|  |  |
| --- | --- |
| **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. |

**Пояснительная записка №3**

**«ПЗ по feature extraction (ПЗ3)»**

**Этап 1. Подготовка обзоров на существующие подходы к решению   
задачи и построение технологического стека**

**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 Введение 3](#_Toc5491345)

[2 Модели сверточных сетей 4](#_Toc5491346)

[2.1 VGG-16, 19 4](#_Toc5491347)

[2.2 Inception v1,2,3 5](#_Toc5491348)

[2.3 ResNet \* 6](#_Toc5491349)

[2.4 InceptionResNet v1,2 7](#_Toc5491350)

[2.5 OpenFace models (основаны на Google’s FaceNet [Inception ResNet]) 7](#_Toc5491351)

[2.6 Выводы 7](#_Toc5491352)

[3 Методы расчета расстояния между признаками 8](#_Toc5491353)

[3.1 Евклидово расстояние 8](#_Toc5491354)

[3.2 Метрика Минковского 8](#_Toc5491355)

[3.3 Косинусная метрика 8](#_Toc5491356)

[4 Список источников 9](#_Toc5491357)

# Введение

В данной пояснительной записке мы рассмотрим ряд моделей сверточных нейронных сетей, которые наиболее популярны в задачах распознавания лиц на текущий момент. Задача извлечения признаков выполняется нейронными сетями. Входные данные для каждой реализации нейронный сети - картинка любого разрешения и одно из форматов (jpg, png) так как в непосредственно перед выполнением распознавания есть возможность подкорректировать до требуемого разрешения для конкретной модели сети. Кроме того, мы рассмотрим способы расчета расстояния между признаками.

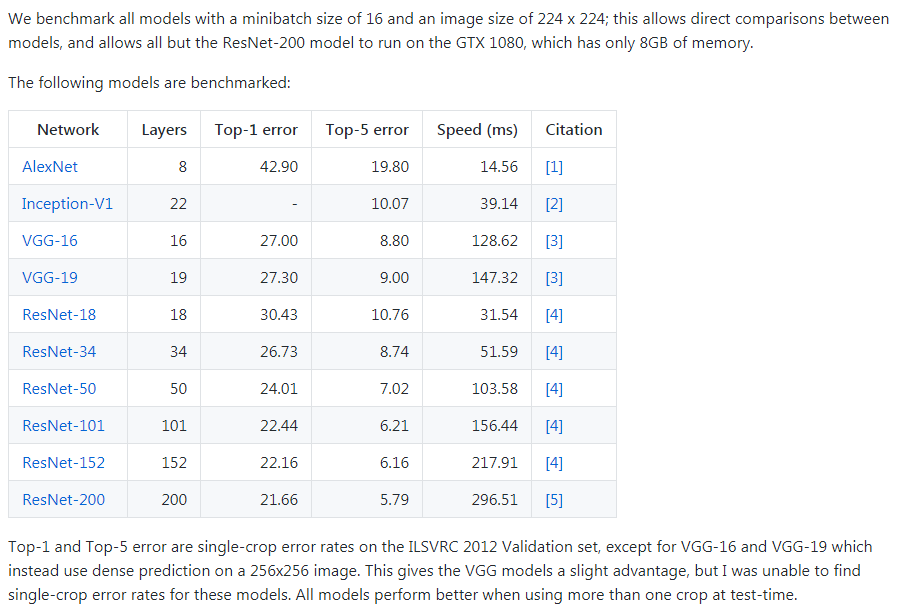


Рисунок 1 Top-1, Top-5 ошибки и скорость распознавания на каждой из реализаций нейронных сетей используя набор из 15 изображений разрешением 224х224

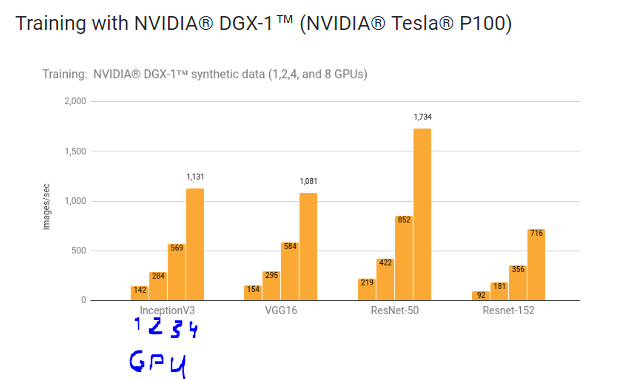


Рисунок 2 Количество изображений в секунду на разных архитектурах, на разных количествах графических процессоров

# Модели сверточных сетей

## 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. <http://www.robots.ox.ac.uk/~vgg/research/very_deep/>

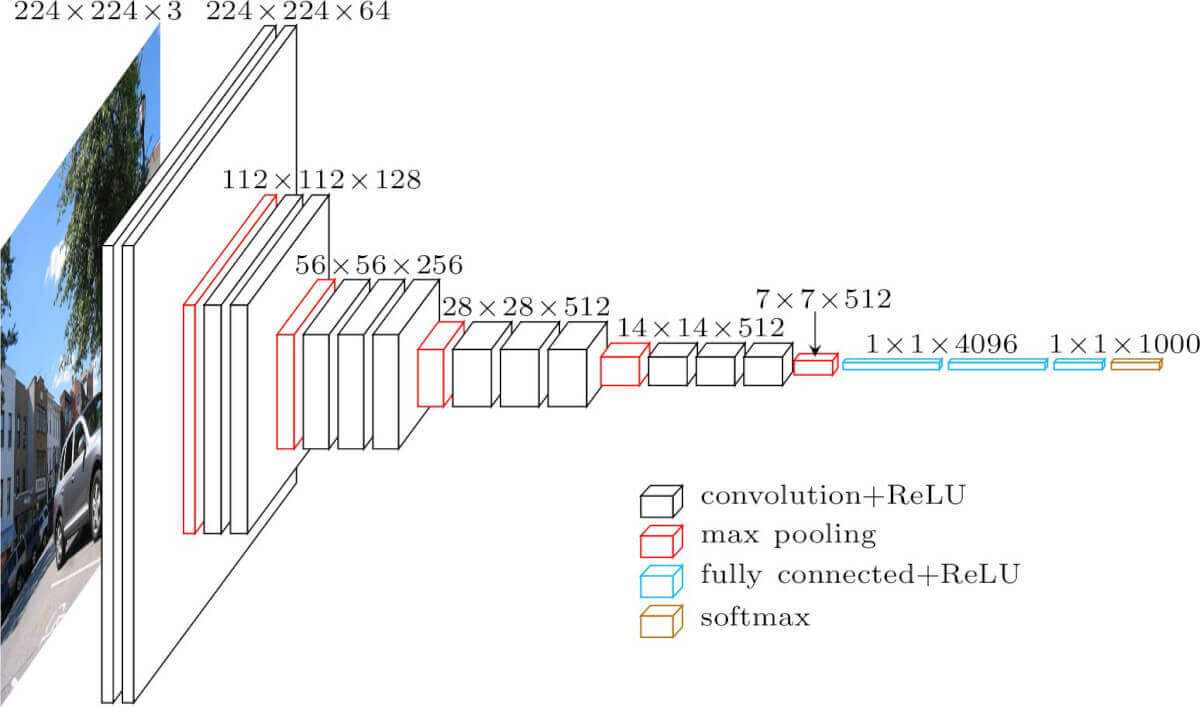


Рисунок Архитектура VGG-16

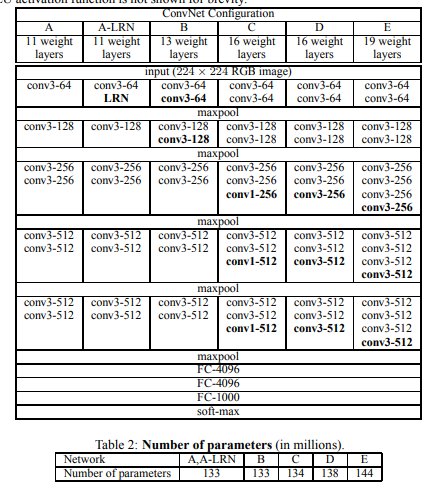


Рисунок 4 Различные конфигурации сетей VGG-16,19

performance: [5]

(вход: 224х224, выход:1000)

## Inception v1,2,3

В первой версии модели исследователи Google руководствовались принципом увеличения ширины нейронной сети в каждом слое, напротив увеличения глубины(слоев). Разработали идею Inception модулей для расширения сети в ширину.

Использовались несколько сверточных слоев для входа, они 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]

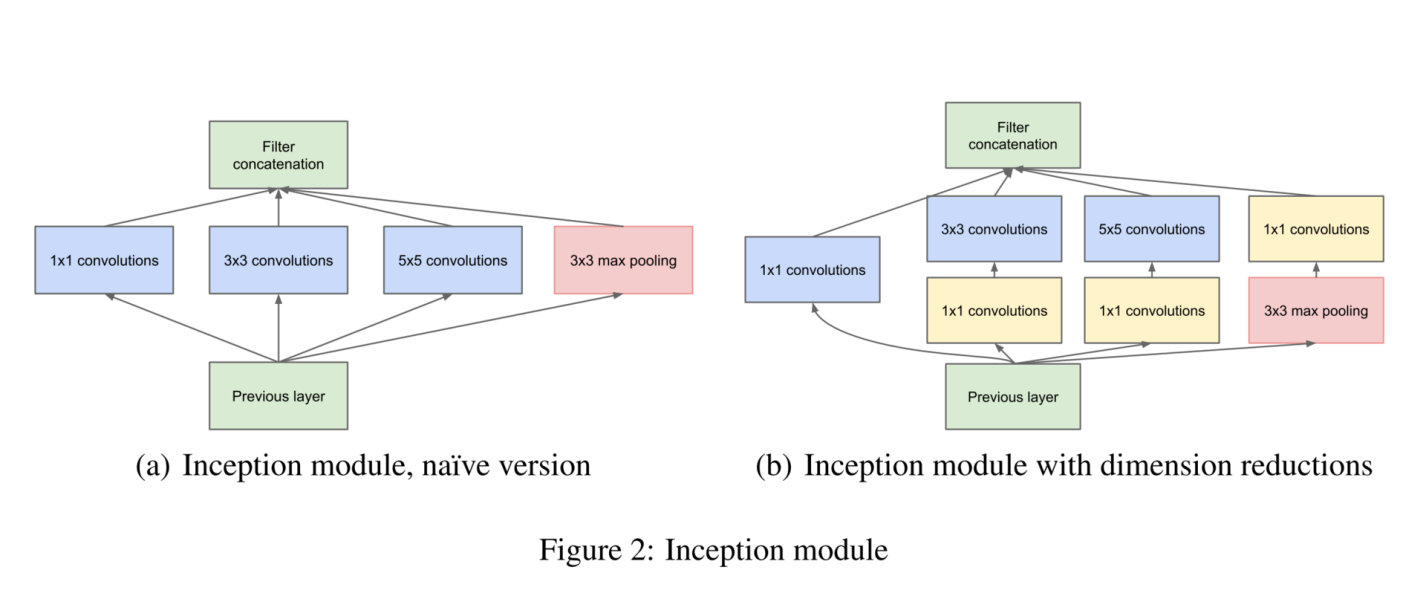


Рисунок Модуль Inception

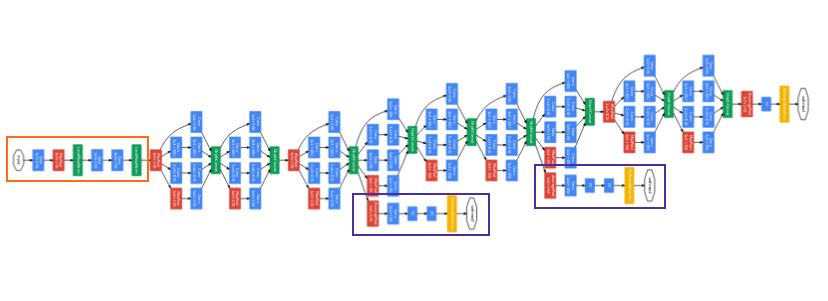


Рисунок Архитектура Inception v1

## 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 (VanishingGradientProblem). 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 фиче вектор.

3) производительность: выше на картинке для некоторых сетей из семейства

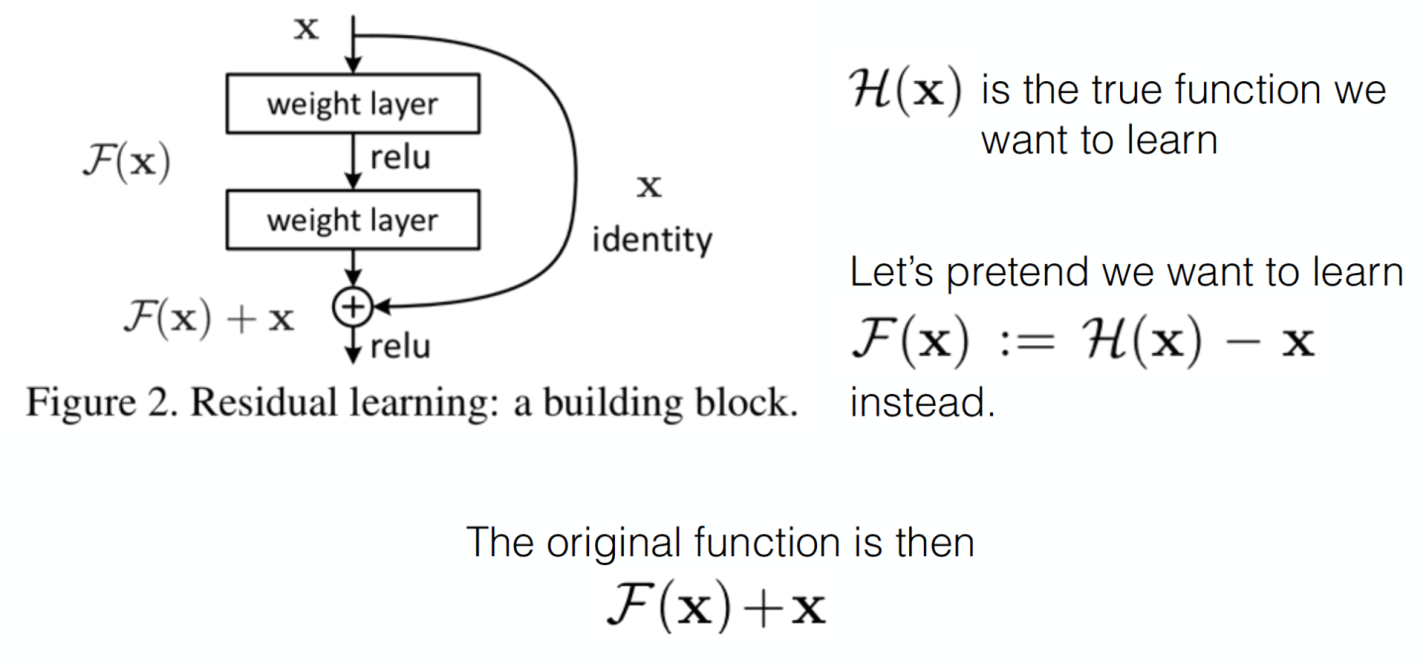


Рисунок Shortcut соединения

## InceptionResNet 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: выше на картинке для некоторых сетей из семейства

## OpenFace models (основаны на Google’s FaceNet [Inception ResNet])

Предлагают точности распознавания сравнимые с популярными сетями распознавания лиц.

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

Есть возможность заимпортитьпредобученые модели и использовать их без API:

1) входные данные: (96, 96)

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

3) производительность:

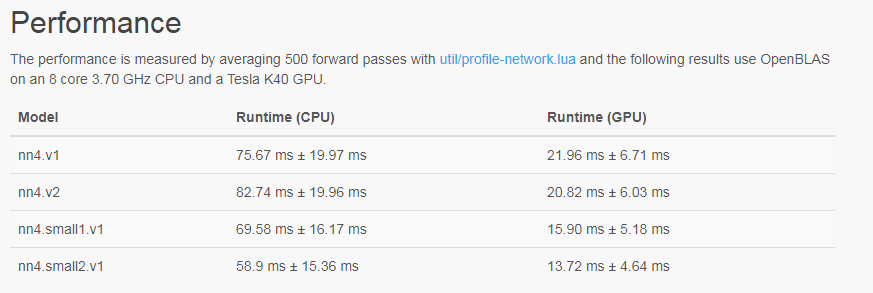


Рисунок 8 Производительность предобученых моеделей OpenFace[7]

## Выводы

Существуют уже обученные реализации модели **InceptionResNet** 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 API OpenFace.

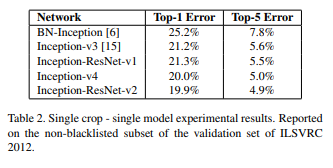


Рисунок 9 Результаты эксперементов на подмножестве датасета ILSVRC 2012

# Методы расчета расстояния между признаками

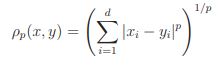
## Евклидово расстояние

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

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

## Метрика Минковского

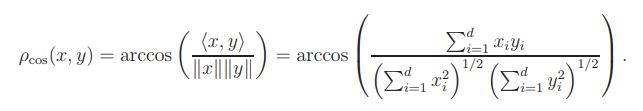
Является обобщением Евклидового расстояния при p=2



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

## 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-2), CosFace[[2]](#footnote-3), ArcFace[[3]](#footnote-4).

# Список источников

1. <https://github.com/davidsandberg/facenet>
2. [http://dlib.net/](http://dlib.net/python/index.html%23dlib.face_recognition_model_v1)
3. [<https://arxiv.org/abs/1602.07261>](https://arxiv.org/pdf/1602.07261.pdf)
4. <http://www.robots.ox.ac.uk/~vgg/research/very_deep/>
5. <https://github.com/jcjohnson/cnn-benchmarks>
6. <https://developer.nvidia.com/deep-learning-performance-training-inference>
7. [https://cmusatyalab.github.io/openface/models-and-accuracies/#pre-trained-models](https://cmusatyalab.github.io/openface/models-and-accuracies/%23pre-trained-models)

1. [Liu et al, "SphereFace: Deep Hypersphere Embedding for Face Recognition", 2018](https://arxiv.org/abs/1704.08063) [↑](#footnote-ref-2)
2. [Wang et al, "CosFace: Large Margin Cosine Loss for Deep Face Recognition", 2018](https://arxiv.org/abs/1801.09414) [↑](#footnote-ref-3)
3. [Deng et al, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition", 2018](https://arxiv.org/abs/1801.07698) [↑](#footnote-ref-4)