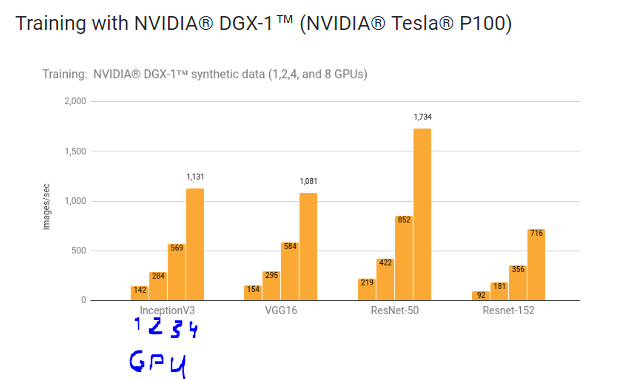
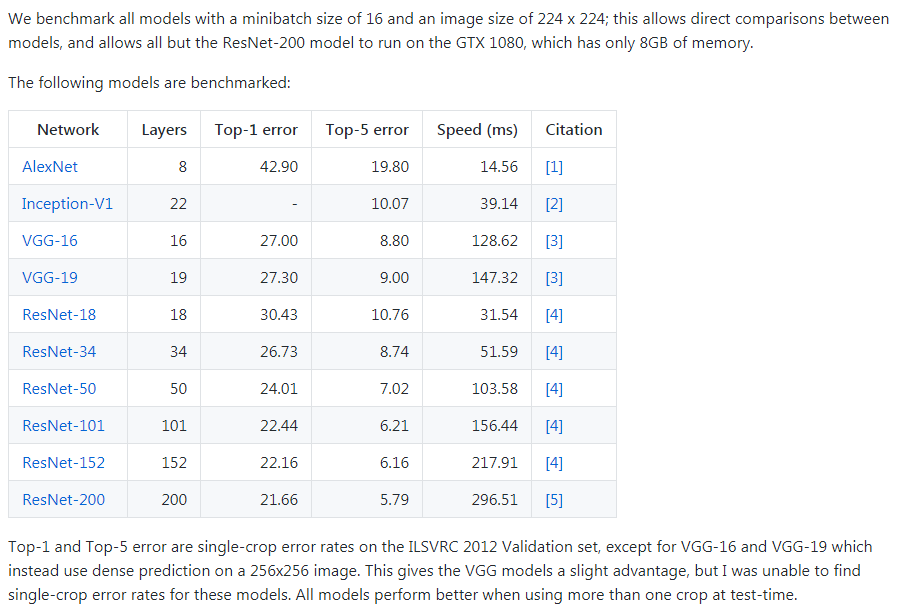
Входные данные для любой: по сути картинка любого разрешения и одно из форматов (jpg, png) так как в коде можно подкорректировать разрешение под нужное для конкретной модели сети. 

**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/>

performance: <https://github.com/jcjohnson/cnn-benchmarks>

(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).

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: <https://developer.nvidia.com/deep-learning-performance-training-inference>

**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: 128-ми мерный фиче вектор.

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

**Inception ResNet v1,2**

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

Результатом статьи становятся модели Inception V4 и Inception ResNet.

Inception V4 не особо отличается от предыдущих поколений, только фактом использования ResNet.

статья: <https://arxiv.org/pdf/1602.07261.pdf>

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.

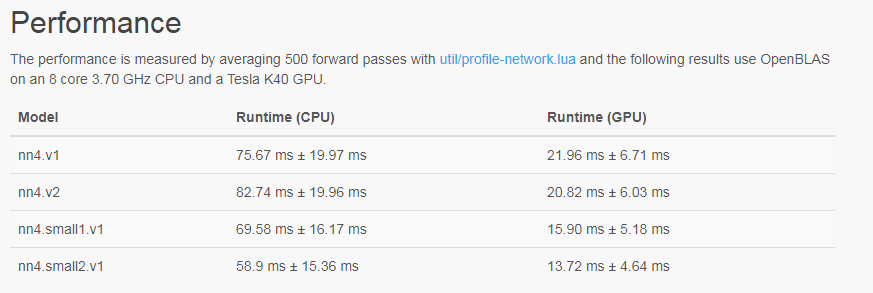
Предлагают достаточно высокоуровневый API для feature extraction (и для всего пайплайна распознавания лиц).

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:

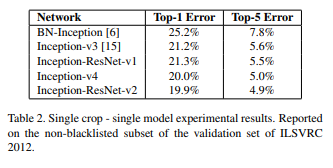
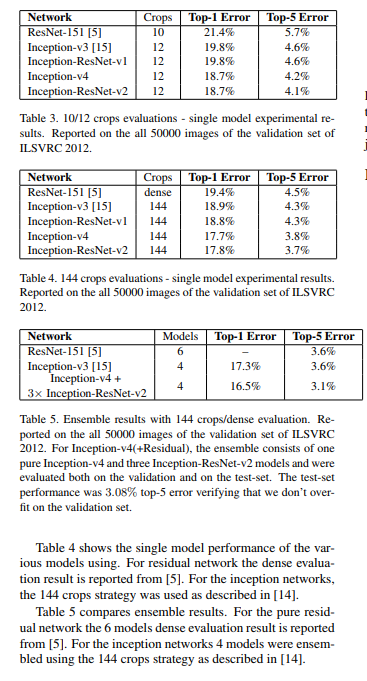
[https://cmusatyalab.github.io/openface/models-and-accuracies/#pre-trained-models](https://cmusatyalab.github.io/openface/models-and-accuracies/%23pre-trained-models)

**Личные предпочтения и по количеству готовых реализаций:**

Существуют уже обученные реализации модели **Inception ResNet** с достаточно неплохой точностью ~99%, но тут требуется среднее количество знаний для корректной реализации (TenserFlow). <https://github.com/davidsandberg/facenet>

Кроме того библиотека dlib(<http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html>) представляет весь требуемый пайплайн распознавания лиц, включая фиче экстрашен ([http://dlib.net/python/index.html#dlib.face\_recognition\_model\_v1](http://dlib.net/python/index.html%23dlib.face_recognition_model_v1)) на базе модели ResNet.

Для совсем высокоуровнего подхода можно использовать API OpenFace.

****