментов на подмножестве датасета ILSVRC 2012

# 3.2 Методы расчета расстояния между признаками

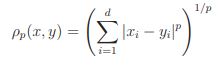
## 3.2.1 Евклидово расстояние

Евклидово расстояние есть геометрическое расстояние между многомерными точками в многомерном пространстве. Для точек xи yв n-мерном пространстве оно рассчитывается так:

# ÐµÐ²ÐºÐ»Ð¸Ð´Ð¾Ð²Ð¾ ÑÐ°ÑÑÑÐ¾ÑÐ½Ð¸Ðµ ÑÐ¾ÑÐ¼ÑÐ»Ð°

## 3.2.2 Метрика Минковского

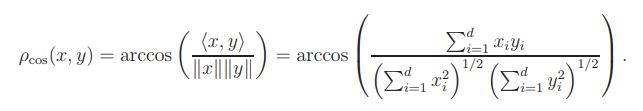
Является обобщением Евклидового расстояния при p=2



, где p>=1. При разном параметре pполучаются такие метрики как: Манхэттенское расстояние, Метрика Чебышева.

## 3.2.3 Косинусная метрика

Расстоянием в данной метрике является угол между двумя многомерными точками.



Используется для определения расстояния между признаками в угловом пространстве с сетями, обученными на таких функциях потерь как: SphereFace[[1]](#footnote-1), CosFace[[2]](#footnote-2), ArcFace[[3]](#footnote-3).

4 Методы выравнивания лиц

# 4.1 Общее описание метода

Метод поворота лица с использованием оценки ориентиров лица (face landmark estimation).

Будем использовать подход, изобретенный в 2014 году Вахидом Каземи и Джозефин Салливан. Основная идея заключается в том, что мы найдем 68 конкретных точек (называемых ориентирами), которые существуют на каждом лице - верхняя часть подбородка, внешний край каждого глаза, внутренний край каждой брови и т.д. Затем мы обучим модель, чтобы иметь возможность найти эти 68 конкретных точек на любом лице.

Для нахождения меток используется рекуррентная сеть, которая обеспечивает сквозное обучение каскадной структуры. На каждом этапе сеть принимает в качестве входных данных патчи вокруг ориентиров, предсказанных предыдущим этапом. Но, с другой стороны, с помощью рекуррентной структуры каждый этап также принимает в качестве входных данных функции скрытого уровня предыдущего этапа. Следовательно, градиент может распространяться от последней ступени к первой через связь между скрытыми слоями смежных ступеней. Таким образом, позволяется проводить сквозное обучение всей сети, а не обучать все этапы по отдельности.

Теперь, когда мы знаем, координаты глаз и рта, мы будем вращать, масштабировать и сдвигать изображение так, чтобы глаза и рот были отцентрированы как можно лучше. Мы не будем делать трехмерных деформаций, потому что это внесет искажения в изображение. Мы будем использовать только базовые преобразования изображений, такие как вращение и масштабирование, которые сохраняют параллельные линии (так называемые аффинные преобразования)

Теперь независимо от того, как повернуто лицо, мы можем центрировать глаза, а рот находится примерно в одном и том же положении на изображении. Это сделает распознавание лиц более точным.

# 4.2 Нахождение меток на лице

## Обзор полностью сквозной каскадной сверточной нейронной сети для обнаружения лицевых ориентиров:

Сеть содержит несколько подчиненных CNN, каждая из которых в качестве входных данных принимает патчи вокруг ориентиров, предсказанных предыдущей стадией. S0 обозначает начальную форму расположения ориентиров, ΔSt, t = 1, ..., T обозначает ошибку t-й формы, а St, t = 1, ..., T обозначает t-ю уточненную форму, которая является суммой St − 1 и ΔSt. Вся сеть полностью обучается, распространяясь вниз по градиенту через каждую стрелку, показанную на рисунке.

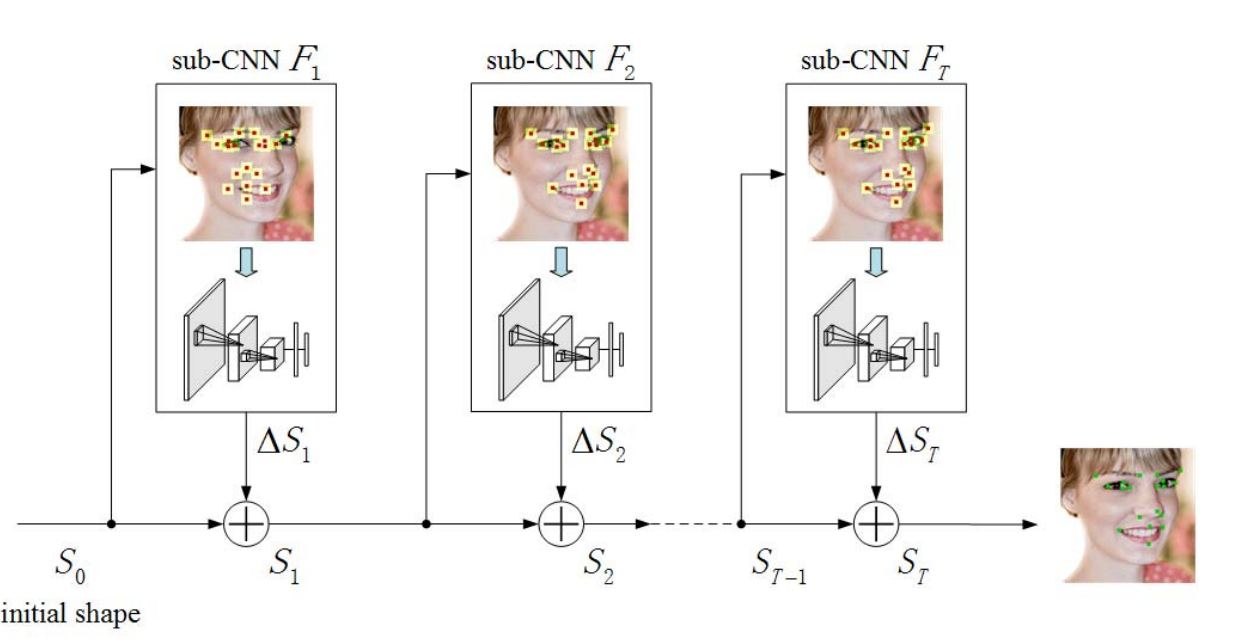


Рис. 1 Иллюстрация сквозной каскадной сверточной нейронной сети для обнаружения лицевых ориентиров.

Сначала вычисляется градиент параметров, а затем параметры обновляются в направлении снижения градиента.

Как видно из рисунка, структуры всех стадий одинаковы, и поэтому процессы обратного распространения градиента всех стадий также одинаковы.

# 4.3 Выравнивание на основе меток

Собственно, поворот лица на основе 68 меток реализован в библиотеке openface. В ней имеется 2 режима поворота лиц: INNER\_EYES\_AND\_BOTTOM\_LIP и OUTER\_EYES\_AND\_NOSE, которые основаны на выравнивании основных точек для выравнивания (внутренние углы глаз и нижняя точка на губе; и внешний угол глаз и нос соответственно).

Используются методы поворота с помощью треугольника, обозначенного тремя точками, что упоминались выше. Используя методы OpenCV, находится необходимое афинное преобразование, чтобы повернуть координаты исходного треугольника в конечные координаты, одинаковые для каждого изображения.

Находится матрица преобразования.

, где src(i) = (xi, yi) – исходные координаты трех ключевых точек, dst(i) = (x’i, y’i) – целевые координаты точек (i = 0, 1, 2), а map\_matrix – собственно, матрица преобразования размерности 2×3.

После, согласно этой матрице, используя опять же методы OpenCV, все изображение поворачивается.

Они выравниваются до одинакового расположения трех ключевых точек для всех изображений, что облегчает нам будущий процесс распознавания.

# 4.4 Выводы

В работе будем использовать выравнивание лиц с помощью 68 меток, найденных по методу Каземи и Салливана. Для этих целей будем использовать библиотеки dlib и OpenCV, а также созданную обертку специально для наших целей openface.

5 Нефункциональные требования к идентификации лиц

Система может распознавать изображения форматов “.jpg” или “.png” различного разрешения в диапазоне от 300x300 до 4000x4000 пикселов, удовлетворяющие условию: минимальный размер лица 100х100 пикселов, при изменении исходного разрешения на разрешение в 300x300 пикселов, на лица людей, присутствующие на изображении, приходится область не менее 20х20 пикселов. Отсутствует числовая недостаточность или избыточность цвета изображения, т.е. нет слишком тёмных или слишком светлых участков. Область лица от линии бровей до начала подбородочной области не перекрыта посторонними объектами, цвет кожи не подвержен изменениям, лицо различимо, т.е. находится в фокусе/не размыто.

5.1 Требования по точности:

5.1.1 Значение метрики mAP[[4]](#footnote-4) должно быть не менее 0.8. Количество ошибок детекции должно составлять не более 1% от общего количества изображений. Под ошибками детекции понимается определение такой области на изображении, для которой значение параметра IOU[[5]](#footnote-5) относительно ground truth[[6]](#footnote-6) bounding box каждого лица на изображении меньше 0,5

5.1.2 Количество правильно детектированных лиц должно составлять не менее 98% от суммарного числа всех лиц на всех изображениях. Лицо считается детектировано правильно, если значение параметра IOU2 относительно его ground truth3 bounding box больше или равно 0,5.

5.1.3 Точность распознавания лиц должна составлять не менее 97% от всех правильно детектированных лиц.

5.2 Требования по скорости:

5.2.1 ПО «AFR» должно обрабатывать изображение, на котором имеется не более трех лиц, не более чем за 20 секунд.

5.2.2 Входное изображение ПО «AFR» должно содержать не более 20 лиц.

5.3 Система должна корректно обрабатывать изображения на которых лица повернуты:

* 1. До 45% влево/вправо
  2. До 15% вверх/вниз
  3. До 30% по/против часовой стрелки

6. Тестовый базис

6.1 Набор тестовых данных

Задача распознавания лиц затрагивает две широкие проблемы: локализация лица на изображении и последующая его идентификация. Для решения таких задач необходимы данные, на которых будет обучаться решающий их алгоритм, включающие в себя информацию, необходимую для идентификации лиц, а именно примеры их изображения. Ни один из существующих наборов данных такой информацией не обладает, поэтому возникает необходимость его создания.

6.2 Назначение и область применения

Набор данных используется для обучения, валидации и тестирования алгоритма распознавания лиц.

6.3 Технические характеристики

В набор данных входят изображения формата “.jpg” . Разрешение каждого лица превышает 100x100 пикселей. Допустимы изображения лиц, отклоненных от горизонтальной оси не более чем на 45° в левую и правую стороны, а также поворот/наклон лица от положения анфас не более чем на 45° по всем направлениям (вправо, влево, вверх, вниз).

Набор состоит из следующих частей:

* 1. Данные для обучения нейронной сети.

Содержит подкаталоги с изображениями разных классов: по 50 изображений лиц каждого из людей, которых необходимо распознать; 250 изображений лиц людей, отличных от тех, кого требуется распознать; 500 изображений, на которых отсутствуют лица.

* 1. Данные для тестирования и валидации, необходимые для контроля качества сети в процессе и после окончания её обучения.

Содержит подкаталоги с изображениями разных классов: по 50 изображений лиц людей, которые необходимо распознать; 300 изображений лиц людей, отличных от тех, кого требуется распознать; 100 изображений, на которых отсутствуют лица.

1. [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)
3. [↑](#footnote-ref-3)
4. mАР (mean average precision) - Gordon V. Cormack and Thomas R. Lynam David R. Cheriton School of Computer Science University of Waterloo Waterloo, Ontario N2L 3G1, Canada «Cheriton Statistical Precision of Information Retrieval Evaluation» [↑](#footnote-ref-4)
5. IOU(intersection over union or jaccard distance) - Sven Kosub Department of Computer & Information Science, University of Konstanz Box 67, D-78457 Konstanz, Germany «A note on the triangle inequality for the Jaccard distance» [↑](#footnote-ref-5)
6. Ground truth – Krig, Scott «Computer Vision Metrics Survey, Taxonomy, and Analysis» [↑](#footnote-ref-6)